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

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОДУВАННЯ ВІДЕОЗОБРАЖЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ СТИСНЕННЯ У СИСТЕМАХ ОБРОБКИ ВІДЕОІНФОРМАЦІЇ»

Виконала: студентка IV курсу, групи АКІТ-21бз
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Поліна КОСТРИЦЯ
(прізвище та ініціали)
Керівник: доцент кафедри АІІТ
Володимир ГАРМАШ
(прізвище та ініціали)
«17» 06 2025 р.
Рецензент: к.т.н, доцент, професор кафедри КСУ
Микола БИКОВ
(прізвище та ініціали)
«18» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
Олег БІСІКАЛО
«19» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

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

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
_____ д.т.н., проф. Олег БІСІКАЛО
«19» 06 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Кострицi Поліні Русланівні
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації»
керівник роботи Гармаш Володимир Володимирович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи:
   - тип цифрового зображення – півтонові та кольорові;
   - мінімальна роздільна здатність зображення 128 x 128 пікселів, максимальна роздільна здатність 2048 x 2048 пікселів;
   - коефіцієнт стиснення від 1.1 до 3.
   - Виграш у стисненні від 0,5% до 10% у порівнянні з класичним алгоритмом RLE.
4. Зміст текстової частини (перелік питань, які потрібно розробити) Вступ. Аналіз методів кодування зображень. Алгоритм кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації. Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації. Висновки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) діаграма активності роботи кодера, діаграма активності процедури виділення групових послідовностей, діаграма класів системи стиснення зображень, діаграма класів процесу стискання, тестові зображення.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

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

Студентка _____ Поліна КОСТРИЦЯ
(підпис) (ім'я та прізвище)

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

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 81 сторінки формату А4, включаючи додатки, що містить 17 рисунків. Список використаних джерел містить 23 найменування.

Робота присвячена розробці програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації на основі адаптивного алгоритму групового кодування. Метою є вдосконалення існуючих алгоритмів кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації, їх програмна реалізація та експериментальне дослідження. Проведено аналіз існуючих алгоритмів стиснення, таких як RLE та H.264, а також досліджено можливості адаптації групового кодування до потреб відеоінформаційних систем. Запропоновано модифікований метод, який перевершує RLE за якістю компресії в 1,28 раза, зберігаючи адаптивність до різноманітних типів вмісту. Розроблено програмне забезпечення з використанням MATLAB, яке реалізує цей метод, та проведено експерименти на тестовій колекції CIPR Still Images, що підтверджують його ефективність у автоматизованих системах обробки відеоінформації.

Ключові слова: стиснення, зображення; групове кодування, алгоритм RLE, коефіцієнт стиснення.

ABSTRACT

The bachelor's qualification work consists of 81 pages of A4 format, including appendices, containing 17 figures. The list of used sources contains 23 names.

The work is devoted to the development of video image coding software for automating the compression process in video information processing systems based on the adaptive group coding algorithm. The goal is to improve existing video image coding algorithms for automating the compression process in video information processing systems, their software implementation and experimental research. An analysis of existing compression algorithms, such as RLE and H.264, was conducted, and the possibilities of adapting group coding to the needs of video information systems were also investigated. A modified method is proposed that surpasses RLE in compression quality by 1.28 times, while maintaining adaptability to various types of content. Software using MATLAB has been developed to implement this method, and experiments have been conducted on the CIPR Still Images test collection, confirming its effectiveness in automated video information processing systems.

Keywords: compression, image; group coding, RLE algorithm, compression ratio.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі, де обсяги відеоконтенту стрімко зростають, автоматизація процесу стиснення відеозображень стає надзвичайно актуальною задачею. Відеодані займають значну частину цифрового трафіку, що зумовлено їхнім широким використанням у соціальних мережах, онлайн-освіті, потокових платформах і системах відеоспостереження. Збільшення кількості пристроїв із високою роздільною здатністю, таких як 4K і 8K телевізори, смартфони та камери, призводить до експоненційного зростання обсягів інформації, що передається. Це створює значне навантаження на комунікаційні мережі, обчислювальні ресурси та сховища даних, що потребує ефективних рішень для оптимізації.

Одним із ключових викликів є забезпечення швидкого доступу до відеоконтенту в умовах обмеженої пропускної здатності, особливо в мобільних мережах. Наприклад, перегляд відео у високій якості на смартфонах із повільним інтернетом може бути ускладнений через великі розміри файлів. Розробка програмного забезпечення для кодування відеозображень із автоматизацією стиснення дозволяє зменшувати розмір файлів без значної втрати якості, що сприяє прискоренню завантаження та економії трафіку. Це особливо важливо для користувачів у регіонах із нестабільним з’єднанням, де кожна секунда затримки впливає на досвід перегляду.

Крім того, зростання популярності потокового відео, таких як Netflix, YouTube та Twitch, вимагає від систем обробки відеоінформації високої ефективності. Сучасні алгоритми кодування, такі як H.265/HEVC і H.266/VVC, пропонують кращу компресію порівняно з попередніми стандартами, але їхнє впровадження потребує складного програмного забезпечення, здатного адаптуватися до різноманітних умов. Автоматизація цього процесу дозволяє зменшити участь людини, прискорити обробку та

забезпечити узгодженість результатів, що є критичним для великих платформ, які щодня обробляють мільйони годин відео.

Іншим важливим аспектом є інтеграція відеосистем у комп'ютерно-інтегровані платформи, наприклад, у промисловості чи охороні. Системи відеоспостереження генерують величезні обсяги даних, які необхідно зберігати протягом тривалого часу. Ефективне стиснення дозволяє оптимізувати використання пам'яті та зменшити витрати на зберігання, зберігаючи при цьому ключову інформацію, таку як обличчя чи номери автомобілів. Розробка програмного забезпечення, яке автоматизує кодування, стає необхідним кроком для підвищення продуктивності таких систем і забезпечення їхньої роботи в реальному часі.

Розвиток технологій штучного інтелекту та машинного навчання також впливає на актуальність теми. Сучасні алгоритми кодування можуть використовувати нейронні мережі для адаптивного стиснення, що враховує специфіку вмісту відео (наприклад, статичні сцени чи динамічні кадри). Це відкриває нові можливості для створення інтелектуальних систем, які оптимізують якість і розмір залежно від контексту. Однак реалізація таких рішень потребує розробки спеціалізованого програмного забезпечення, яке здатне інтегрувати передові технології з існуючими стандартами.

Таким чином, актуальність розробки програмного забезпечення для кодування відеозображень полягає в необхідності вирішення проблем, пов'язаних із зростанням обсягів даних, обмеженнями мереж і вимогами до швидкості обробки. Автоматизація стиснення у системах обробки відеоінформації не лише покращує ефективність ресурсів, але й сприяє розширенню доступу до відеоконтенту для широкої аудиторії. У майбутньому ця галузь матиме ще більший вплив завдяки інтеграції інноваційних технологій, що зробить її пріоритетною для наукових досліджень і промислових розробок.

Метою даної роботи є вдосконалення існуючих алгоритмів кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації, їх програмна реалізація та експериментальне дослідження.

Об’єктом дослідження даної бакалаврської роботи є процес кодування відеозображень.

Предметом дослідження даної бакалаврської роботи є алгоритмічні, математичні та програмно-технічні засоби системи обробки відеозображень.

Для досягнення мети в роботі поставлені наступні основні **задачі**:

- Огляд відомих методів і алгоритмів кодування відеокадрів;
- Покращення та створення нових методів кодування відеозображень;
- Удосконалення та розробка інноваційних алгоритмів декодування відеокадрів;
- Тестування та експериментальна оцінка створених програмних інструментів.

Практична цінність роботи полягає в розробці програмного забезпечення для кодування відеозображень, що значно підвищує ефективність автоматизованого стиснення у системах обробки відеоінформації. Створений модуль дозволяє зменшити обсяг даних, зберігаючи прийнятну якість зображень, що є критично важливим для потокового передавання. Використання запропонованого рішення сприяє оптимізації ресурсів обчислювальних систем і комунікаційних каналів. Програмне забезпечення може бути інтегроване в існуючі платформи для обробки відео, забезпечуючи прискорення завантаження контенту. Розробка адаптується до умов низькошвидкісних мереж, що особливо корисно для мобільних користувачів. Рішення має потенціал для застосування в індустріях, таких як телекомунікації, охорона та розваги. Робота відкриває можливості для подальшого вдосконалення технологій стиснення відео в реальному часі.

1 АНАЛІЗ МЕТОДІВ КОДУВАННЯ ЗОБРАЖЕНЬ

1.1 Аналіз об’єкту автоматизації

Система обробки відеоінформації є складним об’єктом автоматизації, який відіграє ключову роль у сучасних технологіях, охоплюючи такі сфери, як телекомунікації, розваги, безпека та промисловість. Її основною функцією є прийом, аналіз, обробка та передача відеоданих у реальному часі або з відкладеним доступом. Ця система складається з кількох взаємопов’язаних компонентів, включаючи апаратне забезпечення (камери, сервери, процесори), програмне забезпечення (алгоритми кодування/декодування, інтерфейси) та комунікаційні канали. Об’єктом автоматизації в даному контексті є процес стиснення відеозображень, який спрямований на оптимізацію використання ресурсів і підвищення ефективності роботи системи.

Система обробки відеоінформації функціонує як інтегрований комплекс, де кожна складова має специфічне призначення. Джерела відеоданих, такі як камери високої роздільної здатності або датчики, генерують великі обсяги інформації, які потребують обробки. Ці дані передаються через мережеві канали до центрального процесора, де здійснюється їхній аналіз і кодування. Апаратне забезпечення, зокрема графічні процесори (GPU) і спеціалізовані чипи, забезпечує високу швидкість обчислень, необхідну для реального часу. Програмне забезпечення, у свою чергу, включає алгоритми стиснення (наприклад, H.264, H.265/HEVC), які зменшують розмір відеофайлів без значної втрати якості.

Важливим аспектом є взаємодія між компонентами системи. Наприклад, затримки в передачі даних або недостатня обчислювальна потужність можуть призводити до втрати кадрів або погіршення якості відео. Тому об’єкт автоматизації має бути адаптованим до змін навантаження, якості з’єднання та

специфіки відеоконтенту (наприклад, динамічні сцени чи статичні зображення).

Об’єкт автоматизації – процес стиснення відеозображень – характеризується високою складністю через обсяги даних і вимоги до швидкості обробки. Відео складається з послідовності кадрів, кожен із яких є цифровим зображенням, що містить мільйони пікселів. Стиснення вимагає аналізу просторових і тимчасових кореляцій між кадрами, що ускладнює алгоритми кодування. Основними параметрами є ступінь стиснення, якість відтворення та час обробки. Наприклад, стандарт H.265 забезпечує кращу компресію порівняно з H.264, але потребує більше обчислювальних ресурсів.

Система також залежить від зовнішніх факторів, таких як пропускна здатність мережі та характеристики кінцевого пристрою (смартфон, телевізор, комп’ютер). У реальному часі, наприклад, під час відеоконференцій, важливим є мінімізувати затримки, що вимагає адаптивного стиснення. Крім того, об’єкт піддається впливу шумів, артефактів і змін освітлення, що ускладнює підтримання стабільної якості.

Однією з ключових проблем є баланс між якістю відео та розміром файлу. Високий рівень стиснення може призводити до появи блокових артефактів або втрати деталей, особливо в динамічних сценах. Іншою проблемою є енергоефективність, особливо для портативних пристроїв, де ресурси обмежені. Автоматизація процесу стиснення має враховувати ці фактори, адаптуючи алгоритми до умов роботи системи.

Також важливим є забезпечення сумісності з різними платформами та стандартами кодування. Наприклад, старіші пристрої можуть не підтримувати нові формати, такі як AV1, що вимагає гнучкості в реалізації програмного забезпечення. Викликом залишається й захист даних, адже відеоконференції чи відеоспостереження часто передбачають шифрування, яке додає додаткове навантаження на систему.

