Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень»

Виконав: студент 2 курсу, групи 2ПІ-22м спеціальності 121 – Інженерії програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Матвійчук О. В.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри ПЗ
Катєльніков Д. І.
(прізвище та ініціали)
«08» грудня 2023 р.

Рецензент: к.т.н., доцент кафедри ОТ
Колесник І. С.
(прізвище та ініціали)
«08» грудня 2023 р.

Допущено до захисту
Зав. кафедри ПЗ Романюк О. Н.
«08» грудня 2023р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти другий (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«19» вересня 2023 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Матвійчуку Олександру Васильовичу

1. Тема роботи – «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень»

Керівник роботи: Катєльніков Денис Іванович, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Термін подання студентом роботи : 5 грудня 2023 р.

3. Вихідні дані: тип квантування – цілочислове, векторне; тип перетворення: дискретне косинус не перетворення, перетворення Уолша-Адамара; операційна система –Windows; очікуваний коефіцієнт ущільнення зображення – 3-4; середовище розробки – MS Visual Studio; мова програмування – C/C++.

4.Зміст текстової частини: вступ; аналіз застосування дискретних ортогональних перетворень для ущільнення зображень; розробка методів і алгоритмів квантування коефіцієнтів двовимірних ортогональних перетворень; розробка модулів квантування та зонального відбору коефіцієнтів ДОП; експериментальні дослідження квантування компонент зображення; економічна частина; висновки; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): мета і задачі роботи; наукова новизна та практична цінність роботи; схема ущільнення зображень на основі дискретних ортогональних перетворень; дискретне косинусне перетворення та перетворення Уолша-Адамара; блок-схема алгоритму групування трансформант дискретних ортогональних перетворень; блок-схема алгоритму векторного квантування згрупованих трансформант; вікна програми; класи і методи додатку; результати досліджень групування трансформант; дослідження векторного квантування компонент зображення; формати файлів з трансформантами перетворення; основні результати роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	завдання прийняв
1–4	Катєльніков Д. І., доц. каф. ПЗ, к.т.н	19.09.2023	05.12.202
5	Кавецький В. В., доц. каф. ЕПВМ, к.т.н	11.11.2023	25.11.202

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23	вип
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23	вип
3	Аналіз і вибір мови програмування та середовища розробки	16.10.23 – 23.10.23	вип
4	Розробка програмного продукту	24.10.23 – 13.11.23	вип
5	Тестування програми	14.11.23 – 21.11.23	вип
6	Розробка економічної частини	17.11.23 –25.11.23	вип
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23	вип

Студент **Матвійчук О. В.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи **Катєльніков Д. І.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.932

Матвійчук О. В. Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. - 98 с.

На укр. мові. Бібліогр.: 31 назва; рис.: 18; табл.: 14.

Метою роботи є підвищення коефіцієнту ущільнення зображень з використанням дискретних ортогональних перетворень за рахунок оптимізації методів квантування компонент зображення.

У роботі проведено детальний аналіз методів і засобів кодування зображень та методів квантування компонент зображення. Отримав подальший розвиток метод кодування зображень, який ґрунтується на застосуванні двовимірних ортогональних перетворень, цілочисловому квантуванні та групуванні трансформант цих перетворень. Показано, що подальшого підвищення коефіцієнта ущільнення можна досягти за рахунок векторного квантування компонент зображення.

З використанням середовища програмування Microsoft Visual Studio мовою програмування C++ розроблено програмне забезпечення для дослідження методів квантування компонент зображення з використанням перетворення дискретного косинусного перетворення та перетворення Уолша-Адамара. При прийнятній якості об’єм зображення зменшується в 3-5 раз.

Ключові слова: Windows, Visual Studio, C/C++, ущільнення зображень, перетворення Уолша-Адамара, дискретне косинусне перетворення, квантування, векторне квантування, програма.

ABSTRACT

Matviychuk O. V. Methods and software tools for quantization of components of two-dimensional orthogonal transformations during image compression. Master's thesis on the specialty 121 - Software engineering, educational program - Software engineering. Vinnytsia: VNTU, 2023. - 98 p.
In Ukrainian language. Bibliographer: 31 titles; fig.: 18; tabl.: 14.

The aim of the work is to increase the image compression ratio using discrete orthogonal transformations due to the optimization of image component quantization methods.

The paper provides a detailed analysis of methods and means of image coding and methods of quantization of image components. The method of image coding, which is based on the application of two-dimensional orthogonal transformations, integer quantization and grouping of transformants of these transformations, was further developed. It is shown that a further increase in the compression ratio can be achieved due to vector quantization of image components.

Using the Microsoft Visual Studio programming environment in the C++ programming language, software was developed for researching image component quantization methods using the discrete cosine transform and the Walsh-Hadamard transform. With acceptable quality, the volume of the image is reduced by 3-5 times.

Keywords: Windows, Visual Studio, C/C++, image compression, Walsh-Hadamard transform, discrete cosine transform, quantization, vector quantization, program.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Коефіцієнт ущільнення і якість відновленого зображення залежать від вирішення задачі квантування коефіцієнтів двовимірних ортогональних перетворень (ДОП) [1-3].

Рівномірне квантування [4] використовує однаковий інтервал для всіх значень. Найкраще підходить для застосувань, де вимагається рівномірна точність.

Нерівномірне квантування використовує різні інтервали для різних діапазонів значень коефіцієнтів за рахунок чого покращується ефективність стиснення. Застосовується у випадках, коли деякі частини зображення важливіші за інші [5-9].

Квантування з втратами (Lossy Quantization) [2, 10-13] зменшує точність квантування для досягнення вищого ступеня стиснення. Втрачається частина інформації, але при цьому значно зменшується обсяг даних. Знаходить застосування У випадках, де деякі втрати якості припустимі (наприклад, відеостиснення).

Адаптивне квантування [2] застосовує різні рівні квантування для різних областей зображення. Дозволяє краще враховувати властивості окремих частин зображення. Застосовується при кодуванні графічних або відеоданих, де різні частини можуть мати різну важливість.

Дельта кодування - квантування виконується відносно попередніх значень (дельта). Зберігає лише різницю між поточним і попереднім значеннями. Застосовується в аудіо- та відеостисненні для ефективної роботи з сигналами [7].

В свою чергу нерівномірне квантування поділяється на декілька методів, наприклад [7-10]:

- компандування — це метод, при якому сигнал попередньо перетворюється за допомогою нелінійної функції, що збільшує малі значення і зменшує великі. Після цього застосовується рівномірне квантування, а потім

обернена функція компандування. Прикладами компандування є А-закон і μ-закон, які використовуються в телефонній мережі.

- квантування зі змінною довжиною кодового слова (VQ) — це метод, при якому рівні квантування представлені кодовими словами різної довжини, так що частіші значення мають коротші коди, а рідші — довші. Це дозволяє зменшити середню довжину коду і збільшити ступінь стиснення. Прикладами VQ є кодування Гаффмана і арифметичне кодування.

- квантування зі змінною роздільною здатністю (VRQ) — це метод, при якому рівні квантування мають різну точність, так що малі значення мають більшу роздільну здатність, а великі — меншу. Це дозволяє зменшити помилку квантування і покращити якість сигналу. Прикладом VRQ є кодування зі змінною швидкістю передачі (VBR), яке використовується в кодуванні мовлення.

Ці методи можуть застосовуватися як окремо, так і в комбінації для досягнення оптимального ступеня квантування та стиснення в залежності від конкретних вимог застосування.

У більшості випадків при кодуванні зображення розмірність значень багатьох коефіцієнтів перетворення є порівняно малою. Коефіцієнти такого типу часто взагалі можна відкинути (зональний відбір коефіцієнтів, порогове кодування) або, у разі їх кодування, відвести на це кодування невелику кількість двійкових розрядів (зональне кодування).

Звичайно при ущільненні зображень квантування виконується шляхом цілочислового ділення кожного коефіцієнта перетворення на свій «коефіцієнт квантування». Подальше збільшення коефіцієнта ущільнення може бути досягнуте через векторне квантування коефіцієнтів ДОП [5].

Ідеальними для вирішення завдань векторного квантування є нейронні мережі, що самоорганізуються, запропоновані фінським вченим Т. Кохоненом (Self-Organizing Feature Map – SOFM), а саме, мережа, що самоорганізується, у вигляді двовимірної карти Кохонена [5, 14-15].

Найчастіше в якості ДОП при кодуванні зображень використовуються дискретне косинусне перетворення (ДКП) та перетворення Уолша-Адамара [16-26] і вирішення задачі оптимального квантування трансформант цих перетворень є дуже важливим. Саме ці питання розглядаються в роботі, тому її тема є вкрай актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є підвищення коефіцієнту ущільнення зображень за рахунок оптимізації квантування трансформант дискретних ортогональних перетворень.

Основними задачами дослідження є:

- обґрунтування доцільності розробки нового технічного рішення;
- розробка та модифікація методів квантування;
- розробка алгоритмів і програм для дослідження методів квантування компонент зображень;
- експериментальні дослідження залежності ступеня ущільнення від методів квантування компонент зображень;
- розрахунок економічних показників розробки.

Об'єкт дослідження – процес квантування компонент зображень при ущільненні з використанням ортогональних перетворень.

Предмет дослідження – методи та програмні засоби квантування трансформант дискретних ортогональних перетворень при ущільненні зображень.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів, теорія імовірності та математична статистика, дискретна математика, теорія цифрової обробки сигналів та зображень для розробки моделей та методів квантування зображень; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод квантування трансформант двовимірних ортогональних перетворень при ущільненні зображень, у якому на відміну від існуючих, використано поблочне групування цілочисельно квантованих трансформант однакової частоти, що підвищує коефіцієнт ущільнення в 1,5-2 рази порівняно з методами без групування.
2. Вперше запропоновано комбінований метод виконання квантування трансформант дискретних ортогональних перетворень, особливість якого полягає у використанні цілочислового квантування і групування низькочастотних трансформант та групування і векторного квантування високочастотних трансформант, що дозволило підвищити коефіцієнт ущільнення Grayscale зображень на 30 % порівняно з методом JPEG.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено мовою програмування C/C++ програмні засоби для дослідження квантування зображень та трансформант дискретних ортогональних перетворень.

Особистий внесок здобувача. Усі результати, викладені магістерській кваліфікаційній роботі, отримані автором особисто. У друкованій праці, опублікованій у співавторстві, автору належать такі результати: порівняльний аналіз дискретних ортогональних перетворень [5].

Апробація матеріалів бакалаврської дипломної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Міжнародній науково-практичній конференції «Інформаційні технології і автоматизація» (Одеса, 2023).

Публікації. Основні результати досліджень опубліковано в одній статті у матеріалах конференції.

1 АНАЛІЗ ЗАСТОСУВАННЯ ДИСКРЕТНИХ ОРТОГОНАЛЬНИХ ПЕРЕТВОРЕНЬ ДЛЯ УЩІЛЬНЕННЯ ЗОБРАЖЕНЬ

У даному розділі виконано аналіз основних дискретних ортогональних перетворень при їх застосуванні при кодуванні зображень.

1.1 Застосування перетворень для кодування зображень

Згідно цьому підходу, оборотне лінійне перетворення (наприклад, перетворення Фур'є) використовується для відображення зображення в набір коефіцієнтів перетворення, які потім квантуються і кодуються. Для більшості реальних зображень значне число коефіцієнтів мають малу величину, і можуть бути достатньо грубо квантовані (або повністю видалені) ціною невеликого спотворення зображення. Для перетворення даних зображення можуть використовуватися різні перетворення, включаючи *дискретне перетворення Фур'є* (ДПФ) дискретне косинусне перетворення (ДКП), перетворення Адамара та інші [10, 16-24].

На рис. 1.1 показана схема звичайної системи трансформаційного (з перетвореннями) кодування. Кодер виконує чотири достатньо зрозумілі операції: розбиття зображення на блоки, перетворення, квантування і кодування. Декодер виконує зворотну послідовність операцій (за винятком квантування).

Первинне зображення розмірами NxN розбивається на $(N/n)^2$ блоків розмірами nxn, які потім і піддаються перетворенням. Метою процесу перетворення є декореляція значень елементів в кожному блоці, або ущільнення як можна більшої кількості інформації в найменше число коефіцієнтів перетворення. На етапі квантування ті коефіцієнти, які несуть мінімальну інформацію, видаляються або ж квантуються грубо (вони дають найменший внесок в якість відновлюваного блоку). На завершальному етапі

здійснюється кодування квантованих коефіцієнтів, як правило, за допомогою нерівномірних кодів. Всі або деякі з вказаних етапів можуть бути адаптовані до вмісту блоку, тобто до локальних характеристик зображення; такий варіант називають адаптивним трансформаційним кодуванням. Інакше говорять про неадаптивне трансформаційне кодування.

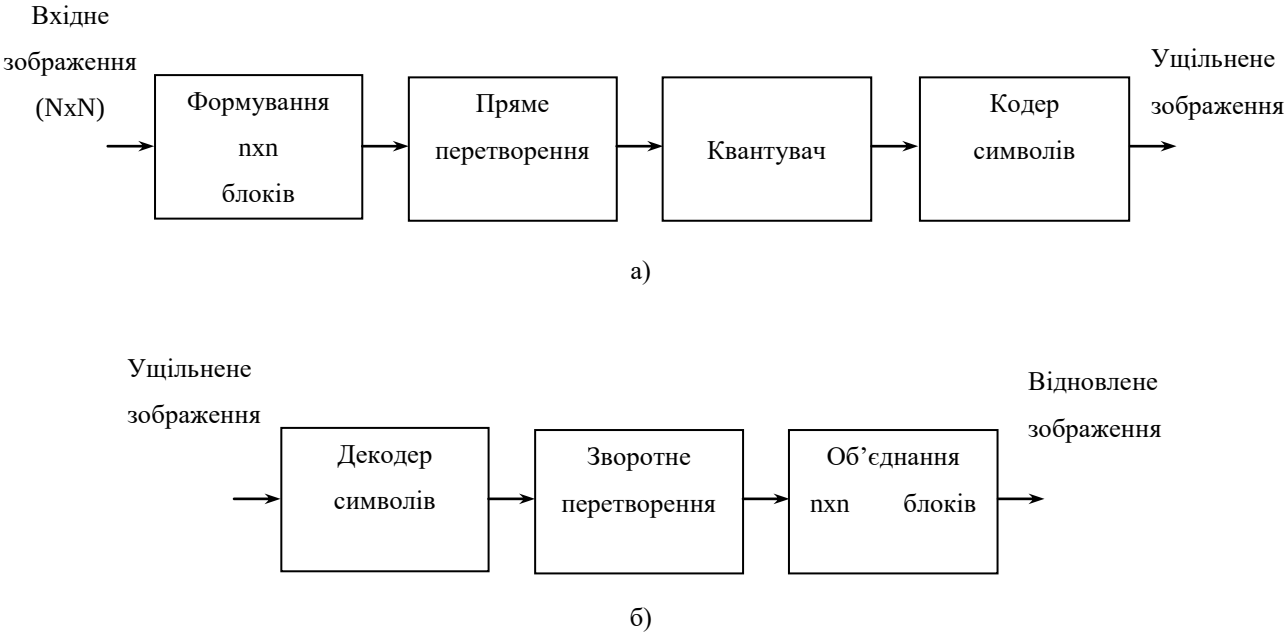


Рисунок 1.1 – Схема кодування з перетвореннями (трансформаційне кодування: а) кодер; б) декодер

Розглянемо основні перетворення, які використовуються при обробці і кодуванні зображень.

1.2 Перетворення Уолша-Адамара

Простим методом перетворення початкового сигналу є перетворення Адамара. Матриця перетворення *H* для двох випадкових величин матиме вигляд [10]:

$$H = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \qquad (1.1)$$

Якщо з елементів матриці *H* скласти базисні вектори

$$e_1 = [h_{11}, h_{12}] = [1,1] \quad e_2 = [h_{21}, h_{22}] = [1,-1],$$

то вони характеризуватимуть поворот ортогональної системи координат на 45° щодо одиничного базису (рис. 1.2). Якщо випадкові величини x_1 і x_2 мають кореляційну залежність, то проекція вектора X $[x_1, x_2]^T$ на базисний вектор e_1 буде, в середньому, більша, ніж проекція цього ж вектора на базисний вектор e_2. Завдяки цьому інформація буде зосереджена в першому коефіцієнті перетворення. Другий коефіцієнт служить для уточнення представлення вектора *X* в новому базисі.

При збільшенні розмірності простору велика частка інформації буде зосереджена в малому числі коефіцієнтів, які потрібно зберегти з найменшими втратами. Решту коефіцієнтів можна або відкинути, або квантувати з крупнішим кроком.

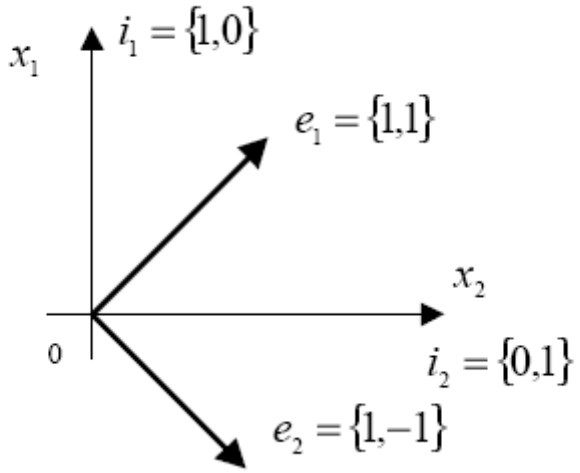


Рисунок. 1.2 - Розташування базисних векторів перетворення Адамара щодо одиничного базису

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

$$H = \begin{vmatrix} H & H \\ H & -H \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix} \quad (1.2)$$

Користуючись співвідношенням (2.2), можна побудувати матрицю Адамара будь-якої розмірності $N = 2^n$, де n - будь-яке ціле позитивне число.

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

$$Y = HAH^T,$$

де A - N x N матриця початкового зображення.

Оскільки матриця H є оператором ортогонального перетворення, то зворотне перетворення з Y в A запишеться у вигляді:

$$A = H^T YH.$$

Перевагою такого перетворення є простота реалізації і низька обчислювальна складність. Це перетворення даватиме добрі результати для кусочно-постійних функцій, оскільки базисні функції перетворення Адамара виділяють постійні складові сигналу. В результаті для кусочно-постійних сигналів велика частка коефіцієнтів розкладання перетворення Адамара буде близька до нуля, що приведе до зменшення ентропії перетворених даних і збільшення коефіцієнта ущільнення. Проте на практиці такі сигнали зустрічаються достатньо рідко, тому виникає необхідність визначити таке перетворення, яке давало б оптимальне або близьке до оптимального розкладання сигналів, точніше характеризуючи реальні зображення. При цьому під оптимальністю розуміється мінімум середньої дисперсії помилки

відновлення, при відкиданні частки високочастотних коефіцієнтів перетворення.

1.3 Перетворення Карунена-Лоева

Виконаємо синтез алгоритму оптимального перетворення в сенсі мінімуму квадрата помилки, при відкиданні незначущих коефіцієнтів.

Для ортогонального перетворення з набором N базисних векторів φ_i, $i = \overline{1, N}$, перетворення з Y в X можна записати як:

$$X = A^T Y = \sum_{i=1}^{N} \phi_i y_i ,$$

де *A* - *N* xN матриця ортогонального перетворення.

Із всієї множини значень $Y = [y_1, y_2, ..., y_N]^T$ запам'ятовуються перші M, причому M < N. Після відновлення отримуємо оцінку вектора $\hat{X}$:

$$\hat{X} = \sum_{i=1}^{M} \phi_i y_i + \sum_{i=M+1}^{N} \phi_i b_i ,$$

де b_i - деяке константне значення відкинутих коефіцієнтів розкладу y_i, $i = \overline{M+1, N}$. Помилка при відкиданні N - M коефіцієнтів визначається як

$$e = X - \sum_{i=1}^{M} \phi_i y_i - \sum_{i=M+1}^{N} \phi_i b_i = \sum_{i=M+1}^{N} \phi_i (y_i - b_i), \qquad (1.3)$$

а дисперсія цієї помилки

$$M\{e^2\} = M\left\{\left(\sum_{i=M+1}^{N} \phi_i (y_i - b_i)\right)^2\right\} = M\left\{\sum_{i=M+1}^{N}\sum_{j=M+1}^{N} (y_i - b_i)(y_j - b_j)\phi_i \phi_j\right\}. \qquad (1.4)$$

Аналіз виразу (1.4) показує, що для мінімізації дисперсії помилки необхідно підібрати відповідні значення b_i і вектори φ_i.

Очевидно, значення $b_i = M\{y_i\}$ або, для центрованих випадкових величин x_i, $i = \overline{1, N}$, $b_i=0$.

Для знаходження оптимальних векторів φ_i, $i = \overline{1, N}$ перепишемо вираз (1.4) з урахуванням $b_i = \phi_i^T M\{X\} = \phi_i^T \overline{X}$ і $\phi_i^T \phi_j = 0$, при $i \neq j$ у вигляді:

$$M\{e^2\} = \sum_{i=M+1}^{N} M\left\{(y_i - b_i)(y_i - b_i)^T\right\} = \sum_{i=M+1}^{N} M\left\{(\phi_i^T X - \phi_i^T \overline{X})(\phi_i^T X - \phi_i^T \overline{X})^T\right\} =$$
$$= \sum_{i=M+1}^{N} \phi_i^T M\left\{(X - \overline{X})(X - \overline{X})^T\right\} \phi_i = \sum_{i=M+1}^{N} \phi_i^T R_{xx} \phi_i, \qquad (1.5)$$

де $R_{xx} = M\left\{(X - \overline{X})(X - \overline{X})^T\right\}$ є коваріаційною матрицею.

Для мінімізації (1.5) застосовується метод множників Лагранжа [ ]. В результаті отримуємо:

$$R_{xx} \phi_i = \beta_i \phi_i. \qquad (1.6)$$

Вираз (1.6) визначає вектори φ_i і значення β_i як власні вектори і власні значення кореляційної матриці R_{xx}. Таким чином, матриця перетворення A, складена з власних векторів кореляційної матриці R_{xx}, дозволяє виконувати перетворення Карунену-Лоева (ПКЛ) [ ], яке мінімізує середню дисперсію помилки при відновленні сигналу за неповними даними.

Недоліком описаного методу є необхідність апріорних відомостей про кореляційну матрицю R_{xx} початкового сигналу X, а також відносно висока обчислювальна складність. Так, в разі застосування роздільного ПКЛ, для обчислення всіх коефіцієнтів перетворення потрібно N^2 арифметичних операцій, де N кількість відліків початкового сигналу X.

Таким чином, метод, що розглянуто, може бути застосований в завданнях, в яких апріорі відома кореляційна матриця R_{xx}, а також немає великих обмежень на обчислювальні ресурси [16].

1.4 Дискретне косинусне перетворення

Розглянуті способи перетворення зображень не повністю задовольняють вимогам, що зазвичай пред'являються, до алгоритмів ущільнення, таким як низька обчислювальна складність, високі коефіцієнти ущільнення, хороша візуальна якість відновленого зображення. Ці вимоги обумовлюють вибір перетворень, які, з одного боку, мають низьку обчислювальну складність, а з іншою, добре описують кодований сигнал. Традиційно як одне з таких перетворень застосовують розкладання сигналу по косинусним гармоніках ряду Фур'є [10, 16]. Даний метод набув широкого поширення і використовується в алгоритмі ущільнення зображень JPEG.

Відомо, що ряд Фур'є для будь-якої неперервної дійсної і симетричної (парної) функції містить тільки дійсні коефіцієнти, відповідні косинусним членам ряду. У відповідній інтерпретації цей результат можна розповсюдити і на перетворення зображень. Існує простий спосіб отримання симетричних зображень, при якому до початкового зображення впритул пристроюють його дзеркальні відображення. При цьому з первинного масиву, що містить NxN елементів, отримують масив з 2Nx2N елементів.

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

$$f_s(j,k) = \begin{cases} f(j,k) \text{ при } j \geq 0, k \geq 0 \\ f(-1-j,k) \text{ при } j < 0, k \geq 0 \\ f(j,-1-k) \text{ при } j \geq 0, k < 0 \\ f(-1-j,-1-k) \text{ при } j < 0, k < 0 \end{cases}$$

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

$$F_s(u,v) = \frac{1}{2N}\sum_{j=-N}^{N-1}\sum_{k=-N}^{N-1} f_s(j,k)\exp\left\{-\frac{2\pi i}{2N}\left[u\left(j+\frac{1}{2}\right)+v\left(k+\frac{1}{2}\right)\right]\right\}, \qquad (1.7)$$

де $u, v = \overline{-N, N-1}$. Оскільки масив f_s (j, k) симетричний і складається з дійсних чисел, співвідношення (1.7) можна записати у вигляді:

$$F(u,v) = \frac{2}{N}C(u)C(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\cos(\frac{(2i+1)u\pi}{2n})\cos(\frac{(2j+1)v\pi}{2n}),$$

Оскільки базисна функція при $u = 0$, $v = 0$ є постійною складовою і в два рази більша по відношенню до решти базисних функцій парного косинусного перетворення, то при виконанні зворотного перетворення необхідно також ввести нормуючий коефіцієнт C(t), який описується таким виразом [21]:

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & x <> 0, \end{cases}$$

$$u, v = 0,1,2 \ldots n-1.$$

В цьому випадку, зворотне перетворення запишеться таким чином:

$$f(i,j) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos(\frac{(2i+1)u\pi}{2n})\cos(\frac{(2j+1)v\pi}{2n}), \qquad (1.8)$$

$$i, j = 0,1,2 \ldots n-1.$$

Перевагою ДКП є те, що базисні функції цього перетворення належать поліному Чебишева і можуть бути обчислені рекурентно. Крім того, це дозволяє синтезувати алгоритми швидкого ортогонального перетворення.

Недоліком ДКП є нерівномірна збіжність на краях інтервалу розкладання N при відкиданні коефіцієнтів, що позначається на візуальній якості відновленого зображення у вигляді артефактів, тобто областей, які не були присутні раніше на початковому зображенні.

Метод, що розглянуто, набув широкого 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].

1.5 Порівняльний аналіз методів кодування з перетворенням

Зменшення дисперсії помилок оцінювання приводить до зменшення ентропії і збільшення коефіцієнта ущільнення. Порівняємо якість відновлення перетворень Адамара, ПКЛ і ДКП для сигналів з КФ $R_x(i) = \sigma_x^2 \, r^{|i|}$, де σ_x – дисперсія сигналу; r - коефіцієнт кореляції між сусідніми відліками. Середню дисперсію помилок відновлення по K коефіцієнтам і заданому виді базисних функцій φ_i можна обчислити за формулою (1.5). На рис. 1.3, а показані розподіли виграшів ПКЛ щодо перетворення Адамара при різному значенні коефіцієнтів кореляції. На рис. 1.3, б розподіли виграшів ПКЛ відносно ДКП.

З рис. 1.3 видно, що ПКЛ показує кращі результати відновлення в порівнянні з перетворенням Адамара і ДКП. При цьому найбільший виграш досягається при коефіцієнтах кореляції r = 0,8 - 0,9 . Разом з тим ДКП програє ПКЛ всього на 0,5-1%, а перетворення Адамара на 20-40%. Отже алгоритм на основі ДКП повинен показувати близькі результати стискування до алгоритму на основі ПКЛ.

На рис. 1.4 представлені результати відновлення кодованих зображень залежно від коефіцієнтів ущільнення. Тут по осі ординат відкладені втрати *l*, по осі абсцис - коефіцієнт ущільнення *k*. Аналіз результатів ущільнення показує на 5-10% кращі коефіцієнти ущільнення алгоритму на основі ПКЛ в порівнянні з алгоритмом на основі ДКП і на 5-20% в порівнянні з алгоритмом на основі перетворення Адамара для зображень рис. 1.4, а, б і в. Для зображення рис. 1.4, г результати ущільнення для всіх алгоритмів близькі, але алгоритм на основі перетворення Адамара показує на 1-2% кращі коефіцієнти стискування порівняно з алгоритмами на основі ПКЛ і ДКП.

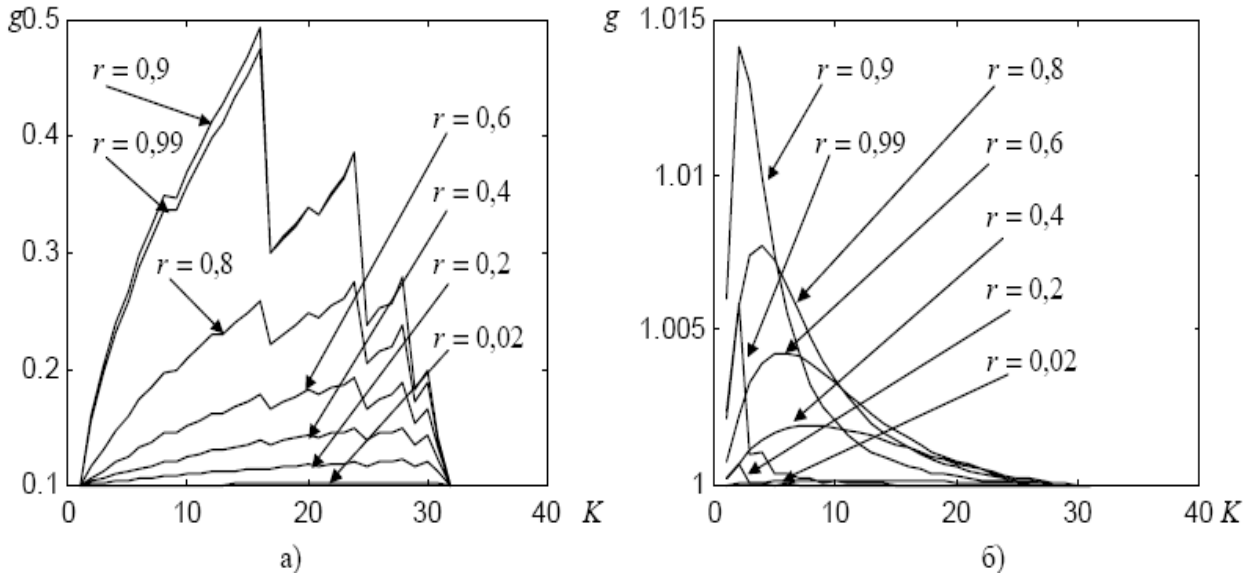


Рисунок 1.3 - Розподіл виграшу ПКЛ щодо перетворень Адамара і ДКП

Таким чином, у випадках, коли апріорі відома кореляційна функція кодованого сигналу і не пред'являється високих вимог до обчислювальної складності, доцільно застосовувати перетворення Карунену-Лоева. У випадках, коли кодоване зображення близьке по своїй структурі до кусочно-постійних функцій, то добрі результати ущільнення можуть бути досягнуті за допомогою перетворення Адамара. При апріорній невизначеності про властивості кодованого сигналу і високі вимоги до швидкості обчислення доцільно застосовувати ДКП.