Автоматизація стиснення відеозображень відкриває можливості для покращення продуктивності системи. Використання штучного інтелекту дозволяє адаптивно регулювати рівень стиснення залежно від змісту кадру – наприклад, зменшуючи якість у фонових областях і зберігаючи деталі в ключових зонах. Інтеграція з хмарними технологіями може розподілити обчислювальне навантаження, зменшивши вимоги до локальних пристроїв.

Таким чином, система обробки відеоінформації є динамічним об’єктом автоматизації, який потребує комплексного підходу до розробки програмного забезпечення для кодування. Аналіз її структури, характеристик і викликів є основою для створення ефективних рішень, що забезпечать оптимальне стиснення та високу якість обробки відео.

1.2 Теоретичні основи стиснення інформації

Стиснення інформації є фундаментальним процесом у комп’ютерних науках, спрямованим на зменшення обсягу даних для ефективнішого зберігання та передачі. Теоретична основа цього процесу базується на концепціях надлишковості даних та ентропії, які були сформульовані в інформаційній теорії, зокрема працями Клода Шеннона. Надлишковість виникає тоді, коли дані містять повторювані або передбачувані елементи, які можна представити компактніше. Ентропія ж визначає мінімальну кількість бітів, необхідних для кодування повідомлення без втрати змісту, відображаючи ступінь випадковості в даних.

Процес стиснення ґрунтується на видаленні надлишкової інформації або її компактному представленні. У текстах чи зображеннях часто зустрічаються повторювані послідовності, які алгоритми можуть замінити коротшими кодами. Наприклад, у текстах слово, що часто повторюється, може бути закодовано одним символом, а в зображеннях однорідні ділянки можна

описати єдиним значенням. Цей підхід дозволяє зменшити розмір файлу, зберігаючи основний зміст.

Інформаційна теорія Шеннона вводить поняття ентропійного кодування, яке визначає оптимальну довжину кодового слова для кожного символу залежно від його ймовірності появи. Чим нижча ймовірність символу, тим довшим є його код, що забезпечує економію бітів у середньому. Наприклад, у тексті англійською мовою літера "e" з'являється частіше, ніж "z", тому її код може бути коротшим, що оптимізує загальний обсяг даних.

Існують два основних підходи до стиснення: статистично-базовані та словникові методи. Статистичні методи, такі як кодування Хаффмана, аналізують частоту появи символів у повідомленні та присвоюють їм змінної довжини коди. Це дозволяє ефективно стискати дані з нерівномірним розподілом символів. Словникові методи, наприклад, LZW, створюють таблицю повторюваних послідовностей і замінюють їх посиланнями на цю таблицю, що особливо корисно для текстів із частими повтореннями.

У зображеннях стиснення часто включає перетворення просторових даних у частотну область за допомогою методів, таких як дискретне косинусне перетворення (DCT). Це дозволяє виділити низькочастотні компоненти, які несуть основну інформацію, і відкинути високочастотні деталі, що менш помітні для людського ока. Такий підхід лежить в основі формату JPEG і є прикладом стиснення з втратами.

Теоретичні межі стиснення визначаються ентропією джерела інформації. За законом Шеннона, жоден алгоритм не може стиснути дані нижче їхньої ентропійної межі без втрат, якщо дані є абсолютно випадковими. Для даних із високою надлишковістю, як-от текст чи зображення, можливе значне стиснення, але для випадкових послідовностей, таких як криптографічні ключі, стиснення неефективне.

Стиснення без втрат обмежене лише видаленням надлишковості, тоді як стиснення з втратами використовує моделі сприйняття людини для усунення

менш значущих деталей. Наприклад, людське око менш чутливе до змін у високочастотних областях зображення, що дозволяє алгоритмам JPEG видаляти ці дані без значного впливу на візуальну якість.

Теоретичні основи стиснення мають практичне застосування в комп’ютерно-інтегрованих системах, де економія ресурсів є критичною. У веброзробці стиснення зменшує час завантаження сторінок, у відеоконференціях – обсяг переданих даних, а в архівації – місце на носіях. Розуміння принципів ентропії та надлишковості дозволяє розробляти алгоритми, які балансують між якістю та розміром, адаптуючись до специфіки даних.

Розвиток теорії стиснення також впливає на появу нових методів, таких як адаптивне кодування чи алгоритми на основі машинного навчання. Ці підходи дозволяють динамічно аналізувати вміст і оптимізувати процес стиснення, що відкриває нові можливості для обробки великих обсягів інформації в реальному часі [2-5].

Цифрові зображення розрізняються за типами, форматами та призначенням, що дозволяє класифікувати їх залежно від структури, способу кодування та сфери застосування. Класи зображень відображають їхні технічні характеристики та специфіку використання в комп’ютерно-інтегрованих системах, від вебдизайну до наукових досліджень. Розуміння цих категорій є важливим для вибору оптимального формату та методів обробки.

Першим класом є растрові зображення, які складаються з пікселів – окремих точок із заданими значеннями кольору чи яскравості. Растрові зображення, такі як JPEG, PNG чи BMP, є найпоширенішими в цифровій фотографії та графіці. Їхня якість залежить від роздільної здатності: чим більше пікселів, тим детальніше зображення. Однак при масштабуванні такі зображення можуть втрачати чіткість, оскільки пікселі стають видимими. Растрові формати ідеально підходять для реалістичних зображень, таких як

портрети чи пейзажі, але їхній обсяг даних може бути значним без стиснення [3].

Другим класом є векторні зображення, які базуються на математичних рівняннях для опису ліній, кривих та фігур. Формати, такі як SVG або AI, дозволяють масштабувати зображення без втрати якості, що робить їх популярними в логотипах, ілюстраціях та дизайні інтерфейсів. Векторні зображення займають менше місця, оскільки не зберігають інформацію про кожен піксель, а лише про геометричні об’єкти. Проте вони менш придатні для складних фотографій із градієнтами чи текстурами.

Зображення також класифікуються за кольоровими моделями, які визначають спосіб представлення кольорів. Монохромні зображення використовують лише один канал – яскравість, що робить їх простими для обробки та зберігання. Вони застосовуються в скануванні документів чи медичних знімках, де важлива контрастність. Кольорові зображення зазвичай використовують модель RGB (червоний, зелений, синій), яка є стандартною для екранів. Кожен піксель тут складається з трьох компонентів, що дозволяє відтворювати широкий спектр кольорів. Інший клас – зображення в моделі CMYK (ціан, магента, жовтий, чорний), призначені для друку, де кольори змішуються субтрактивно [4].

За призначенням зображення поділяються на кілька груп. Фотографічні зображення відображають реальні сцени та об’єкти, часто мають високу деталізацію та використовують стиснення з втратами, наприклад, JPEG. Графічні зображення створюються штучно і включають логотипи, іконки чи діаграми, де перевагу віддають векторним форматам або PNG для збереження прозорості. Технічні зображення, такі як рентгенівські знімки чи схеми, потребують високої точності та часто зберігаються у форматах без втрат, наприклад, TIFF.

Окремий клас становлять анімаційні зображення, представлені форматами GIF чи WebP. Вони складаються з послідовності кадрів, що

дозволяє створювати рухомі ефекти. Такі зображення широко використовуються в рекламі та соціальних мережах. Іншим прикладом є зображення з високим динамічним діапазоном (HDR), які зберігають деталі в темних і світлих областях, що робить їх придатними для професійної фотографії та кіновиробництва. Формати, такі як EXR чи Radiance, підтримують HDR і вимагають спеціального програмного забезпечення для обробки.

Кожен клас має свої технічні вимоги. Растрові зображення з високою роздільною здатністю, наприклад 4K, потребують значного обсягу пам’яті, що робить їх складними для швидкої передачі. Векторні зображення, навпаки, легкі, але менш гнучкі для реальних сцен. Монохромні зображення економічніші за обсягом, але обмежені в передачі кольорової інформації. Анімаційні формати балансують між розміром і кількістю кадрів, а HDR-зображення потребують розширеної глибини кольору (10-16 біт на канал).

Вибір класу залежить від контексту. Для вебсайтів оптимальними є JPEG для фотографій і SVG для іконок через їхню компактність і сумісність. У медицині перевага віддається TIFF або DICOM для збереження всіх деталей. Анімаційні GIF підходять для коротких візуальних ефектів, тоді як HDR використовується в кіноіндустрії. Розуміння цих класів дозволяє розробникам адаптувати зображення до конкретних задач, оптимізуючи як якість, так і продуктивність систем [5-6].

Таким чином, класи зображень відображають їхню різноманітність і специфіку, що визначається структурою, кольоровими моделями та призначенням. Ця класифікація є основою для ефективного використання зображень у сучасних технологіях.

1.3 Кодування зображень

Кодування зображень є фундаментальним етапом у процесі обробки та стиснення цифрових даних, що дозволяє представити візуальну інформацію у компактному та придатному для зберігання чи передачі вигляді. Цей процес включає перетворення піксельних даних у послідовність бітів, які можуть бути інтерпретовані комп'ютерними системами. Ефективність кодування напряму впливає на розмір файлів, швидкість завантаження та якість зображень, що робить його ключовим аспектом у сучасних технологіях [5].

Процес кодування починається з аналізу структури зображення, яке складається з пікселів, кожен із яких має числові значення, що відображають колір або яскравість. У кольорових зображеннях найчастіше використовується модель RGB, де кожен піксель кодується трьома компонентами – червоним, зеленим і синім. Кожна компонента зазвичай представлена 8 бітами, що дає 256 рівнів інтенсивності на канал, а загалом – мільйони можливих комбінацій. У монохромних зображеннях кодування обмежується одним значенням яскравості на піксель [6].

Для зменшення обсягу даних застосовуються різні методи кодування. Одним із базових підходів є бінарне представлення, де кожне значення пікселя перетворюється в біти. Наприклад, 8-бітне кодування дозволяє представити 256 відтінків, але для зображень із вищою роздільною здатністю чи глибшою кольоровою палітрою (наприклад, 16 біт на канал) обсяг інформації зростає, що вимагає додаткової оптимізації [6-7].

Існують різні стратегії кодування, залежно від цілей стиснення. Кодування ентропії, наприклад, використовує статистику даних для заміни частіше повторюваних значень на коротші біти. Алгоритм Хаффмана є класичним прикладом такого підходу: він будує дерево кодів, де найпоширеніші значення отримують найкоротші біти. Цей метод часто застосовується у форматах, таких як PNG, для стиснення без втрат.

Інший підхід – перетворення у частотну область, як у JPEG. Тут зображення розбивається на блоки, а дискретне косинусне перетворення (DCT) переводить просторові дані в частотні компоненти. Потім менш значущі високочастотні дані відкидаються, а решта кодується з меншою точністю. Це дозволяє досягти високого ступеня стиснення, хоча й із втратами якості.

Різні формати зображень використовують унікальні методи кодування. Формат JPEG застосовує DCT разом із квантуванням і кодуванням Хаффмана для стиснення з втратами, що робить його ідеальним для фотографій. PNG, навпаки, використовує DEFLATE – комбінацію LZ77 і кодування Хаффмана – для стиснення без втрат, зберігаючи прозорість і деталі. GIF обмежується 256 кольорами через використання індексованої палітри, що полегшує кодування, але знижує гнучкість [8].

Сучасні формати, такі як WebP, комбінують кілька методів: стиснення з втратами на основі VP8 і без втрат із використанням передових алгоритмів. Це дозволяє адаптувати кодування до конкретних потреб, балансуючи якість і розмір файлу. Такі інновації розширюють можливості обробки зображень у веброзробці та потоковому передаванні.

Одним із викликів у кодуванні є збереження балансу між розміром файлу та візуальною якістю. Надмірне стиснення може призвести до артефактів, таких як блоковість чи розмиття, тоді як недостатнє стиснення не вирішує проблему обсягу даних. Оптимізація включає вибір правильного алгоритму кодування та налаштування параметрів, таких як рівень квантування в JPEG [9].

Інший аспект – сумісність із пристроями. Старіші системи можуть не підтримувати нові формати, як-от AVIF, що вимагає використання універсальних методів кодування. Крім того, кодування має враховувати особливості людського сприйняття: наприклад, око менш чутливе до змін у високочастотних областях, що дозволяє ефективніше відкидати ці дані.

Розвиток штучного інтелекту відкриває нові горизонти для кодування зображень. Алгоритми машинного навчання можуть аналізувати зміст зображення та адаптувати кодування до його особливостей, наприклад, зберігаючи деталі на обличчях, але стискаючи однорідні фони. Такі підходи, як нейронні компресори, обіцяють підвищити ефективність і якість у майбутньому.

Отже, кодування зображень є складним і багатогранним процесом, що поєднує математичні методи, алгоритми стиснення та адаптацію до потреб користувачів. Його вдосконалення залишається актуальним завданням для забезпечення ефективної роботи сучасних цифрових систем [10].

У сучасних алгоритмах кодування також застосовується загальна схема (рисунок 1.1)

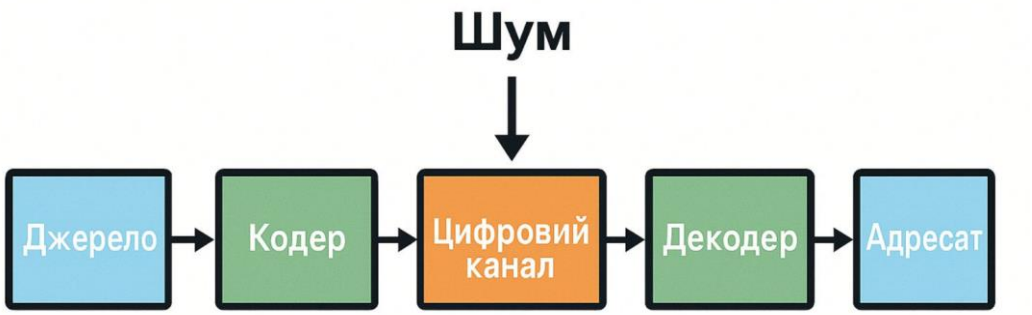


Рисунок 1.1 - Загальна схема процесу стиснення зображення

Крім того, Шенон довів, що завдання надійного зв'язку можна розкласти на дві підзадачі без применшення її ефективності. Ці дві підзадачі називаються кодуванням джерела і кодуванням каналу (рисунок 1.2).

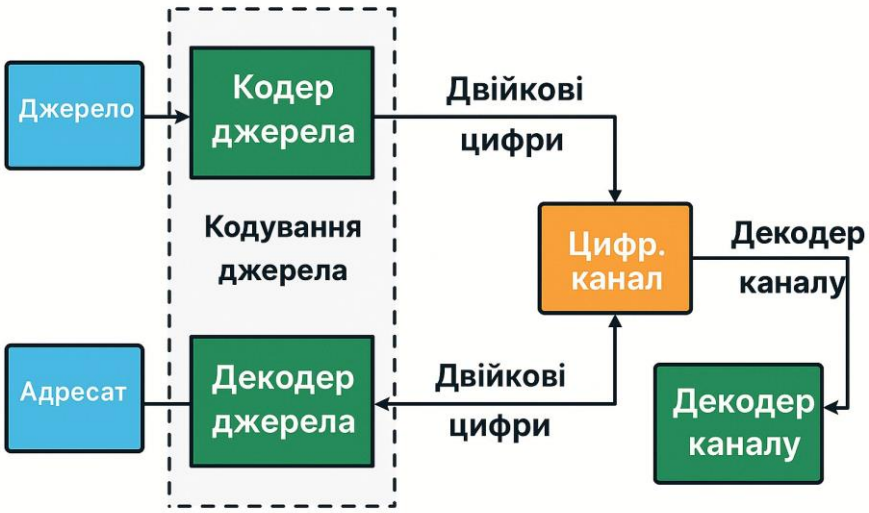


Рисунок 1.2 – Системи зв'язку з роздільним кодуванням

У використовуваних в даний час алгоритмах кодування також працює за загальною схемою, що називається предикативним кодуванням (рисунок 1.3)

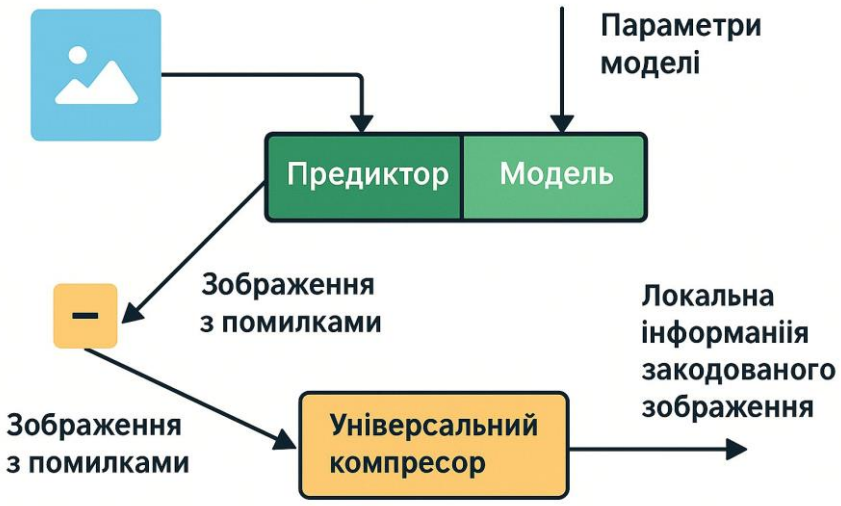


Рисунок 1.3 – Загальна схема предиктивного кодування

Предикативне кодування є одним із ефективних методів стиснення даних, який активно застосовується для обробки цифрових зображень. Цей підхід базується на передбаченні значень пікселів на основі їхніх сусідів, що

дозволяє видаляти надлишкову інформацію та зменшувати обсяг даних без значних втрат якості. На відміну від традиційних методів, таких як дискретне косинусне перетворення, предикативне кодування фокусується на просторовій кореляції між пікселями, що робить його особливо корисним для зображень із плавними градієнтами чи однорідними областями.

Основна ідея предикативного кодування полягає в тому, щоб передбачити значення поточного пікселя на основі вже відомих даних, таких як значення сусідніх пікселів. Наприклад, у двовимірному зображенні значення поточного пікселя може бути оцінене як середнє арифметичне чи лінійна комбінація значень верхнього, лівого та діагонального пікселів. Різниця між фактичним значенням і передбаченим (так звана залишкова помилка) кодується та зберігається. Оскільки ці різниці зазвичай мають менший діапазон значень, їх можна стиснути ефективніше, ніж усі дані зображення [11].

Процес включає кілька етапів. Спочатку визначається модель передбачення, яка може бути простою (лінійною) або складною (з урахуванням контексту). Потім обчислюється різниця між реальним і передбаченим значенням, яка називається предикторною помилкою. Нарешті, ці помилки кодуються за допомогою ентропійного кодування, наприклад, арифметичного чи кодування Хаффмана, щоб мінімізувати розмір файлу. Під час декодування зворотний процес дозволяє відтворити зображення з високою точністю.

Однією з головних переваг предикативного кодування є його здатність зберігати високу якість зображення при відносно низькому рівні стиснення. Цей метод особливо ефективний для зображень із природними сценами, де пікселі мають сильну кореляцію. Крім того, він не вимагає складних математичних перетворень, як у JPEG, що робить його менш ресурсоємним для реалізації на простих пристроях.

Проте є й обмеження. Ефективність цього методу залежить від однорідності зображення: у випадках із різкими переходами чи деталізованими областями (наприклад, текст або графіки) предикція стає менш точною, що може призвести до погіршення стиснення. Крім того, вибір оптимальної моделі передбачення є складним завданням, яке потребує врахування специфіки зображення [12].

Предикативне кодування використовується в кількох відомих форматах стиснення. Наприклад, у стандарті JPEG-LS (Lossless JPEG) цей метод поєднується з контекстним моделюванням для забезпечення стиснення без втрат. JPEG-LS є популярним у медичних системах, де важлива точність даних, таких як рентгенівські знімки. Також цей підхід застосовується в FAX-стандартах, де передача документів вимагає компактного представлення без втрати деталей.

Інший приклад – формат FLIF (Free Lossless Image Format), який використовує адаптивне предикативне кодування для покращення ефективності стиснення без втрат. Цей метод адаптується до локальних особливостей зображення, що дозволяє досягти кращих результатів порівняно з традиційними алгоритмами, такими як PNG [11].

Сучасні дослідження в галузі предикативного кодування спрямовані на інтеграцію машинного навчання. Нейронні мережі можуть аналізувати великі набори даних і створювати динамічні моделі передбачення, які враховують складні патерни в зображеннях. Це дозволяє підвищити точність предикції та зменшити розмір залишкових помилок, що, у свою чергу, покращує загальну ефективність стиснення.

Крім того, комбінація предикативного кодування з іншими методами, такими як вейвлет-перетворення, відкриває нові можливості для створення гібридних алгоритмів. Такі підходи можуть бути корисними для обробки зображень із високою роздільною здатністю чи динамічним діапазоном, що

стає дедалі актуальнішим із розвитком технологій відеострімінгу та віртуальної реальності.

Предикативне кодування є потужним інструментом для стиснення зображень, який базується на просторовій кореляції пікселів. Його гнучкість і ефективність роблять його цінним у різних галузях, від медицини до телекомунікацій. Незважаючи на певні обмеження, подальший розвиток цього методу, зокрема з використанням штучного інтелекту, обіцяє значні покращення в обробці та зберіганні цифрових даних [13].

Лінійне предиктивне кодування (LP-кодування) — це тип предиктивного кодування, за якого значення поточного символу (пікселя) прогнозується як лінійна комбінація раніше закодованих пікселів. Це можна виразити рівнянням (1.4):

$$P(i) = \sum_{j=i-1}^{i-n} k_j \, v(j) \tag{1.1}$$

де *P(i)* - значення, що передбачається, для і-го пікселя;

v(j) - відоме значення j-го пікселя;

k_j - деякі коефіцієнти, в загальному випадку не постійні;

n - кількість вже закодованих пікселів, використовуваних для прогнозу.

Лінійне предиктивне кодування (ЛПК) є одним із ефективних методів стиснення даних, який широко застосовується для обробки цифрових зображень, звукових сигналів та інших послідовностей даних. Ця техніка базується на передбаченні поточних значень даних на основі попередніх, використовуючи лінійну комбінацію. ЛПК особливо цінне у випадках, коли дані мають сильну кореляцію між сусідніми значеннями, як, наприклад, у градаціях пікселів зображення чи амплітуді звукового сигналу. Завдяки цьому метод дозволяє значно зменшити надлишкову інформацію, зберігаючи при цьому ключові характеристики оригінальних даних [12].

Основна ідея ЛПК полягає в тому, щоб передбачити наступне значення пікселя або зразка на основі лінійної комбінації попередніх значень. Для цього використовується математична модель, яка визначається коефіцієнтами передбачення. Наприклад, якщо розглядати зображення, значення поточного пікселя може бути обчислено як сума зважених значень сусідніх пікселів (зверху, зліва тощо) з додаванням залишкової помилки. Ця залишкова помилка, або різниця між фактичним і передбаченим значенням, кодується та зберігається, що й забезпечує стиснення.

Модель передбачення зазвичай будується на основі автокореляційного аналізу даних. Коефіцієнти обчислюються таким чином, щоб мінімізувати середньоквадратичну помилку між реальними та передбаченими значеннями. У найпростішому випадку лінійне передбачення може використовувати лише одне попереднє значення (наприклад, попередній піксель), але в складніших системах враховуються кілька сусідів, що підвищує точність.

У контексті зображень ЛПК часто застосовується як частина гібридних методів стиснення, таких як JPEG-LS – безвтратового аналога стандартного JPEG. У JPEG-LS лінійне передбачення комбінується з ентропійним кодуванням (наприклад, кодуванням Голомба-Риса), що дозволяє досягти високого ступеня стиснення. Алгоритм аналізує локальні особливості зображення, такі як градієнти чи однорідні області, і адаптує модель передбачення відповідно до них. Це робить ЛПК особливо ефективним для зображень із плавними переходами кольорів або текстур.