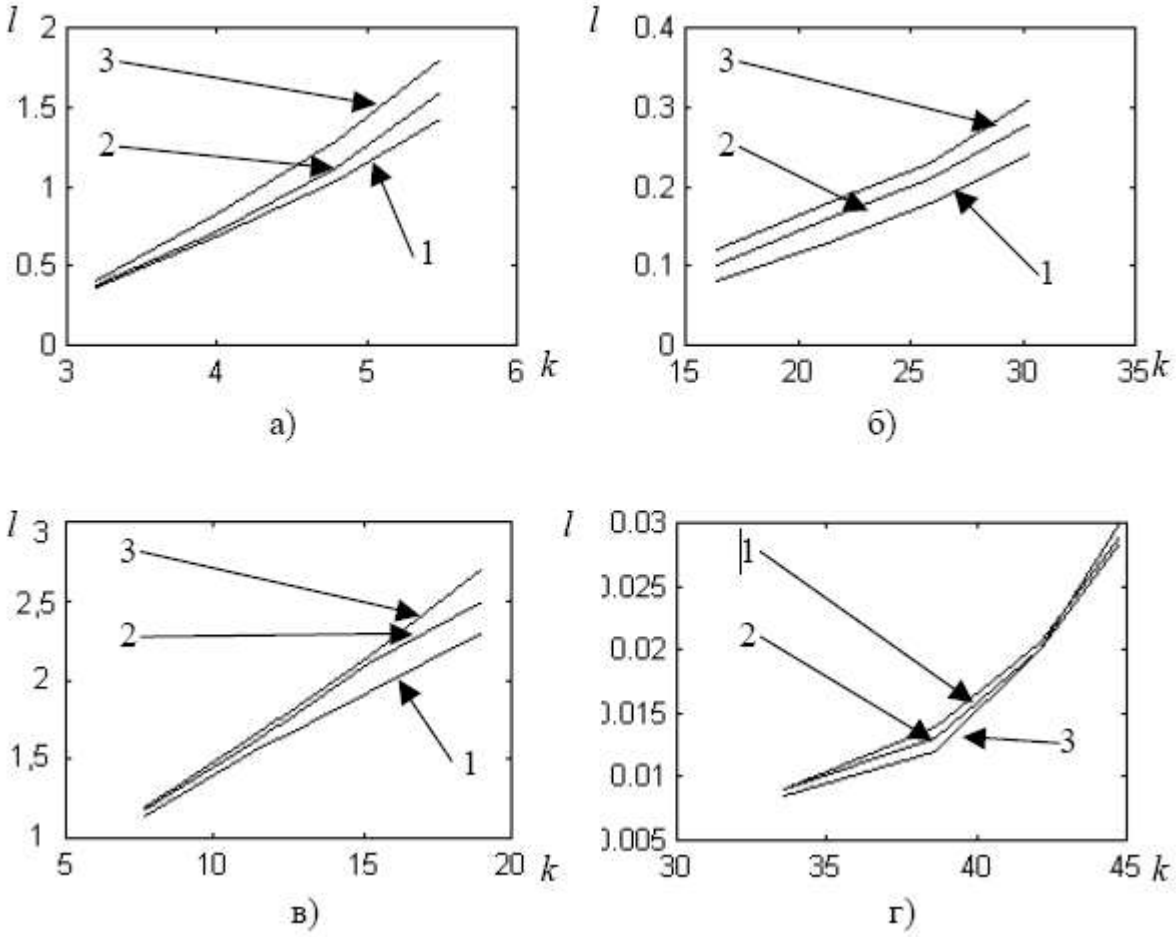


Рисунок 1.4 - Результати ущільнення тестових зображень: 1 – перетворення Карунену-Лоева; 2 – ДКП; 3 – перетворення Адамара.

1.6 Аналіз методів квантування коефіцієнтів ДОП та зональний відбір

Квантування коефіцієнтів двовимірних перетворень використовується при ущільненні зображень для зменшення обсягу даних та покращення стиснення. Розглянемо деякі методи квантування:

Рівномірне квантування [4]:

Рівномірно розподіляє діапазон значень коефіцієнтів.

Використовує однаковий інтервал для всіх значень.

Застосування:

Для простих застосувань, де рівномірна точність важлива.

Нерівномірне квантування [5-9]:

Використовує різні інтервали для різних діапазонів значень коефіцієнтів.

Дозволяє краще адаптуватися до розподілу важливості для покращення ефективності стиснення.

Застосування:

У випадках, коли деякі частини зображення важливіші за інші.

Квантування з втратами (Lossy Quantization) [2, 10-13]:

Зменшує точність квантування для досягнення вищого ступеня стиснення.

Втрачає частину інформації, але при цьому значно зменшує обсяг даних.

Застосування:

У випадках, де деякий втрати якості припустимі (наприклад, відеостиснення).

Адаптивне квантування [2]:

Застосовує різні рівні квантування для різних областей зображення.

Дозволяє краще враховувати властивості окремих частин зображення.

Застосування:

В графічних або відеоданих, де різні частини можуть мати різну важливість.

Кодування дельта [7]:

Квантування відбувається відносно попередніх значень (дельта).

Зберігає лише різницю між поточним і попереднім значеннями.

Застосування:

В аудіо- та відеостисненні для ефективної роботи з сигналами.

Нерівномірне квантування — це процес, при якому рівні квантування не розподілені рівномірно, а залежать від амплітуди сигналу. Це дозволяє

зменшити помилку квантування і підвищити ефективність стиснення даних. Існує декілька методів нерівномірного квантування, наприклад [7-10]:

- Компандування — це метод, при якому сигнал попередньо перетворюється за допомогою нелінійної функції, що збільшує малі значення і зменшує великі. Після цього застосовується рівномірне квантування, а потім обернена функція компандування. Прикладами компандування є A-закон і μ-закон, які використовуються в телефонній мережі.

- Квантування зі змінною довжиною кодового слова (VQ) — це метод, при якому рівні квантування представлені кодовими словами різної довжини, так що частіші значення мають коротші коди, а рідші — довші. Це дозволяє зменшити середню довжину коду і збільшити ступінь стиснення. Прикладами VQ є кодування Гаффмана і арифметичне кодування.

- Квантування зі змінною роздільною здатністю (VRQ) — це метод, при якому рівні квантування мають різну точність, так що високі значення мають більшу роздільну здатність, а низькі — меншу. Це дозволяє зменшити помилку квантування і покращити якість сигналу. Прикладом VRQ є кодування зі змінною швидкістю передачі (VBR), яке використовується в кодуванні мовлення.

Ці методи можуть застосовуватися як окремо, так і в комбінації для досягнення оптимального ступеня квантування та стиснення в залежності від конкретних вимог застосування.

У більшості випадків при кодуванні зображення розмірність значень багатьох коефіцієнтів перетворення є порівняно малою. Коефіцієнти такого типу часто взагалі можна відкинути (зональний відбір коефіцієнтів, порогове кодування) або, у разі їх кодування, відвести на це кодування невелику кількість двійкових розрядів (зональне кодування).

Подальше збільшення коефіцієнта ущільнення може бути досягнуте через векторне квантування трансформант ДОП. Ідея векторного квантування дуже проста. Зображення розбивається на квадратні блоки, наприклад 2x2, 4x4 або 8x8. Кожен блок розглядається як вектор в 4-мірному, 16-мірному або 64-

мірному просторі. З цього простору вибирається обмежена кількість векторів, які утворюють кодову книгу, але так, щоб з найбільшою точністю апроксимувати вектори, які вилучаються з вхідного зображення. Оскільки векторів в кодовій книзі значно менше загальної кількості векторів в початковому зображенні, то для представлення номера вектора витрачається менше біт, чим для початкового вектора. За рахунок цього і досягається ущільнення [25].

Ідеальними для вирішення завдань векторного квантування є нейронні мережі, що самоорганізуються, запропоновані фінським вченим Т. Кохоненом (Self-Organizing Feature Map – SOFM), а саме, мережа, що самоорганізується, у вигляді двовимірної карти Кохонена [25]. Карта Кохонена має дві важливі властивості, які можуть бути використані при ущільненні зображень. По-перше, вона дуже схожа на інші методи векторного квантування, які застосовуються при ущільненні зображень з втратами, а по-друге близьким кластерам вхідних векторів відповідають близько розташовані нейрони, що збільшує ефективність ущільнення без втрат, яке застосовується до отриманих компонентів зображення. Кожний нейрон мережі представляється ваговими коефіцієнтами $w_{ij.}$ Векторне квантування з використанням карти Кохонена виконується за два проходи початкового зображення: перший прохід - навчання мережі; другий прохід – власне векторне квантування. Після навчання ця мережа може апроксимувати вектори вхідного простору найкращим способом.

Базова архітектура мережі SOFM наведена на рис. 1.5.

Вихідні елементи називаються кластерними елементами. Кластерні елементи або кодові слова розміщуються в виді одно або двовимірного масиву. Звичайно кількість кластерних елементів значно менша в порівнянні з кількістю навчальних зразків, оскільки метою є отримання спрощеної характеристики вхідних даних. Це і дає можливість використання SOFM як векторного квантувача.

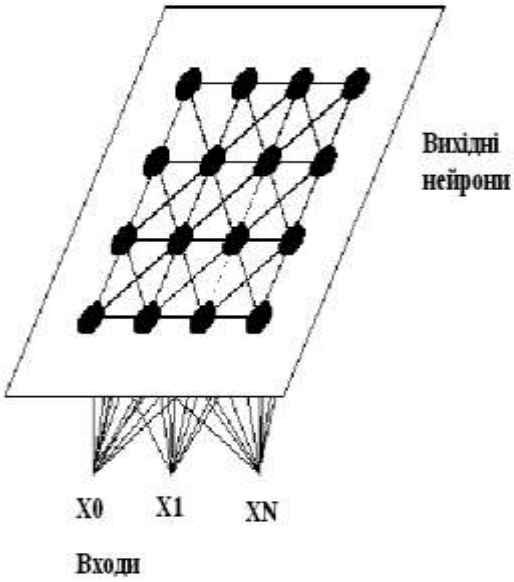


Рисунок 1.5 - Базова архітектура мережі SOFM

Алгоритм навчання мережі такий:

1. Ініціалізувати вагові коефіцієнти випадковими значеннями.
2. Для кожного кластерного елемента обчислити відстань до навчального вектора:

$$d_j = \sum_i (w_{ij} - x_i)^2$$

3. Найти кластерний елемент j для якого d_j мінімально.
4. Для кластерних елементів із круга заданого радіуса з центром в j елементі обновити вагові коефіцієнти згідно формули:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)],$$

де η - норма навчання, x_i – координата навчального вектора.

5. Обновити норму навчання η і радіус при необхідності і повторити пункти 1-5 для наступного навчального вектора.

Норма навчання з часом змінюється. Вона може, наприклад, мати значення 0,9, а потім змінюватись лінійно до деякого фіксованого значення, наприклад 0,01, після чого залишатися незмінною. Радіус також спочатку

вибирається достатньо великим, щоб обновлялись всі елементи. З часом радіус зменшується і в кінці повинен обновлятись тільки сам елемент-переможець.

Отже, тема кодування зображень є актуальною задачею і потребує пошуку нових методів кодування.

Метою роботи є підвищення коефіцієнту ущільнення зображень за рахунок розробки програмного забезпечення придатного для дослідження залежності коефіцієнта ущільнення від параметрів квантування коефіцієнтів двовимірного перетворення.

1.7 Висновки

1. Показано актуальність ущільнення зображень та перспективність дискретного косинусного перетворення (ДКП) та перетворенням Уолша-Адамара, відмінною особливістю якого є те, що розрахунок базується лише на операціях додавання.
2. Коефіцієнт ущільнення і якість відновленого зображення залежать від вирішення задачі квантування трансформант дискретного ортогонального перетворення. Показано, що серед методів квантування перспективним є застосування векторного квантування, оскільки векторне квантування потенційно може забезпечити збільшення коефіцієнта ущільнення.
3. Ідеальними для вирішення завдань векторного квантування є нейронні мережі, що самоорганізуються, запропоновані фінським вченим Т. Кохоненом (Self-Organizing Feature Map – SOFM), а саме, мережа, що самоорганізується, у вигляді двовимірної карти Кохонена, оскільки близьким кластерам вхідних векторів відповідають близько розташовані нейрони, що збільшує ефективність ущільнення без втрат, яке застосовується до отриманих компонентів зображення.

2 РОЗРОБКА МЕТОДІВ І АЛГОРИТМІВ КВАНТУВАННЯ КОЕФІЦІЄНТІВ ДВОВИМІРНИХ ОРТОГОНАЛЬНИХ ПЕРЕТВОРЕНЬ

2.1 Розробка загальної схеми ущільнення зображень

Загальна схема кодування наведена на рис. 2.1. Пряме ДОП (дискретне ортогональне перетворення) і цілочислове квантування виконуються з використанням дискретного косинусного перетворення (ДКП) аналогічно JPEG або з використанням перетворення Уолша-Адамара. Після виконання групування коефіцієнти однакової частоти утворюють у двовимірному масиві даних блоки однакової частоти, елементи яких можуть мати близькі значення, що підвищує точність векторного квантування. На етапі кодування найбільш доцільним є застосування арифметичного кодування, оскільки цей метод забезпечує найкращий коефіцієнт ущільнення серед відомих методів ущільнення без втрат (рис. 1).

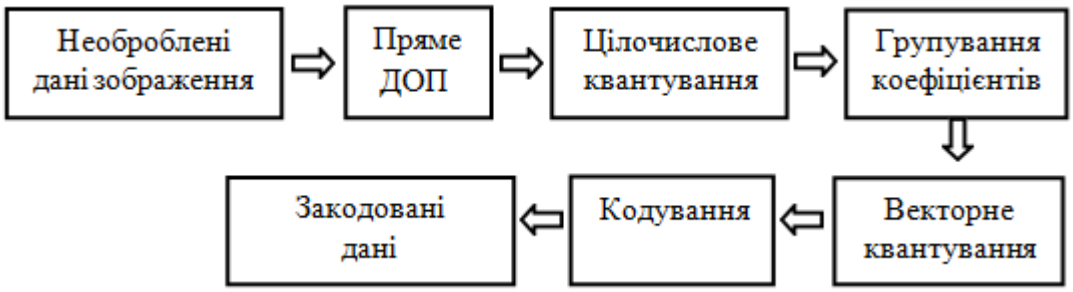


Рисунок 2.1 – Загальна схема кодування з векторним квантування трансформант

Групування коефіцієнтів є альтернативою зигзагоподібному скануванню JPEG і може не тільки підвищити точність векторного квантування, але збільшити коефіцієнт ущільнення на етапі ущільнення без втрат за рахунок підвищення точності роботи моделювальника в адаптивних схемах кодування без втрат, що і може збільшувати коефіцієнт ущільнення. Також після

групування коефіцієнтів можна легко організувати зональний відбір коефіцієнтів, оскільки коефіцієнти з однаковими частотами локалізовані у просторі зображення в певних ділянках, що дає можливість виключити їх з процесу синтезу зображення.

У подальшому при дослідженні квантування коефіцієнтів можна буде розглянути і дослідити різні схеми кодування, які є похідними від базової схеми (рис. 2.1).

Зокрема, представляють інтерес такі схеми:

- ДОП – Цілочислове квантування – Кодування;
- ДОП – Цілочислове квантування – Групування - Кодування;
- ДОП – Цілочислове квантування – Групування – Зональний відбір-Кодування;
- ДОП – Групування – Зональний відбір-Кодування;
- ДОП – Цілочислове квантування – Групування – Векторне квантування – Кодування;
- Векторне квантування – Кодування.

2.2 Розробка методу та алгоритму групування коефіцієнтів ДОП

При групуванні коефіцієнтів ДОП виконується відбір коефіцієнтів однакової частоти. Цей відбір виконує модуль групування коефіцієнтів (МГК). Координати коефіцієнтів для відбору обчислюються у відповідності з такими виразами:

$$\begin{aligned} tmpj &:= (j \bmod n) \cdot (N \cdot \operatorname{div} n) + j \cdot \operatorname{div} n; \\ tmpi &:= (i \bmod n) \cdot (M \cdot \operatorname{div} n) + i \cdot \operatorname{div} n, \end{aligned} \tag{2.1}$$

де i, j – поточні координати в площині зображення;

tmpi, tmpj - нові координати;

N, M – розміри зображення;

n – розмір сторони фрагмента в межах якого виконується ДОП;

div – операція цілочислового ділення.

При декодуванні коефіцієнти знову переміщуються на свої позиції в площині зображення, що необхідно для виконання процедури зворотного перетворення ДОП. Їх координати обчислюються так:

$$\begin{aligned} tmpj &:= (j \bmod (N \cdot div\ n)) \cdot n + j\ div\ (N \cdot div\ n); \\ tmpi &:= (i \bmod (M \cdot div\ n)) \cdot n + i\ div\ (M \cdot div\ n). \end{aligned} \qquad (2.2)$$

Граф-схема алгоритму роботи блоку групування коефіцієнтів приведена на рис. 2.1. Вхідними даними є двовимірний масив цілочисельно квантованих коефіцієнтів ДОП. Оператор (2) виконує обчислення координат коефіцієнта, в новому масиві. Оператор (3) читає поточні значення з вхідного масиву і переписує їх в новий масив.

Якщо тепер візуалізувати коефіцієнти, то у верхньому лівому куті буде відображена зменшена копія зображення розміром (N div n) x (m div M), а в цілому площина зображення буде розбита на (nxn) блоків кожний розміром (N div n) x (m div M). Тепер з’являється можливість відкинути будь-який із цих блоків, тобто застосувати комбінацію квантування і зонального відбору.

При виведенні згрупованих коефіцієнтів у файл вони виводяться послідовно, спочатку всі коефіцієнти F(0.0) рядок за рядком, потім F (0,1) і так дальше. Тобто виконується перетворення двовимірного масиву в одновимірний. Координати у площині згрупованих коефіцієнтів для виведення у файл обчислюються згідно виразу:

$$\begin{aligned} tmpj &:= (j \bmod (N\ div\ n)) + ((i \bmod (M\ div\ n) * (N\ div\ n); \\ tmpi &:= i + (j\ div\ (N\ div\ n)). \end{aligned} \qquad (2.2)$$

У файлі коефіцієнти записані в такому порядку:

$$F_{00}(0,0), \ldots F_{kl}(0,0), F_{00}(0,1), \ldots F_{kl}(0,1), \ldots F_{00}(7,7), \ldots F_{kl}(7,7)$$

Такий порядок зберігання особливо важливий, коли на етапі кодування без втрат застосовують адаптивні кодери, наприклад, кодери, які використовують технології передбачення PPM або DMC [29]. Слід очікувати зростання коефіцієнту ущільнення в 1,5 – 2 рази. Як показує практика зигзагоподібне сканування коефіцієнтів ДОП в JPEG не може дати такого ефекту.

2.3 Розробка методу і алгоритму ущільнення при векторному квантуванні

Для розробки алгоритму необхідно вирішити декілька питань:

1. Який розмір карти Кохонена вибрати.
2. Яку вибрати розмірність вхідного вектора.
3. Яка розрядність вагових коефіцієнтів карти Кохонена.

Розрядність вагових коефіцієнтів вибирається з урахуванням розрядності вхідних даних. Оскільки зображення звичайно представляються в форматі 8x8x8, тобто складові кольорів RGB представляються одним байтом, то і розрядність вагових коефіцієнтів вибираємо рівною 8-и бітам.

Але розрядність коефіцієнтів ДОП більша чим 8 розрядів, оскільки при прямому дискретному косинусному перетворенні кожний коефіцієнт ділиться на 4 згідно виразу (1.7), а якщо використовує перетворення Уолша-Адамара, то на 8 згідно виразу в [20]. Тоді додатні значення при фрагменті перетворення 8x8 можуть досягати максимального значення:

$$(256x64)/8=2048.$$

А від’ємні значення

$$-(32x256)/8=-1024.$$

Тому розрядність вагових коефіцієнтів вибираємо рівною 16 біт.

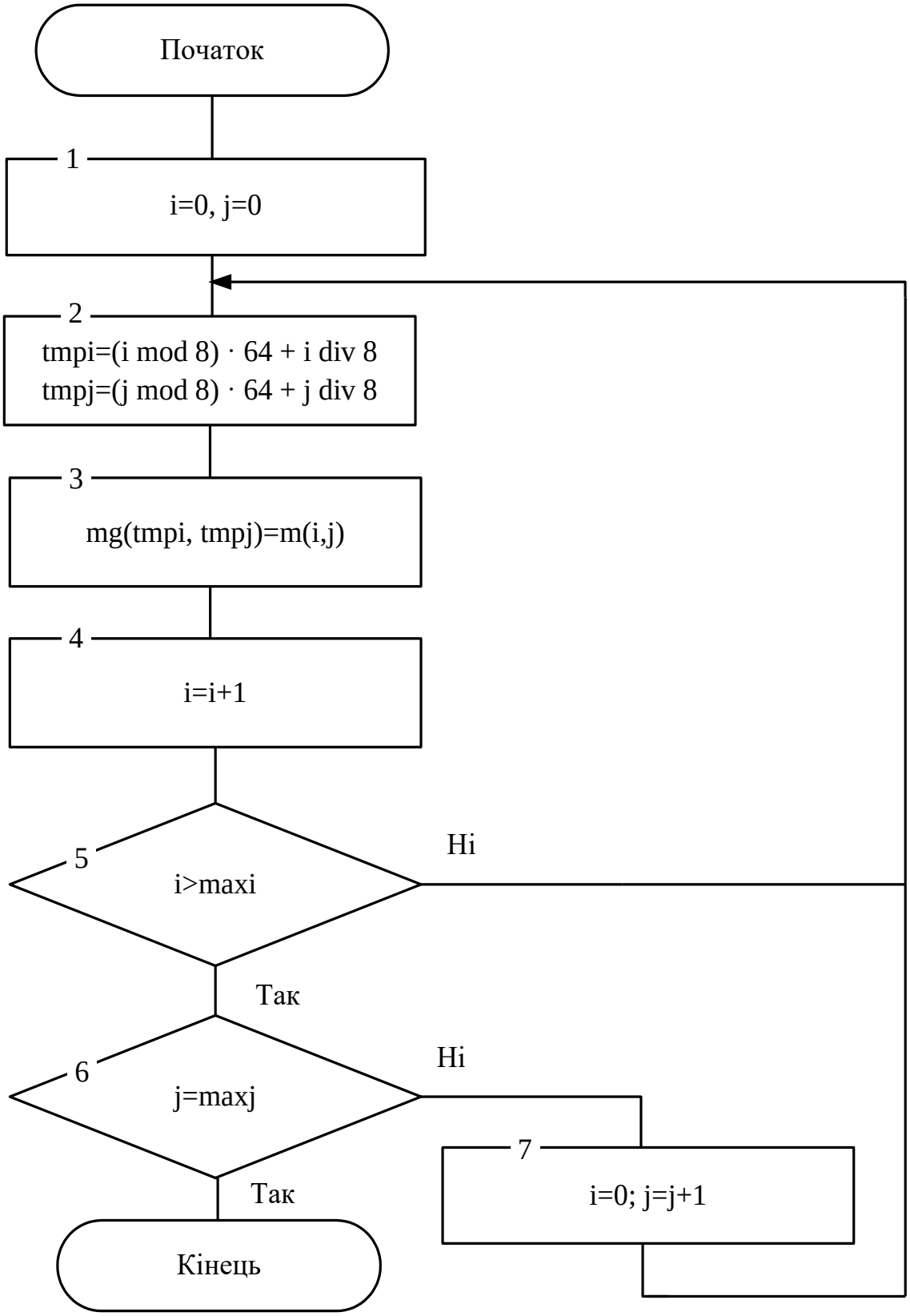


Рисунок 2.1 - Блок-схема алгоритму роботи групування трансформант ДОП

Розмірність вхідних векторів вибирається з урахуванням кореляції вхідних даних. Відомо, що найбільшими кореляційними зв'язками характеризуються сусідні відліки зображення, які можуть мати близькі значення [25]. Тому зображення розбивається на примикаючі квадрати розміром 2x2, кожний з яких розглядається як вектор в 4-вимірному просторі.

Розмір карти Кохонена визначається мінімальною кількістю розрядів, яка представляє елемент зображення і забезпечує достатню якість зображення. З наукової літератури відомо, що менше 2 біт на елемент зображення при будь-яких перетвореннях досягти проблематично при заданих високих вимогах до якості зображення. До того ж збільшення розміру карти призводить до значної втрати швидкодії навчання мережі. Швидкість навчання прямо пропорційна квадрату розміру сторони карти , тобто:

$$T_н = kN^2,$$

де N – розмір сторони карти, k – коефіцієнт пропорційності, залежить від конкретної реалізації.

З урахуванням вище сказаного розмір карти Кохонена, який задовольняє цим суперечливим вимогам, вибираємо 16x16. Оскільки розмір вхідного вектора рівний чотирьом, така карта забезпечує при векторному квантуванні таку кількість біт на елемент зображення:

$$M = \log_2 N^2 / n = 8/4 = 2 \text{ біти/ел.}$$

Що є прийнятним як з точки зору швидкодії, так і з точки зору забезпечення високої якості відновленого зображення.

Векторне квантування з використанням карти Кохонена виконується за два проходи початкового зображення:

- перший прохід - навчання мережі;
- другий прохід – векторне квантування.

Причому навчальними векторами є всі фрагменти зображення з розмірами 2x2.

Таким чином алгоритм кодування буде таким:

Етап навчання

1. Ініціалізувати вагові нейронів коефіцієнти випадковими значеннями.
2. Вибрати з зображення перший фрагмент 2x2
3. Представити його у вигляді навчального вектора.
4. Для кожного кластерного елемента карти обчислити відстань до навчального вектора:

$$d_j = \sum_{i=0}^{3} (w_{ij} - x_i)^2$$

Знайти кластерний елемент j для якого d_j мінімально.

5. Для даного кластерного елемента обновити вагові коефіцієнти згідно формули:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)],$$

де η - норма навчання, x_i – координата навчального вектора.

6. Обновити норму навчання η і вибрати з зображення наступний фрагмент 2x2 і повторяти пункти 3-6 для наступних навчальних векторів до тих пір поки не будуть вибрані всі фрагменти.

Векторне квантування

7. Вибрати з зображення перший фрагмент 2x2
8. Представити його у вигляді навчального вектора.
9. Для кожного кластерного елемента карти обчислити відстань до навчального вектора:

$$d_j = \sum_{i=0}^{3} (w_{ij} - x_i)^2$$

10. Номер кластерного елемента з мінімальним d_j записати в вихідний файл.

11. Вибрати з зображення наступний фрагмент 2x2 і повторяти пункти 8-11 до тих пір поки не будуть вибрані всі фрагменти.

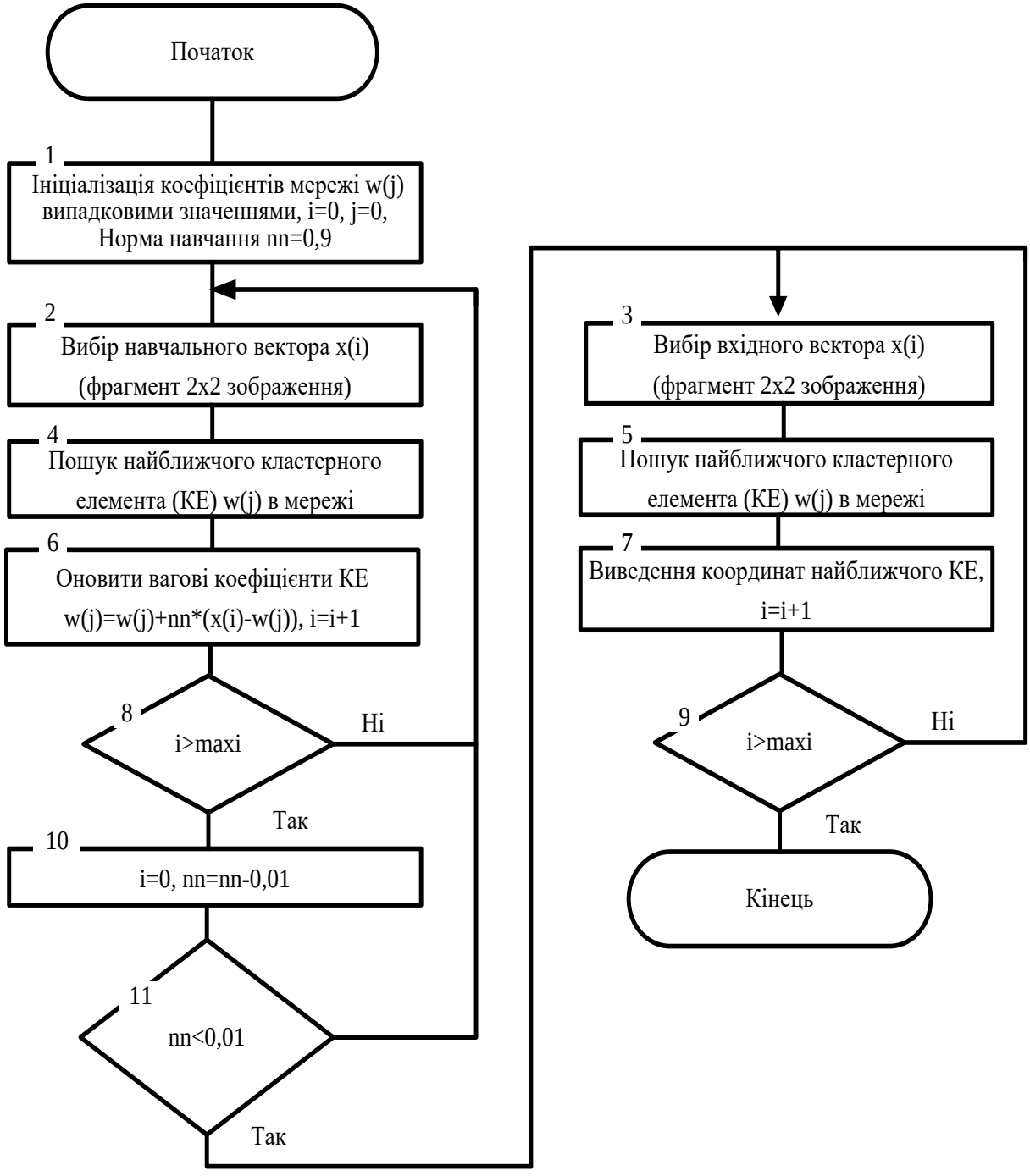


Рисунок 2.2 - Блок-схема алгоритму векторного квантування трансформант ДОП

12. Записати у вихідний файл значення коефіцієнтів w_{ij} – всього 2048 байт.

13. Записати у вихідний файл розмір початкового файлу.

14. Ущільнити отриманий файл методом арифметичного кодування.

Декодування виконується значно швидше і включає такі етапи:

1. Декодувати ущільнений файл арифметичним методом.

2. Прочитати з вхідного файлу і записати в масив значення номерів кластерних елементів, коефіцієнтів w_{ij} і розмір початкового зображення.

3. Вибрати з масиву номер кластерного елементу для першого фрагменту 2x2.

4. Коефіцієнти w_{ij} даного кластерного елемента (4 коефіцієнти) записати в вихідне зображення як значення елементів відповідного фрагменту 2x2.

5. Вибрати наступний номер кластерного елементу і повторяти пункти 4-5 до тих пір поки не будуть відновлені всі фрагменти зображення (рис. 2.2).

2.4 Висновки

1. Розроблено метод та алгоритм квантування трансформант двовимірних ортогональних перетворень при ущільненні зображень, у якому на відміну від існуючих, використано поблочне групування цілочисельно квантованих трансформант однакової частоти, що підвищить коефіцієнт ущільнення в у порівнянні з методами без групування.
3. Розроблено комбінований метод квантування трансформант ДОП та алгоритм його виконання із використанням цілочислового квантування та групування низькочастотних трансформант і групування та векторного квантування високочастотних трансформант, що дозволить підвищити коефіцієнт ущільнення зображень.

3 РОЗРОБКА МОДУЛІВ КВАНТУВАННЯ ТА ЗОНАЛЬНОГО ВІДБОРУ КОЕФІЦІЄНТІВ ДОП

3.1 Аналіз та вибір середовища і мови програмування

Від правильного вибору мови програмування та середовища розробки залежить якість розробки та час реалізації проекту. Основним критерієм вибору мови програмування для даної роботи є можливість автоматизованої розробки інтерфейсу користувача, оскільки час, відведений на цю роботу, відносно невеликий. Крім того, ще одним виміром є швидкість складних математичних обчислень. Маючи це на увазі, розглянемо на найважливіші мови програмування в операційній системі Windows:

- Assembler;
- C++ (використовується середовище Visual Studio);
- Java (середовище Eclipse);
- C# (середовище Visual Studio);
- Мови сценаріїв (Python, Perl, Ruby).

Асемблер використовується для низькорівневого програмування. Програмування мовою Асемблера забезпечує максимальну швидкість, але програми складно реалізувати та зрозуміти, а написання займає багато часу. Крім того, розробка графічного інтерфейсу користувача цією мовою є тривалим процесом.

Мови C# і Java полегшують проектування графічних інтерфейсів користувача, вони мають зручне середовище розробки, але через те, що їх код транслюється в байт-код, вони швидко програють традиційним мовам.

Мови сценаріїв не мають зручних інструментів для проектування користувацьких інтерфейсів і характеризуються відносно низькою продуктивністю, оскільки програмний код вимагає динамічного аналізу під час виконання або перекладу в байт-код з подальшою інтерпретацією.

C++ – це універсальна мова програмування високого рівня, що характеризується високою швидкістю та розширюваністю [30]. З іншого боку, сучасне інтегроване середовище розробки Microsoft Visual Studio забезпечує простий інтерфейс користувача та прості засоби створення користувацького інтерфейсу.

Тому обираємо C++ як мову програмування та середовище розробки Microsoft Visual Studio 2019 для автоматизації розробки інтерфейсу користувача.

3.2 Розробка архітектури та інтерфейсу користувача додатку

Використаємо проект Windows Forms [30], який дозволяє створювати користувацькі інтерфейси за допомогою простих елементів управління мишею. Виходячи з визначеного набору функцій досліджуваної програми, головне вікно програми повинно містити поле для виведення початкового та обробленого зображень, командні кнопки для відкриття початкового файлу, збереження обробленого файлу та відправки інших команд - інформації про програму та інше. Крім того, слід також враховувати експериментальний характер роботи.

З огляду на це базова форма програми показана на рис. 3.1 і реалізує концепцію одновіконної програми, що є максимально зручним для користувача.

Для програмної реалізації алгоритмів квантування коефіцієнтів двовимірних перетворень під час стиснення зображення використовуються наступні компоненти керування Visual Studio (System.Windows.Forms):

Рисунок 3.1 – Головна форма програми

1. Вкладка Windows Form — це основна форма програми, яка містить інші візуальні компоненти.
2. Дві форми PictureBox, призначені для завантаження вхідного зображення та виведення зображення після кодування.
3. Шістнадцять кнопок Button для надсилання команд.
4. Три форми DataGridView, призначені для введення та відображення матриць квантування трансформант двовимірного ортогонального перетворення для розміру блоків 8x8. Зокрема, різні коефіцієнти квантування передбачені для колірних складових (RGB) зображення.
5. П'ять форм Label для виведення інформації про призначення деяких компонентів.
6. Крім того, вікно «Про програму» містить такі форми:
   - Form;

- Label — стандартна інформація про назву продукту, версію, права та назву компанії;
- кнопку Button для закриття вікна.

Оскільки додаток призначений для дослідження різних методів квантування компонент зображення, що вимагає великої кількості експериментальних досліджень і змін у коді у процесі цих досліджень, то доцільно побудувати додаток як набір повністю незалежних автономних модулів, які не використовують спільні методи та функції, тобто додаток будується за мультимодульною архітектурою.

Діаграма класів додатку, що реалізує запропоновані алгоритми, наведено на рис. 3.2.

3.3 Розробка модулів виконання дискретного ортогонального перетворення з цілочисловим квантуванням трансформант перетворення

Розроблено модулі виконання прямого і зворотного дискретного косинусного перетворення (ДКП) та перетворення Уолша-Адамара. Код цих модулів відрізняється лише у частині обчислення власне перетворення, тому розглянемо більш детально модулі виконання прямого і зворотного ДКП.

Вхідними даними для модуля прямого ДКП (метод button22_Click ()) є зображення у будь-якому форматі, який підтримує Windows. При виборі файлу з використанням стандартного вікна відкриття файлу дані зображення записуються в масив типу image1 Bitmap^:

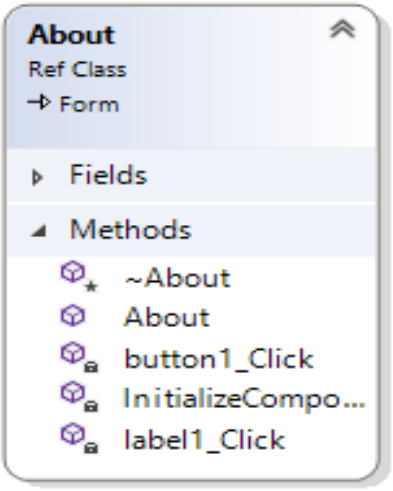
```
image1 = gcnew Bitmap(openFileDialog1->FileName, true);
```

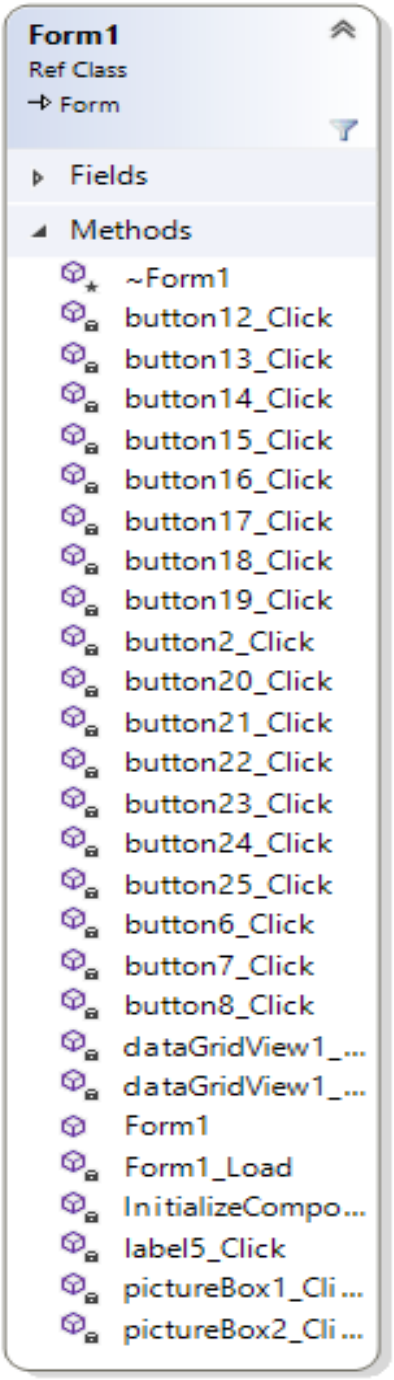
Також створюється файл для запису трансформант перетворення з розширенням .dct:

```
name1 = name1 + ".dct";
ofstream out_file(name1, ios::binary);
```

В цей файл на початку записуються коефіцієнти цілочислового квантування для компонент зображення в послідовності RGB, значення яких беруться з форм dataGridView:

- форма dataGridView2 містить коефіцієнти квантування для компоненти R;
- форма dataGridView5 містить коефіцієнти квантування для компоненти G;





Команда	**Метод**
About	button6_click()
CopyImage	button2_click()
DCTCoder	button22_click()
DCTDecoder	button23_click()
DeGroup8	button16_click()
DeVectIm	button20_click()
Group8	button15_click()
LoadImage	button8_click()
NetworkTr	button21_click()
SaveD	button7_click()
SD	button12_click()
VectDKVCm	Button25_Click()
VectIm	Button19_Click()
VectKVCm	Button24_Click()
VectorDKV	Button18_Click()
VectorKV	Button17_Click()
WHCoder	Button13_Click()
WHDecoder	Button14_Click()

Рисунок 3.2 – Класи і методи додатку

- форма dataGridView6 містить коефіцієнти квантування для компоненти B.

Дані в ці форми може вносити користувач. Розмір цих форм 8x8, що відповідає розміру фрагментів зображення в межах яких виконується ДКП або перетворення Уолша-Адамара. Кожна трансформанта ДКП або перетворення Уолша-Адамара цілочисельно ділиться на відповідний коефіцієнт квантування.

Таким чином на початку вихідного файлу 192 байти займають коефіцієнти цілочислового квантування.

Далі виконується перетворення ДКП для фрагментів вхідного зображення розміром 8x8, що примикають один до одного, у послідовності - спочатку для фрагменту з компоненти R, потім G і B. Для компоненти R значення пікселів записуються в масив tr розміром 8x8:

```
Color pixelColor = image1->GetPixel(8 * x + j, 8 * y + i);
tr[i][j] = pixelColor.R;
```

Для кожного фрагменту послідовно обчислюються коефіцієнти перетворення:

```
t = t + (tr[k][l] * cos(((2 * k + 1) * j * Math::PI) / 16) * cos(((2 * l + 1) * i * Math::PI ) / 16));
```

Який записується в масив коефіцієнтів ДКП tr1 за відповідною позицією:

F (0,0) – лівий верхній кут, …, F (7,7) – правий нижній кут:

```
tr1[j][i] = short int (t*c[j]*c[i]*0.25);
```

Коефіцієнти c[j], c[i] приймають значення у відповідності до виразу (1.7).

Кожний коефіцієнт ДКП або перетворення Уолша-Адамара цілочисельно ділиться на свій коефіцієнт квантування і записується у вихідний файл:

```
tr1[k][l] = tr1[k][l] / kv8[k][l];
out_file.write((char*)&tr1[k][l], 2);
```

Після чого виконується деквантування для виконання зворотного перетворення і виведення відновленого зображення для контролю у праве поле виведення зображень:

```
tr1[k][l] = tr1[k][l] * kv8[k][l];
```

Вхідним для зворотного перетворення є масив tr1. Послідовно обчислюються значення пікселів компоненти R:

```
t = t + (c[k]*c[l]*tr1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 * i + 1) * l * Math::PI) / 16));
```

Результати обчислення записуються у масив tr:

```
tr[j][i] = short int(t * 0.25);
```

Аналогічні перетворення виконуються для компонент G та B. Для цього використовуються масиви:

- tg, tg1 – для компоненти G;
- tb, tb1 – для компоненти B.

Для візуалізації відновленого зображення виконується перевірка знаходження значення кожного відновленого пікселя по кожній з компонент зображення в межах 0…255 і приведення його до цього діапазону, оскільки через квантування, округлення або зональний відбір такі явища можуть мати місце:

```
t = tr[i][j];
if (t < 0)  t = 0;
if (t > 255)  t = 255;
```

```
r = t;
```

Після цього значення пікселя записується у масив image2 типу Bitmap^ на свою позицію:

```
newColor1 = Color::FromArgb(r, g, b);
image2->SetPixel(8 * x + j, 8 * y + i, newColor1);
```

Після виведення відновленого зображення у праве поле для виведення зображень і закриття файлу з коефіцієнтами ДКП :

```
pictureBox2->Image = image2;
out_file.close();
```

виконується ущільнення файлу з коефіцієнтами ДКП або перетворення Уолша-Адамара арифметичним методом. Ущільнені коефіцієнти ДКП записуються у файл з розширенням .dctc.

Вхідними даними для модуля для декодування зображень методом ДКП (метод button23_Click ()) є саме ці файли. Тому на початку роботи виконується арифметичне декодування і результати записуються у файл з розширенням .dct, яки й після відкривається для читання у бінарному форматі:

```
ifstream in_file(name1, ios::binary);
```

З файлу зчитуються 192 байти коефіцієнтів квантування і виводяться dataGridView. Потім послідовно з вхідного файлу читаються коефіцієнти для фрагментів зображення по кожній компоненті RGB розміром 8x8 і деквантуються:

```
in_file.read((char*)&tr1[k][l], 2);
tr1[k][l] = tr1[k][l] * kv8[k][l];
```

Аналогічно описаному вище виконується зворотне перетворення, результати якого записуються у масив tr для R, tg для G tb для B.

Візуалізація відновленого зображення також виконується аналогічно як і в модулі прямого перетворення.

Текст програми для цих модулів наведено в додатку В:

- метод button22_Click() - модуль кодування з використанням ДКП;
- метод button23_Click() - модуль декодування з використанням ДКП;
- метод button13_Click() - модуль кодування з використанням перетворення Уолша-Адамара;
- метод button14_Click() - модуль декодування з використанням перетворення Уолша-Адамара.

3.4 Розробка модуля групування та зонального відбору

Вхідними є файли ущільненого зображення, з цілочисельно квантованими коефіцієнтами перетворення Уолша-Адамара. Оскільки цей файл після перетворення Уолша-Адамара ущільнювався адаптивним арифметичним кодером з алгоритмом передбачення DMC (Dynamic Markov compression), то спочатку потрібно його розархівувати. Для цього використовується модуль dmc,exe, який знаходиться в папці DOTCodec. Цей модуль запускається на виконання як зовнішній процес вхідними даними для якого є назва ущільненого файлу:

```
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL,
NULL, &si, &pi))
    {
        printf("CreateProcess failed (%d).\n", GetLastError());
```

```
return;
}
```

В результаті деархівації утворюється файл коефіцієнтів з розширенням wh8, формат якого такий:

00000000 – 000000BF – коефіцієнти квантування (три матриці 8x8 байт для квантування по кожній складовій кольору RGB);

000000C0 – END OF FILE – коефіцієнти ДОП.

Цей файл і буде вхідним файлом. Для запису згрупованих коефіцієнтів відкривається файл з розширення .grp:

```
ifstream in_file(name1, ios::binary);
ofstream out_file(name0, ios::binary);
```

Далі читаємо з вхідного файлу значення коефіцієнтів цілочислового квантування і ініціалізуємо, візуалізуємо матриці квантування, а також записуємо їх у вихідний файл:

```
n = 8;
for (int k = 0; k < n; k++) {
    for (int l = 0; l < n; l++)
    {
        in_file.read((char*)&kv8[k][l], 1);
        dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
        out_file.write((char*)&kv8[k][l], 1);
…   …
    }
}
```

Візуальний компонент dataGridView2 візуалізує коефіцієнти квантування для компоненти R кольору, для компоненти G використовується dataGridView5, а для компоненти кольору B – dataGridView6.

Для групування коефіцієнтів формуємо блоки коефіцієнтів 8x8, читаючи вхідний файл. Оскільки на етапі дослідження використовуються зображення у форматі 512x512, то кількість таких блоків буде 64x64. Тоді процес формування блоків можна подати так:

```
for (int x = 0; x <= 63; x++) {
    for (int y = 0; y <= 63; y++){
        //----RRRR-----
        for (k = 0; k < n; k++) {
            for (l = 0; l < n; l++) {
                in_file.read((char*)&tr[k][l], 2);
// …}}}}
```

Формування блоків виконується по кожній компоненті кольору RGB. Масив tr[k][l] містить коефіцієнти для складової кольору R, tg[k][l] – G, tb[k][l] – B.

Згруповані коефіцієнти записуються в масиви gwhr – складова кольору R, gwhg – G, gwhb – B:

```
gwhr[64 * j + x][64 * i + y] = tr[i][j];
gwhg[64 * j + x][64 * i + y] = tg[i][j];
gwhb[64 * j + x][64 * i + y] = tb[i][j];
```

де x, y – координати блока в площині 64x64;

j, i – координати значення коефіцієнту в блоці 8x8.

Крім того, виконується візуалізація коефіцієнтів дискретного ортогонального перетворення:

```
newColor1 = Color::FromArgb(r, g, b);
image2->SetPixel(64*j + x, 64*i + y, newColor1);
```

де image2 – структура даних типу Bitmap^;

newColor1 структура даних типу Color.

Для відображення цих коефіцієнтів у вікні додатку необхідний лише один рядок коду:

pictureBox2->Image = image2;

Тобто зображення коефіцієнтів з’явиться у правому полі для виведення зображень.

Запис згрупованих коефіцієнтів у файл виконується послідовно – спочатку всі коефіцієнти F(0,0) рядок за рядком, потім F(0,1) і т. д.:

```
out_file.write((char*)&gwhr[64*x+k][64*y+l], 2);
out_file.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
out_file.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
```

Тут змінні x, y можуть приймати в циклі значення 0…7, а змінні k, l в циклі приймають значення 0…63.

Для виконання зонального відбору в цикл запису у файл необхідно внести ряд умов, які забороняють запис у файл блоків коефіцієнтів певної частоти розміром 64x64. Наприклад так:

if ((x>=znx)&(x<=zvx)& (y>=zny)&(y<=zvy) {//запису немає} else {//запис у файл};

де znx, zny – нижня границя;

zvx, zvy – верхня границя.

На першому етапі налагодження цей рядок за коментовано.

І на останньому етапі згруповані коефіцієнти записуються у файл .grpc в ущільненому виді з використанням арифметичного кодера dmc.exe, якому необхідно передати такі параметри:

```
name3 = "d:/DOTCodec/dmc.exe c " + name + ".grp " + name + ".grpc";
```

де «c» - режим ущільнення;

name + ".grp " – назва файлу із згрупованими коефіцієнтами;

name + ".grpc" – назва ущільненого файлу.

Текст програми для цього модуля наведено в додатку В – метод button15_Click().

3.5 Розробка модуля дегрупування

Вхідними є файли ущільненого зображення з розширенням .grpc, із згрупованими цілочисельно квантованими коефіцієнтами перетворення Уолша-Адамара. Оскільки цей файл після перетворення Уолша-Адамара ущільнювався адаптивним арифметичним кодером з алгоритмом передбачення DMC (Dynamic Markov compression), то спочатку потрібно його розархівувати. Для цього використовується модуль dmc,exe, який знаходиться в папці DOTCodec. Цей модуль запускається на виконання як зовнішній процес вхідними даними для якого є назва ущільненого файлу аналогічно модулю групування.

В результаті деархівації утворюється файл коефіцієнтів з розширенням .grp, формат аналогічний формату файла описаному в пункті 3.2 за виключенням того, що коефіцієнти згруповані, а також можливо і виконано зональний відбір:

00000000 – 000000BF – коефіцієнти квантування (три матриці 8x8 байт для квантування по кожній складовій кольору RGB);

000000C0 – END OF FILE – згруповані коефіцієнти ДОП.

Цей файл і буде вхідним файлом. Для запису дегрупованих коефіцієнтів відкривається файл з розширення .dgrp.

Далі читаємо з вхідного файлу значення коефіцієнтів цілочислового квантування і ініціалізуємо, візуалізуємо матриці квантування, а також записуємо їх у вихідний файл. Аналогічно попередньому пункту візуальний компонент dataGridView2 візуалізує коефіцієнти квантування для компоненти

R кольору, для компоненти G використовується dataGridView5, а для компоненти кольору B – dataGridView6.

Згруповані коефіцієнти читаються із вхідного файлу записуються в масиви gwhr – складова кольору R, gwhg – G, gwhb – B:

```
in_file.read((char*)&gwhr[64*x+k][64*y+l], 2);
in_file.read((char*)&gwhg[64 * x + k][64 * y + l], 2);
in_file.read((char*)&gwhb[64 * x + k][64 * y + l], 2);
```

де x, y – координати блока в площині 8x8;

k, l – координати значення коефіцієнту в блоці 64x64.

Якщо виконувався зональний відбір, то при формуванні масивів коефіцієнтів потрібно заповнити нулями відповідні ділянки. Для цього в цикл зчитування коефіцієнтів з файлу необхідно додати такий код:

if ((x>=znx)&(x<=zvx)& (y>=zny)&(y<=zvy) {gwhr[64*x+k][64*y+l]=0; gwhg[64*x+k][64*y+l]=0 gwhb[64*x+k][64*y+l]=0;} else {//читати файл};

де znx, zny – нижня границя;

zvx, zvy – верхня границя.

Для дегрупування коефіцієнтів формуємо блоки де групованих коефіцієнтів 8x8. Оскільки на етапі дослідження використовуються зображення у форматі 512x512, то кількість таких блоків буде 64x64. Тоді процес формування блоків можна подати так:

```
for (int x = 0; x <=63; x++)
{for (int y = 0; y <= 63; y++)
        {n = 8;
                for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++)
                        {
                                tr[i][j]=gwhr[64 * j + x][64 * i + y];
// … }}}}
```

І дальше в межах цього циклу дегруповані блоки коефіцієнтів розміром 8x8 по кожній компоненті кольору RGB записуються у вихідний файл:

```
out_file.write((char*)&tr[k][l], 2);
out_file.write((char*)&tg[k][l], 2);
out_file.write((char*)&tb[k][l], 2);
```

Але візуалізовані вони будуть у згрупованому вигляді аналогічно пункту 3.2.

І на останньому етапі дегруповані коефіцієнти записуються у файл .dgrpc в ущільненому виді з використанням арифметичного кодера dmc.exe/

Текст програми для цього модуля наведено в додатку В – метод button16_Click().

3.6 Розробка модуля векторного квантування

Вхідними даними є кольрові та чорно-білі (grayscale) зображення у форматі .BMP розміром 512x512. У якості векторного квантувача використовується двовимірна карта Кохонена. Початковий розмір карти Кохонена 16x16 при розмірності кластерного елемента 2x2. Навчальними векторами будуть всі фрагменти початкового зображення розміром 2x2, що примикають один до одного.

При відкритті вхідного зображення відкривається стандартне вікно вибору файла і зображення записується у структуру image1 типу Bitmap^:

```
image1 = gcnew Bitmap(openFileDialog1->FileName, true);
```

Три компоненти кольору RGB записуються в циклі в три масиви:

```
Color pixelColor = image1->GetPixel(x, y);
    gwhr[x][y] = pixelColor.R;
    gwhg[x][y] = pixelColor.G;
    gwhb[x][y] = pixelColor.B;
```

Також виконується початкова ініціалізація векторного квантувача випадковими значеннями:

```
for (i = 0; i < 32; i++) {
        for (j = 0; j < 32; j++) {
                w[i][j] = gwhr[(i)][(j)]; }}
```

Далі виконується навчання мережі (карти Кохонена). Для цього змінній dmin присвоюється значення більше ніж сума квадратів значень фрагмента 8x8 при значенні кожного 256. Тобто

dmin>256^2 *64> 4 194 304.

Норма навчання nn=0.9. Для кожного фрагмента початкового зображення розміром 2x2 по кожній компоненті RGB обчислюється середньоквадратичне відхилення кожного кластерного елемента від початкового фрагмента:

```
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +(w[2
* i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y])
+
(w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y
+ 1]) +(w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] -
gwhr[2 * x + 1][2 * y + 1]);
```

І якщо d<dmin, то в змінні ci та cj зберігаються координати цього кластерного елемента, а значення dmin стає рівним d:

```
dmin = d;  ci = i; cj = j;
```

Після порівняння всіх кластерних елементів мережі з вхідним вектором значення коефіцієнтів кластерного елемента з найменшим dmin коригується:

w[2 * ci][2 * cj] = w[2 * ci][2 * cj] + short int(nn * (gwhr[2 * x][2 * y] - w[2 * ci][2 * cj]));

w[2 * ci + 1][2 * cj] = w[2 * ci + 1][2 * cj] + short int(nn * (gwhr[2 * x + 1][2 * y] - w[2 * ci + 1][2 * cj]));

w[2 * ci][2 * cj + 1] = w[2 * ci][2 * cj + 1] + short int(nn * (gwhr[2 * x][2 * y + 1] - w[2 * ci][2 * cj + 1]));

w[2 * ci + 1][2 * cj + 1] = w[2 * ci + 1][2 * cj + 1] + short int(nn * (gwhr[2 * x + 1][2 * y + 1] - w[2 * ci + 1][2 * cj + 1]));

Після порівняння всіх векторів з вхідного зображення з кожним з кластер ним елементом значення норми навчання зменшуються на 0.01:

nn=nn-0.01;

Процес повторюється в циклі до тих пір поки норма навчання nn не стане меншою 0.01.

Після навчання виконується процес векторного квантування аналогічно етапу навчання з тією різницею, що не виконується корекція значень коефіцієнтів кластерного елемента. Тобто для кожного фрагменту зображення розміром 2x2 по кожній компоненті кольору RGB знаходиться кластерний елемент, який забезпечує найменше середньоквадратичне відхилення. Номер цього кластерного елемента зберігається в масиві:

gwhrvk[x][y] = ci + cj * 16;

gwhrvk[x][y] = ci + cj * 16;

gwhrvk[x][y] = ci + cj * 16;

де gwhrvk – результати векторного квантування для компоненти R;

gwhgvk – результати векторного квантування для компоненти G;

gwhbvk – результати векторного квантування для компоненти B.

Увихідний файл записується карта Кохонена після навчання і двовимірні масиви gwhrvk, gwhgvk, gwhbvk з результатами векторного квантування. При розмірі карти Кохонена 16x16, розмірності кластерного елемента 2x2 і двухбайтних коефіцієнтах цієї карти кількість байт, яка витрачається на зберігання карти Кохонена складе:

16x16x4x2=2048 байт.

А кількість байт, яка витрачається у цьому випадку на зберігання квантованих значень зображення при початковому розмірі зображення 512x512 складе:

256x256x3=196608 байт.

Байти компонент записуються у файл у такому порядку: RGBRGB…

Таким чином, розмір файлу з даними зображення після векторного квантування складе:

196608+2048=198656 байт.

Цей файл має розширення .vim. Після ущільнення його арифметичним кодером dmc.exe додається нове розширення .vimc. Для ущільнення файлу .vim необхідно функції CreateProcess () передати назву вхідного і вихідного файлів, а також задати режим ущільнення “c”:

String^ name3 = "d:/DOTodec/dmc.exe c " + name + ".vim " + name + ".vimc";

Крім того цей модуль для контролю виконує декватування компонет зображення і результат виводить у вікно додатку в праве поле для виведення зображень.

Текст програми для цього модуля наведено в додатку В – метод button19_Click().

3.7 Розробка модуля векторного деквантування

Вхідними для роботи модуля є файли з розширенням .vimc, які були сформовані на етапі векторного квантування. Векторне деквантування виконує модуль button20_Click ().

На початку роботи виконується арифметичне декодування і результати записуються у файл з розширенням .vim. Перші 2048 байт у цьому файлі коефіцієнти мережі Кохонена, які читаються з цього файлу і записуються у масив w[32][32]:

```
in_file.read((char*)&w[i][j], 2);
```

З вхідного файлу читаються квантовані значення трьох складових кольору, які записуються у відповідні масиви:

```
in_file.read((char*)&gwhrvk[x][y], 1);
in_file.read((char*)&gwhgvk[x][y], 1);
in_file.read((char*)&gwhbvk[x][y], 1);
```

Для виконання векторного деквантування з кожного з цих масивів необхідно вибрати координати кластерного елемента карти Кохонена, який є найближчим до чергового фрагмента зображення розміром 2x2 і значення коефіцієнтів цього кластерного елемента і будуть деквантованими значеннями пікселів зображення:

```
ci = gwhrvk[x - 0][y - 0] & 15;
cj = (gwhrvk[x - 0][y - 0] / 16) & 15;
gwhr[2 * x][2 * y] = w[2 * ci][2 * cj];
gwhr[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
gwhr[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
gwhr[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
```

Аналогічні дії виконуються для кожного фрагменту зображення розміром 2x2 для кожної складової кольору.

Для виведення декодованого зображення у праве поле для виведення зображення спочатку значення неквантованих пікселів записуються у масив image2:

```
newColor1 = Color::FromArgb(r, g, b);
image2->SetPixel(x, y, newColor1);
```

Коли масив image2 буде повністю сформований, власне візуалізація зображення виконується за рахунок простого присвоювання властивості Image форми pictureBox2 значень цього масиву:

```
pictureBox2->Image = image2;
```

Завершується робота модуля закриттям файлу.

Текст програми для цього модуля наведено в додатку В.

3.8 Розробка модулів навчання, векторного квантування та деквантування коефіцієнтів ДОП

Для дослідження векторного векторного квантування коефіцієнтів дискретних ортогональних перетворень розреблено три методи:

- метод button21_Click (), призначений для навчання мережі типу двовимірна карта Кохонена;
- метод button17_Click (), призначений дослідження ущільнення зображень з векторним квантуванням коефіцієнтів ДОП;
- метод button18_Click (), призначений векторного деквантування файлів коефіцієнтів ДОП.