Процес включає кілька етапів: спочатку визначається контекст пікселя (його сусіди), потім обчислюються коефіцієнти передбачення, а згодом кодуються різниці. Завдяки цьому метод адаптується до змін у структурі зображення, що дозволяє зберігати деталі без значних втрат якості.

Однією з головних переваг ЛПК є його здатність працювати з різними типами даних, включаючи як монохромні, так і кольорові зображення. Метод не вимагає складних обчислень, що робить його придатним для реалізації на

пристроях із обмеженими ресурсами. Крім того, ЛПК забезпечує гнучкість, дозволяючи регулювати рівень стиснення залежно від потреб [11].

Проте є й певні обмеження. Ефективність ЛПК залежить від ступеня кореляції між даними. Якщо зображення має високу деталізацію чи різкі контрасти (наприклад, текст або краї), передбачення стає менш точним, що знижує ступінь стиснення. У таких випадках доцільніше комбінувати ЛПК із іншими техніками, такими як перетворення Фур'є чи вейвлети.

Нелінійні предиктори відіграють важливу роль у сучасних методах стиснення зображень, особливо в алгоритмах, які прагнуть досягти високої ефективності при збереженні візуальної якості. На відміну від лінійних предикторів, які базуються на простих математичних співвідношеннях між сусідніми пікселями, нелінійні предиктори використовують складніші моделі для прогнозування значень пікселів. Ці методи враховують не лише просторову близькість, але й локальні особливості зображення, такі як текстури, краї чи градієнти, що дозволяє покращити точність прогнозування та зменшити обсяг даних, необхідних для кодування.

Нелінійні предиктори базуються на аналізі складних залежностей між пікселями, які не можуть бути адекватно описані лінійними рівняннями. Вони часто використовують адаптивні алгоритми, які змінюють свою поведінку залежно від змісту зображення. Наприклад, у областях із різкими переходами (контрастами) предиктор може акцентувати увагу на градієнтах, тоді як у однорідних зонах – на середніх значеннях. Такий підхід дозволяє ефективніше видаляти надлишкову інформацію, що є ключовим для стиснення без втрат або з мінімальними втратами [10].

Одним із прикладів є метод MED (Median Edge Detector), який визначає медіанне значення пікселів у локальному вікні та коригує прогноз залежно від виявлення країв. Цей алгоритм аналізує різницю між горизонтальними та вертикальними градієнтами, обираючи оптимальний шлях для прогнозування. Інші нелінійні підходи можуть включати використання штучних нейронних

мереж або рекурсивних функцій, які адаптуються до статистик зображення в реальному часі.

Перевагою нелінійних предикторів є їхня здатність адаптуватися до складних структур зображення. Наприклад, у фотографіях із деталізованими текстурами (ліси, хмари) вони демонструють кращу ефективність порівняно з лінійними аналогами. Це дозволяє досягти вищого ступеня стиснення без значного погіршення якості. Крім того, такі методи є гнучкими, що дає змогу їх використовувати в різних форматах, таких як JPEG-LS або CALIC (Context-based Adaptive Lossless Image Codec).

Разом із тим, нелінійні предиктори мають і недоліки. Їхня обчислювальна складність вища, що може уповільнити процес кодування та декодування, особливо на пристроях із обмеженими ресурсами. Крім того, ефективність залежить від правильного налаштування параметрів, а помилки в адаптації можуть призвести до артефактів або зниження якості прогнозу [13].

Адаптивне кодування є сучасним підходом до стиснення даних, зокрема зображень, який динамічно адаптується до характеристик вхідного контенту. На відміну від традиційних статичних методів, де параметри кодування залишаються незмінними, адаптивне кодування аналізує структуру зображення в реальному часі, оптимізуючи процес стиснення залежно від локальних особливостей. Ця технологія набуває популярності завдяки своїй гнучкості та здатності забезпечувати високий ступінь компресії без значних втрат якості [14].

Адаптивне кодування базується на аналізі статистичних властивостей даних під час їхньої обробки. Алгоритми цього методу автоматично підбирають оптимальні параметри кодування, такі як довжина кодових слів чи рівень квантування, залежно від змісту зображення. Наприклад, однорідні ділянки (як небосхил чи гладкі поверхні) можуть стискатися з меншою деталізацією, тоді як області з різкими переходами (контури чи текст)

зберігають більше даних. Такий підхід дозволяє ефективно розподіляти ресурси, фокусуючись на збереженні візуально значущих елементів.

Одним із ключових етапів є використання адаптивних моделей ентропії, які оновлюють ймовірності символів на основі вже оброблених даних. Наприклад, алгоритми, подібні до арифметичного кодування, можуть адаптивно регулювати інтервали ймовірностей, що підвищує ефективність стиснення. Ця технологія особливо корисна для зображень із складною структурою, де статичні методи можуть бути менш продуктивними.

Однією з головних переваг адаптивного кодування є його здатність до самонавчання. Алгоритми можуть адаптуватися до специфічних характеристик зображення, таких як текстури, кольорові градієнти чи шум, що дозволяє досягати кращого балансу між розміром файлу та якістю. Це особливо актуально для сучасних форматів, таких як WebP чи AVIF, де адаптивні методи інтегруються для підвищення ефективності [15].

Крім того, адаптивне кодування зменшує потребу в попередній обробці зображень, що спрощує автоматизацію процесу. Наприклад, замість ручного налаштування параметрів стиснення користувач може покладатися на алгоритм, який самостійно визначає оптимальні настройки. Це економить час і ресурси, особливо в системах із великим обсягом даних, таких як хмарні платформи чи потокові сервіси.

Адаптивне кодування широко застосовується в комп’ютерно-інтегрованих системах, де важлива швидка обробка зображень. У веброзробці воно дозволяє оптимізувати завантаження сторінок, адаптуючи стиснення до умов мережі та пристрою користувача. У медичних системах адаптивні методи допомагають зберігати критичні деталі зображень, таких як рентгенівські знімки, одночасно зменшуючи обсяг даних для передачі.

Проте цей підхід має свої виклики. Адаптивні алгоритми вимагають більших обчислювальних ресурсів для аналізу та обробки даних у реальному часі. Це може бути проблемою для пристроїв із обмеженою потужністю, таких

як смартфони. Крім того, складність реалізації адаптивного кодування може ускладнити його інтеграцію в існуючі системи, вимагаючи додаткових зусиль для тестування та налаштування [12].

Розвиток адаптивного кодування тісно пов’язаний із технологіями штучного інтелекту. Алгоритми машинного навчання можуть прогнозувати оптимальні параметри стиснення на основі попереднього аналізу великої кількості зображень. Це відкриває можливості для створення ще ефективніших методів, які враховуватимуть не лише технічні характеристики, а й сприйняття зображень людиною.

У майбутньому адаптивне кодування може стати основою для нових стандартів стиснення, таких як наступні версії JPEG чи вдосконалені формати для віртуальної реальності. Завдяки своїй гнучкості та адаптивності ця технологія має потенціал значно вплинути на обробку мультимедійних даних у цифровому світі.

Таким чином, адаптивне кодування є потужним інструментом для оптимізації стиснення зображень, що поєднує динамічність і ефективність. Його розвиток обіцяє нові рішення для вирішення сучасних викликів у сфері обробки даних [13].

Алгоритми цієї групи можна поділити на два основні класи. Перший клас включає предиктори, що комбінують результати прогнозування статичних предикторів з адаптивно змінюваними вагами, тобто які працюють за формулою (1.2).

$$p(i) = \sum_{j=1}^{N} a_j \, p_j(i), \tag{1.2}$$

де $p(i)$ - значення поточного пікселя, передбачене адаптивним пре диктором;

$p_j(i)$ - значення поточного пікселя, передбачене j-м статичним пре диктором;

a_j - деякі змінні коефіцієнти.

Основним методом для визначення коефіцієнтів a_j є так званий метод «штрафування» предикторов [15]. Суть методу полягає в наступному. Нехай необхідно передбачити *i*-й піксель. Тоді для кожного предиктора розраховується штрафний коефіцієнт:

$$G_j = \sum_{k=i-1}^{i-L} \left| v(k) - p_j(k) \right|, \tag{1.3}$$

де L - кількість вже переглянутих пікселів, що враховується;

$v(k)$ - значення k-го пікселя;

$p_j(k)$ - значення k-го пікселя, передбачене j-м предиктором.

Після цього значення коефіцієнтів a_j розраховуються за формулою:

$$a_j = \frac{1/G_j}{\sum_{i=1}^{N} 1/G} \tag{1.4}$$

До другого класу відносяться предиктори, в яких для прогнозу використовується формула звичайного LP-кодування:

$$p(i) = \sum_{j=1}^{N} k_j \, v(i-j) \tag{1.5}$$

але, на відміну від нього, динамічно розраховуються коефіцієнти k_j.

1.4 Методи стиснення зображень

Кодування зображень є ключовим етапом у процесі їхньої обробки та стиснення, спрямованим на ефективне представлення даних для зберігання та передачі. Існує кілька основних методів кодування, які розрізняються за принципами роботи, ступенем стиснення та впливом на якість. Аналіз цих

методів дозволяє зрозуміти їхні сильні сторони, обмеження та сфери застосування [6].

Кодування Хаффмана належить до методів стиснення без втрат і базується на присвоєнні коротших бітових послідовностей найчастіше повторюваним символам. Алгоритм будується на створенні дерева кодів, де кожному символу (наприклад, значенню пікселя) відповідає унікальний код. Цей метод ефективний для зображень із великою кількістю однорідних областей, де певні значення повторюються часто.

Переваги:

- Забезпечує повне відновлення даних після декодування.
- Простота реалізації на апаратному рівні.

Недоліки:

- Ефективність залежить від розподілу даних; при рівномірному розподілі стиснення мінімальне.
- Не підходить для високого ступеня стиснення зображень із складними деталями.

Використовується в комбінації з іншими алгоритмами, наприклад, у форматі PNG, де воно інтегрується з DEFLATE для підвищення ефективності.

DCT є основою кодування з втратами, широко застосовуваним у форматі JPEG. Цей метод перетворює піксельні дані з просторової області в частотну, розбиваючи зображення на блоки (зазвичай 8x8 пікселів). Потім високі частоти, які менш помітні для людського ока, відкидаються, що дозволяє зменшити обсяг даних [7].

Переваги:

- Висока ступінь стиснення, особливо для фотографій із градієнтами.
- Швидке кодування та декодування за умови оптимальних налаштувань.

Недоліки:

- Виникнення артефактів (блоковість) при сильному стисненні.
- Втрата деталей, що може бути критичним для технічних зображень.

DCT є стандартом для вебграфіки та потокового передавання, де пріоритетом є швидкість і розмір файлу.

Вейвлет-перетворення використовується в сучасних форматах, таких як JPEG 2000. На відміну від DCT, яке працює з фіксованими блоками, вейвлет аналізує зображення в різних масштабах, розкладаючи його на низькочастотні та високочастотні компоненти. Це дозволяє досягти кращого балансу між якістю та розміром.

Переваги:

- Гнучкість у виборі ступеня стиснення, включаючи режим без втрат.
- Менше артефактів порівняно з DCT при високому стисненні.

Недоліки:

- Висока обчислювальна складність, що вимагає потужних ресурсів.
- Обмежена сумісність із застарілими пристроями.

Цей метод підходить для професійних застосувань, де потрібна висока якість, наприклад, у цифровій архітектурі чи картографії.

Арифметичне кодування є вдосконаленням кодування Хаффмана, яке представляє весь потік даних як єдине число з плаваючою комою. Замість присвоєння окремих кодів кожному символу, воно визначає ймовірності їхнього виникнення, що дозволяє досягти кращого стиснення.

Переваги:

- Вища ефективність порівняно з кодуванням Хаффмана.
- Підходить для зображень із складними розподілами даних.

Недоліки:

- Вимагатиме точного обчислення ймовірностей, що може бути складним.
- Потребує більше обчислювальних ресурсів для реалізації.

Використовується в гібридних форматах, таких як JPEG 2000, для підвищення компресії.

Кожен метод кодування має свої особливості. Кодування Хаффмана та арифметичне кодування оптимальні для стиснення без втрат, але їхня

ефективність залежить від однорідності даних. DCT і вейвлет-перетворення домінують у стисненні з втратами, де DCT швидше, а вейвлет забезпечує кращу якість. Вибір методу залежить від цільового використання: веброзробка віддає перевагу JPEG із DCT, тоді як професійні системи обирають JPEG 2000 із вейвлетами. Розвиток технологій, зокрема інтеграція ШІ, обіцяє створення нових методів кодування, які поєднуватимуть високу ефективність із збереженням деталей [9].

Стиснення зображень є фундаментальним процесом у цифрових технологіях, спрямованим на зменшення обсягу даних для ефективного зберігання та передачі. Базові стратегії стиснення базуються на аналізі структури зображень та видаленні надлишкової або менш значущої інформації. Ці підходи поділяються на кілька ключових категорій, кожна з яких має свої принципи та методи реалізації. Розуміння цих стратегій дозволяє оптимізувати продуктивність комп'ютерно-інтегрованих систем і адаптувати їх до різних потреб [8].

Ентропійне кодування є однією з основних стратегій стиснення, яка фокусується на видаленні надлишковості в даних. Цей метод базується на теорії інформації, розробленій Клодом Шенноном, і передбачає присвоєння коротших кодів частіше повторюваним символам, а довших – рідкісним. Таким чином досягається компактніше представлення даних без втрати їхньої змістової цінності.

- Принцип роботи: Алгоритми, такі як кодування Хаффмана або арифметичне кодування, аналізують частоту появи піксельних значень або їхніх груп у зображенні. Наприклад, у зображенні з великими однорідними областями повторювані значення кодуються коротшими бітами.
- Переваги: Не впливає на якість зображення, що робить цей метод ідеальним для стиснення без втрат.

- Застосування: Використовується в форматах, таких як PNG (разом із DEFLATE) або TIFF.
- Обмеження: Ефективність залежить від однорідності даних; для складних зображень із великою кількістю унікальних значень ступінь стиснення може бути низьким.

Трансформаційне стиснення базується на перетворенні зображення з просторової області в іншу, де дані можна ефективніше стискати. Ця стратегія використовує математичні перетворення для виділення частотних компонентів, що дозволяє видаляти менш значущу інформацію [9].

- Принцип роботи: Найпоширенішим є дискретне косинусне перетворення (DCT), яке застосовується в JPEG. Зображення розбивається на блоки (зазвичай 8x8 пікселів), а потім перетворюється в частотну область. Низькі частоти, що відповідають основним деталям, зберігаються, тоді як високі частоти (дрібні деталі) можуть бути відкинуті.
- Переваги: Дозволяє досягти високого ступеня стиснення, особливо для фотографій із плавними градієнтами.
- Застосування: Широко використовується в мультимедійних форматах, таких як JPEG, MPEG і WebP.
- Обмеження: При занадто сильному стисненні виникають артефакти, такі як блоковість або розмиття.

Прогнозувальне кодування базується на передбаченні значень пікселів на основі їхніх сусідів. Ця стратегія ефективна для зображень із локальною кореляцією, де значення пікселів у близьких областях подібні [10].

- Принцип роботи: Алгоритм аналізує попередні пікселі (наприклад, у рядку або блоці) і обчислює різницю між фактичним значенням і прогнозованим. Ці різниці зазвичай менші за вихідні значення і кодуються більш компактно.

- Переваги: Забезпечує стиснення без втрат і добре працює з градієнтами чи текстурами.
- Застосування: Використовується в форматах, таких як GIF або у комбінації з іншими методами у TIFF.
- Обмеження: Менш ефективне для зображень із різкими змінами або шумом.

Ця стратегія передбачає використання математичних або статистичних моделей для опису структури зображення. Вона часто застосовується у векторних форматах або сучасних алгоритмах із використанням штучного інтелекту [11].

- Принцип роботи: Замість збереження всіх пікселів зображення кодуються параметри моделі, які можуть відтворити його. Наприклад, у векторних зображеннях (SVG) використовуються криві Безьє для опису контурів.
- Переваги: Дозволяє безкінечне масштабування без втрати якості, що корисно для логотипів чи ілюстрацій.
- Застосування: Популярне в графічному дизайні (SVG) і перспективно для AI-оптимізованих форматів, таких як JPEG XL.
- Обмеження: Не підходить для фотографій із складними деталями через складність моделювання [12].

Кожна стратегія має свої сильні сторони залежно від типу зображення та цілей стиснення. Ентропійне кодування ідеально для однорідних даних і стиснення без втрат, тоді як трансформаційне стиснення домінує в мультимедії завдяки високому ступеню компресії. Прогнозувальне кодування підходить для градієнтів, а моделі – для векторної графіки. Вибір залежить від балансу між якістю, розміром файлу та обчислювальними ресурсами.

Серед викликів – адаптація стратегій до різноманітних зображень, таких як HDR або 3D-графіку, де потрібна висока деталізація. Перспективи пов’язані з інтеграцією машинного навчання, яке може адаптивно обирати найкращу

стратегію для кожного зображення. Наприклад, нейронні мережі вже використовуються для прогнозування та оптимізації стиснення в форматах на кшталт AVIF.

Базові стратегії стиснення, такі як ентропійне кодування, трансформаційне стиснення, прогнозувальне кодування та методи на основі моделей, формують основу сучасних технологій обробки зображень. Їхнє розумне застосування дозволяє оптимізувати ресурси, зберігаючи необхідну якість, і відкриває можливості для подальшого розвитку в епоху штучного інтелекту [9].

Алгоритми компресії зображень відіграють вирішальну роль у сучасних інформаційних системах, де ефективне зберігання та передача даних є пріоритетом. Вимоги до цих алгоритмів значною мірою залежать від специфіки додатків, у яких вони застосовуються. Від веброзробки до медичних систем – кожен напрямок має унікальні потреби, що впливають на вибір методу стиснення, його швидкості, якості результату та сумісності з апаратним забезпеченням.

Першою ключовою вимогою є ступінь стиснення. У додатках, таких як вебсайти чи мобільні платформи, де швидкість завантаження є критичною, алгоритми повинні максимально зменшувати розмір файлів. Наприклад, для потокового відео чи соціальних мереж формат JPEG або WebP з високим рівнем стиснення з втратами є оптимальним вибором, оскільки користувачі готові пожертвувати частиною якості заради швидкості. Натомість у професійних сферах, таких як архітектура чи дизайн, де деталізація важлива, потрібні методи з мінімальними втратами або навіть без втрат, як PNG [16].

Другою важливою характеристикою є швидкість обробки. Додатки в реальному часі, наприклад, відеоконференції чи ігри, вимагають алгоритмів із низькою затримкою при кодуванні та декодуванні. Це означає, що обчислювальна складність повинна бути оптимізованою для роботи на пристроях із обмеженими ресурсами, таких як смартфони чи вбудовані

системи. Алгоритми, такі як JPEG, завдяки простоті дискретного косинусного перетворення (DCT), добре справляються з цією задачею, тоді як складніші методи, як у JPEG 2000, можуть бути менш придатними.

Якість зображення після компресії є ще одним визначальним фактором. У медичних додатках, де аналіз зображень (наприклад, рентгенівських знімків) вимагає точності, алгоритми без втрат, такі як DICOM, є стандартом. Навпаки, у рекламних платформах чи онлайн-галереях допустимі невеликі артефакти, якщо це дозволяє значно скоротити обсяг даних. Баланс між якістю та розміром часто досягається через гнучкі налаштування, як у форматі WebP, де користувач може регулювати рівень стиснення [15].

У веброзробці пріоритет віддається сумісності з браузерами та швидкості завантаження. Формати, такі як WebP чи AVIF, підтримують сучасні платформи і забезпечують високу ефективність стиснення. Додатки також потребують адаптивності до різноманітних екранів, що вимагає алгоритмів, які зберігають прийнятну якість при масштабуванні. Наприклад, стиснення з втратами може бути налаштоване так, щоб уникати помітних дефектів на дисплеях із високою роздільною здатністю.

У медичних системах вимоги значно жорсткіші. Алгоритми повинні гарантувати збереження всіх деталей, навіть найдрібніших, оскільки від цього залежить діагностична точність. Формати без втрат, такі як TIFF чи DICOM, використовуються для зберігання зображень, а стиснення з втратами застосовується лише з обережністю та спеціальними профілями, затвердженими медичними стандартами. Крім того, важлива безпека даних, що може вимагати інтеграції шифрування з процесом компресії [12].

Для мобільних додатків ключовими є енергоспоживання та обчислювальна ефективність. Алгоритми повинні бути оптимізованими для роботи на процесорах із низькою потужністю, а також адаптованими до нестабільних мереж. Наприклад, стиснення з втратами у форматі JPEG є популярним завдяки швидкості, але новіші формати, такі як AVIF, можуть

бути обмеженими через вимоги до обчислювальних ресурсів на старих пристроях.

Сумісність із апаратним забезпеченням є важливим для вбудованих систем, таких як камери чи промислові пристрої. Алгоритми компресії повинні підтримуватися апаратними кодеками, щоб зменшити навантаження на центральний процесор. Наприклад, багато сучасних чипів мають вбудовану підтримку JPEG, що прискорює обробку в реальному часі.

Гнучкість налаштувань дозволяє адаптувати алгоритми до конкретних задач. У додатках із динамічним контентом, таких як онлайн-редактори зображень, користувачі можуть потребувати вибору між стисненням із втратами чи без втрат. Це вимагає від алгоритмів модульності та можливості регулювання параметрів, таких як рівень якості чи розмір блоку для обробки.

Стійкість до помилок є критичною для додатків, що працюють у нестабільних умовах, наприклад, у супутникових системах чи бездротових мережах. Алгоритми повинні включати механізми корекції помилок або сегментації даних, щоб уникнути повної втрати зображення при пошкодженні частини файлу [11].

Зростання обсягів даних і розвиток технологій, таких як віртуальна реальність чи високодинамічний діапазон (HDR), ставлять нові вимоги до алгоритмів компресії. Додатки майбутнього потребуватимуть методів, які поєднують високу ефективність стиснення з підтримкою складних форматів. Штучний інтелект уже використовується для адаптивного стиснення, де алгоритми аналізують зміст зображення для оптимального видалення даних. Однак це підвищує обчислювальні вимоги, що може бути проблемою для старих пристроїв.

Отже, вимоги додатків до алгоритмів компресії охоплюють широкий спектр аспектів – від ступеня стиснення та швидкості до якості та сумісності. Розуміння цих потреб дозволяє розробляти рішення, які оптимізують

використання ресурсів і відповідають специфіці кожної сфери застосування, від вебплатформ до медичних систем [14].

1.5 Висновки до розділу 1

У першому розділі проаналізовано методи кодування зображень у сучасних системах обробки відео, а також класифікацію відеоматеріалів і програмного забезпечення, що допомогло визначити вимоги до алгоритмів кодування. Ці особливості детально розглядатимуться в наступному розділі дипломної роботи. Зокрема, досліджено ключові підходи до стиснення відеоданих, зокрема внутрішньокадрове та міжкадрове кодування, їхні сильні та слабкі сторони щодо ефективності обробки й передачі відео. Особливу увагу зосереджено на властивостях відеоформатів і стандартів, які впливають на вибір кодувальних методів. Цей аналіз заклав основу для подальшого вивчення та створення вдосконалених алгоритмів, здатних гарантувати високу якість зображень за умови зменшення обсягу даних і зниження навантаження на ресурси систем обробки відео.

2 АЛГОРИТМ КОДУВАННЯ ВІДЕОЗОБРАЖЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ СТИСНЕННЯ У СИСТЕМАХ ОБРОБКИ ВІДЕОІНФОРМАЦІЇ

2.1 Удосконалений метод групового кодування

Типовим прикладом алгоритмів групового кодування є RLE [12]. У цьому методі зазвичай усі рядки зображення впорядковуються в одну суцільну послідовність (рисунок 2.1).