Для навчання нейронної мережі використовуються файли з ущільненим зображенням на основі ДОП, в якості яких використовується дискретне косинус не перетворення або перетворення Уолша-Адамара. Оскільки ці файли ущільнені арифметичним методом, то спочатку виконужться арифметичне декодування і результати записуються у файл до назви якого додається розширення .wh8.

У цього файлі перші 192 байти займають матриці, які використовувались для цілочислового квантування коефіцієнтів ДОП. Далі читаються і групуються коефіцієнти ДОП по кожній складовій кольору, які записуються у двовимірні масиви gwhr, gwhg та gwhb:

```
gwhr[64 * j + x][64 * i + y] = tr[i][j];
gwhg[64 * j + x][64 * i + y] = tg[i][j];
gwhb[64 * j + x][64 * i + y] = tb[i][j];
```

Ці згруповані коефіцієнти використовуються для навчання нейронної мережі, коефіцієнти якої зберігаються у масиві w[i][j].

Навчена мережа зберігається у файлі map.kh, тому спочатку перевіряємо наявність цього файлу в робочій директорії додатку і якщо він відсутній, то

створюємо його. А якщо присутній, то ініціалізуємо коефіцієнти мережі w[i][j], читаючи цей файл. Після навчання мережі вона знову зберігається у файлі map.kh.

Вхідними даними для модуля векторного квантування коефіцієнтів ДОП є файли з ущільненими коефіцієнтами ДКП або Уоша-Адамара та файл map.kh.

Тому спочатку виконується арифметичне декодування, а потім групування коефіцієнтів, векторне квантування і арифметичне кодування (дивись пп. 3.4-3.3.7). Результати записуються у файл з розширенням .vkkc.

Модуль векторного деквантування коефіцієнтів ДОП виконує всі операції у зворотному порядку, тільки замість векторного квантування виконується векторне деквантування. Результати зберігаються у вихідному форматі модулів виконання ДКП або Уолша-Адамара.

Текст програми для цих модулів наведено в додатку В.

3.9 Розробка модулів комбінованого квантування та деквантування коефіцієнтів ДОП

Комбіноване квантування виконує метод button24_Click (), який викликається на виконання командою VectKVCm. Вхідними даними для цього модуля є файли з виходу модулів ДКП та перетворення Уолша-Адамара з розширеннями ,dctc або .dm8 , які викликаються на виконання командами DCTCoder або WHCoder .

Схема роботи модуля така:

Арифметичне декодування->Групування коефіцієнтів-> Навчання векторного квантувача для високочастотних коефіцієнтів->Векторне квантування високочастотних коефіцієнтів->Запис у вихідний файл коефіцієнтів цілочислового та векторного квантувача, низькочастотних трансформант без змін, векторноквантованих височастотних трансформант->Арифметичне кодування вихідного файлу.

Оскільки файли .dctc або .dm8 ущільнені арифметичним методом, то на початку роботи модуля виконується декодування цих файлів і результати записуються в проміжний файл. Потім трансформанти перетворення поблочно зчитуються із цього проміжного файлу і записуються у двовимірні масиви розміром 8x8:

- масив tr – складова кольору R;
- масив tg – складова кольору G;
- масив tb – складова кольору B/

При групуванні значення з цих масивів переписуються у двовимірні масиви розміром 512x512:

- масив gwhr – складова кольору R;
- масив gwhg – складова кольору G;
- масив gwhb – складова кольору b;

Індекси цих масивів змінюються згідно виразам

$$64 * j + x,\ \ 64 * i + y,$$

що і забезпечує групування трансформант.

Навчання мережі Кохонена розміром 4x4 для векторного кантувача виконується у циклі для високочастотних компонент:

```
for (int y = 64; y < 256; y++)
    {for (int x = 64; x < 256; x++)
        { // Навчання мережі }}
```

Після навчання мережі її коефіцієнти w[i][j] можна використовувати для векторного квантування.

Векторне квантування також виконується в аналогічному циклі, тобто низькочастотні трансформанти F (0,0)…F (0,7); F (1,0)…F(1,7); F(2,0)…F (7,0); F (2,1)…F(7,1) не піддаються векторному квантуванню, а залишаються цілочисельно квантованими. Векторноквантовані значення трансформант записуються в масиви:

- масив gwhrvk – складова кольору R;
- масив gwhgvk – складова кольору G;

- масив gwhbvk – складова кольору b.

Оскільки розмір двовимірної карти Кохонена 4x4, то для указання координат найближчого кластерного елемента після векторного квантування потрібно 4 біта для номера рядка і 4 біта для номера стовпця, тобото 1 байт. Тому запис в ці масиви виконується так:

gwhrvk[x - 64][y - 64] = ci + cj * 16;

В старшу тетраду байта записується номер рядка, а в молодшу номер стовпця (або навпаки).

Формат вихідного файлу з груповаими та квантованими трансформантами (має розширення .cm) такий:

- 192 байти коефіцієнти для цілочислового квантування;
- 2048 байт коефіцієнти векторного кантувача;
- 688128 цілочисельно квантовані згруповані трансформанти 0,1 рядок, 0,1 стовпець;
- 110592 байти векторно квантовані високочастотні трансформанти.

Тобто розмір вихідного не ущільненого файлу з розширенням .cm 800960 байт. Після арифметичного кодування формується ущільнений файл з розширенням .cmc.

Комбіноване деквантування виконує метод button25_Click (), який викликається на виконання командою VectDKVCm. Вхідними даними для цього модуля є файли з виходу модуля комбінованого векторного квантування з розширеннями ,cmc.

Схема роботи модуля така:
Арифметичне декодування->Векторне деквантування високочастотних трансформант->Дегрупування трансформант->Запис у вихідний файл коефіцієнтів цілочислового квантування, трансформант ДОП->Арифметичне кодування вихідного файлу.

Перед початком роботи векторного деквантування з вхідного файлу зчитуються коефіцієнти карти Кохонена і записуються у масив w. Також

зчитуються значення трансформант 0,1 рядка, 0,1 колонок, які записуються у відповідні масиви та значення векторноквантованих трансформант, які також записуються у свої масиви.

При виконанні векторного деквантування формуються значення індексів для вибору кластерного елемента карти Кохонена:

```
ci = gwhrvk[x - 64][y - 64] & 15;
cj = (gwhrvk[x - 64][y - 64] / 16) & 15;
```

Далі з масиву w вибараюся значення коефіцієнтів відповідного кластерного елемента карти Кохонена і записуються як трансформанти ДОП у відповідні масиви.

Формат вихідного файлу такий:

- 192 байти коефіцієнти для цілочислового квантування;
- 1 572 864 байти Трансформанти ДОП.

Розмір вихідного файлу до ущільнення 1573056 байт. Після ущільнення цей файл записується з розширенням .rcmc і тепер його можна переглядати з використанням команд WHDecoder або DCTDecoder.

Текст програми для цих модулів наведено в додатку В

3.10 Висновки

1. В якості мови програмування обрано мову C/C++ та середовище розробка Microsoft Visual Studio, оскільки ці засоби забезпечують найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.
2. Виконано розробку коду основних модулів для дослідження квантування трансформант ДОП. Додаток побудовано за мультимодульною архітектурою, оскільки дослідження різних методів квантування компонент зображення вимагає великої кількості експериментальних досліджень і змін у коді у процесі цих досліджень.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ КВАНТУВАННЯ КОМПОНЕНТ ЗОБРАЖЕННЯ

4.1 Керівництво користувача

Програмний продукт розроблений під платформу Windows. Комп’ютер повинен не менше 4 Гб оперативної пам’яті та 80 Мб вільного місця на жорсткому диску.

Представлена програма призначена для дослідження застосування різних методів квантування зображень та коефіцієнтів двовимірних ортогональних перетворень зображень і була розроблена в середовищі Visual Studio 2019 мовою програмування C++. Програма може працювати на комп’ютері з операційною системою Windows XP, Windows 7, Windows 10. Програма може знайти застосування в навчальному процесі при вивченні методів кодування зображень.

Для використання програми потрібно завантажити папку DOTCodec на будь-який диск комп’ютера. Файловий склад програми такий:

файл DOTCodec.exe – головний модуль програми;

файл DMC.exe – арифметичний кодер-декодер;

lena512color.tiff, lena512.bmp – технологічні зображення.

Щоб запустити програму на виконання потрібно вибрати файл DOTCodec.exe, що знаходиться у папці DOTCodec. Після запуску програми з’явиться головне вікно програми (рис. 4.1).

Головне вікно програми містить такі елементи:

- два поля для виведення зображень;
- Три таблиці розміром 8x8 (матриці квантування) для введення коефіцієнтів для цілочислового квантування для кожної компоненти кольору;
- кнопка LoadImage для завантаження зображення в ліве поле для виведення зображень;
- кнопка CopyImage для копіювання зображення з лівого поля в праве;
- кнопка WHCoder для виконання перетворення Уолша-Адамара над зображеннями при розмірах блоків 8x8, коефіцієнти перетворення записуються у файл коефіцієнтів з розширенням .whc;
- кнопка WHDecoder для відновлення зображення з файлу коефіцієнтів перетворення Уолша-Адамара при розмірах блоків 8x8, виводиться в праве поле;
- кнопка DCTCoder для виконання ДКП над зображеннями при розмірах блоків 8x8, коефіцієнти перетворення записуються у файл коефіцієнтів з розширенням .dct, а після ущільнення у файл з розширеннм .dctc;
- кнопка DCTDecoder для відновлення зображення з файлу коефіцієнтів ДКП при розмірах блоків 8x8, виводиться в праве поле;
- Кнопка Group8 для групування коефіцієнтів ДКП або перетворення Уолша-Адамара. Вхідними є файл из ущільненими коефіцієнтами ДКП або перетворення Уолша-Адамара (файл из розширенням .dm8 або .dctc). В результаті роботи модуля утворюються такі робочі файли:

  - файл з розширенням .wh8 – не ущільнені коефіцієнти ДКП або перетворення Уолша-Адамара;

- файл з розширенням .grp - згруповані коефіцієнти ДОП;

- файл з розширенням .grpc – ущільнені арифметичним методом згруповані коефіцієнти ДОП.

- Кнопка DeGroup8 для дегрупування коефіцієнтів ДКП або перетворення Уолша-Адамара. Вхідними є файл из згрупованими ущільненими коефіцієнтами ДКП або перетворення Уолша-Адамара (файл из розширенням .grpc). В результаті роботи модуля утворюються такі робочі файли:

  - файл з розширенням .grp – не ущільнені згруповані коефіцієнти ДКП або перетворення Уолша-Адамара;

  - файл з розширенням .dgrp - дегруповані коефіцієнти ДОП;

  - файл з розширенням .dgrpc – ущільнені арифметичним методом дегруповані коефіцієнти ДОП.

- кнопка VectIm виконує векторне квантування початкового зображенняю В результаті роботи програми утворюється:

  - файл з роширенням .vim – векторно-квантовані дані, отримані після векторного квантування; файл з роширенням ,vimc – ущільнений арифметичним методом файл .vim;

- кнопка DeVectIm виконує векторне деквантування файлу .vimc В результаті роботи програми утворюється

  - файл з роширенням .vim – векторно-деквантовані дані, які виводяться у праве поле виведення зображень;

- кнопка NetworkTr виконує навчання мережі типу двовимірна карта Кохонена. Результати навчання зберігаються у файлі map.kh;

- кнопка VectorKV виконує векторне квантування коефіцієнтів ДОП, вхідними даними є ущільнені файли з коефіцієнтами ДКП або перетворення Уолша-Адамара, результати векторного квантування

записуються у файл з розширенням .vkk, а після арифметичного кодування у файл з розширенням .vkkc.

- кнопка VectorDKV виконує векторне деквантування коефіцієнтів ДОП, результатом є файл у вхідному форматі для векторного квантування (кнопка VectorKV);
- кнопка VectKVCm виконує цілочислове квантування низькочастотних коефіцієнтів ДОП і векторне квантування високочастотних коефіцієнтів ДОП, результатом є файл з квантованими коефіцієнтами ДОП з розширенням .cm і цей же файл ущільнений арифметичним методом з розширенням .cmc. Вхідними даними є файли з виходу модулів виконная ДКП або перетворення Уолша-Адамара;
- кнопка VectDKVCm виконує цілочислове деквантування низькочастотних коефіцієнтів ДОП і векторне деквантування високочастотних коефіцієнтів ДОП, результатом є файл з деквантованими коефіцієнтами ДОП і цей же файл ущільнений арифметичним методом з розширенням .rcmc;
- кнопка SD виконує обчислення середньоквадратичного відхилення зображення у лівому полі від зображення у правому полі по кожній компоненті кольору;
- кнопка SaveD записує зображення з правого поля у файл у форматі .BMP, найчастіше в це поле виводиться декодоване зображення;
- кнопка About для виведення відомостей про роботу і автора розробки (рис. 4.2).

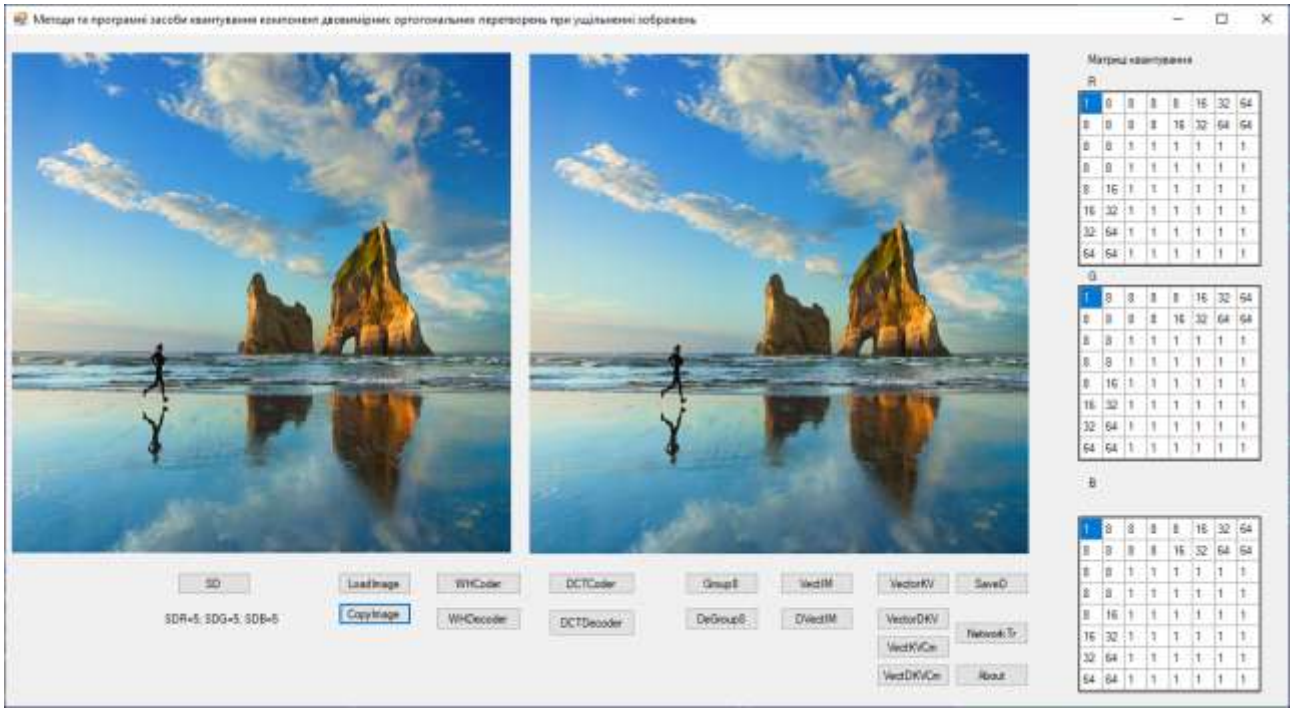


Рисунок 4.1 – Головне вікно програми

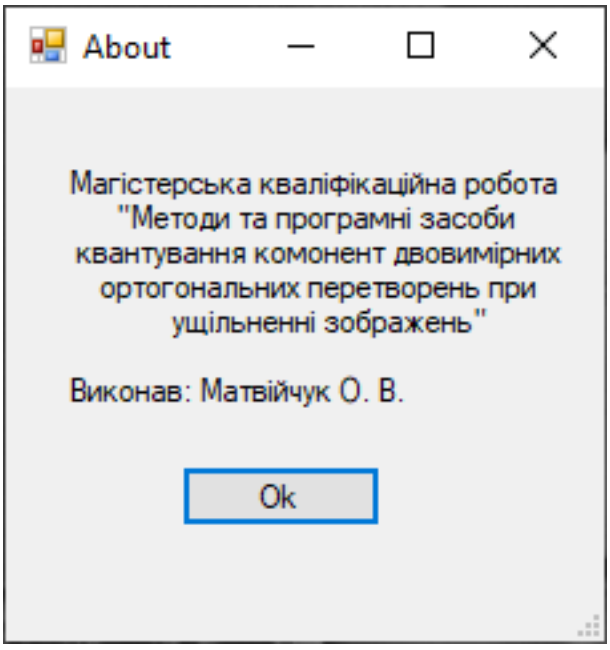


Рисунок 4.2 – Вікно About

Вибір будь-якої команди (крім SD і About) відкриває стандартне вікно вибору файлу. Необхідно вибрати файл, натиснути мишкою кнопку «Відкрити». Додаткові функції залежать від призначення кнопки.

При виборі команди SD розраховується та відображається середнє квадратичне відхилення лівого зображення від правого зображення для кожної компоненти кольору: SDR – червоний; SDG – зелений; SDB синього кольору (Малюнок 4.1 під кнопкою SD).

Для проведення дослідження можна змінити коефіцієнт квантування для кожного кольору (RGB), клацнувши лівою кнопкою миші у відповідному полі матриці квантування та ввівши нове значення (рис. 4.1).

Використання інших програмних команд є інтуїтивно зрозумілим і не потребує додаткових пояснень.

4.2 Дослідження цілочислового квантування і групування коефіцієнтів ДОП

Початкове напівтонове або кольорове зображення представлене у форматі 512x512 з глибиною 24 Біт/піксель.

Буде досліджуватись залежність ступеня ущільнення зображення від коефіцієнтів цілочислового квантування та від їх групування за частотами для таких ДОП:

- Перетворення Уолша-Адамара при розмірі блоків 8x8;
- Дискретне косинусне перетворення при розмірі блоків 8x8;

Цілочислове квантування, групування та зональний відбір коефіцієнтів. Проведено тестування для декількох зображень при значеннях коефіцієнтів квантування наведених на рис. 4.3 (R) з групуванням коефіцієнтів ДОП і без групування.

Згруповані коефіцієнти ДКП та перетворення Уолша-Адамара, отримані в процесі роботи програми при кодуванні зображення, розміром 512 x 512 у з глибиною 24 біт/піксель при розмірі блоків 8x8 продемонстровані на рис. 4.4. Як показує аналіз цього рисунка суттєві відмінності в цих коефіцієнтах

відсутні, тому можна очікувати, що квантування цих коефіцієнтів можна виконувати за однаковою схемою.

Для ущільнення без втрат файлів коефіцієнтів з розширенням будемо використовувати арифметичний кодер dmc.exe. Результати наведено в табл. 4.1.

Як показує аналіз цієї таблиці завдяки групуванню коефіцієнтів перетворення файл з ущільненими трансформантами перетворення зменшився у 1,5-2,5 рази у порівнянні з файлом без групування коефіцієнтів.

Ще один висновок – ДКП при тих же коефіцієнтах цілочислового квантування забезпечує стиснення на 5-10 % більше.

Таблиця 4.1 – Тестування програми на прикладі файлів Lena

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групування, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale)	Уолша-Адамара	180782	78 307	4	Відмінна
	ДКП	169534	72312	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
716 794 (color)	Уолша-Адамара	214070	170 338	5	Відмінна
	ДКП	203859	161 505	4	
716 794 (color)		95 646 (jpeg)	95 646 (jpeg)	4	Відмінна

Матриці квантування

R

1	1	1	1	1	1	1	1
1	1	8	8	16	32	64	64
1	8	8	16	32	64	64	64
1	8	16	32	64	64	64	64
1	16	32	64	64	64	64	64
1	32	64	64	64	64	64	64
1	64	64	64	64	64	64	64
1	64	64	64	64	64	64	64

G

2	4	4	4	4	4	4	4
4	4	8	8	8	8	8	16
4	8	8	8	8	8	16	16
4	8	8	8	8	16	16	16
4	8	8	8	16	16	16	16
4	8	8	16	16	16	16	16
4	8	16	16	16	16	32	32
4	16	16	16	16	16	32	32

Рисунок 4. 3 – Матриці квантування

Групування коефіцієнтів дозволяє просто організувати зональний відбір. Внесемо зміни в програмне забезпечення, які дозволять не записувати у вихідний файл такі коефіцієнти:

F (4,4), F (4,5), F (4,6), F (4,7), F (5,4), F (6,4), F (7,4) F (5,5), F (5,6), F (5,7), F (6,5), F (7,5), F (6,6), F (6,7), F(7,6), F(7,7),

Тобто в площині згрупованих коефіцієнтів (рис. 4.4) це 4-а четверть. Зміни дуже прості:

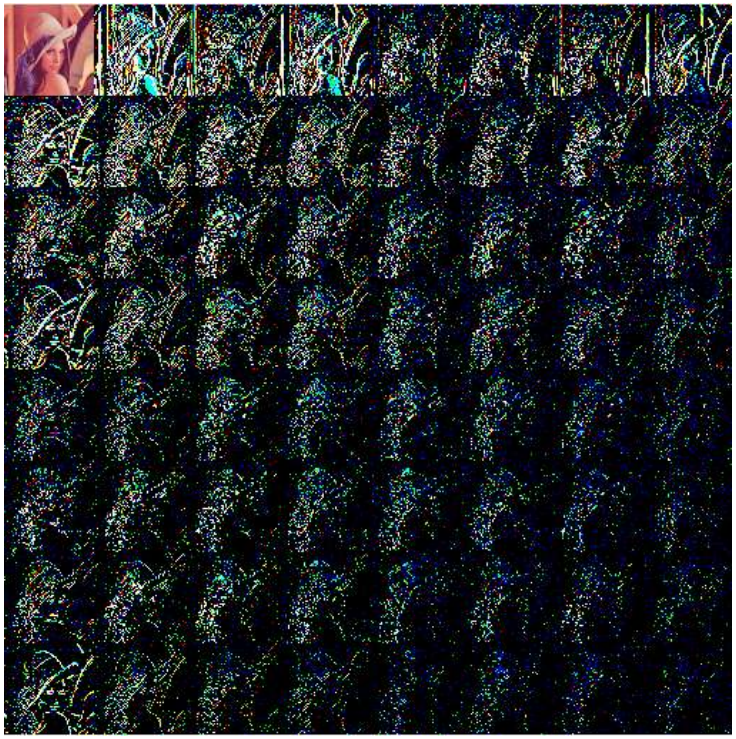
Для модуля групування

if ((x > 3) & (y > 3)) {//} else {//Писати у файл};

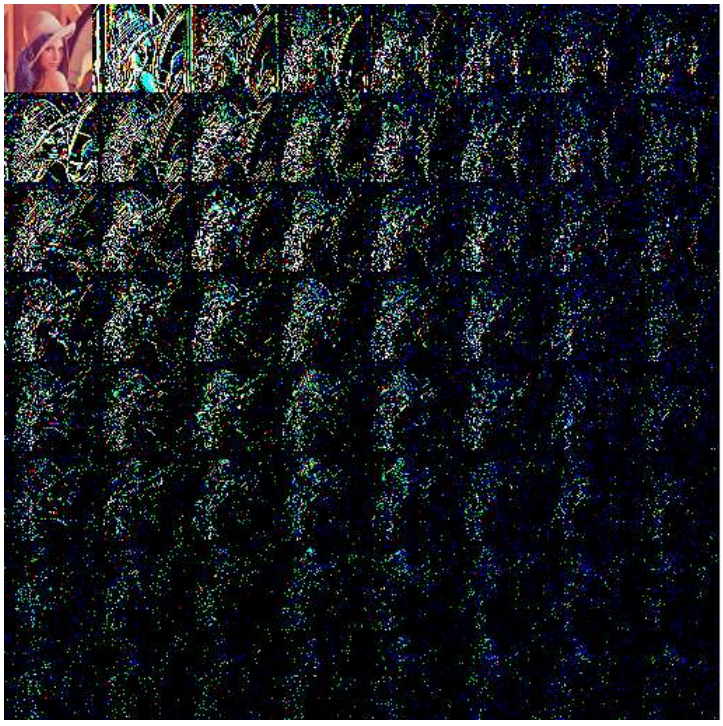
Для модуля дегрупування

if ((x > 3) & (y > 3)) {//запис «0»} else {//Читати з файлу}.

Результати наведено в табл. 4.2, де МК R, МК G – матриці квантування відповідні рис. 4.3.



а)



б)

Рисунок 4.4 – Згруповані коефіцієнти ДОП: а)коефіцієнти перетворення Уолша-Адамара; б) коефіцієнти ДКП

Таблиця 4.2 – Дослідження ущільнення зображень з зональним відбором

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групування та зональним відбором, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale) МК R	Уолша-Адамара	180782	77 389	4	Відмінна
	ДКП	169534	71848	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
263 222 (grayscale) МК G	Уолша-Адамара	146 323	63 589	3	Відмінна
	ДКП	126 635	55 985	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна

Таким чином, цілочислове квантування з групуванням і зональним відбором коефіцієнтів дозволяє досягти достатньо високих коефіцієнтів ущільнення при високій якості зображення (рис. 4.5). Групування коефіцієнтів особливо важливо, якщо на етапі ущільнення без втрат застосовуються адаптивні алгоритми з передбаченням. Наприклад, технологія PPM або DMC [23-25]. Для цих алгоритмів групування є дуже важливим, оскільки вони використовують для ущільнення модель Ріссанена-Ленгдона, яка передбачає розділення кодера на два модулі – моделювальник і кодувальник.

Аналіз результатів показує, що об’єм зображення зменшується в 3-5 разів, що відповідає технічному завданню на роботу.

(A)

(B)

Рисунок 4.5 - Початкове (А – розмір файлу 786486 байт) та відновлене (В – розмір файлу файлу 115604 байт) зображення. Перетворення ДКП, матриця квантування МК G, СКВ=3

4.3 Дослідження векторного квантування зображення

Вхідним для модуля векторного квантування є напівтонове або кольорове зображення представлене у форматі 512x512 з глибиною 24 Біт/піксель.

Векторне квантування виконується при виборі команди VectIm. Параметри векторного кантувача такі:

- двовимірна карта Кохонена розміром 16x16 кластерних елементів;
- формат вхідного вектора – двовимірний вектор розміром 2x2.

Очікувані результати роботи програми:

- файл з квантованими компонентами зображення з розширенням .vim;
- ущільнений арифметичним методом файл .vim з розширенням .vimc.

Розмір файлу .vim дорівнює сумі розміру карти Кохонена і векторно квантованих компонент зображення:

16x16x4x2+(512/2)*(512/2)*1*3 =198 656 байт.

Для перевірки правильності роботи цього модуля виберемо файл кольорового зображення стандартної заставки Windows (рис. 4.5). Результати такі:

- файл .vim – має розмір 198 656 байт, що відповідає очікуваним результатам;

- ущільнений файл .vimc має розмір 84 802 байт, СКВ=6, візуальна якість зображення відмінна. Для порівняння розмір цього файлу у форматі ,jpg 88 041 байт СКВ=5.

Тобто результати порівняні з форматом ,jpg.

Однак дослідження виконані для інших зображень показали, що розміру карти Кохонена 16x16 недостатньо для отримання високоякісного відновленого зображення. Тому потрібно збільшити розмір карти Кохонена, але збільшення розміру карти Кохонена зменшує коефіцієнт ущільнення, оскільки навчена карта Кохонена також записується у вихідний файл. Наприклад, для карти Кохонена 32x32 розмір пам’яті для зберігання налаштувань векторного кантувача складе:

32*32*4*2=8192 байти.

З одної сторони це не так і багато при ущільненні файлів розміром 1 МГбайт і більше, але з іншої сторони для указання позиції найближчого кластерного елемента уже потрібно 10 біт, що збільшує файл квантованих компонент зображення в 1,5 рази і відповідно зменшується коефіцієнт ущільнення зображення.

4.4 Дослідження векторного квантування трансформант ортогональних перетворень

Для виконання дослідження будуть задіяні такі модулі:

- модуль навчання векторного кантувача коефіцієнтів ДОП у вигляді двовимірної карти Кохонена. Запускається на виконання командою NetworkTr (рис. 4.1);
- модуль векторного квантувача для трансформант ДОП. У якості векторного квантувача використовується карта Кохонена після навчання. Запускається на виконання командою VectorKV (рис. 4.1);
- модуль векторного деквантування для трансформант ДОП. У якості векторного деквантувача використовується карта Кохонена після навчання. Запускається на виконання командою VectorDKV (рис. 4.1).

Кодування зображення в експерименті виконується за такою схемою:

ДОП->Групування трансформант->Навчання мережі->Векторне квантування->Арифметичне кодування.

Для навчання мережі вхідними є файли з виходу модуля ДКП (команда DCTCoder) та модуля перетворення Уолша-Адамара (команда WHCoder).

Для навчання нейронної мережі типу карта Кохонена сформуємо файл трансформант ДОП без цілочислового квантування, тобто значення елементів матриць квантування задамо рівними «1».

Для цього запустимо на виконання додаток і встановимо значення елементів матриці квантування рівними «1», виберемо команду DCTCoder і введемо назву досліджуваного ущільненого файлу test. В результаті після ущільнення буде створено файл test.dctc розмір якого складе 404027 байт. Для порівняння початковий файл LenaGrey512.bmp має розмір 263222 байт, тобто замість ущільнення вихідний файл збільшився у 1,5 рази. Аналогічні результати отримаємо і при використанні перетворення Уолша-Адамара. Це пояснити просто – без квантування коефіцієнтів ДОП на меншу кількість рівнів ущільнення неможливе або іншими словами без втрат інформації ущільнення не працює.

Але цей файл потрібен для навчання векторного квантувача, оскільки містить всі трансформанти ДОП у початковому вигляді, у даному випадку ДКП при розмірі блоків 8x8.

Початковий розмір мережі 64x64 кластерних елементи. Якщо запустити процес навчання командою NetworkTr, то очікувані результати такі:

- буде сформовано файл map.kh з навченою мережею розміром: 64x64x4x2=32768 байт;
- візуалізовано згруповані трансформанти ДКП.

Для перевірки виберемо команду NetworkTr і задамо вхідний файл test.dctc. Отримані результати відповідають очікуваним:

- розмір файла map.kh – 32768;
- візуалізація також відповідає очікуваним результатам.

Для виконання векторного квантування потрібно вибрати команду VectorKV. Вхідними даними для роботи цього модуля виберемо файл test.dctc. В результаті отримаємо ущільнений файл зображення з векторно-квантованими коефіцієнтами ДКП, але без запису у файл самої карти Кохонена. Утворено файл test.dctc.vkkc розміром 93699 байт. Для порівняння розмір файлу JPEG 82912 байт, тобто векторне квантування при розмірах карти Кохонена 64x64 уступає за коефіцієнтом ущільнення JPEG.

Перевіримо якість відновленого зображення. Для цього виберемо команду VectorDKV, яка відновить файл test.dctc.vkkc до початкового формату з урахуванням втрат при векторному квантуванні. В результаті буде утворено файл test.dctc.vkkc.rdm8, який можна візуалізувати командою DCTDecoder (рис. 4.6). Відновлене зображення характеризується такими параметрами:

- візуальна якість – відмінна, спотворення непомітні;
- СКВ=2, що також в межах норми.

Рисунок 4.6 – Відновлене зображення після векторного квантування

Подібні результати отримані і для трансформант перетворення Уолша-Адамара.

Необхідно відзначити, що при розмірах карти Кохонена 64x64 час навчання мережі складає 600 сек, тому для зменшення цього часу необхідні мережі меншого розміру. Однак експерименти з картою Кохонена менших розмірів для квантування коефіцієнтів ДОП показали, що не забезпечуються вимоги до якості зображень, хоча векторне квантування безпосередньо елементів зображення показало, що уже при розмірі карти Кохонена 4x4 можна

отримати прийняту якість зображення. Тому доцільно було б провести експерименти застосування комбінації різних методів квантування до трансформант ДОП.

4.5 Дослідження комбінованого методу квантування трансформант ортогональних перетворень

Пропонується така схема ущільнення зображення:

- виконання ДОП;
- групування коефіцієнтів ДОП;
- для низькочастотних трансформант F (0,0)…F (0,7); F (1,0)…F(1,7); F(2,0)…F (7,0); F (2,1)…F(7,1) виконується цілочислове квантування;
- для інших трансформант векторне квантування при розмірах карти Кохонена 4x4.

Оскільки високочастотні компоненти несуть інформацію про дрібні деталі зображення до спотворення яких око людини менш чутливо, то можна зменшити вимоги до точності квантування і перейти до карти Кохонена розміром 4x4. Такий алгоритм реалізують методи button24_Click() (викликається на виконання командою VectKVCm), який забезпечує навчання мережі і векторно квантування високочастотних трансформант, та метод button25_Click() (викликається на виконання командою VectDKVCm), який забезпечує векторне деквантування високочастотних трансформант.

Для проведення досліджень потрібно сформувати файл коефіцієнтів ДОП. Для цього задіємо команду WHCoder і вибираємо файл з напівтоновим (grayscale) зображенням Lena512.bmp розміром 512x512, 24 біти на піксель. А потім виконується векторне квантування високочастотних трансформант перетворення з цього файлу (команда VectKVCm). Перед виконанням квантування виконується групування коефіцієнтів аналогічно параграфу 4.2. Згруповані трансформанти перетворення наведено на рис. 4.7 а, а коефіцієнти

цілочислового квантування на рис. 4.7 б. Тобто у цьому файлі над високочастотними коефіцієнтами не виконувалось цілочислове квантування.

а)

1	8	8	8	8	16	32	64
8	8	8	8	16	32	64	64
8	8	1	1	1	1	1	1
8	8	1	1	1	1	1	1
8	16	1	1	1	1	1	1
16	32	1	1	1	1	1	1
32	64	1	1	1	1	1	1
64	64	1	1	1	1	1	1

б)

Рисунок 4.7 – Результати роботи модуля комбінованого векторного квантування: а – згруповані коефіцієнти перетворення Уолша-Адамара; б – матриця квантування

Низькочастотні цілочисельно квантовані трансформанти передаються у вихідний файл без змін, а над високочастотними виконується векторне квантування і результати також записуються у цей файл, який ущільнюється арифметичним методом і результати записуються у файл з розширенням .cmc. Для виконання деквантування вибираємо команду VectDKVCm, результатом роботи якої є файл у форматі з виходу модуля ДОП, для перегляду якого потрібно задіяти команду WHDecoder (або команду DCTDecoder, якщо використовується ДКП). Для даного випадку результати такі:

- розмір вхідного зображення у форматі .BMP– 263222 байт;
- розмір ущільненого зображення – 57876 байт;
- візуальна якість зображення – відмінна, СКВ=5;
- час квантування – 15 сек;
- для порівняння розмір файлу .jpg – 82912 байт, візуальна якість відмінні, СКВ=1.

Результати векторного квантування для матриці квантування згідно рис. 4.7 наведено в табл. 4.3, відновлене зображення на рис. 4.8.

Як показує аналіз цієї таблиці завдяки групуванню та векторному квантуванню трансформант перетворення файл з ущільненими трансформантами перетворення для напівтонових зображень менший у порівнянні з файлом jpeg на 30 %, а файл кольорового зображення навпаки більший на 30 % у порівнянні з jpeg. Пояснення цього факту в цілому вимагає додаткових досліджень, які виходять за межі даної роботи.

Також підтверджується висновок, що ДКП при тих же коефіцієнтах цілочислового квантування забезпечує стиснення на 5-10 % більше, а також кращу візуальну якість зображення. Хоча середньоквадратичне відхилення і більше.

а)

б)

Рисунок 4.7 – Оригінальне (а) та відновлене (б) зображення при комбінованому квантуванні на основі ДКП

Рисунок 4.7 –Таблиця 4.3 – Комбіноване квантування трансформант ДОП

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групуванням та векторним квантуванням ВЧ трансформант, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale)	Уолша-Адамара	302352	57876	5	Відмінна
	ДКП	284509	50658	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
716 794 (color)	Уолша-Адамара	371283	138433	5	Відмінна
	ДКП	203008	128229	4	
716 794 (color)		95 646 (jpeg)	95 646 (jpeg)	4	Відмінна

4.6 Висновки

1. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп’ютер та її експлуатації.

2. Тестування та дослідження програми показало, що групування коефіцієнтів ДОП збільшує коефіцієнт ущільнення у 1,5-2 рази у порівнянні з методами без групування коефіцієнтів.

3. Показано, що подальше збільшення коефіцієнта ущільнення зображень можливе при застосуванні векторного квантування трансформант ДОП. Дослідження показали, що кращі результати забезпечує комбіноване квантування – для низькочастотних трансформант цілочислове квантування, для високочастотних – векторне з використанням двовимірної карти Кохонена, що дозволило підвищити коефіцієнт ущільнення Grayscale зображень на 30 % у порівнянні з методом JPEG.

Таким чином можна зробити висновок, що поставлені задачі та цілі магістерської кваліфікаційної роботи виконано.

5 ЕКОНОМІЧНА ЧАСТИНА

Виконання науково-дослідної роботи завжди передбачає отримання певних результатів і вимагає відповідних витрат. Результати виконаної роботи завжди дають нам нові знання, які в подальшому можуть бути використані для удосконалення та/або розробки (побудови) нових, більш продуктивних зразків техніки, процесів та програмного забезпечення.

Дослідження на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» може бути віднесено до фундаментальних і пошукових наукових досліджень і спрямоване на вирішення наукових проблем, пов’язаних з практичним застосуванням. Основою таких досліджень є науковий ефект, який виражається в отриманні наукових результатів, які збільшують обсяг знань про природу, техніку та суспільство, які розвивають теоретичну базу в тому чи іншому науковому напрямку, що дозволяє виявити нові закономірності, які можуть використовуватися на практиці.

Для цього випадку виконаємо такі етапи робіт:

1) здійснимо проведення наукового аудиту досліджень, тобто встановлення їх наукового рівня та значимості;

2) проведемо планування витрат на проведення наукових досліджень;

3) здійснимо розрахунок рівня важливості наукового дослідження та перспективності, визначимо ефективність наукових досліджень.

5.1 Оцінювання наукового ефекту

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР на тему «Методи та програмні засоби квантування компонент двовимірних

ортогональних перетворень при ущільненні зображень» можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання науково-дослідної роботи в балах наведені в табл. 5.1 та 5.2.

Таблиця 5.1 – Показники ступеня новизни науково-дослідної роботи виставлені експертами

Ступінь новизни	Характеристика ступеня новизни	Значення ступеня новизни, бали		
		Експерти (ПІБ, посада)		
		Катєльніков Д. І., доцент каф. ПЗ	Майданюк В. П., доцент каф. ПЗ	Черноволик Г. О., доцент каф. ПЗ
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в даній галузі науки і техніки. Отримані принципово нові факти, закономірності; розроблена нова теорія. Створено принципово новий пристрій, спосіб, метод	0	0	0
Нова	Отримана нова інформація, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснені відомі факти, закономірності,	0	57	53

	впроваджені нові поняття, розкрита структура змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів			
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв'язок (або знайдено новий зв'язок) між явищами. В принципі відомі положення розповсюджені на велику кількість об'єктів, в результаті чого знайдено ефективне рішення. Розроблені більш прості способи для досягнення відомих результатів. Проведена часткова раціональна модифікація (з ознаками новизни)	40	0	0
Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджені або поставлені під сумнів відомі факти та твердження, які	0	0	0

	потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг в порівнянні з існуючим			
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі, та не був відомий авторам	0	0	0
Середнє значення балів експертів		50,0		

Згідно отриманого середнього значення балів експертів ступінь новизни характеризується як нова, тобто отримана нова інформація, яка суттєво зменшує невизначеність наявних знань (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту) та проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

Таблиця 5.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи виставлені експертами

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали		
	Експерт (ПІБ, посада)		
	Катєльніков Д. І., доцент каф. ПЗ	Майданюк В. П., доцент каф. ПЗ	Черноволик Г. О., доцент каф. ПЗ
Відкриття закону, розробка теорії	0	0	0
Глибоке опрацювання проблеми: багатоаспектний аналіз зв’язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного	60	61	60

прогнозу			
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	0	0	0
Елементарний аналіз зв'язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	0	0	0
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	0	0	0
Середнє значення балів експертів	60,3		

Згідно отриманого середнього значення балів експертів рівень теоретичного опрацювання науково-дослідної роботи характеризується як глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу.

Показник, який характеризує рівень наукового ефекту, визначаємо за формулою [31]:

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}, \quad (5.1)$$

де $k_{нов}, k_{теор}$ - показники ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи, $k_{нов} = 50,0, k_{теор} = 60,3$ балів;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор} = 0,6 \cdot 50,0 + 0,4 \cdot 60,33 = 54,13 \text{ балів.}$$

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів виходячи з граничних значень, які наведені в табл. 5.3.

Таблиця 5.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до визначеного рівня наукового ефекту проведеної науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень», даний рівень становить 54,13 балів і відповідає статусу - середній рівень. Тобто у даному випадку можна вести мову про потенційну фактичну ефективність науково-дослідної роботи.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [31]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 26500{,}00 \cdot 32 / 22 = 38545{,}45 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-дослідної роботи	26500,00	1204,55	32	38545,45
Ст. науковий співробітник	18400,00	836,36	21	17563,64
Інженер-програміст	22300,00	1013,64	21	21286,36
Аналітик систем обробки графічних даних	24500,00	1113,64	5	5568,18
Консультант-аналітик обробки цифрових зображень	24500,00	1113,64	7	7795,45
Провідний фахівець	8300,00	377,27	11	4150,00
Всього				94909,09

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \,, \qquad (5.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М \cdot К_i \cdot К_с}{Т_р \cdot t_{зм}} \,, \qquad (5.4)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

$К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [31];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}35 / (22 \cdot 8) = 56{,}53$ *грн.*

$$З_{р1} = 56{,}53 \cdot 8{,}00 = 452{,}25 \text{ грн.}$$

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення допоміжного обладнання робочих місць дослідників	8,00	2	1,10	56,53	452,25

Інсталяція програмного забезпечення	7,50	3	1,35	69,38	520,34
Встановлення цифрових обчислювальних систем	5,60	4	1,50	77,09	431,69
Відлагодження інтерполяційних модулів графічної системи	6,35	5	1,70	87,37	554,78
Підготовка цифрової експериментальної моделі обробки зображень	5,00	5	1,70	87,37	436,83
Формування бази даних двовимірних зображень	16,00	3	1,35	69,38	1110,07
Тренування системи	5,00	2	1,10	56,53	282,66
Всього					3788,62

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (94909,09 + 3788,62) · 11 / 100% = 10856,75 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (94909,09 + 3788,62 + 10856,75) · 22 / 100% = 24101,98 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень».

Витрати на матеріали на даному етапі проведення досліджень в основному пов'язані з використанням моделей елементів та моделювання роботи і досліджень за допомогою комп'ютерної техніки та створення експериментальних математичних моделей або програмного забезпечення, тому дані витрати формуються на основі витратних матеріалів характерних для офісних робіт.

Витрати на матеріали (M), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \qquad (5.7)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, ($K_j = 1{,}1 \ldots 1{,}15$);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 3{,}0 \cdot 212{,}00 \cdot 1{,}06 - 0{,}0 \cdot 0{,}0 = 674{,}16$ *грн.*

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір	212,00	3,0	0,0	0,0	674,16

канцелярський офісний (А4-500/80)					
Папір для заміток (А5)-500/70	112,00	3,0	0,0	0,0	356,16
Начиння канцелярське Lezard Ultra	216,00	4,0	0,0	0,0	915,84
Органайзер офісний Lezard Ultra	260,00	3,0	0,0	0,0	826,80
Картридж для принтера Canon 750AF-DX	2120,00	2,0	0,0	0,0	4494,40
Диск оптичний LG-10 (CD-R)	23,00	5,0	0,0	0,0	121,90
Диск оптичний LG-W (CD-RW)	32,00	5,0	0,0	0,0	169,60
USB-пам'ять Kingstar (32 ГБ) Class 10	139,00	1,0	0,0	0,0	147,34
USB-пам'ять Kingstar (64 ГБ) Class 10 A	189,00	2,0	0,0	0,0	400,68
Всього					8106,88

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (5.8)$$

де H_j – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$).

$К_в = 1 \cdot 3152,00 \cdot 1,06 = 3341,12$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн.	Сума, грн
Мережева карта Mikrotik R11e-LoRa	1	3152,00	3341,12
Відеокарта ASUS GeForce RTX3060 12Gb DUAL OC V2 LHR (DUAL-RTX3060-O12G-V2)	1	12299,00	13036,94
Кабелі інтерфейсів	3	850,00	2703,00
Всього			19081,06

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10 \ldots 1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 18960{,}00 \cdot 1 \cdot 1{,}06 = 20097{,}60$ *грн.*

Отримані результати зведемо до таблиці 5.8.

Таблиця 5.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Мультимедійна проекційна система	1	18960,00	20097,60
Монітор Samsung LC27G55TQBIXCI	1	8999,00	9538,94
Всього			29636,54

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1{,}10...1{,}12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 7646{,}00 \cdot 1 \cdot 1{,}06 = 8104{,}76$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Прикладний пакет Microsoft Visual Studio Community 2019	1	7646,00	8104,76
Середовище програмування мови	1	4850,00	5141,00

програмування – C/C++			
База даних тестових зображень у форматі .BMP	1	4350,00	4611,00
Модель перетворення - Дискретні ортогональні перетворення	1	1800,00	1908,00
Модель перетворення - Дискретне косинусне перетворення	1	1690,00	1791,40
Модель перетворення - Уолша-Адамара	1	1969,00	2087,14
Всього			23643,30

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (45389{,}00 \cdot 2) / (3 \cdot 12) = 2521{,}61$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Електронний комплекс аналітичної системи	45389,00	3	2	2521,61

ПК DELL OPTIPLEX 7010 MFF / I5-13500T (N007O7010MFF)				
Персональний комп'ютер HP PRODESK 405 G6 SFF / RYZEN3 4300G (294D5EA)	19699,00	3	2	1094,39
Спеціалізоване робоче місце дослідника	8699,00	5	2	289,97
Пристрій виводу текстової інформації	6899,00	5	2	229,97
Оргтехніка	8400,00	5	2	280,00
Приміщення лабораторії досліджень	399000,00	25	2	2660,00
ОС Windows 11	6500,00	3	2	361,11
Прикладний пакет Microsoft Office 2019	6540,00	3	2	363,33
Всього				7800,38

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вні}}{\eta_i}, \quad (5.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,50 грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,32 \cdot 220,0 \cdot 7,50 \cdot 0,95 / 0,97 = 528,00$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Електронний комплекс аналітичної системи ПК DELL OPTIPLEX 7010 MFF / I5-13500T (N007O7010MFF)	0,32	220,0	528,00
Персональний комп'ютер HP PRODESK 405 G6 SFF / RYZEN3 4300G (294D5EA)	0,25	220,0	412,50
Спеціалізоване робоче місце дослідника	0,08	220,0	132,00
Пристрій виводу текстової інформації	0,25	4,5	8,44
Оргтехніка	0,50	2,0	7,50
Мультимедійна проекційна система	0,23	100,0	172,50
Монітор Samsung LC27G55TQBIXCI	0,06	100,0	45,00
Всього			1305,94

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також

витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» відсутні.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (94909,09 + 3788,62) · 50 / 100% = 49348,86 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 110%.

$В_{нзв}$ = (94909,09 + 3788,62) · 110 / 100% = 108567,48 грн.

Витрати на проведення науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

$В_{заг}$= 94909,09 +3788,62 +10856,75 +24101,98 +8106,88 +19081,06 +29636,54 + 23643,30 + 7800,38 +1305,94 +0,00 +0,00 +49348,86 +108567,48 = 381146,88 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 381146,88 / 0,9 = 423496,53 грн.

5.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним

процесом і часто базується на експертних оцінках, тому має вірогідний характер.

Для обґрунтування доцільності виконання науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» використовується спеціальний комплексний показник, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник K_P рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \qquad (5.17)$$

де I – коефіцієнт важливості роботи. Приймемо $I = 4$;

n – коефіцієнт використання результатів роботи; $n = 0$, коли результати роботи не будуть використовуватись; $n = 1$, коли результати роботи будуть використовуватись частково; $n = 2$, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; $n = 3$, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок. Приймемо n=3;

T_C – коефіцієнт складності роботи. Приймемо $T_C = 2$;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то $R = 4$; якщо результати роботи відповідають відомому рівню, то $R = 3$; якщо нижче відомих результатів, то $R = 1$. Приймемо $R = 4$;

B – вартість науково-дослідної роботи, тис. грн. Приймемо B =423496,53 грн;

t – час проведення дослідження. Приймемо t = 0,17 років, (2 міс.).

Визначення показників *I*, *n*, T_C, *R*, *B*, *t* здійснюється експертним шляхом або на основі нормативів [31].

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = 4^3 \cdot 2 \cdot 4 \,/\, 423{,}5 \cdot 0{,}17 = 7{,}25.$$

Якщо $K_P > 1$, то науково-дослідну роботу на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» можна вважати ефективною з високим науковим, технічним і економічним рівнем.

5.4 Висновок

Витрати на проведення науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» складають 423496,53 грн. Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

ВИСНОВКИ

У магістерській кваліфікаційній роботі проводилося дослідження залежності коефіцієнта ущільнення зображень від методу квантування трансформант двовимірних ортогональних перетворень, у результаті якого розроблено програмне забезпечення, яке виконує квантування трансформант ДКП та перетворення Уолша-Адамара.

Основні результати роботи такі:

1. Показано, що коефіцієнт ущільнення зображень з втратами на основі двовимірних ортогональних перетворень повністю визначається методом квантування трансформант цих перетворень.

2. Розроблено схему ущільнення зображень на основі двовимірних ортогональних перетворень та методи і алгоритми виконання квантування трансформант цих перетворень, які ґрунтуються на цілочисловому та векторному квантуванні.

3. Аналіз середовищ розробки та мов програмування показав, що для цієї розробки доцільним є використання середовища розробки Microsoft Visual Studio та мови програмування C++, оскільки ці засоби забезпечують найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.

4. Мовою програмування C++ розроблено додаток для дослідження квантування компонент двовимірних ортогональних перетворень. Додаток побудовано за мультимодульною архітектурою, оскільки дослідження різних методів квантування компонент зображення вимагає великої кількості експериментальних досліджень і змін у коді у процесі цих досліджень. В якості перетворень використовувалось дискретне косинусне перетворення та перетворення Уолша-Адамара.

5. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп’ютер та її експлуатації.

6. Тестування програми показало, що застосування групування трансформант ДОП після цілочислового квантування за частотами дозволяє підвищити коефіцієнт ущільнення 1,5-2 рази у порівнянні з файлами, де не використовувалось групування.

7. Показано, що подальше збільшення коефіцієнту ущільнення можна досягнути за рахунок векторного квантування трансформант перетворення з використанням нейронної мережі типу двовимірна карта Кохонена.

8. Дослідження показали, що кращі результати з точки зору швидкості ущільнення забезпечує комбіноване квантування – для низькочастотних трансформант цілочислове квантування, для високочастотних – векторне з використанням двовимірної карти Кохонена розмірністю 16x16 при розмірності вхідних векторів 2x2, що дозволило підвищити коефіцієнт ущільнення Grayscale зображень на 30 % у порівнянні з методом JPEG. Час векторного квантування зображення розміром 512x512 складає 15 сек.

9. Розрахунки затрат на роботу показали доцільність даної розробки з економічної точки зору, оскільки комплексний коефіцієнт доцільності виконання науково-дослідної роботи $K_P > 1$.

Таким чином можна зробити висновок, що поставлені задачі та цілі магістерської кваліфікаційної роботи виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кобилін О.А., Творошенко І.С. Методи цифрової обробки зображень: навч. посібник. – Харків: ХНУРЕ, 2021. – 124 с.
2. Майданюк В. П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень. Навчальний посібник. - Вінниця: ВНТУ. [Електронний ресурс]. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=122346.pdf&x=1&card_id=32351&id=122346.
3. Перелигін Б.В. Цифрова обробка зображень: конспект лекцій. Одеса : Одеський державний екологічний університет, 2023. 186 с.
4. Грицишин В. О., Майданюк В. П. Ущільнення зображень з використанням перетворення Уолша-Адамара. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17603/14625.
5. Матвійчук О. В., Майданюк В. П. Квантування трансформант двовимірних ортогональних перетворень при ущільненні зображень / Інформаційні технології і автоматизація – 2023 / Матеріали XVI міжнародної науково-практичної конференції. Одеса, 19-20 жовтня 2023 р. - Одеса, Видавництво ОНТУ, 2023 р. – С. 93-96.
6. Квантування (обробка сигналів). Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Квантування_(обробка_сигналів).
7. Дельта-кодування. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Дельта-кодування.
8. Теорія сигналів [Електронний ресурс] : навч. посіб. для студ. спеціальності 153 «Мікро- та наносистемна техніка» / КПІ ім. Ігоря Сікорського ; уклад.: А.О. Попов. – Електронні текстові данні (1 файл: 7399 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 268 с. URL: https://ela.kpi.ua/bitstream/123456789/41437/3/Teoria_sygnaliv.pdf.

9. Компандування. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Компандування.
10. Майданюк В. П. Обробка сигналів. Навчальний посібник. - Вінниця: ВНТУ. [Електронний ресурс]. URL: https://iq.vntu.edu.ua/method/getfile.php?fname=63704.pdf&x=1&card_id=7303&id=63704.
11. Лавер В.О., Левчук О.М. Обробка зображень: навч.-метод. посіб. / В.О. Лавер, О.М. Левчук. – Ужгород : вид-во ПП «АУТДОР - ШАРК», 2021. – 51 с.
12. Котомчак О. Ю. Комп’ютерна обробка зображень та мультимедіа : навч. посіб., метод. розроб. до лаб.робіт./ О. Ю. Котомчак – К. : Редакційно-видавничий центр Державного університету телекомунікацій, 2018. – 124с. : іл.
13. Borko Furht, Esad Akar, Whitney Angelica Andrews. Digital Image Processing: Practical Approach. ISSN 2191-5768 ISSN 2191-5776 (electronic) SpringerBriefs in Computer Science ISBN 978-3-319-96633-5 ISBN 978-3-319- 96634-2 (eBook) https://doi.org/10.1007/978-3-319-96634-2 Library of Congress Control Number: 2018952345 © The Author(s), under exclusive licence to Springer Nature Switzerland AG 2018.
14. Bogdan, H. E. Kohonen neural network for image coding based on iteration transformation theory, Proceedings from SPIE Neural and Stochastic Methods in Image and Signal Processing, Vol. 1766, pp. 425-436.
15. S. Welstead, Self-Organizing Neural Network Domain Classification for Fractal Image Coding, Proc. of the IASTED International Conference on Artificial Intelligence and Soft Computing, July 27-31, 1997, Banff, Canada pp. 248-251.
16. Discrete Fourier Analysis and Wavelets Applications to Signal and Image Processing Second Edition S. Allen Broughton Kurt Bryan. This second edition first published 2018 © 2018 John Wiley & Sons, Inc. – 465 p.
17. Digital image processing / William K. Pratt., 1993. – 790 p.

18. Digital image processing: ISBN 0132345633, Gonzalez, Rafael C; Woods, Richard E. – 2012, 1072 p.
19. Salomon D. A Guide to data compression methods / David Salomon. - Original english language edition published by Springer-Verlag © 2002 All Rights Reserved. - 368 p.
20. Ing. Milan Ptacek. DrSc. Digitalni zpracovani a pfenos obrazove informace Nakladatelstvi dopravy a spoju Praha.
21. Бабак В. П. Обробка сигналів: Підручник / В. П. Бабак, В.С. Хандецький, Е. Шрюфер. – К.: Либідь, 1996. – 392с.
22. Творошенко І. С. Конспект лекцій з дисципліни «Цифрова обробка зображень» для студентів 4 курсу денної форми навчання напряму 6.080101 – Геодезія, картографія та землеустрій / І. С. Творошенко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова. [Електронний ресурс]. URL: https://core.ac.uk/download/pdf/83144138.pdf.
23. Системи збору даних та їх компактного представлення: конспект лекцій для здобувачів освітнього ступеня бакалавра з спеціальності 151 «Автоматизація та комп`ютерно-інтегровані технології» денної форми навчання [Електронний ресурс] / [Упорядники О. В. Нечипоренко, Я. В. Корпань]; М-во освіти і науки України, Черкас. держ. технол. унт. – Черкаси: ЧДТУ, 2018. – 240 с.
24. Денисюк Н. В., Майданюк В. П.Ущільнення зображень на основі перетворень. [Електронний ресурс]. URL: http://conf.vntu.edu.ua/allvntu/2012/initki/txt/denisyuk.pdf
25. Майданюк В.П., Кожем’яко К.В., Арсенюк І.Р. Нейроподібні методи ущільнення зображень. [Електронний ресурс]. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/32213/06 -Maidanyuk. pdf?sequence=1.
26. Цифрова обробка сигналів. Вікіпедія. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Цифрова_обробка_сигналів.

27. Rahman C., Rahman A. A New Approach for Compressing Color Images using Neural Network //CIMCA 2003 Proceedings/ISBN 1740880684: M. Mohammadian (ed). – 2003. – pp. 315-326.
28. Namhol A., Steven H.S., Arozullah M. Image compression with a hierarchcal neural network // IEEE Transactions on Aerospace and Electronic Systems. – 1996.- № 32. – pp. 326 – 337.
29. Основи теорії інформації та кодування. Конспект лекцій: [Електронний ресурс]: навч. посіб. для студ. спеціальності 171 «Електроніка», спеціалізації «Електронні та інформаційні системи і технології телебачення, кінематографії та звукотехніки»/ М.І. Романюк; Ю. Г. Савченко; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 1,86 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. –70 с. URL: ttps://ela.kpi.ua/bitstream/123456789/27880/1/OTIK_konsp._Romaniuk_Savchenko.pdf.
30. MSDN Library for Visual Studio 2019. [Електронний ресурс]. URL: https://msdn.microsoft.com.
31. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

Додаток А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

"20" вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доцент Д. І. Катєльніков

"20" вересня 2023 р.

Виконав:

_____ студент гр. 2ПІ-22м О. В. Матвійчук

"20" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень».

Галузь застосування – ущільнення зображень, навчальний процес.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на БДР та наказ № 247 від «18» вересня 2023 р. ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення коефіцієнту ущільнення зображень за рахунок оптимізації квантування трансформант ДОП.

Основними задачами дослідження є:

- обґрунтування доцільності розробки нового технічного рішення;
- розробка алгоритмів і програм для дослідження методів квантування компонент зображень;
- експериментальні дослідження залежності ступеня ущільнення від методів квантування компонент зображень;
- розрахунок економічних показників розробки.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Майданюк В. П. Методи і засоби комп’ютерних інформаційних технологій. Кодування зображень. Навчальний посібник. - Вінниця: ВНТУ. Режим доступу:https://iq.vntu.edu.ua/method/getfile.php?fname=122346.pdf&x=1&card_id=3 2351&id=122346 (дата звернення 11.01.2023). — Назва з екрана.
2. Кобилін О.А., Творошенко І. С. Методи цифрової обробки зображень: навч. посібник. – Харків: ХНУРЕ, 2021. – 124 с.

3. Перелигін Б.В. Цифрова обробка зображень: конспект лекцій. Одеса : Одеський державний екологічний університет, 2023. 186 с.
4. Discrete Fourier Analysis and Wavelets Applications to Signal and Image Processing Second Edition S. Allen Broughton Kurt Bryan. This second edition first published 2018 © 2018 John Wiley & Sons, Inc. – 465 p.
5. Майданюк В. П. Обробка сигналів. Навчальний посібник. - Вінниця: ВНТУ. Режим доступу: https://iq.vntu.edu.ua/method/getfile.php?fname = 63704.pdf&x=1&card_id=7303&id=63704 (дата звернення 11.01.2023). — Назва з екрана.
6. Грицишин В. О., Майданюк В. П. Ущільнення зображень з використанням перетворення Уолша-Адамара. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view /17603 /14625 (дата звернення 21.10.2023). - Назва з екрана.
7. Матвійчук О. В., Майданюк В. П. Квантування трансформант двовимірних ортогональних перетворень при ущільненні зображень / Інформаційні технології і автоматизація – 2023 / Матеріали XVI міжнародної науково-практичної конференції. Одеса, 19-20 жовтня 2023 р. - Одеса, Видавництво ОНТУ, 2023 р. – С. 93-96.