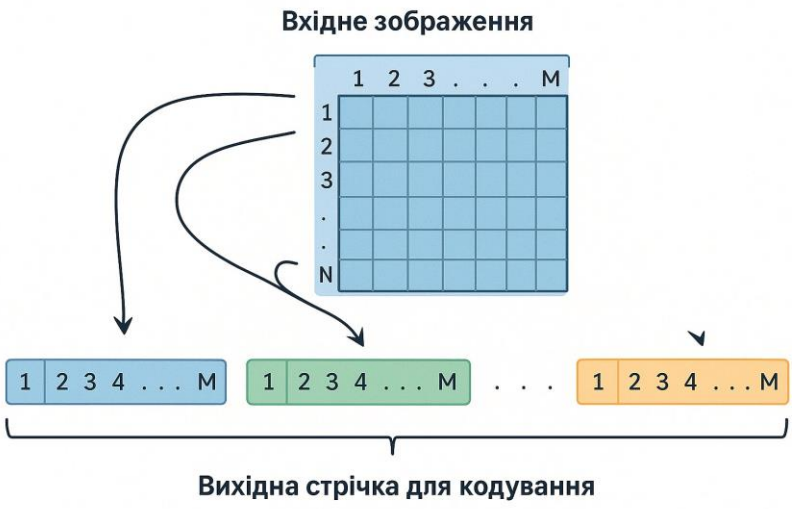


Рисунок 2.1 – Формування стрічки для групового кодування

Після чого застосовуються різні способи усунення надлишкових повторів символів або чисел шляхом заміни їх на стиснуті послідовності.

В алгоритмі RLE послідовність складається з двох байтів: перший байт містить один біт, що вказує на наявність повторюваних значень у стрічці кодування, а решта 7 біт використовуються для позначення кількості повторів (максимум 128). Другий байт представляє значення кольору пікселя, що повторюється в стрічці. Для відтворення двовимірної структури зображення

під час декодування необхідно також зберігати додаткові байти, які визначають розміри зображення *MxN* (рисунок 2.2).

Рисунок 2.2 – Процес декодування зображення

Отже, в алгоритмі групового кодування RLE, для кожної повторюваної або неповторюваної послідовності виділяється по одному додатковому байту, до яких додаються байти для збереження розмірності зображення (2.1):

$$redudancy = \sum_{i=1}^{K} R_i + \sum_{j=1}^{L} \bar{R}_j + S_{NxM}, \quad (2.1)$$

де *redundancy* – надлишковість байт;

R_i – кількість повторюючих послідовностей;

$R\,j$ - кількість неповторюючих послідовностей;

S_{NxM} – кількість байт відведених на визначення розмірності зображення із N рядками та M стовбцями.

Оскільки для кодування повторюваних або унікальних фрагментів використовується 7 бітів, то це зумовлює обов’язкове додавання спеціальних

байтів приблизно кожні 128 елементів. У випадку повної відсутності однакових пікселів такі вставки створюють додаткове навантаження, що може призвести до збільшення обсягу даних після стискання. Це ж стосується і повторів, якщо довжина послідовностей перевищує 128 символів — з’являються зайві службові байти. Такий підхід іноді призводить до ефекту зворотного стискання, коли розмір архіву перевищує початковий. Подібне може траплятися і при частих змінах між повтореннями та унікальними даними.

Щоб покращити ефективність традиційного групового стискання, запропоновано вдосконалений варіант, орієнтований на підвищення щільності збереження.

Суть методу полягає у збільшенні блоків однакових елементів та зменшенні надмірності завдяки використанню повторюваних кольорів. На відміну від класичного RLE, де створюється єдиний потік, новий підхід аналізує зображення у двох площинах — рядках і стовпцях — для виявлення зон із повторюваними значеннями. Замість зберігання кожного фрагмента окремо, метод шукає великі прямокутні області з ідентичним кольором і зберігає їх як єдину сутність. Це дозволяє досягти вищого рівня стискання, особливо у випадках, коли кадри містять великі одноколірні ділянки, характерні для зображень реальних об’єктів.

Для реалізації запропонованого методу групового стискання зображень пропонується використовувати спеціальну структуру ознак, яка дозволяє ідентифікувати однакові блоки пікселів (рисунок 2.3).

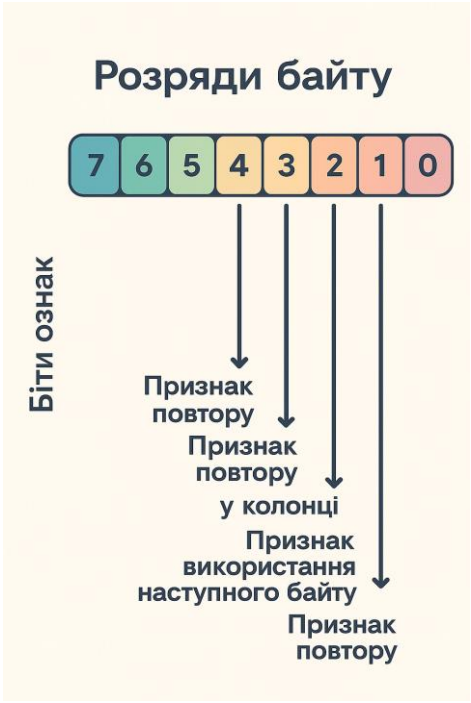


Рисунок 2.3 – Значення розрядів одного байту для позначення признаків повтору

Для позначення повторюваних груп пікселів у запропонованому алгоритмі використовується один байт, де старші біти виконують функцію ознак:

– 1-й біт вказує на наявність або відсутність повторень кольору: якщо дорівнює 1 — повторення є, якщо 0 — немає.
– 2-й біт активується лише у випадку, коли перший біт дорівнює 1, і сигналізує про наявність вертикальних (стовпчикових) повторів: значення 1 означає їх присутність, а 0 — відсутність.
– 3-й біт також активний при першому бітові, рівному 1, і використовується для вказання повторів у горизонтальному (рядковому) напрямку: 1 — повторення є, 0 — відсутні.
– 4-й біт повідомляє про використання додаткового байта у випадках, коли кількість повторів перевищує 8.

Біти з 5-го по 8-й несуть інформаційне навантаження.

Якщо ж ідентифікується область зображення, що включає кілька рядків і стовпців із повторюваними значеннями, тоді застосовується розширена структура ознак, представлена на рисунку 2.4.

Рисунок 2.4 – Структура ознак для позначення групового повторення прямокутної форми

У цьому випадку зона повторюваних кольорів на зображенні має прямокутну форму. Для її кодування застосовуються три байти: перший байт містить ознаки повтору в старших 4 бітах та частину даних про кількість повторень, другий байт уточнює кількість повторів, третій байт вказує колір, який повторюється.

У даній ситуації область, де повторюються кольори, має форму прямокутника. Для її кодування застосовується трибайтна структура:

- перший байт містить службові біти, що описують тип повторення,
- другий байт задає розміри області (кількість рядків і колонок),
- третій байт визначає сам колір, який повторюється у вказаній ділянці.

Перший біт встановлюється в 1, що вказує на наявність повторення кольорів. Другий і третій біти також дорівнюють одиниці, оскільки повтори охоплюють кілька рядків і стовпців прямокутної області. Четвертий біт також має значення 1, бо для визначення розмірів області потрібно додатково використати ще один байт.

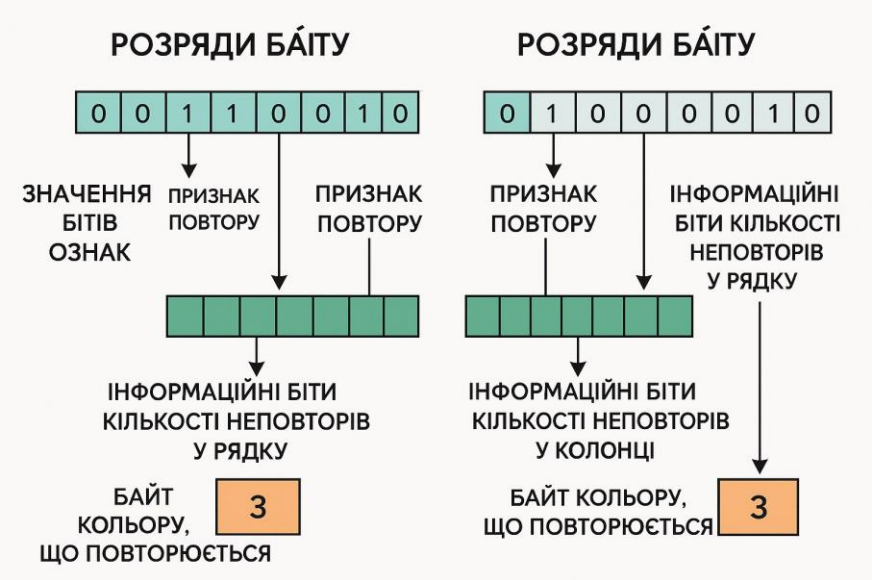
Наступні 6 інформаційних бітів першого байта визначають кількість колонок у прямокутнику, а шість інформаційних бітів другого байта задають кількість рядків (на рисунку зображено область із п’ятьма рядками). Отже, максимальні розміри області, які можна закодувати цією структурою, складають до 64 рядків і 64 колонок (тобто до 4096 пікселів). Якщо область перевищує ці розміри, її потрібно поділити на кілька менших частин.

Останній байт містить значення кольору пікселів, що повторюються в зазначеній прямокутній зоні.

Максимальне стискання при такому підході досягається, коли вся область займає 4096 пікселів, і для її зберігання потрібно лише 3 байти, тобто коефіцієнт стискання становить $K_{стис}=3/4096=0,0007$. Водночас, мінімальний доцільний розмір області для застосування цього методу має бути не меншим за 4 пікселі (наприклад, 2x2), оскільки для кодування все одно потрібно 3 байти. У такому випадку коефіцієнт стискання становитиме $K_{стис}=3/4=0,75$, що все ще означає вигідність стиснення, оскільки коефіцієнт менший за 1.

Інший можливий варіант передбачає, що повторювана послідовність розміщується в межах лише одного рядка або одного стовпця (рисунок 2.5).

На відміну від стискання прямокутних ділянок зображення, кодування повторюваних пікселів, що розташовані в одному рядку або стовпці, передбачає гнучке використання обсягу даних — від двох до трьох байтів, залежно від довжини послідовності.



а) б)

Рисунок 2.5 – Значення розрядів при груповому кодуванні піксеслів: а) - колонки (б) - рядка

Якщо кількість повторів не перевищує 16 елементів, достатньо закодувати цю інформацію в 4 молодших бітах першого байта, а другий байт використовувати для збереження кольору повторюваного пікселя. У цьому випадку біт, що вказує на потребу в додатковому байті для зберігання довжини, дорівнює нулю.

Якщо ж довжина групової послідовності перевищує 15, але не перевищує 4095 (тобто в межах 12 біт), розбивати її на менші фрагменти недоцільно. Замість цього варто скористатися ще одним байтом для точного збереження кількості повторень.

Для кодування такої послідовності використовується три байти. Варто зазначити, що біт, який сигналізує про використання додаткового байта для вказання кількості повторів, встановлений у значення 1.

Таким чином, застосування цього підходу до групового стискання дозволяє зменшити послідовність обсягом 4096 пікселів до лише 3 байтів. Тобто коефіцієнт стиснення рівний $K_{стис}=3/4096=0,0007$. Слід також підкреслити, що стискання є доцільним лише для послідовностей, які містять

понад два пікселі, оскільки мінімально для групового кодування використовується два байти. Таким чином, при кодуванні трьох пікселів коефіцієнт стиснення становитиме $K_{стис}=2/3=0,6667$, що свідчить про вищу ефективність у порівнянні зі стисканням прямокутних областей. Це забезпечується завдяки адаптивному використанню байтів і є співмірним із результатами, які дає стандартний алгоритм RLE.

Ще одним варіантом застосування запропонованого методу групового кодування є його використання для кодування неповторюваних послідовностей. Такі фрагменти зображення можуть охоплювати як окремі пікселі, так і великі ділянки, що не мають повторів. У цьому випадку доцільно використовувати змінну кількість байтів залежно від довжини послідовності.

Таким чином, для збереження послідовностей, у яких кольори не повторюються, можна застосовувати один або два байти — залежно від кількості пікселів, після яких розміщується відповідна послідовність байтів із кольоровими значеннями. У такому випадку надлишкові дані становитимуть лише один або два байти на всю неповторювану послідовність.