5. Технічні вимоги

– середовище розробки – Microsoft Visual Studio;
– мова програмування – C++;
– метод ущільнення – перетворення Уолша-Адамара;
– коефіцієнт ущільнення – 3-6;

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до МКР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів дипломного проєкту	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задач	20.09.23 – 02.10.23
2	Розробка архітектури та алгоритмів програмного продукту	03.10.23 – 23.10.23
3	Аналіз і вибір мови програмування та середовища розробки	16.10.23 – 23.10.23
4	Розробка програмного продукту	24.10.23 – 13.11.23
5	Тестування програми	14.11.23 – 21.11.23
6	Розробка економічної частини	17.11.23 –25.11.23
7	Оформлення матеріалів до захисту МКР	22.11.23 – 01.12.23

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Катєльніков Д. І.

Оригінальність	92
Схожість	8

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Матвійчук О. В.

Керівник роботи _____ Катєльніков Д. І.

Додаток В

(довідниковий)

Лістинг програмного коду квантування трансформант ДОП

```
// Модуль прямого ДКП перетворення та цілочислового квантування
private: System::Void button22_Click(System::Object^ sender, System::EventArgs^ e) {

	Bitmap^ image1;
	Bitmap^ image2;
	Bitmap^ image3;
	Color newColor1;
	String^ name;
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
		image1 = gcnew Bitmap(openFileDialog1->FileName, true);

	image2 = gcnew Bitmap("lena512color.tiff", true);
	for (int x = 0; x < 512; x++)
	{
		for (int y = 0; y < 512; y++)
		{

			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);

		}

	}
	if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
		name = saveFileDialog1->FileName;

	string name1 = "";
	for (int i = 0; i < name->Length; i++)
		name1 += (char)name[i];
	name1 = name1 + ".dct";

	//Відкриття файлу для запису в бінарному режимі
	ofstream out_file(name1, ios::binary);
	//Перевірка на помилку відкриття

	if (!out_file) {
		cout << "\nCouldn't open file";
		return;
	}

	Byte r, g, b;
	int  i, j, k, l, n;
	double t;

	float c[8] = { 0.7,1,1,1,1,1,1,1};
	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];

	n = 8;

	for (int k = 0; k < n; k++) {
```

```
                for (int l = 0; l < n; l++)
                {

                kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
                        out_file.write((char*)&kv8[k][l], 1);
                kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
                        out_file.write((char*)&kv8[k][l], 1);
                kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
                        out_file.write((char*)&kv8[k][l], 1);

                }
        }

        for (int x = 0; x <= 63; x++)
        {
                for (int y = 0; y <= 63; y++)
                {

                        //------RRRRRRR-------

                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {

                kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
                                }
                        }

                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                Color pixelColor = image1->GetPixel(8 * x + j, 8 * y + i);
                                        tr[i][j] = pixelColor.R;
                                }
                        }

                        for (j = 0; j < n; j++) {
                                for (i = 0; i < n; i++)
                                {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                                for (int l = 0; l < n; l++)
                                                {
      t = t + (tr[k][l] * cos(((2 * k + 1) * j * Math::PI) / 16) * cos(((2 * l + 1) * i *
Math::PI ) / 16));
                                                }
                                        }

                                        tr1[j][i] = short int (t*c[j]*c[i]*0.25);
                                }
                        }

                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {

                                tr1[k][l] = tr1[k][l] / kv8[k][l];
                                        out_file.write((char*)&tr1[k][l], 2);
                                        tr1[k][l] = tr1[k][l] * kv8[k][l];
                                }
                        }


                        for (j = 0; j < n; j++) {
                                for (i = 0; i < n; i++)
                                {
```

```
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                               for (int l = 0; l < n; l++)
                                               {
      t = t + (c[k]*c[l]*tr1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 * i +
1) * l * Math::PI) / 16));
                                               }
                                        }

                                        tr[j][i] = short int(t * 0.25);

                                 }
                          }

                          //-------GGGGGGGGGG--------

                          for (int k = 0; k < n; k++) {
                                 for (int l = 0; l < n; l++)
                                 {

             kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
                                 }
                          }

                          for (int i = 0; i < n; i++) {
                                 for (int j = 0; j < n; j++)
                                 {
                       Color pixelColor = image1->GetPixel(8 * x + j, 8 * y + i);
                                        tg[i][j] = pixelColor.G;
                                 }
                          }

                          for (j = 0; j < n; j++) {
                                 for (i = 0; i < n; i++)
                                 {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                               for (int l = 0; l < n; l++)
                                               {
      t = t + (tg[k][l] * cos(((2 * k + 1) * j * Math::PI) / 16) * cos(((2 * l + 1) * i *
Math::PI) / 16));
                                               }
                                        }

                                        tg1[j][i] = short int(t * c[j] * c[i] * 0.25);
                                 }
                          }


                          for (k = 0; k < n; k++) {
                                 for (l = 0; l < n; l++) {

                                        tg1[k][l] = tg1[k][l] / kv8[k][l];
                                        out_file.write((char*)&tg1[k][l], 2);
                                        tg1[k][l] = tg1[k][l] * kv8[k][l];
                                 }
                          }

                          for (j = 0; j < n; j++) {
                                 for (i = 0; i < n; i++)
                                 {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                               for (int l = 0; l < n; l++)
                                               {
```

```
            t = t + (c[k] * c[l] * tg1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) *
cos(((2 * i + 1) * l * Math::PI) / 16));
                                        }
                                  }

                                  tg[j][i] = short int(t * 0.25);

                            }
                      }

                      //-----BBBBBBB----------------

                      for (int k = 0; k < n; k++) {
                            for (int l = 0; l < n; l++)
                            {

                kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
                            }
                      }

                      for (int i = 0; i < n; i++) {
                            for (int j = 0; j < n; j++)
                            {
                            Color pixelColor = image1->GetPixel(8 * x + j, 8 * y + i);
                                  tb[i][j] = pixelColor.B;
                            }
                      }

                      for (j = 0; j < n; j++) {
                            for (i = 0; i < n; i++)
                            {
                                  t = 0;
                                  for (int k = 0; k < n; k++) {
                                        for (int l = 0; l < n; l++)
                                        {
      t = t + (tb[k][l] * cos(((2 * k + 1) * j * Math::PI) / 16) * cos(((2 * l + 1) * i *
Math::PI) / 16));
                                        }
                                  }

                                  tb1[j][i] = short int(t * c[j] * c[i] * 0.25);
                            }
                      }


                      for (k = 0; k < n; k++) {
                            for (l = 0; l < n; l++) {

                                  tb1[k][l] = tb1[k][l] / kv8[k][l];
                                  out_file.write((char*)&tb1[k][l], 2);
                                  tb1[k][l] = tb1[k][l] * kv8[k][l];
                            }
                      }


                      for (j = 0; j < n; j++) {
                            for (i = 0; i < n; i++)
                            {
                                  t = 0;
                                  for (int k = 0; k < n; k++) {
                                        for (int l = 0; l < n; l++)
                                        {
      t = t + (c[k] * c[l] * tb1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 *
i + 1) * l * Math::PI) / 16));
                                        }
```

```
                                }

                                tb[j][i] = short int(t * 0.25);

                        }
                }

                //-----Виведення відновленого зображення----
                for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++)
                        {
                                t = tr[i][j];
                                if (t < 0)  t = 0;
                                if (t > 255)  t = 255;
                                r = t;

                                t = tg[i][j];
                                if (t < 0)  t = 0;
                                if (t > 255)  t = 255;
                                g = t;

                                t = tb[i][j];
                                if (t < 0)  t = 0;
                                if (t > 255)  t = 255;
                                b = t;

                                newColor1 = Color::FromArgb(r, g, b);
                                image2->SetPixel(8 * x + j, 8 * y + i, newColor1);

                        }
                }


        }

    }
    pictureBox2->Image = image2;
    out_file.close();

    String^ name3 = "d:/WHCodec/dmc.exe c " + name + ".dct " + name + ".dctc";
    char name2[512];
    i = 0;
    for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
    name2[i] = NULL;

    PROCESS_INFORMATION pi;
    STARTUPINFO si;

    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

//Арифметичне кодування

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
    {
        printf("CreateProcess failed (%d).\n", GetLastError());
        return;
    }

    WaitForSingleObject(pi.hProcess, INFINITE);
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
```

```
//Модуль зворотного ДКП
private: System::Void button23_Click(System::Object^ sender, System::EventArgs^ e) {

	Bitmap^ image2;
	Color newColor1;
	String^ name;
	//Вибір файлу
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		name = openFileDialog1->FileName;

	String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".dct";
	char name2[512];
	int i = 0;
	for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
	name2[i] = NULL;

	PROCESS_INFORMATION pi;
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));
//Арифметичне декодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

	name3 = name + ".dct";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	//Відкриття файлу для читання в бінарному режимі
	ifstream in_file(name1, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}

	image2 = gcnew Bitmap("lena512color.tiff", true);

	for (int x = 0; x < 512; x++)
	{
		for (int y = 0; y < 512; y++)
		{

			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);

		}

	}

	Byte r, g, b;
	int  j, k, l, n;
	double t;
```

```
float c[8] = { (1.0 / sqrt(2)) ,1.0,1.0,1.0,1.0,1.0,1.0,1.0 };
short int tr[8][8];
short int tr1[8][8];
short int tg[8][8];
short int tg1[8][8];
short int tb[8][8];
short int tb1[8][8];

n = 8;

for (int k = 0; k < n; k++) {
	for (int l = 0; l < n; l++)
	{//Ініціалізація коефіцієнтів цілочислового квантування

		in_file.read((char*)&kv8[k][l], 1);
		dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
		in_file.read((char*)&kv8[k][l], 1);
		dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
		in_file.read((char*)&kv8[k][l], 1);
		dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];



	}
}

for (int x = 0; x <= 63; x++)
{
	for (int y = 0; y <= 63; y++)
	{

		//----RRRR-----
		for (int k = 0; k < n; k++) {
			for (int l = 0; l < n; l++)
			{

	kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
			}
		}

		for (k = 0; k < n; k++) {
			for (l = 0; l < n; l++) {
				in_file.read((char*)&tr1[k][l], 2);
				tr1[k][l] = tr1[k][l] * kv8[k][l];
			}
		}

		//----GGGG-----

		for (int k = 0; k < n; k++) {
			for (int l = 0; l < n; l++)
			{

	kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
			}
		}

		for (k = 0; k < n; k++) {
			for (l = 0; l < n; l++) {
				in_file.read((char*)&tg1[k][l], 2);
				tg1[k][l] = tg1[k][l] * kv8[k][l];
			}
		}
		//----BBBB-----
```

```
                        for (int k = 0; k < n; k++) {
                                for (int l = 0; l < n; l++)
                                {
        kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
                                }
                        }

                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {
                                        in_file.read((char*)&tb1[k][l], 2);
                                        tb1[k][l] = tb1[k][l] * kv8[k][l];
                                }
                        }
                        //Відновлення RRRR
                        n = 8;
                        for (j = 0; j < n; j++) {
                                for (i = 0; i < n; i++)
                                {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                                for (int l = 0; l < n; l++)
                                                {
t = t + (c[k] * c[l] * tr1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 * i + 1)
* l * Math::PI) / 16));
                                                }
                                        }

                                        tr[j][i] = short int(t * 0.25);

                                }
                        }
                        //Відновлення GGGG

                        for (j = 0; j < n; j++) {
                                for (i = 0; i < n; i++)
                                {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                                for (int l = 0; l < n; l++)
                                                {
                                                        t = t + (c[k] * c[l] * tg1[k][l] *
cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 * i + 1) * l * Math::PI) / 16));
                                                }
                                        }

                                        tg[j][i] = short int(t * 0.25);

                                }
                        }

                        //Відновлення BBBB

                        for (j = 0; j < n; j++) {
                                for (i = 0; i < n; i++)
                                {
                                        t = 0;
                                        for (int k = 0; k < n; k++) {
                                                for (int l = 0; l < n; l++)
                                                {
        t = t + (c[k] * c[l] * tb1[k][l] * cos(((2 * j + 1) * k * Math::PI) / 16) * cos(((2 *
i + 1) * l * Math::PI) / 16));
                                                }
                                        }

                                        tb[j][i] = short int(t * 0.25);
```

```
                        }
                    }

                    //-----Виведення відновленого зображення----

                    for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++)
                        {
                            t = tr[i][j];
                            if (t < 0)  t = 0;
                            if (t > 255)  t = 255;
                            r = t;

                            t = tg[i][j];
                            if (t < 0)  t = 0;
                            if (t > 255)  t = 255;
                            g = t;

                            t = tb[i][j];
                            if (t < 0)  t = 0;
                            if (t > 255)  t = 255;
                            b = t;

                            newColor1 = Color::FromArgb(r, g, b);
                            image2->SetPixel(8 * x + j, 8 * y + i, newColor1);

                        }
                    }

                }

            }
            pictureBox2->Image = image2;
            in_file.close();

        }

        //Модуль Групування коефіцієнтів ДОП
        private: System::Void button15_Click(System::Object^ sender, System::EventArgs^ e) {
        //Модуль групування коефіцієнтів ДОП
            Bitmap^ image2;
            Color newColor1;
            String^ name;

            if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

                name = openFileDialog1->FileName;

            //Вибір файлу і арифметичне декодування
            String^ name3 = "d:/WHCodec/dmc.exe e " + name + " " + name + ".wh8";
            char name2[512];
            int i = 0;
            for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
            name2[i] = NULL;

            PROCESS_INFORMATION pi;
            STARTUPINFO si;

            ZeroMemory(&si, sizeof(si));
            si.cb = sizeof(si);
            ZeroMemory(&pi, sizeof(pi));

            if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si,
&pi))
```

```
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

	name3 = name + ".wh8";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	name3 = name + ".grp";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

	//Відкриття файлу коефіцієнтів ДОП і файлу для запису їх у згрупованому виді
	ifstream in_file(name1, ios::binary);
	ofstream out_file(name0, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}



	Byte r, g, b;
	int  j, k, l, t, n;

	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];

//Ініціалізація матриці квантування
	n = 8;

	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{

			in_file.read((char*)&kv8[k][l], 1);
			dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);



		}
	}
//Формування блоків коефіцієнтів 8x8
	for (int x = 0; x <= 63; x++)
	{
		for (int y = 0; y <= 63; y++)
```

```
		{

			//----RRRR-----

			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tr[k][l], 2);
				}
			}

			//----GGGG-----


			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tg[k][l], 2);
				}
			}
			//----BBBB-----


			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tb[k][l], 2);
				}
			}

	//-----Групування і візуалізація трансформант зображення----

			for (int i = 0; i < n; i++) {
				for (int j = 0; j < n; j++)
				{
					gwhr[64 * j + x][64 * i + y] = tr[i][j];
					t = tr[i][j]/8;
					r = t;

					gwhg[64 * j + x][64 * i + y] = tg[i][j];
					t = tg[i][j]/8;
					g = t;

					gwhb[64 * j + x][64 * i + y] = tb[i][j];
					t = tb[i][j]/8;
					b = t;

					newColor1 = Color::FromArgb(r, g, b);
					image2->SetPixel(64*j + x, 64*i + y, newColor1);

				}
			}

		}

	}
	pictureBox2->Image = image2;

//Запис у файл згрупованих коефіцієнтів - послідовно F(0,0) (64x64), F(0,1) (64x64)...
	for (int x = 0; x < 8; x++)
	{
		for (int y = 0; y < 8; y++)
		{
			for (k = 0; k < 64; k++) {
				for (l = 0; l < 64; l++) {
					if ((x > 3) & (y > 3)) {} else {
				out_file.write((char*)&gwhr[64*x+k][64*y+l], 2);
				out_file.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
```

```
                                out_file.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
                                }
                        }
                }
        }
        in_file.close();
        out_file.close();

        //Кодування файлу арифметичним кодером
        name3 = "d:/WHCodec/dmc.exe c " + name + ".grp " + name + ".grpc";

        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;


        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
}

//Модуль дегрупування коефіцієнтів ДОП
private: System::Void button16_Click(System::Object^ sender, System::EventArgs^ e) {
        Bitmap^ image2;
        Color newColor1;
        String^ name;
        //Вибір вхідного файлу
        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

        name = openFileDialog1->FileName;
        //Арифметичне декодування
        String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".grp";
        char name2[512];
        int i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;

        PROCESS_INFORMATION pi;
        STARTUPINFO si;

        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
```

```
	name3 = name + ".grp";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	name3 = name + ".dgrp";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

//Відкриття файлу групованих коефіцієнтів і файлу для дегрупованих коефіцієнтів
	ifstream in_file(name1, ios::binary);
	ofstream out_file(name0, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}

	image2 = gcnew Bitmap("lena512color.tiff", true);

	Byte r, g, b;
	int  j, k, l, t, n;

	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];

	n = 8;
//Читати і писати матрицю квантування
	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{

			in_file.read((char*)&kv8[k][l], 1);
			dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);

		}
	}
//Читати файл і створити двовимірні матриці групованих кофіцієнтів
	for (int x = 0; x < 8; x++)
	{
		for (int y = 0; y <8; y++)
		{

			for (k = 0; k < 64; k++) {
				for (l = 0; l < 64; l++)
				{
					if ((x > 3) & (y > 3)) {
						gwhr[64 * x + k][64 * y + l] = 0;
						gwhg[64 * x + k][64 * y + l] = 0;
						gwhb[64 * x + k][64 * y + l] = 0;
					}
					else {
				in_file.read((char*)&gwhr[64 * x + k][64 * y + l], 2);
```

```
                        in_file.read((char*)&gwhg[64 * x + k][64 * y + l], 2);
                        in_file.read((char*)&gwhb[64 * x + k][64 * y + l], 2);
                                }
                        }
                }
        }
}

//-----Дегрупування і виведення трансформант зображення----
for (int x = 0; x <=63; x++)
{
        for (int y = 0; y <= 63; y++)
        {

                n = 8;
                for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++)
                        {
                                tr[i][j]=gwhr[64 * j + x][64 * i + y];
                                t = tr[i][j] / 8;
                                r = t;

                                tg[i][j]=gwhg[64 * j + x][64 * i + y];
                                t = tg[i][j] / 8;
                                g = t;

                                tb[i][j]=gwhb[64 * j + x][64 * i + y];
                                t = tb[i][j] / 8;
                                b = t;

                                newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

                        }
                }
//Запис у файл дегрупованих коефіцієнтів
                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {

                                out_file.write((char*)&tr[k][l], 2);
                        }
                }
                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {
                                out_file.write((char*)&tg[k][l], 2);
                        }
                }
                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {

                                out_file.write((char*)&tb[k][l], 2);
                        }
                }


        }

}
pictureBox2->Image = image2;

in_file.close();
out_file.close();

//Арифметичне кодування дегрупованих коефіцієнтів
name3 = "d:/DOTCodec/dmc.exe c " + name + ".dgrp " + name + ".dgrpc";
```

```
	i = 0;
	for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
	name2[i] = NULL;


	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

}

//Модуль векторного квантування у просторі значень пікселів зображення
private: System::Void button19_Click(System::Object^ sender, System::EventArgs^ e) {


	Bitmap^ image1;
	Bitmap^ image2;
	Bitmap^ image3;
	Color newColor1;
	String^ name;
	//Вибір зображення
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		image1 = gcnew Bitmap(openFileDialog1->FileName, true);

	image2 = gcnew Bitmap("lena512color.tiff", true);

	for (int x = 0; x < 512; x++)
	{
		for (int y = 0; y < 512; y++)
		{



			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);

		}

	}

	if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		name = saveFileDialog1->FileName;

	string name1 = "";
	for (int i = 0; i < name->Length; i++)
		name1 += (char)name[i];
	name1 = name1 + ".vim";

	//Відкриття файлу для запису в бінарному режимі
	ofstream out_file(name1, ios::binary);

	//Перевірка на помилку відкриття
```

```
if (!out_file) {
	cout << "\nCouldn't open file";
	return;
}


Byte r, g, b;
int  i, j, k, l, t, n;
short int tr[8][8];
short int tr1[8][8];
short int tg[8][8];
short int tg1[8][8];
short int tb[8][8];
short int tb1[8][8];
short int w[32][32];		//Карта Кохонена

n = 8;
for (int x = 0; x <512; x++)
{
	for (int y = 0; y <512; y++)
	{//Запис у масиви складових кольору
			Color pixelColor = image1->GetPixel(x, y);
			gwhr[x][y] = pixelColor.R;
			gwhg[x][y] = pixelColor.G;
			gwhb[x][y] = pixelColor.B;
		}
	}

//Ініціалізація коефіцієнтів мережі
for (i = 0; i < 32; i++) {
	for (j = 0; j < 32; j++) {
	}
}

//Навчання мережі 16x16, розмірність вхідного вектора 4

int d, dmin, ci, cj;
float nn;
d = 0;
dmin = 5000000;
nn = 0.9;
while (nn > 0.01)
{
	for (int x = 0; x < 256; x++)
	{
		for (int y = 0; y < 256; y++)
		{

			for (i = 0; i < 16; i++) {
				for (j = 0; j < 16; j++) {
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
					(w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) +
					(w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) +
					(w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
					if (d < dmin) { dmin = d;  ci = i; cj = j; }
				}
			}
w[2 * ci][2 * cj] = w[2 * ci][2 * cj] + short int(nn * (gwhr[2 * x][2 * y] - w[2 * ci][2 * cj]));
```

```
	w[2 * ci + 1][2 * cj] = w[2 * ci + 1][2 * cj] + short int(nn * (gwhr[2 * x + 1][2 * y] - w[2 * ci + 1][2 * cj]));
	w[2 * ci][2 * cj + 1] = w[2 * ci][2 * cj + 1] + short int(nn * (gwhr[2 * x][2 * y + 1] - w[2 * ci][2 * cj + 1]));
	w[2 * ci + 1][2 * cj + 1] = w[2 * ci + 1][2 * cj + 1] + short int(nn * (gwhr[2 * x + 1][2 * y + 1] - w[2 * ci + 1][2 * cj + 1]));
                                dmin = 5000000;
                        }
                }

                nn = nn - 0.01;
                dmin = 5000000;
        }


	//Квантування

	//RRR
        d = 0; dmin = 5000000;

        for (int x = 0; x < 256; x++)
        {
                for (int y = 0; y < 256; y++)
                {

                        for (i = 0; i < 16; i++) {
                                for (j = 0; j < 16; j++) {
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
                                        (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) +
                                        (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) +
                                        (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
                                        if (d < dmin)
                                        {
                                                dmin = d;
                                                ci = i; cj = j;
                                        }

                                }
                        }
                        gwhrvk[x][y] = ci + cj * 16;
                        dmin = 5000000;
                }
        }
	//GGG
        d = 0; dmin = 5000000;

        for (int x = 0; x < 256; x++)
        {
                for (int y = 0; y < 256; y++)
                {

                        for (i = 0; i < 16; i++) {
                                for (j = 0; j < 16; j++) {
d = (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) +
                                        (w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 * y]) +
                                        (w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y + 1]) +
                                        (w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2 * y + 1]);
                                        if (d < dmin)
                                        {
```

```
                                        dmin = d;
                                        ci = i; cj = j;
                                }

                        }
                }
                gwhgvk[x][y] = ci + cj * 16;
                dmin = 5000000;
        }
    }

//BBB
    d = 0; dmin = 5000000;

    for (int x = 0; x < 256; x++)
    {
        for (int y = 0; y < 256; y++)
        {

            for (i = 0; i < 16; i++) {
                for (j = 0; j < 16; j++) {

d = (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) +
                                        (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 *
y]) * (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 * y]) +
                                        (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y +
1]) * (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y + 1]) +
                                        (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2
* y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2 * y + 1]);
                                if (d < dmin)
                                {
                                        dmin = d;
                                        ci = i; cj = j;
                                }

                        }
                }
                gwhbvk[x][y] = ci + cj * 16;
                dmin = 5000000;
        }
    }

//Векторне деквантування


    for (int x = 0; x < 256; x++)
    {
        for (int y = 0; y < 256; y++)
        {
//RRR
                ci = gwhrvk[x - 0][y - 0] & 15;
                cj = (gwhrvk[x - 0][y - 0] / 16) & 15;
                gwhr[2 * x][2 * y] = w[2 * ci][2 * cj];
                gwhr[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                gwhr[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                gwhr[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
//GGG
                ci = gwhgvk[x - 0][y - 0] & 15;
                cj = (gwhgvk[x - 0][y - 0] / 16) & 15;
                gwhg[2 * x][2 * y] = w[2 * ci][2 * cj];
                gwhg[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                gwhg[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                gwhg[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
//BBB
                ci = gwhbvk[x - 0][y - 0] & 15;
```

```
                        cj = (gwhbvk[x - 0][y - 0] / 16) & 15;
                        gwhb[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhb[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhb[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhb[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

                }
        }

//-----Виведення відновленого зображення----
        for (int x = 0; x < 512; x++)
        {
                for (int y = 0; y<512; y++)
                {

                                        t = gwhr[x][y];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        r = t;

                                        t = gwhg[x][y];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        g = t;

                                        t = gwhb[x][y];
                                        if (t < 0)  t = 0;
                                        if (t > 255)  t = 255;
                                        b = t;

                                        newColor1 = Color::FromArgb(r, g, b);
                                        image2->SetPixel(x,  y, newColor1);

                                }
                        }


        pictureBox2->Image = image2;
//Запис у файл карти Кохонена
        for (i = 0; i < 32; i++) {
                for (j = 0; j < 32; j++) {

                        out_file.write((char*)&w[i][j], 2);
                }
        }

//Запис у фай квантованих компонент зображення
        for (int x = 0; x < 256; x++) {
                for (int y = 0; y < 256; y++)
                {

                        out_file.write((char*)&gwhrvk[x][y], 1);
                        out_file.write((char*)&gwhgvk[x][y], 1);
                        out_file.write((char*)&gwhbvk[x][y], 1);

                }
        }
        out_file.close();

//Ущільнення файлу арифметичним кодером
        String^ name3 = "d:/DOTCodec/dmc.exe c " + name + ".vim " + name + ".vimc";
        char name2[512];
        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;
```

```
	PROCESS_INFORMATION pi;
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);
}
//Модуль векторного деквантування в просторі пікселів
private: System::Void button20_Click(System::Object^ sender, System::EventArgs^ e) {

	Bitmap^ image2;
	Color newColor1;
	String^ name;
//Вибір файлу і арифметичне декодування
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		name = openFileDialog1->FileName;

	String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".vim";
	char name2[512];
	int i = 0;
	for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
	name2[i] = NULL;

	PROCESS_INFORMATION pi;
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);
//Відкрити квантований файл для читання
	name3 = name + ".vim";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	//Відкриття файлу для читання в бінарному режимі
	ifstream in_file(name1, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}
```

```
	image2 = gcnew Bitmap("lena512color.tiff", true);

	for (int x = 0; x < 512; x++)
	{
		for (int y = 0; y < 512; y++)
		{



			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);

		}

	}

	Byte r, g, b;
	int  j, k, l, t, n;


	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];
	short int w[32][32];              //Карта Кохонена
	n = 8;
//Ініціалізація карти Кохонена із файлу
	for (i = 0; i < 32; i++) {
		for (j = 0; j < 32; j++) {

			in_file.read((char*)&w[i][j], 2);
		}
	}
	for (int x = 0; x < 256; x++) {
		for (int y = 0; y < 256; y++)
		{

			in_file.read((char*)&gwhrvk[x][y], 1);
			in_file.read((char*)&gwhgvk[x][y], 1);
			in_file.read((char*)&gwhbvk[x][y], 1);

		}
	}


//Векторне деквантування
	int ci, cj;

	for (int x = 0; x < 256; x++)
	{
		for (int y = 0; y < 256; y++)
		{
			ci = gwhrvk[x - 0][y - 0] & 15;
			cj = (gwhrvk[x - 0][y - 0] / 16) & 15;
			gwhr[2 * x][2 * y] = w[2 * ci][2 * cj];
			gwhr[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
			gwhr[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
			gwhr[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

			ci = gwhgvk[x - 0][y - 0] & 15;
			cj = (gwhgvk[x - 0][y - 0] / 16) & 15;
```

```
                        gwhg[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhg[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhg[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhg[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

                        ci = gwhbvk[x - 0][y - 0] & 15;
                        cj = (gwhbvk[x - 0][y - 0] / 16) & 15;
                        gwhb[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhb[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhb[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhb[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

                }
        }



        //-----Виведення відновленого зображення----
        for (int x = 0; x < 512; x++)
        {
                for (int y = 0; y < 512; y++)
                {

                        t = gwhr[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        r = t;

                        t = gwhg[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        g = t;

                        t = gwhb[x][y];
                        if (t < 0)  t = 0;
                        if (t > 255)  t = 255;
                        b = t;

                        newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(x, y, newColor1);

                }
        }

        pictureBox2->Image = image2;
        in_file.close();
}
```

//Навчання мережі 64x64 на трансформантах ДОП з записом результатів у файл map.kh

```
private: System::Void button21_Click(System::Object^ sender, System::EventArgs^ e) {

        Bitmap^ image2;
        Color newColor1;
        String^ name;
        //Вибір файлу з ущільненими коефіцієнтами ДОП
        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

                name = openFileDialog1->FileName;

        String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".wh8";
        char name2[512];
        int i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;
```

```
	PROCESS_INFORMATION pi;
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));
	//Арифметичне декодування файлу з коефіцієнтами ДОП
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

	name3 = name + ".wh8";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	name3 = name + ".gwh8";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

	//Відкриття файлів для читання та запису у бінарному режимі
	ifstream in_file(name1, ios::binary);
	ofstream out_file(name0, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}
	image2 = gcnew Bitmap("lena512color.tiff", true);

	for (int x = 0; x < 512; x++)
	{
		for (int y = 0; y < 512; y++)
		{

			Color newColor = Color::FromArgb(0, 0, 0);
			image2->SetPixel(x, y, newColor);
		}

	}

	Byte r, g, b;
	int  j, k, l, t, n;
	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];
	short int w[128][128];                    //Карта Кохонена
	float lsp[16][16];

	n = 8;
	//Ініціалізація матриці квантування
	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{
```

```
                in_file.read((char*)&kv8[k][l], 1);
                dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
                out_file.write((char*)&kv8[k][l], 1);
                in_file.read((char*)&kv8[k][l], 1);
                dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
                out_file.write((char*)&kv8[k][l], 1);
                in_file.read((char*)&kv8[k][l], 1);
                dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
                out_file.write((char*)&kv8[k][l], 1);
        }
}
//Читати з файлу трансформанти (коефіцієнти) ДОП блоками 8x8
for (int x = 0; x <= 63; x++)
{
        for (int y = 0; y <= 63; y++)
        {

                //----RRRR-----

                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {
                                in_file.read((char*)&tr[k][l], 2);
                        }
                }

                //----GGGG-----

                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {
                                in_file.read((char*)&tg[k][l], 2);
                        }
                }
                //----BBBB-----

                for (k = 0; k < n; k++) {
                        for (l = 0; l < n; l++) {
                                in_file.read((char*)&tb[k][l], 2);
                        }
                }

                //-----Групування і виведення трансформант зображення----

                for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++)
                        {
                                gwhr[64 * j + x][64 * i + y] = tr[i][j];
                                t = tr[i][j] / 8;
                                r = t;

                                gwhg[64 * j + x][64 * i + y] = tg[i][j];
                                t = tg[i][j] / 8;
                                g = t;

                                gwhb[64 * j + x][64 * i + y] = tb[i][j];
                                t = tb[i][j] / 8;
                                b = t;

                                newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

                        }
                }

        }
```

```
	}
	pictureBox2->Image = image2;
//запис у вихідний файл згрупованих коефіцієнтів ДОП (для порівняння з групуванням)
	for (int x = 0; x < 8; x++)
	{
		for (int y = 0; y < 8; y++)
		{
			for (k = 0; k < 64; k++) {
				for (l = 0; l < 64; l++) {
		out_file.write((char*)&gwhr[64 * x + k][64 * y + l], 2);
		gwhr[64 * x + k][64 * y + l] = gwhr[64 * x + k][64 * y + l];
		out_file.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
		gwhg[64 * x + k][64 * y + l] = gwhg[64 * x + k][64 * y + l];
		out_file.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
		gwhb[64 * x + k][64 * y + l] = gwhb[64 * x + k][64 * y + l];

				}
			}
		}
	}
	in_file.close();
	out_file.close();
	//Арифметичне кодування
	name3 = "d:/DOTCodec/dmc.exe c " + name + ".gwh8 " + name + ".gdm8";

	i = 0;
	for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
	name2[i] = NULL;


	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);


	name3 = "d:/DOTCodec/map.kh";
	name0 = "";
	for (int i = 0; i < name3->Length; i++)
	name0 += (char)name3[i];

	//Відкриття файлу map.kh для запису в бінарному режимі
	ofstream out_fvkv(name0, ios::binary);

	//Ініціалізація коефіцієнтів мережі

 for (i = 0; i < 128; i++) {
     for (j = 0; j < 128; j++) {

			w[i][j] = gwhr[(1*i)][(1*j)];
	}
}

	//Навчання мережі 64x64 при розмірності вхідного вектора 4 (2x2)

	int d, dmin, ci, cj;
```

```
		float nn;
		d = 0;
		dmin = 2000000;
		nn = 1;
				while (nn > 0.01)
		{
			for (int y = 0; y < 256; y++)
			{
				for (int x = 0; x < 256; x++)
				{

					for (j = 0; j < 64; j++) {
						for (i = 0; i < 64; i++) {
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
(w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) + (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) + (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
		if (d < dmin) { dmin = d;  ci = i; cj = j; }
						}
					}
w[2 * ci][2 * cj] = w[2 * ci][2 * cj] + short int(nn * (gwhr[2 * x][2 * y] - w[2 * ci][2 * cj]));
	w[2 * ci + 1][2 * cj] = w[2 * ci + 1][2 * cj] + short int(nn * (gwhr[2 * x + 1][2 * y] - w[2 * ci + 1][2 * cj]));
	w[2 * ci][2 * cj + 1] = w[2 * ci][2 * cj + 1] + short int(nn * (gwhr[2 * x][2 * y + 1] - w[2 * ci][2 * cj + 1]));
	w[2 * ci + 1][2 * cj + 1] = w[2 * ci + 1][2 * cj + 1] + short int(nn * (gwhr[2 * x + 1][2 * y + 1] - w[2 * ci + 1][2 * cj + 1]));
					dmin = 2000000;
				}
			}

			nn = nn - 0.01;
			dmin = 2000000;
		}

		//Запис у файл результатів навчання
	for (j = 0; j < 128; j++) {
			for (i = 0; i < 128; i++) {
				out_fvkv.write((char*)&w[i][j], 2);
			}
		}

		out_fvkv.close();
	}

	//Модуль векторного квантування коефіцієнтів ДОП 64x64

	private: System::Void button17_Click(System::Object^ sender, System::EventArgs^ e) {
		Bitmap^ image2;
		Color newColor1;
		String^ name;
		//Вибір файлу з коефіцієнтами ДОП
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

			name = openFileDialog1->FileName;

		String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".wh8";
		char name2[512];
		int i = 0;
		for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
		name2[i] = NULL;

		PROCESS_INFORMATION pi;
```

```
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));
	//Арифметичне декодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}

	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

	name3 = name + ".wh8";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	name3 = name + ".gwh8";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

	//Відкриття файлу для читання в бінарному режимі з коефіцієнтами ДОП
	ifstream in_file(name1, ios::binary);
	//Відкриття файлу для запису квантованих коефіцієнтів в бінарному режимі
ofstream out_file(name0, ios::binary);

	if (!in_file) {
		cout << "\nCouldn't open file";
		return;
	}


	Byte r, g, b;
	int  j, k, l, t, n;
	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];
	short int w[128][128];                    //Карта Кохонена
	float lsp[16][16];

	n = 8;

	//Читати коефіцієнти цілочислового квантування
for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{

			in_file.read((char*)&kv8[k][l], 1);
			dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
			in_file.read((char*)&kv8[k][l], 1);
			dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
```

```
		}
	}
	//Читати коефіцієнти ДОП блоками 8x8
	for (int x = 0; x <= 63; x++)
	{
		for (int y = 0; y <= 63; y++)
		{

			//----RRRR-----

			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tr[k][l], 2);
				}
			}

			//----GGGG-----


			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tg[k][l], 2);
				}
			}
			//----BBBB-----


			for (k = 0; k < n; k++) {
				for (l = 0; l < n; l++) {
					in_file.read((char*)&tb[k][l], 2);
				}
			}

			//-----Групування і виведення трансформант зображення----

			for (int i = 0; i < n; i++) {
				for (int j = 0; j < n; j++)
				{
					gwhr[64 * j + x][64 * i + y] = tr[i][j];
					t = tr[i][j] / 8;
					r = t;

					gwhg[64 * j + x][64 * i + y] = tg[i][j];
					t = tg[i][j] / 8;
					g = t;

					gwhb[64 * j + x][64 * i + y] = tb[i][j];
					t = tb[i][j] / 8;
					b = t;

					newColor1 = Color::FromArgb(r, g, b);
				image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

				}
			}

		}

	}
	pictureBox2->Image = image2;
//Запис згрупованих коефіцієнтів у файл іїх арифметичне кодування (для контролю)
	for (int x = 0; x < 8; x++)
	{
```

```
			for (int y = 0; y < 8; y++)
			{
				for (k = 0; k < 64; k++) {
					for (l = 0; l < 64; l++) {

			out_file.write((char*)&gwhr[64 * x + k][64 * y + l], 2);
			gwhr[64 * x + k][64 * y + l] = gwhr[64 * x + k][64 * y + l];
			out_file.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
			gwhg[64 * x + k][64 * y + l] = gwhg[64 * x + k][64 * y + l];
			out_file.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
			gwhb[64 * x + k][64 * y + l] = gwhb[64 * x + k][64 * y + l];

					}
				}
			}
		}
		in_file.close();
		out_file.close();

		name3 = "d:/DOTCodec/dmc.exe c " + name + ".gwh8 " + name + ".gdm8";

		i = 0;
		for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
		name2[i] = NULL;


		ZeroMemory(&si, sizeof(si));
		si.cb = sizeof(si);
		ZeroMemory(&pi, sizeof(pi));

	if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
		{
			printf("CreateProcess failed (%d).\n", GetLastError());
			return;
		}

		WaitForSingleObject(pi.hProcess, INFINITE);
		CloseHandle(pi.hProcess);
		CloseHandle(pi.hThread);

	//Відкриття файлу мережі 64x64
		name3 = "d:/DOTCodec/map.kh";
		string name20 = "";
		for (int i = 0; i < name3->Length; i++)
			name20 += (char)name3[i];


		//Відкриття файлу для читання в бінарному режимі
		ifstream in_fkv(name20, ios::binary);

		//Ініціалізація коефіцієнтів мережі 64x64

		for (j = 0; j < 128; j++) {
			for (i = 0; i < 128; i++) {
				in_fkv.read((char*)&w[i][j], 2);
			}
		}

		in_fkv.close();

	    name3 = name + ".vkk";
	    name0 = "";
	   for (int i = 0; i < name3->Length; i++)
			name0 += (char)name3[i];
```

```
	//Відкриття файлу для запису векторно квантованих коефіцієнтів ДОП в бінарному режимі
		ofstream out_fvkv(name0, ios::binary);

		int d, dmin, ci, cj;
		float nn;
		d = 0;
		dmin = 2000000;
		nn = 1;

		//Квантування
		//RRR
	d = 0; dmin = 2000000;

			for (int y = 0; y < 256; y++)
			{
				for (int x = 0; x < 256; x++)
				{

					for (j = 0; j < 64; j++) {
						for (i = 0; i < 64; i++) {
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
(w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) +
(w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) +
(w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
if (d < dmin)
{
dmin = d;
ci = i; cj = j;
}

						}
					}
					gwhrvk[x][y] = ci + cj*64;

					dmin = 2000000;
				}
			}

		//GGG
	d = 0; dmin = 2000000;

			for (int y = 0; y < 256; y++)
			{
				for (int x = 0; x < 256; x++)
				{

					for (j = 0; j < 64; j++) {
						for (i = 0; i < 64; i++) {
	d = (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) +
(w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 * y]) +
(w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y + 1]) +
(w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2 * y + 1]);
if (d < dmin)
{
dmin = d;
ci = i; cj = j;
}

						}
```

```
                                }

                                gwhgvk[x][y] = ci + cj * 64;

                                dmin = 2000000;
                        }
                }

        //BBB
                d = 0; dmin = 2000000;

                for (int y = 0; y < 256; y++)
                {
                        for (int x = 0; x < 256; x++)
                        {

                                for (j = 0; j < 64; j++) {
                                        for (i = 0; i < 64; i++) {
d = (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) +
                                                (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 * y]) +
                                                (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y + 1]) +
                                                (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2 * y + 1]);
                                                if (d < dmin)
                                                {
                                                        dmin = d;
                                                        ci = i; cj = j;
                                                }

                                        }
                                }

                                gwhbvk[x][y] = ci + cj * 64;
                                dmin = 2000000;
                        }
                }

        //Запис у файл коефіцієнтів цілочислового квантування
        n = 8;

                for (int k = 0; k < n; k++) {
                        for (int l = 0; l < n; l++)
                        {
        kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
        out_fvkv.write((char*)&kv8[k][l], 1);
        kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
        out_fvkv.write((char*)&kv8[k][l], 1);
                kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
                                out_fvkv.write((char*)&kv8[k][l], 1);

                        }
                }

                //Запис у файл векторно квантованих коефіцієнтів ДОП
                for (int y = 0; y < 256; y++)
                {
                        for (int x = 0; x < 256; x++)
                        {
                                        out_fvkv.write((char*)&gwhrvk[x][y], 2);
                                           out_fvkv.write((char*)&gwhgvk[x][y], 2);
                                           out_fvkv.write((char*)&gwhbvk[x][y], 2);
                                }
                        }
```

```
		out_fvkv.close();

		name3 = "d:/DOTCodec/dmc.exe c " + name + ".vkk " + name + ".vkkc";

		i = 0;
		for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
		name2[i] = NULL;


		ZeroMemory(&si, sizeof(si));
		si.cb = sizeof(si);
		ZeroMemory(&pi, sizeof(pi));
	//Арифметичне кодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
		{
			printf("CreateProcess failed (%d).\n", GetLastError());
			return;
		}

		WaitForSingleObject(pi.hProcess, INFINITE);
		CloseHandle(pi.hProcess);
		CloseHandle(pi.hThread);
}

//Модуль векторного деквантування коефіцієнтів ДОП 64x64
private: System::Void button18_Click(System::Object^ sender, System::EventArgs^ e) {
	Bitmap^ image2;
	Color newColor1;
	String^ name;

	Byte r, g, b;
	int  j, k, l, t, n;
	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];
	short int w[128][128];                    //Карта Кохонена
	float lsp[16][16];
	//Вибір файлу
	if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

		name = openFileDialog1->FileName;

	String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".dvkk";
	char name2[512];
	int i = 0;
	for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
	name2[i] = NULL;

	PROCESS_INFORMATION pi;
	STARTUPINFO si;

	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));
	//Арифметичне декодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}
```

```
	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);

	//Ініціалізація карти Кохонена
	name3 = "d:/DOTCodec/map.kh";
	string name00 = "";
	for (int i = 0; i < name3->Length; i++) name00 += (char)name3[i];
	ifstream in_map(name00, ios::binary);

	for (int j = 0; j < 128; j++) {
		for (int i = 0; i < 128; i++) {
			in_map.read((char*)&w[i][j], 2);
		}
	}
	in_map.close();

//Відкриття файлу векторно-квантованих коефіцієнтів та файлу для запису деквантованих

	name3 = name + ".dvkk";
	string name1 = "";
	for (int i = 0; i < name3->Length; i++)
		name1 += (char)name3[i];

	name3 = name + ".dkvkk";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

	ifstream in_fkv(name1, ios::binary);
	ofstream out_fdkv(name0, ios::binary);


	if (!in_fkv) {
		cout << "\nCouldn't open file";
		return;
	}


//Ініціалізація матриці цілочислового квантування
	n = 8;
	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{

			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_fdkv.write((char*)&kv8[k][l], 1);
			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_fdkv.write((char*)&kv8[k][l], 1);
			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_fdkv.write((char*)&kv8[k][l], 1);
		}
	}


//Запис квантованих коефіцієнтів в масиви
	for (int y = 0; y < 256; y++)
	{
		for (int x = 0; x < 256; x++)
		{
			in_fkv.read ((char*)&gwhrvk[x][y], 2);
			in_fkv.read ((char*)&gwhgvk[x][y], 2);
```

```
                        in_fkv.read ((char*)&gwhbvk[x][y], 2);
                }
        }

//Векторне деквантування

        int ci, cj;

        for (int y = 0; y < 256; y++)
        {
                for (int x = 0; x < 256; x++) {
                        //RRR
ci = gwhrvk[x - 0][y - 0] & 63;
                        cj = (gwhrvk[x-0][y-0] / 64) & 63;
                        gwhr[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhr[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhr[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhr[2 * x + 1][2 * y + 1]= w[2 * ci + 1][2 * cj + 1];
                        //GGG
                        ci = gwhgvk[x - 0][y - 0] & 63;
                        cj = (gwhgvk[x - 0][y - 0] / 64) & 63;
                        gwhg[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhg[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhg[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhg[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
                        //BBB
                        ci = gwhbvk[x - 0][y - 0] & 63;
                        cj = (gwhbvk[x - 0][y - 0] / 64) & 63;
                        gwhb[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhb[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhb[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhb[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

                }
        }


        for (int x = 0; x <= 63; x++)
        {
                for (int y = 0; y <= 63; y++)
                {

                        //-----Дегрупування і виведення трансформант зображення----
                        n = 8;
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                        tr[i][j] = gwhr[64 * j + x][64 * i + y];
                                        t = tr[i][j] / 8;
                                        r = t;

                                        tg[i][j] = gwhg[64 * j + x][64 * i + y];
                                        t = tg[i][j] / 8;
                                        g = t;

                                        tb[i][j] = gwhb[64 * j + x][64 * i + y];
                                        t = tb[i][j] / 8;
                                        b = t;

                                        newColor1 = Color::FromArgb(r, g, b);
                                image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

                                }
                        }
```

```
                        for (k = 0; k < n; k++) {
                               for (l = 0; l < n; l++) {

                                      out_fdkv.write((char*)&tr[k][l], 2);
                               }
                        }
                        for (k = 0; k < n; k++) {
                               for (l = 0; l < n; l++) {
                                      out_fdkv.write((char*)&tg[k][l], 2);
                               }
                        }
                        for (k = 0; k < n; k++) {
                               for (l = 0; l < n; l++) {

                                      out_fdkv.write((char*)&tb[k][l], 2);
                               }
                        }


                }

        }

        pictureBox2->Image = image2;


        in_fkv.close();
        out_fdkv.close();

        name3 = "d:/DOTCodec/dmc.exe c " + name + ".dkvkk " + name + ".rdm8";

        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;


        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));
        //Арифметичне кодування деквантованих коефіцієнтів
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
}

//Модуль комбінованого векторного квантування 4x4
private: System::Void button24_Click(System::Object^ sender, System::EventArgs^ e) {
        Bitmap^ image2;
        Color newColor1;
        String^ name;
        //Вибір файлу
        if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

                name = openFileDialog1->FileName;

        String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".wh8";
        char name2[512];
        int i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
```

```
		name2[i] = NULL;

		PROCESS_INFORMATION pi;
		STARTUPINFO si;

		ZeroMemory(&si, sizeof(si));
		si.cb = sizeof(si);
		ZeroMemory(&pi, sizeof(pi));
		//Аривметичне декодування
	if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
		{
			printf("CreateProcess failed (%d).\n", GetLastError());
			return;
		}

		WaitForSingleObject(pi.hProcess, INFINITE);
		CloseHandle(pi.hProcess);
		CloseHandle(pi.hThread);

		name3 = name + ".wh8";
		string name1 = "";
		for (int i = 0; i < name3->Length; i++)
			name1 += (char)name3[i];

		name3 = name + ".gwh8";
		string name0 = "";
		for (int i = 0; i < name3->Length; i++)
			name0 += (char)name3[i];

//Відкриття файлу для читання коефіцієнтів ДОП в бінарному режимі та запису векторно
//квантованих
		ifstream in_file(name1, ios::binary);
		ofstream out_file(name0, ios::binary);

		if (!in_file) {
			cout << "\nCouldn't open file";
			return;
		}

		image2 = gcnew Bitmap("lena512color.tiff", true);

		for (int x = 0; x < 512; x++)
		{
			for (int y = 0; y < 512; y++)
			{

			Color newColor = Color::FromArgb(0, 0, 0);
				image2->SetPixel(x, y, newColor);

			}

		}

		Byte r, g, b;
		int  j, k, l, t, n;

		short int tr[8][8];
		short int tr1[8][8];
		short int tg[8][8];
		short int tg1[8][8];
		short int tb[8][8];
		short int tb1[8][8];
		short int w[32][32];              //Карта Кохонена
		float lsp[16][16];
```

```
n = 8;
//Ініціалізація матриць квантування
for (int k = 0; k < n; k++) {
	for (int l = 0; l < n; l++)
	{

		in_file.read((char*)&kv8[k][l], 1);
		dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
		out_file.write((char*)&kv8[k][l], 1);
		in_file.read((char*)&kv8[k][l], 1);
		dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
		out_file.write((char*)&kv8[k][l], 1);
		in_file.read((char*)&kv8[k][l], 1);
		dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
		out_file.write((char*)&kv8[k][l], 1);



	}
}
//Читати поблочно 8x8 коефіцієнти ДОП
for (int x = 0; x <= 63; x++)
{
	for (int y = 0; y <= 63; y++)
	{

		//----RRRR-----

		for (k = 0; k < n; k++) {
			for (l = 0; l < n; l++) {
				in_file.read((char*)&tr[k][l], 2);
                                                }
		}

		//----GGGG-----


		for (k = 0; k < n; k++) {
			for (l = 0; l < n; l++) {
				in_file.read((char*)&tg[k][l], 2);
                                                }
		}
		//----BBBB-----


		for (k = 0; k < n; k++) {
			for (l = 0; l < n; l++) {
				in_file.read((char*)&tb[k][l], 2);
                                                }
		}

		//-----Групування і виведення трансформант зображення----

		for (int i = 0; i < n; i++) {
			for (int j = 0; j < n; j++)
			{
				//RRR
				gwhr[64 * j + x][64 * i + y] = tr[i][j];
				t = tr[i][j] / 8;
				//if (t < 0)  t = 0;
				//if (t > 255)  t = 255;
				r = t;
				//GGG
				gwhg[64 * j + x][64 * i + y] = tg[i][j];
				t = tg[i][j] / 8;
```

```
                                //if (t < 0)  t = 0;
                                //if (t > 255)  t = 255;
                                g = t;
                                //BBB
                                gwhb[64 * j + x][64 * i + y] = tb[i][j];
                                t = tb[i][j] / 8;
                                //if (t < 0)  t = 0;
                                //if (t > 255)  t = 255;
                                b = t;

                                newColor1 = Color::FromArgb(r, g, b);
                        image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

                                }
                        }

                }

        }
        pictureBox2->Image = image2;
        //Запис у файл групованих коефіцієнтів ДОП для контролю
        for (int x = 0; x < 8; x++)
        {
                for (int y = 0; y < 8; y++)
                {
                        for (k = 0; k < 64; k++) {
                                for (l = 0; l < 64; l++) {

                                out_file.write((char*)&gwhr[64 * x + k][64 * y + l], 2);
                                out_file.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
                                out_file.write((char*)&gwhb[64 * x + k][64 * y + l], 2);

                                }
                        }
                }
        }
        in_file.close();
        out_file.close();

        name3 = "d:/DOTCodec/dmc.exe c " + name + ".gwh8 " + name + ".gdm8";

        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;


        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));
        //Арифметичне кодування групованих коефіцієнтів ДОП
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);

        name3 = name + ".cm";
        name0 = "";
        for (int i = 0; i < name3->Length; i++)
                name0 += (char)name3[i];
```

```
		//Відкриття файлу для запису в бінарному режимі
		ofstream out_fvkv(name0, ios::binary);

		//Ініціалізація коефіцієнтів мережі

		for (i = 0; i < 32; i++) {
			for (j = 0; j < 32; j++) {

				w[i][j] = gwhr[(8 * i)][(8 * j)];
			}
		}

		//Навчання мережі

		int d, dmin, ci, cj;
		float nn;
		d = 0;
		dmin = 2000000;
		nn = 1;
		while (nn > 0.01)
		{
			for (int y = 64; y < 256; y++)
			{
				for (int x = 64; x < 256; x++)
				{

					for (j = 0; j < 16; j++) {
						for (i = 0; i < 16; i++) {
d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
(w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) +
(w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) +
(w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
if (d < dmin) { dmin = d;  ci = i; cj = j; }
						}
					}
w[2 * ci][2 * cj] = w[2 * ci][2 * cj] + short int(nn * (gwhr[2 * x][2 * y] - w[2 * ci][2 * cj]));
	w[2 * ci + 1][2 * cj] = w[2 * ci + 1][2 * cj] + short int(nn * (gwhr[2 * x + 1][2 * y] - w[2 * ci + 1][2 * cj]));
	w[2 * ci][2 * cj + 1] = w[2 * ci][2 * cj + 1] + short int(nn * (gwhr[2 * x][2 * y + 1] - w[2 * ci][2 * cj + 1]));
	w[2 * ci + 1][2 * cj + 1] = w[2 * ci + 1][2 * cj + 1] + short int(nn * (gwhr[2 * x + 1][2 * y + 1] - w[2 * ci + 1][2 * cj + 1]));
					dmin = 2000000;
				}
			}

			nn = nn - 0.01;
			dmin = 2000000;
		}


	//Квантування

	//RRR
	d = 0; dmin = 2000000;

		for (int y = 64; y < 256; y++)
		{
			for (int x = 64; x < 256; x++)
			{
```

```
                        for (j = 0; j < 16; j++) {
                                for (i = 0; i < 16; i++) {

d = (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhr[2 * x][2 * y]) +
                                                (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 *
y]) * (w[2 * i + 1][2 * j] - gwhr[2 * x + 1][2 * y]) +
                                                (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y +
1]) * (w[2 * i][2 * j + 1] - gwhr[2 * x][2 * y + 1]) +
                                                (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2
* y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhr[2 * x + 1][2 * y + 1]);
                                        if (d < dmin)
                                        {
                                                dmin = d;
                                                ci = i; cj = j;
                                        }

                                }
                        }
                        gwhrvk[x - 64][y - 64] = ci + cj * 16;

                        dmin = 2000000;
                }
        }

        //GGG
                d = 0; dmin = 2000000;

                for (int y = 64; y < 256; y++)
                {
                        for (int x = 64; x < 256; x++)
                        {

        for (j = 0; j < 16; j++) {
                                for (i = 0; i < 16; i++) {

d = (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhg[2 * x][2 * y]) +
                                                (w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 *
y]) * (w[2 * i + 1][2 * j] - gwhg[2 * x + 1][2 * y]) +
                                                (w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y +
1]) * (w[2 * i][2 * j + 1] - gwhg[2 * x][2 * y + 1]) +
                                                (w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2
* y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhg[2 * x + 1][2 * y + 1]);
                                        if (d < dmin)
                                        {
                                                dmin = d;
                                                ci = i; cj = j;
                                        }

                                }
                        }

                        gwhgvk[x - 64][y - 64] = ci + cj * 16;

                        dmin = 2000000;
                }
        }

        //BBB
                d = 0; dmin = 2000000;

                for (int y = 64; y < 256; y++)
                {
                        for (int x = 64; x < 256; x++)
                        {
```

```
                        for (j = 0; j < 16; j++) {
                                for (i = 0; i < 16; i++) {
d = (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) * (w[2 * i][2 * j] - gwhb[2 * x][2 * y]) +
                                        (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 *
y]) * (w[2 * i + 1][2 * j] - gwhb[2 * x + 1][2 * y]) +
                                        (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y +
1]) * (w[2 * i][2 * j + 1] - gwhb[2 * x][2 * y + 1]) +
                                        (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2
* y + 1]) * (w[2 * i + 1][2 * j + 1] - gwhb[2 * x + 1][2 * y + 1]);
                                    if (d < dmin)
                                    {
                                        dmin = d;
                                        ci = i; cj = j;
                                    }

                                }
                        }

                        gwhbvk[x - 64][y - 64] = ci + cj * 16;
                        dmin = 2000000;
                }
        }

    //Запис у вихідний файл
        n = 8;

        for (int k = 0; k < n; k++) {
            for (int l = 0; l < n; l++)
            {     //Коефіцієнтів цілочислового квантування

            kv8[k][l] = Convert::ToInt32(dataGridView2->Rows[k]->Cells[l]->Value);
                out_fvkv.write((char*)&kv8[k][l], 1);
            kv8[k][l] = Convert::ToInt32(dataGridView5->Rows[k]->Cells[l]->Value);
                out_fvkv.write((char*)&kv8[k][l], 1);
            kv8[k][l] = Convert::ToInt32(dataGridView6->Rows[k]->Cells[l]->Value);
                out_fvkv.write((char*)&kv8[k][l], 1);

            }
        }
        //Коефіцієнтів мережі
        for (j = 0; j < 32; j++) {
            for (i = 0; i < 32; i++) {
                out_fvkv.write((char*)&w[i][j], 2);
            }
        }
        //Цілочисельно квантованих коефіцієнтів ДОП
        for (int x = 0; x < 8; x++)
        {
            for (int y = 0; y < 2; y++)
            {
                for (k = 0; k < 64; k++) {
                    for (l = 0; l < 64; l++) {

                    out_fvkv.write((char*)&gwhr[64 * x + k][64 * y + l], 2);
                    out_fvkv.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
                    out_fvkv.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
                    }
                }
            }
        }
        for (int x = 0; x < 2; x++)
        {
            for (int y = 2; y < 8; y++)
            {
                for (k = 0; k < 64; k++) {
```

```
                        for (l = 0; l < 64; l++) {

                        out_fvkv.write((char*)&gwhr[64 * x + k][64 * y + l], 2);
                        out_fvkv.write((char*)&gwhg[64 * x + k][64 * y + l], 2);
                        out_fvkv.write((char*)&gwhb[64 * x + k][64 * y + l], 2);
                        }
                  }
            }
      }
      //Векторно-квантованих коефіцієнтів ДОП
      for (int x = 2; x < 8; x++)
      {
            for (int y = 2; y < 8; y++)
            {
                  for (k = 0; k < 32; k++) {
                        for (l = 0; l < 32; l++) {
      out_fvkv.write((char*)&gwhrvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
      out_fvkv.write((char*)&gwhgvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
      out_fvkv.write((char*)&gwhbvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
                        }
                  }
            }
      }
      out_fvkv.close();

      name3 = "d:/DOTCodec/dmc.exe c " + name + ".cm " + name + ".cmc";

      i = 0;
      for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
      name2[i] = NULL;


      ZeroMemory(&si, sizeof(si));
      si.cb = sizeof(si);
      ZeroMemory(&pi, sizeof(pi));

      //Арифметичне кодування результуючого файлу
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
      {
            printf("CreateProcess failed (%d).\n", GetLastError());
            return;
      }

      WaitForSingleObject(pi.hProcess, INFINITE);
      CloseHandle(pi.hProcess);
      CloseHandle(pi.hThread);

}

//Модуль комбінованого векторного деквантування 4x4
private: System::Void button25_Click(System::Object^ sender, System::EventArgs^ e) {

      Bitmap^ image2;
      Color newColor1;
      String^ name;
      //Вибір файлу
      if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)

            name = openFileDialog1->FileName;

      String^ name3 = "d:/DOTCodec/dmc.exe e " + name + " " + name + ".cm";
      char name2[512];
      int i = 0;
      for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
      name2[i] = NULL;
```

```
	PROCESS_INFORMATION pi;
	STARTUPINFO si;
	ZeroMemory(&si, sizeof(si));
	si.cb = sizeof(si);
	ZeroMemory(&pi, sizeof(pi));
	//Арифметичне декодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
	{
		printf("CreateProcess failed (%d).\n", GetLastError());
		return;
	}
	WaitForSingleObject(pi.hProcess, INFINITE);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);
	name3 = name + ".rwh8";
	string name0 = "";
	for (int i = 0; i < name3->Length; i++)
		name0 += (char)name3[i];

	name3 = name + ".cm";
	string name20 = "";
	for (int i = 0; i < name3->Length; i++)
		name20 += (char)name3[i];
//Відкриття файлу з квантованими коефіцієнта ДОП для читання та запису деквантованих
	ofstream out_file(name0, ios::binary);
	ifstream in_fkv(name20, ios::binary);


	if (!in_fkv) {
		cout << "\nCouldn't open file";
		return;
	}

	image2 = gcnew Bitmap("lena512color.tiff", true);
	Byte r, g, b;
	int  j, k, l, t, n;
	short int tr[8][8];
	short int tr1[8][8];
	short int tg[8][8];
	short int tg1[8][8];
	short int tb[8][8];
	short int tb1[8][8];
	short int w[32][32];              //Карта Кохонена
	float lsp[16][16];

	n = 8;
	//Ініціалізація матриць квантування
	for (int k = 0; k < n; k++) {
		for (int l = 0; l < n; l++)
		{
			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView2->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);

			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView5->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);

			in_fkv.read((char*)&kv8[k][l], 1);
			dataGridView6->Rows[k]->Cells[l]->Value = kv8[k][l];
			out_file.write((char*)&kv8[k][l], 1);
		}
	}

	//Ініціалізація мережі
```

```
for (j = 0; j < 32; j++) {
	for (i = 0; i < 32; i++) {
		in_fkv.read((char*)&w[i][j], 2);
	}
}
//Читати цілочисельно квантовані коефіцієнти ДОП
for (int x = 0; x < 8; x++)
{
	for (int y = 0; y < 2; y++)
	{
		for (k = 0; k < 64; k++) {
			for (l = 0; l < 64; l++) {

			in_fkv.read((char*)&gwhr[64 * x + k][64 * y + l], 2);
			in_fkv.read((char*)&gwhg[64 * x + k][64 * y + l], 2);
			in_fkv.read((char*)&gwhb[64 * x + k][64 * y + l], 2);
			}
		}
	}
}
for (int x = 0; x < 2; x++)
{
	for (int y = 2; y < 8; y++)
	{
		for (k = 0; k < 64; k++) {
			for (l = 0; l < 64; l++) {

			in_fkv.read((char*)&gwhr[64 * x + k][64 * y + l], 2);
			in_fkv.read((char*)&gwhg[64 * x + k][64 * y + l], 2);
			in_fkv.read((char*)&gwhb[64 * x + k][64 * y + l], 2);
			}
		}
	}
}
//Читати векторно квантовані
for (int x = 2; x < 8; x++)
{
	for (int y = 2; y < 8; y++)
	{
		for (k = 0; k < 32; k++) {
			for (l = 0; l < 32; l++) {
	in_fkv.read((char*)&gwhrvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
	in_fkv.read((char*)&gwhgvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
	in_fkv.read((char*)&gwhbvk[32 * (x - 2) + k][32 * (y - 2) + l], 1);
			}
		}
	}
}

//Векторне деквантування

int ci, cj;

for (int y = 64; y < 256; y++)
{
	for (int x = 64; x < 256; x++)
	{	//RRR
		ci = gwhrvk[x - 64][y - 64] & 15;
		cj = (gwhrvk[x - 64][y - 64] / 16) & 15;
		i = w[2 * ci][2 * cj];
		gwhr[2 * x][2 * y] = w[2 * ci][2 * cj];
		gwhr[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
		gwhr[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
		gwhr[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
```

```
                        //GGG
                        ci = gwhgvk[x - 64][y - 64] & 15;
                        cj = (gwhgvk[x - 64][y - 64] / 16) & 15;
                        gwhg[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhg[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhg[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhg[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];
                        //BBB
                        ci = gwhbvk[x - 64][y - 64] & 15;
                        cj = (gwhbvk[x - 64][y - 64] / 16) & 15;
                        gwhb[2 * x][2 * y] = w[2 * ci][2 * cj];
                        gwhb[2 * x + 1][2 * y] = w[2 * ci + 1][2 * cj];
                        gwhb[2 * x][2 * y + 1] = w[2 * ci][2 * cj + 1];
                        gwhb[2 * x + 1][2 * y + 1] = w[2 * ci + 1][2 * cj + 1];

                }
        }

        i = 0;
        for (int x = 0; x <= 63; x++)
        {
                for (int y = 0; y <= 63; y++)
                {
                        //-----Дегрупування і виведення трансформант зображення----
                        n = 8;
                        for (int i = 0; i < n; i++) {
                                for (int j = 0; j < n; j++)
                                {
                                        tr[i][j] = gwhr[64 * j + x][64 * i + y];
                                        t = tr[i][j] / 8;

                                r = t;

                                        tg[i][j] = gwhg[64 * j + x][64 * i + y];
                                        t = tg[i][j] / 8;
                                g = t;

                                        tb[i][j] = gwhb[64 * j + x][64 * i + y];
                                        t = tb[i][j] / 8;

                                b = t;
                                        newColor1 = Color::FromArgb(r, g, b);
                                image2->SetPixel(64 * j + x, 64 * i + y, newColor1);

                                }
                        }

//Запис де групованих і неквантованих коефіцієнтів ДОП у вихідний файл
//RRR
for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {

                                        out_file.write((char*)&tr[k][l], 2);
                                }
                        }
                        //GGG
for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {
                                        out_file.write((char*)&tg[k][l], 2);
                                }
                        }
                        //BBB
                        for (k = 0; k < n; k++) {
                                for (l = 0; l < n; l++) {
```

```
                                        out_file.write((char*)&tb[k][l], 2);
                                }
                        }


                }

        }
        pictureBox2->Image = image2;

        in_fkv.close();
        out_file.close();

        name3 = "d:/DOTCodec/dmc.exe c " + name + ".rwh8 " + name + ".rcmc";
        i = 0;
        for (i = 0; i < name3->Length; i++) name2[i] = (char)name3[i];
        name2[i] = NULL;
        ZeroMemory(&si, sizeof(si));
        si.cb = sizeof(si);
        ZeroMemory(&pi, sizeof(pi));
        //Арифметичне кодування
if (!CreateProcess(NULL, (LPSTR)name2, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                printf("CreateProcess failed (%d).\n", GetLastError());
                return;
        }

        WaitForSingleObject(pi.hProcess, INFINITE);
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
}
```