Існує можливість ефективного стискання надлишкових даних, розташованих у стовпці (зокрема, колонка 28, де всі пікселі мають колір зі значенням 83). У випадку класичного алгоритму RLE така вертикальна послідовність однакових пікселів не підлягає стисканню, оскільки при перетворенні зображення у лінійну стрічку ця колонка втрачає свою цілісність — пікселі розміщуються окремо й розглядаються як незалежні елементи. У результаті жодна компресія не буде застосована.

На відміну від цього, у запропонованому алгоритмі аналіз виконується як по рядках, так і по стовпцях, що дозволяє виявляти надлишкові елементи незалежно від їхнього розташування. Таким чином, вказана колонка з 8 пікселів буде стиснута до 2 байтів, тобто відбудеться зменшення обсягу даних у 4 рази, тоді як традиційний RLE не забезпечить жодного стискання.

У випадках, коли можлива компресія як у рядковому, так і у стовпчиковому напрямках, алгоритм автоматично обирає той варіант, у якому площа однорідної області буде максимальною.

Цей підхід добре ілюструється на ділянці зображення між 29 і 33 колонками, виділеній товстою лінією на рисунку 2.3. Тут спостерігається групова послідовність пікселів із однаковим кольором (значення 86). При використанні стандартного алгоритму RLE дев’ять горизонтальних рядків (за винятком першого) будуть окремо кодуватися з додаванням службового байта для кожної повторюваної послідовності. Таким чином, із загальної кількості 37 пікселів буде стиснуто лише 36, і в результаті отримаємо 16 байтів після кодування. В цьому випадку коефіцієнт стиснення $K_{стис\_RLE} = 16/37 = 0,4324$.

Під час використання запропонованого методу групового стиснення спочатку визначається найбільша прямокутна область з однорідним кольором, яка може бути вписана в задану ділянку зображення. Після цього така зона розбивається на дві частини. У результаті без стискання залишаються лише два пікселі: один у першому рядку послідовності та один на правому краю прямокутника.

Решта 35 пікселів піддаються стисканню у два етапи, внаслідок чого їх представлення займає лише 5 байтів. Таким чином, досягається коефіцієнт стискання $K_{стис\_new} = 5/35 = 0,1429$, що майже втричі ефективніше порівняно з результатом, який забезпечує стандартний алгоритм.

Суть запропонованого методу групового кодування полягає у послідовному формуванні стрічки кодів, які відображають як повторювані, так і унікальні послідовності кольорів зображення — від першого до останнього пікселя (рисунок 2.6).

Іншими словами, процес кодування відбувається циклічно, у результаті чого створюється масив байтів, обсяг якого менший за розмір оригінального зображення.

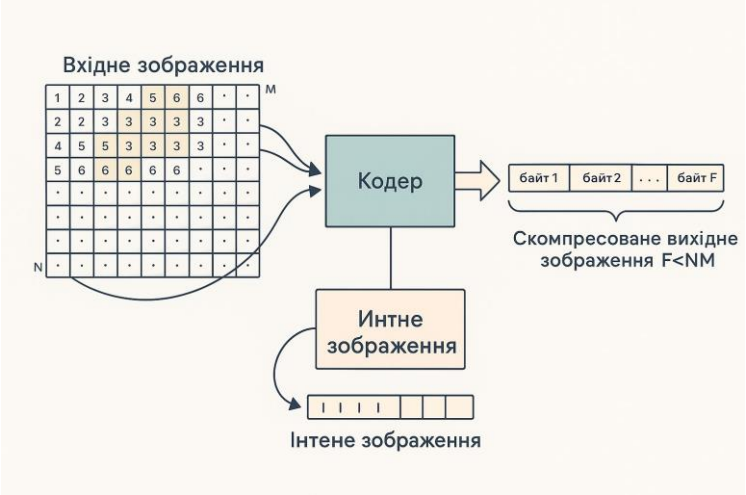


Рисунок 2.6 – Узагальнена схема запропонованого алгоритму групового кодування

Отже, в цьому параграфі було представлено вдосконалений підхід до стиснення відеозображень на основі групового кодування. Завдяки аналізу піксельних послідовностей у двох напрямках — по рядках і стовпцях — запропонований метод забезпечує ефективніше стискання, з меншим коефіцієнтом, ніж традиційний алгоритм RLE.

У наступному параграфі розглядається структура алгоритму реалізації даного способу групового кодування.

2.2. Алгоритм кодування групових послідовностей запропонованого вдосконаленого підходу

Кодер здійснює перетворення вхідного (некомпресованого) зображення у масив байтів, що кодують це зображення. Якщо розмір вихідного масиву менший за розмір початкового зображення, то це свідчить про стиснення вхідної інформації; в іншому випадку відбувається її розширення. Детальніший аналіз величин кодування зображень за допомогою запропонованого методу надано в третьому розділі дипломної роботи.

Важливим аспектом є також можливість відтворення початкового зображення без спотворень, використовуючи стиснуту стрічку, отриману в результаті групового кодування (рисунок 2.7).

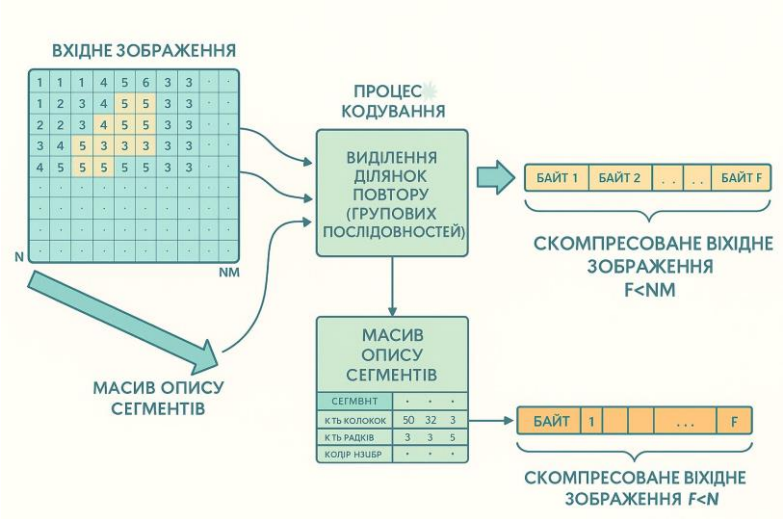


Рисунок 2.7 – Узагальнена структура кодера

На вхід кодера подається початкове зображення у вигляді матриці, де кожен елемент відповідає значенню яскравості в градаціях сірого, з розрахунку один байт на піксель. Таким чином, можливі значення кольору варіюються від 0 до 255. Якщо ж необхідно обробляти кольорові відеозображення у форматі RGB, попередньо потрібно виконати розділення на окремі складові: червону, зелену та синю. У цьому випадку кожен колірний канал обробляється окремо, що може забезпечити ще кращий ступінь стискання.

Першим етапом роботи кодера є виявлення ділянок зображення, де спостерігається повторення одного й того ж кольору. Аналізу підлягають лише ті області, розмір яких відповідає певним критеріям: від 4 елементів і більше — до 64×64 пікселів для прямокутних зон, або ж від 3 до 4096 елементів у разі лінійних ділянок по рядках чи стовпцях. Після цього формується масив, у якому описано знайдені сегменти: зазначено їхній порядковий номер (починаючи з 501, щоб уникнути збігу з діапазоном кольорів, який обмежується 256 значеннями), кількість рядків, колонок і значення кольору.

У самій матриці зображення пікселі, що входять до виявлених сегментів, замінюються на відповідні маркери. Надалі кодер працює з модифікованим зображенням та описом сегментів.

На наступному етапі кодер аналізує кожен піксель. Якщо піксель містить маркер групової послідовності, виконується її кодування, а оброблені пікселі помічаються спеціальним значенням (наприклад, максимально можливим цілим числом), щоб уникнути повторної обробки. Якщо ж піксель не є частиною жодної групової послідовності, він кодується як унікальний (неповторюваний). У результаті формується вихідна стрічка — послідовність байтів, що представляє закодоване зображення.

На завершальному етапі до цієї стрічки додається службова інформація про розмір вхідного зображення. Для цього резервуються 4 байти: два на кількість рядків і два — на кількість колонок. Це дозволяє коректно відновити зображення при декодуванні, адже 16-розрядне представлення кожного параметра дає змогу адресувати до 65 536 рядків і стовпців — що повністю покриває потреби сучасних відеоформатів. Після завершення роботи кодера формується єдина послідовність байтів, яка може бути передана через канали зв’язку або використана для подальшої обробки відеозображень.

Більш детально процес функціонування кодера можна описати у вигляді послідовності кроків, зображеної в додатку В.

Одним із етапів роботи кодера є виявлення областей зображення, які містять повторювані послідовності однакових кольорів (групові повтори).

Основна ідея виділення групових послідовностей – знаходження ділянок зображення, що містять 4-зв’язні послідовності пікселів з однаковими кольорами. Чотири-зв’язність означає зв’язок пікселів по горизонталі або вертикалі.

Пошук є ітераційним: зображення сегментується за кольорами (0–255). Кожен сегмент аналізується на 4-зв’язність пікселів і розмірність. Сегменти

одного кольору можуть мати складну форму, тому їх ділять на частини, кожна з яких формує окрему групову послідовність.

Після цього етапу ділянка з найбільшою площею визначається як групова послідовність, а аналіз сегмента триває для пошуку інших групових послідовностей, виключаючи вже вибрану. Процес є ітераційним для всіх сегментів зображення. Однією з найвимогливіших процедур кодера є кодування групових послідовностей або послідовностей без повторів. Схема роботи цього блоку кодера наведена на рисунках 2.8 та 2.9.