Додаток Г
(довідниковий)

Формати файлів з трансформантами ДОП

Цілочисельно квантовані

.dct, dm8

Зсув	Назва параметру	Тип	Призначення
0	r	byte	Матриці квантування розміром 8x8 для RGB
1	g	Byte	
2	b	Byte	
…	…	…	
189	r	Byte	
190	g	byte	
191	b	byte	
192-319	R	Short int (2 байти)	Значення коефіцієнтів ДОП для складової R першого блоку 8x8
320-447	G	Short int (2 байти)	Значення коефіцієнтів ДОП для складової G першого блоку 8x4
448-575	B	Short int (2 байти)	Значення коефіцієнтів ДОП для складової B першого блоку 8x8
…	…	…	…

Комбіноване – групування, цілочислове та векторне квантування

.cm

Зсув	Назва параметру	Тип	Призначення
0	r	byte	Матриці квантування розміром 8x8 для RGB
1	g	Byte	
2	b	Byte	
…	…	…	
189	r	Byte	
190	g	byte	
191	b	byte	
192-2239	Корта Кохонена 4x4	Short int (2 байти)	Векторний квантувач
2240-395455	RGB	Short int (2 байти)	Трансформанти ДОП F(0,0)-F (7,0); F(0,1)-F(7,1) цілочисельно квантовані.
395456-690367	RGB	Short int (2 байти)	Трансформанти ДОП F(0,2)-F (0,7); F(1,2)-F(1,7) цілочисельно квантовані
690368-800959	RGB	byte	Трансформанти ДОП F(2,2)-F (2,7); F(3,2)-F(3,7)…F(7,2)-F(7,7) векторно квантовані

Додаток Д

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ КВАНТУВАННЯ КОМПОНЕНТ ДВОВИМІРНИХ ОРТОГОНАЛЬНИХ ПЕРЕТВОРЕНЬ ПРИ УЩІЛЬНЕННІ ЗОБРАЖЕНЬ

Магістерська кваліфікаційна робота “Методи та програмні засоби квантування компонент двовимірних ортогональних перетворень при ущільненні зображень”

Виконав: студент 2 курсу,
Групи 2ПІ-22м Матвійчук О. В.
Керівник: Катєльніков Д. І.

Рисунок Д.1 – Титульний слайд

2

МЕТА І ЗАДАЧІ РОБОТИ

Мета роботи:

- підвищення коефіцієнту ущільнення зображень за рахунок оптимізації квантування трансформант дискретних ортогональних перетворень.

Основні задачі дослідження:

- обгрунтування доцільності розробки нового технічного рішення;
- розробка алгоритмів і програм для дослідження методів квантування компонент зображень;
- експериментальні дослідження залежності ступеня ущільнення від методів квантування компонент зображень;
- розрахунок економічних показників розробки.

Об’єкт дослідження – процес квантування компонент зображень при ущільненні з використанням ортогональних перетворень.

Предмет дослідження – методи та програмні засоби квантування трансформант дискретних ортогональних перетворень при ущільненні зображень.

Рисунок Д.2 – Мета і задачі роботи

3

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ РОБОТИ

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод квантування трансформант двовимірних ортогональних перетворень при ущільненні зображень, у якому на відміну від існуючих, використано поблочне групування цілочисельно квантованих трансформант однакової частоти, що підвищує коефіцієнт ущільнення в 1,5-2 рази у порівнянні з методами без групування.

2. Вперше запропоновано комбінований метод виконання квантування трансформант дискретних ортогональних перетворень із використанням цілочислового квантування і групування низькочастотних трансформант та групування і векторного квантування високочастотних трансформант, що дозволило підвищити коефіцієнт ущільнення Grayscale зображень на 30 % у порівнянні з методом JPEG.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено мовою програмування C/C++ програмні засоби для дослідження квантування зображень та трансформант ДОП.

Особистий внесок здобувача. Усі результати, викладені магістерській кваліфікаційній роботі, отримані автором особисто. У друкованій праці, опублікованій у співавторстві, автору належать такі результати: порівняльний аналіз ДОП.

Рисунок Д.3 – Наукова новизна та практична цінність

4

СХЕМА УЩІЛЬНЕННЯ ЗОБРАЖЕНЬ НА ОСНОВІ ДИСКРЕТНИХ ОРТОГОНАЛЬНИХ ПЕРЕТВОРЕНЬ

Схеми кодування для дослідження

- ДОП – Цілочислове квантування – Кодування;
- ДОП – Цілочислове квантування – Групування - Кодування;
- ДОП – Цілочислове квантування – Групування – Зональний відбір-Кодування;
- ДОП – Групування – Зональний відбір-Кодування;
- ДОП – Цілочислове квантування – Групування – Векторне квантування-Кодування;
- Векторне квантування – Кодування.

Рисунок Д.4 - Схема ущільнення зображень на основі дискретних ортогональних перетворень

5

ДИСКРЕТНЕ КОСИНУСНЕ ПЕРЕТВОРЕННЯ ТА ПЕРЕТВОРЕННЯ УОЛША-АДАМАРА

Перетворення Уолша-Адамара

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)*(-1)^{p(x,y,u,v)},$$

$$p(x,y,u,v)=\sum_{i=0}^{n-1}(u_i x_i \oplus v_i y_i)$$

Матриця Адамара упорядкована за Уолшом розмірності 8x8

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

Дискретне косинусне перетворення

Пряме:

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

Зворотне:

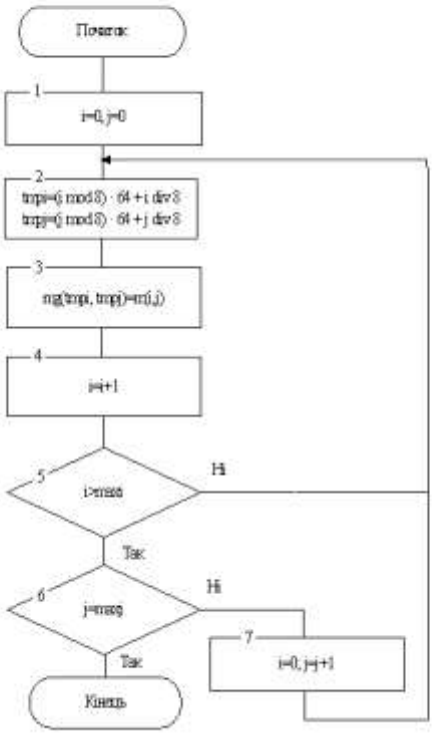
$$f(j,k)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos\left[\frac{\pi}{N}u\left(j+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(k+\frac{1}{2}\right)\right]$$

Дослідження проводились при N=8

Рисунок Д.5 - Дискретне косинусне перетворення та перетворення Уолша-Адамара

6

БЛОК-СХЕМА АЛГОРИТМУ ГРУПУВАННЯ ТРАНСФОРМАНТ ДОП



Згруповані трансформанти ДОП

$$F_{00}(0,0),\ldots F_{kl}(0,0),$$
$$F_{00}(0,1),\ldots F_{kl}(0,1),$$
$$\ldots$$
$$F_{00}(7,7),\ldots F_{kl}(7,7)$$

Рисунок Д.6 – Блок-схема алгоритму групування трансформант ДОП

7

ВЕКТОРНЕ КВАНТУВАННЯ З ВИКОРИСТАННЯМ КАРТИ КОХОНЕНА

Етапи векторного квантування з використанням карти Кохонена :

- навчання мережі;
- векторне квантування.

1. Ініціалізувати вагові коефіцієнти випадковими значеннями.
2. Для кожного кластерного елемента обчислити відстань до навчального вектора:

$$d_j = \sum_i (w_{ij} - x_i)^2$$

3. Найти кластерний елемент j для якого d_j мінімально.
4. Для кластерних елементів із круга заданого радіуса з центром в j елементі обновити вагові коефіцієнти згідно формули:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)] \ .$$

де η - норма навчання, x_i – координата навчального вектора.

5. Обновити норму навчання η і радіус при необхідності і повторити пункти 1-5 для наступного навчального вектора.

Рисунок Д.7 – Векторне квантування з використанням карти Кохонена

8

БЛОК-СХЕМА АЛГОРИТМУ ВЕКТОРНОГО КВАНТУВАННЯ ЗГРУПОВАНИХ ТРАНСФОРМАНТ ДОП

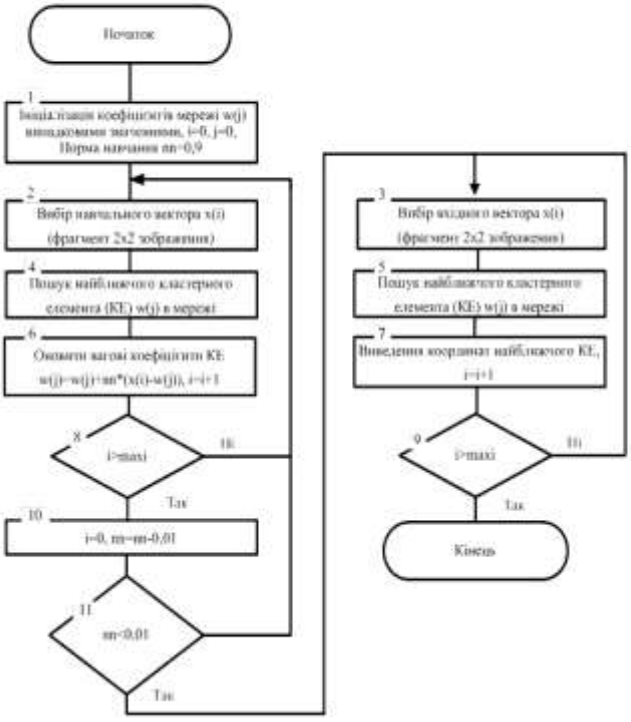


Рисунок Д.8 – Блок-схема алгоритму векторного квантування

9

ГОЛОВНЕ ВІКНО ПРОГРАМИ

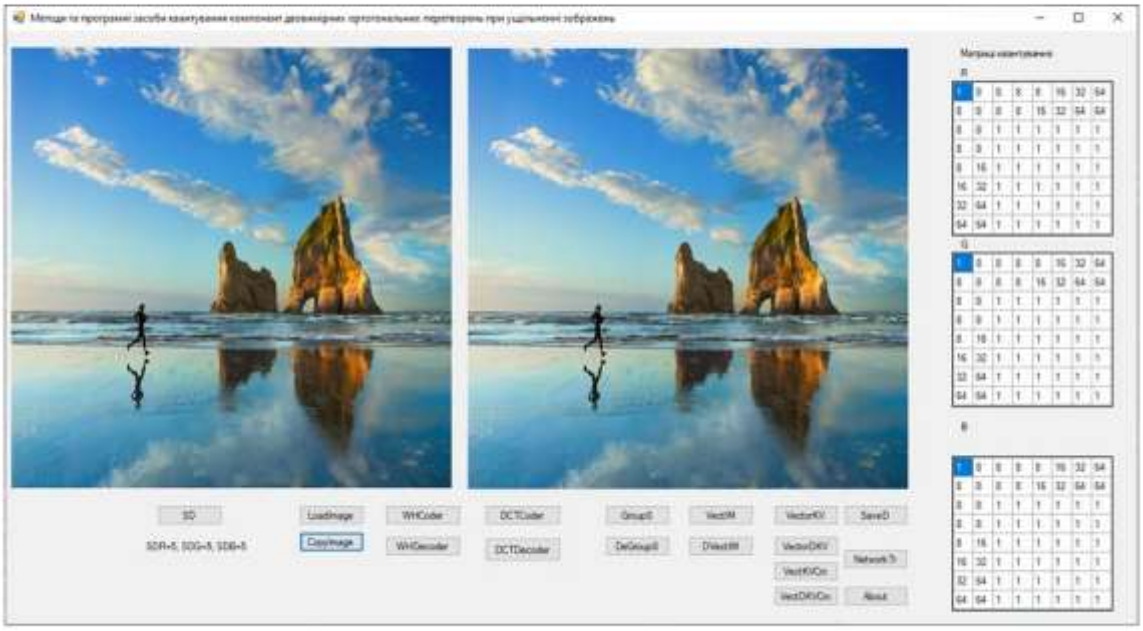


Рисунок Д.9 – Головне вікно програми

10

ІНШІ ВІКНА ДОДАТКУ

Головне вікно програми після запуску

Вікно «Про програму»

Рисунок Д.10 – Інші вікна

11

КЛАСИ І МЕТОДИ ДОДАТКУ

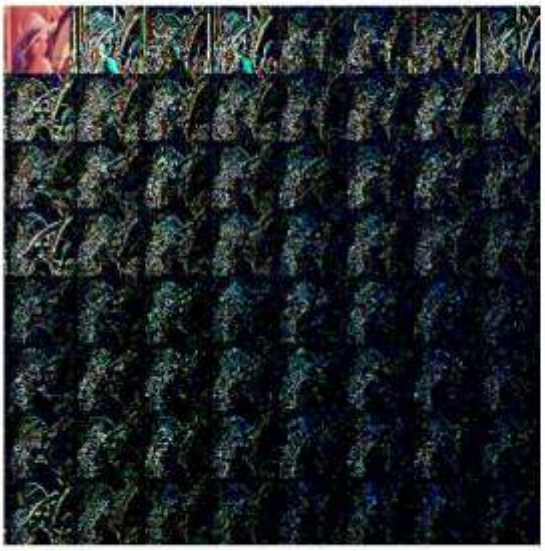
Команда	Метод
About	button6_click()
CopyImage	button2_click()
DCTCoder	button22_click()
DCTDecoder	button23_click()
DeGroup8	button16_click()
DeVectIm	button20_click()
Group8	button15_click()
LoadImage	button8_click()
NetworkTr	button21_click()
SaveD	button7_click()
SD	button12_click()
VectDKVCm	Button25_Click()
VectIm	Button19_Click()
VectKVCm	Button24_Click()
VectorDKV	Button18_Click()
VectorKV	Button17_Click()
WHCoder	Button13_Click()
WHDecoder	Button14_Click()

Рисунок Д.11 – Класи і методи додатку

12

ЗГРУПОВАНІ ТРАНСФОРМАНТИ ДОП

Перетворення Уолша-Адамара



ДКП

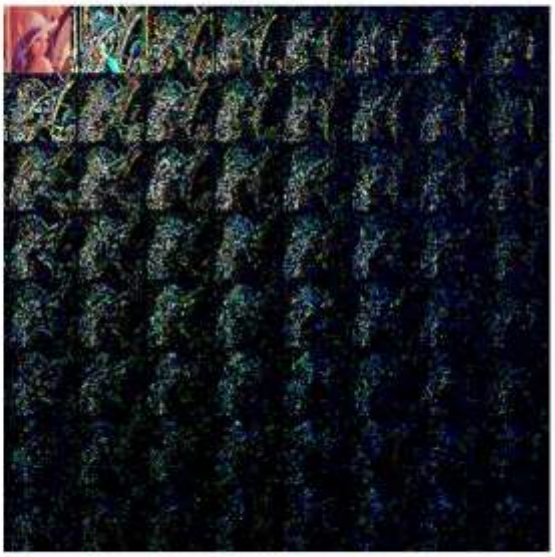


Рисунок Д.12 – Згруповані трансформанти ДОП

13

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ ГРУПУВАННЯ ТРАНСФОРМАНТ

На прикладі файлів Lena

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групування, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale)	Уолша-Адамара	180782	78 307	4	Відмінна
	ДКП	169534	72312	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
716 794 (color)	Уолша-Адамара	214070	170 338	5	Відмінна
	ДКП	203859	161 505	4	
716 794 (color)		95 646 (jpeg)	95 646 (jpeg)	4	Відмінна

Матриці квантування

R

1	1	1	1	1	1	1	1
1	1	8	8	16	32	64	64
1	8	8	16	32	64	64	64
1	8	16	32	64	64	64	64
1	16	32	64	64	64	64	64
1	32	64	64	64	64	64	64
1	64	64	64	64	64	64	64
1	64	64	64	64	64	64	64

Рисунок Д.13 – Результати досліджень групування трансформант

14

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ З ГРУПУВАННЯ ТА ЗОНАЛЬНИМ ВІДБОРОМ ТРАНСФОРМАНТ

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групування та зональним відбором, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale) МК R	Уолша-Адамара	180782	77 389	4	Відмінна
	ДКП	169534	71848	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
263 222 (grayscale) МК G	Уолша-Адамара	146 323	63 589	3	Відмінна
	ДКП	126 635	55 985	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна

G

2	4	4	4	4	4	4	4
4	4	8	8	8	8	8	16
4	8	8	8	8	8	16	16
4	8	8	8	8	16	16	16
4	8	8	8	16	16	16	16
4	8	8	16	16	16	16	16
4	8	16	16	16	16	32	32
4	16	16	16	16	16	32	32

Рисунок Д.14 – Результати досліджень з групуванням та зональним відбором

15

ПОЧАТКОВЕ І ВІДНОВЛЕНЕ ЗОБРАЖЕННЯ ПРИ ГРУПУВАННІ І ЗОНАЛЬНОМУ ВІДБОРІ

Початкове зображення **Відновлене зображення**

Початкове (розмір файлу 786486 байт) та відновлене (розмір файлу файлу 115604 байт) зображення. Перетворення ДКП, матриця квантування МК G, СКВ=3

Рисунок Д.15 – Приклад зображення при групуванні і зональному відборі

16

ДОСЛІДЖЕННЯ ВЕКТОРНОГО КВАНТУВАННЯ КОМПОНЕНТ ЗОБРАЖЕННЯ

Векторне квантування безпосередньо зображення

Параметри векторного кантувача такі:
- двовимірна карта Кохонена розміром 16x16 кластерних елементів;
- розмірність вхідного вектора – 2x2.

Результати:
- Вхідний файл кольорового зображення стандартної заставки розміром 786486 байт;
- файл .vim – має розмір 198 656 байт, що відповідає очікуваним результатам;
- ущільнений файл .vimc має розмір 84 802 байт, СКВ=6, візуальна якість зображення відмінна.
- файл у форматі .jpg має розмір 88 041 байт, СКВ=5.

Векторне кантування коефіцієнтів ДОП

Схема дослідження:
ДОП -> Групування трансформант->Навчання мережі ->Векторне квантування->Арифметичне кодування.

Параметри векторного кантувача такі:
- двовимірна карта Кохонена розміром 64x64 кластерних елементів;
- розмірність вхідного вектора – 2x2.

Результати:
-вхідний файл Lena512.bmp (grayscale) розміром 263222 байт;
-ущільнений файл .vkkc має розмір 93 699 байт без карти Кохонена розміром 32768 байт, СКВ=2, візуальна якість зображення відмінна;
- час ущільнення 600 сек для зображення у форматі 512x512;
- файл у форматі .jpg має розмір 82 012 байт, СКВ=1.

Рисунок Д.16 – Дослідження векторного квантування

17

ЗГРУПОВАНІ КОЕФІЦІЄНТИ ДОП ДЛЯ КОМБІНОВАНОГО КВАНТУВАННЯ ТРАНСФОРМАНТ

Згруповані коефіцієнти перетворення Уолша-Адамара

Матриця квантування

1	8	8	8	8	16	32	64
8	8	8	8	16	32	64	64
8	8	1	1	1	1	1	1
8	8	1	1	1	1	1	1
8	16	1	1	1	1	1	1
16	32	1	1	1	1	1	1
32	64	1	1	1	1	1	1
64	64	1	1	1	1	1	1

Рисунок Д.17 – Згруповані коефіцієнти для комбінованого квантування

18

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ КОМБІНОВАНОГО КВАНТУВАННЯ ТРАНСФОРМАНТ ДОП

Схема ущільнення зображення:
- виконання ДОП;
- групування коефіцієнтів ДОП;
- для низькочастотних трансформант F (0,0)…F (0,7); F (1,0)…F(1,7); F(2,0)…F (7,0); F (2,1)…F(7,1) виконується цілочислове квантування;
- для інших трансформант векторне квантування при розмірах карти Кохонена 16x16, розмірність вхідного вектора 2x2.

Розмір початкового зображення, байт		Розмір зображення після ущільнення без групування, байт	Розмір зображення після ущільнення з групуванням та векторним квантуванням ВЧ трансформант, байт	Середньо-квадратичне відхилення (СКВ)	Візуальна оцінка якості
263 222 (grayscale)	Уолша-Адамара	302352	57876	5	Відмінна
	ДКП	284509	50658	3	
263 222 (grayscale)		82912 (jpeg)	82 912 (jpeg)	1	Відмінна
716 794 (color)	Уолша-Адамара	371283	138433	5	Відмінна
	ДКП	203008	128229	4	
716 794 (color)		95 646 (jpeg)	95 646 (jpeg)	4	Відмінна

Рисунок Д.18 – Результати дослідження комбінованого квантування

19

ПРИКЛАД ЗОБРАЖЕННЯ ДЛЯ КОМБІНОВАНОГО КВАНТУВАННЯ ТРАНСФОРМАНТ ДОП

Початкове зображення

Відновлене зображення

Рисунок Д.19 – Приклад зображення для комбінованого квантування

20

ФОРМАТИ ФАЙЛІВ З ТРАНСФОРМАНТАМИ ПЕРЕТВОРЕННЯ

Цілочисельно квантовані
.dct, dm8

Зсув	Назва параметру	Тип	Призначення
0	r	byte	Матриця квантування розміром 8x8 для RGB
1	g	Byte	
2	b	Byte	
...	...	...	
189	r	Byte	
190	g	byte	
191	b	byte	
192-319	R	Short int (2 байти)	Значення коефіцієнтів ДОП для складової R першого блоку 8x8
320-447	G	Short int (2 байти)	Значення коефіцієнтів ДОП для складової G першого блоку 8x4
448-575	B	Short int (2 байти)	Значення коефіцієнтів ДОП для складової B першого блоку 8x8
...	...	...	...

Комбіноване – групування, цілочислове та векторне квантування
.cm

Зсув	Назва параметру	Тип	Призначення
0	r	byte	Матриця квантування розміром 8x8 для RGB
1	g	Byte	
2	b	Byte	
...	...	...	
189	r	Byte	
190	g	byte	
191	b	byte	
192-2239	Карта Кохонена 4x4	Short int (2 байти)	Векторний квантувач
2240-395455	RGB	Short int (2 байти)	Трансформанти ДОП F(0,0)-F (7,0); F(0,1)-F(7,1) цілочисельно квантовані
395456-690367	RGB	Short int (2 байти)	Трансформанти ДОП F(0,2)-F (0,7); F(1,2)-F(1,7) цілочисельно квантовані
690368-800959	RGB	byte	Трансформанти ДОП F(2,2)-F (2,7); F(3,2)-F(3,7)...F(7,2)-F(7,7) векторно квантовані

Рисунок Д.20 – Формати файлів

21

ОСНОВНІ РЕЗУЛЬТАТИ РОБОТИ

1. Показано, що коефіцієнт ущільнення зображень з втратами на основі двовимірних ортогональних перетворень повністю визначається методом квантування трансформант цих перетворень.

2. Розроблено схему ущільнення зображень на основі двовимірних ортогональних перетворень та методи і алгоритми виконання квантування трансформант цих перетворень, які ґрунтуються на цілочисловому та векторному квантуванні.

3. Аналіз середовищ розробки та мов програмування показав, що для цієї розробки доцільним є використання середовища розробки Microsoft Visual Studio та мови програмування C++, оскільки ці засоби забезпечують найкращі швидкісні характеристики для програм, що розробляються, а також мають розвинені сервісні функції, які пришвидшують розробку.

4. Мовою програмування C++ розроблено додаток для дослідження квантування компонент двовимірних ортогональних перетворень. Додаток побудовано за мультимодульною архітектурою, оскільки дослідження різних методів квантування компонент зображення вимагає великої кількості експериментальних досліджень і змін у коді у процесі цих досліджень. В якості перетворень використовувалось дискретне косинусне перетворення та перетворення Уолша-Адамара.

5. Розроблено керівництво користувача, яке містить необхідні відомості для установки програми на комп'ютер та її експлуатації.

6. Тестування програми показало, що застосування групування трансформант ДОП після цілочислового квантування за частотами дозволяє підвищити коефіцієнт ущільнення 1,5-2 рази у порівнянні з файлами, де не використовувалось групування.

7. Показано, що подальше збільшення коефіцієнту ущільнення можна досягнути за рахунок векторного квантування трансформант перетворення з використанням нейронної мережі типу двовимірна карта Кохонена.

8. Дослідження показали, що кращі результати з точки зору швидкості ущільнення забезпечує комбіноване квантування – для низькочастотних трансформант цілочислове квантування, для високочастотних – векторне з використанням двовимірної карти Кохонена розмірністю 16x16 при розмірності вхідних векторів 2x2, що дозволило підвищити коефіцієнт ущільнення Grayscale зображень на 30 % у порівнянні з методом JPEG. Час векторного квантування зображення розміром 512x512 складає 15 сек.

9. Розрахунки затрат на роботу показали доцільність даної розробки з економічної точки зору, оскільки комплексний коефіцієнт доцільності виконання науково-дослідної роботи $K_D > 1$

Рисунок Д.21 – Основні результати роботи

ДЯКУЮ ЗА УВАГУ!!!

Рисунок Д.22 – Фінальний слайд