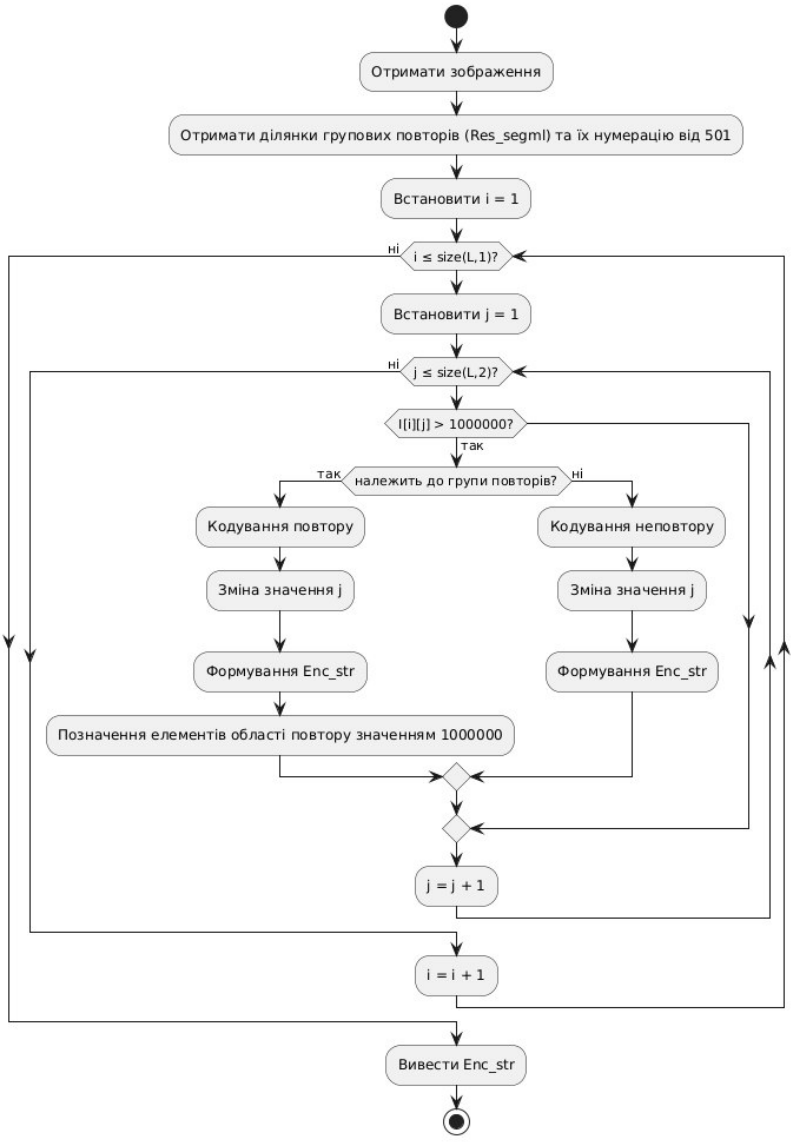


Рисунок 2.8 – Діаграма узагальненої схеми роботи кодера

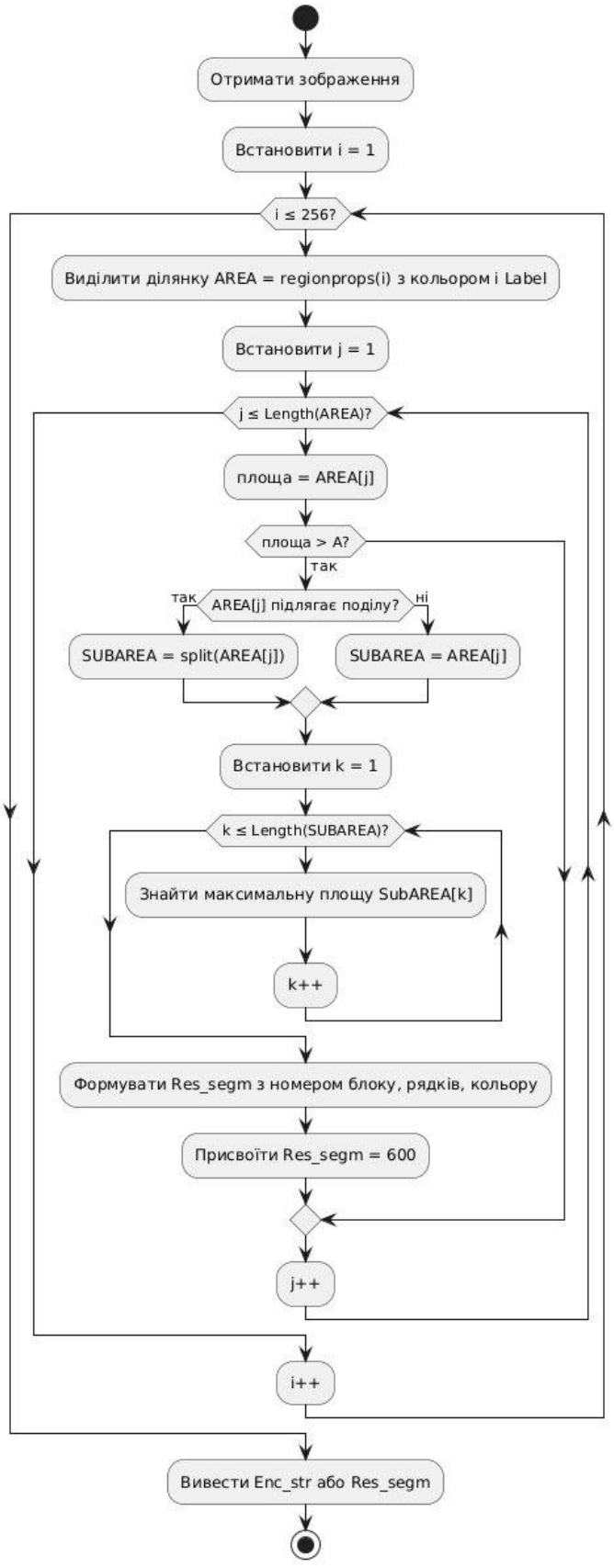


Рисунок 2.9 – Діаграма роботи процедури виділення групових послідовностей

2.3 Висновки до розділу 2

У другому розділі представлено новий алгоритм кодування цифрових відеозображень, який забезпечує більш ефективне стиснення інформації порівняно з відомими методами групового кодування завдяки застосуванню покращеного механізму обробки груп пікселів. Запропоновано нові підходи до кодування, які, на відміну від існуючих, зменшують надлишковість у масивах відеозображень шляхом аналізу групових послідовностей пікселів з однаковими кольоровими значеннями в двох взаємно перпендикулярних напрямках. Розроблено алгоритмічне забезпечення роботи кодера, що дозволяє реалізувати алгоритм без втрат даних, незалежно від програмної чи апаратної платформи та мов програмування. Також створено програмну реалізацію декодера, яка забезпечує точне відновлення закодованої послідовності чисел у вихідне зображення, гарантуючи відсутність втрат і зберігаючи незалежність від конкретного середовища реалізації.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОДУВАННЯ ВІДЕОЗОБРАЖЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ СТИСНЕННЯ У СИСТЕМАХ ОБРОБКИ ВІДЕОІНФОРМАЦІЇ

3.1 Вибір та обґрунтування мови програмування

Розробка програмного забезпечення для кодування відеозображень з метою автоматизації процесу стиснення у системах обробки відеоінформації є складним завданням, що потребує ефективних інструментів для аналізу, обробки та оптимізації даних. Одним із ключових етапів цього процесу є вибір відповідної мови програмування, яка забезпечить високу продуктивність, зручність розробки та відповідність специфіці задачі. У контексті даної теми мова програмування MATLAB була обрана як основний інструмент, і це рішення ґрунтується на її унікальних характеристиках та перевагах.

MATLAB є потужним середовищем для числових обчислень, аналізу даних та обробки зображень і відео, що робить його ідеальним для розробки програмного забезпечення в цій сфері. Однією з головних переваг є наявність вбудованих бібліотек і функцій, спеціально призначених для роботи з мультимедійними даними. Наприклад, інструментарій Image Processing Toolbox і Computer Vision Toolbox дозволяють виконувати операції з кадрами відео, застосовувати алгоритми стиснення, аналізувати частотні характеристики та оптимізувати якість обробки. Це значно скорочує час розробки, оскільки не потрібно створювати ці функції з нуля.

Крім того, MATLAB підтримує обробку великих обсягів даних у реальному часі, що є критичним для систем обробки відеоінформації. Його здатність працювати з матрицями та масивами даних забезпечує високу швидкість виконання операцій, таких як перетворення Фур’є, фільтрація чи кодування. Для автоматизації процесу стиснення відеозображень це дозволяє

ефективно реалізувати алгоритми, наприклад, на основі кодеків H.264 або HEVC, адаптуючи їх до конкретних потреб системи.

Ще однією важливою перевагою MATLAB є його гнучкість у створенні прототипів і моделюванні систем. Розробники можуть швидко тестувати різні алгоритми стиснення, порівнювати їхню ефективність і оптимізувати параметри без необхідності глибокої оптимізації на початкових етапах. Це особливо актуально для задач, де потрібно враховувати баланс між ступенем стиснення та збереженням якості відеокадрів. MATLAB дозволяє візуалізувати результати обробки в реальному часі, що полегшує аналіз і коригування алгоритмів.

Середовище також підтримує інтеграцію з іншими мовами програмування, такими як C або C++, що дає змогу оптимізувати критичні частини коду для підвищення продуктивності. Наприклад, після розробки прототипу в MATLAB можна перевести окремі модулі в C для інтеграції в апаратні системи обробки відео, зберігаючи при цьому переваги початкового дизайну.

Автоматизація процесу стиснення відеозображень вимагає не лише високої обчислювальної потужності, але й зручного інтерфейсу для налаштування параметрів. MATLAB пропонує інтерактивне середовище, де розробники можуть створювати сценарії (скрипти) для автоматичного виконання послідовності операцій, таких як декомпозиція кадрів, застосування перетворень і кодування. Це дозволяє реалізувати конвеєрну обробку відео, що є важливим для систем із великим потоком даних.

Крім того, MATLAB має розвинуті інструменти для роботи з відеофайлами, включаючи підтримку популярних форматів, таких як MP4, AVI та MOV. Це забезпечує сумісність із сучасними платформами і спрощує інтеграцію розробленого програмного забезпечення в існуючі системи. Наприклад, можна імпортувати відеопотік, обробляти його кадр за кадром із

застосуванням модифікованих алгоритмів стиснення, а потім експортувати результат у стиснутому вигляді.

Попри наявність альтернатив, таких як Python із бібліотеками OpenCV чи NumPy, або C++ із бібліотеками FFmpeg, MATLAB вирізняється своєю спеціалізацією на обробці сигналів і зображень. Python, хоч і є гнучким і популярним, вимагає більше часу на налаштування середовища та інтеграцію бібліотек, тоді як MATLAB постачається як готове рішення з укомплектованим набором інструментів. C++ забезпечує високу продуктивність, але його використання потребує значних зусиль для реалізації складних алгоритмів, особливо на етапі прототипування.

Для даної теми MATLAB також вигідний завдяки підтримці симуляцій і тестувань у реальних умовах. Наприклад, можна моделювати вплив стиснення на якість відео при різних рівнях компресії, використовуючи вбудовані функції для оцінки метрик, таких як PSNR (Peak Signal-to-Noise Ratio) або SSIM (Structural Similarity Index). Це дозволяє розробникам обґрунтувати вибір оптимального алгоритму для конкретної системи обробки відеоінформації.

Вибір MATLAB як мови програмування для розробки програмного забезпечення кодування відеозображень обґрунтований її спеціалізацією на обробці мультимедійних даних, наявністю готових інструментів, гнучкістю моделювання та можливостями автоматизації.

3.2 Критерії порівняння алгоритмів

Розробка програмного забезпечення для кодування відеозображень є важливим напрямком у створенні автоматизованих систем обробки відеоінформації. Ефективне стиснення відео дозволяє оптимізувати використання ресурсів, прискорити передачу даних і зменшити навантаження

на обчислювальні платформи. Для оцінки та порівняння алгоритмів кодування відеозображень необхідно визначити ключові критерії, які відображають їхню продуктивність, якість і практичну придатність. Ці критерії включають технічні показники, сумісність із апаратним забезпеченням та адаптивність до різних умов [16].

Одним із основних критеріїв є збереження якості зображення та відео після стиснення. Алгоритми кодування можуть використовувати стиснення з втратами або без втрат, що впливає на кінцевий результат. Для оцінки якості застосовуються метрики, такі як пікова величина сигналу до шуму (PSNR) або структурна схожість (SSIM). Вищий PSNR вказує на менші спотворення, тоді як SSIM враховує сприйняття людиною деталей і контурів. Наприклад, алгоритми, такі як H.264, можуть забезпечувати високу якість при помірному стисненні, тоді як новіші стандарти, як H.265/HEVC, покращують цей показник за рахунок складніших обчислень.

Якість також залежить від типу вмісту: відео з швидкими рухами (наприклад, спортивні трансляції) вимагає меншого стиснення, щоб уникнути артефактів, таких як розмиття чи блоковість. Отже, алгоритм має бути гнучким для адаптації до різноманітних сценаріїв [17].

Ступінь стиснення визначає, наскільки ефективно алгоритм зменшує обсяг даних. Цей критерій є критичним для систем, де важлива економія пам’яті або швидкість передачі, наприклад, у потоковому мовленні чи хмарних платформах. Співвідношення стиснення (compression ratio) розраховується як відношення початкового розміру файлу до стиснутого. Алгоритми, такі як VP9 або AV1, пропонують вищі коефіцієнти стиснення порівняно з попередніми стандартами, такими як MPEG-2, завдяки вдосконаленим методам прогнозування та ентропійного кодування [18].

Однак високий ступінь стиснення часто супроводжується втратою якості, тому необхідно знаходити баланс. Для автоматизованих систем

обробки відеоінформації важливим є забезпечення мінімального розміру файлу без значного погіршення візуального сприйняття.

Швидкість обробки даних є вирішальним фактором для реального часу. Алгоритми кодування оцінюються за часом, необхідним для стиснення та розпакування відео. Наприклад, H.264 забезпечує високу швидкість кодування, що робить його популярним у вебкамерах і відеоконференціях. Натомість H.265/HEVC, хоча й пропонує краще стиснення, потребує більше обчислювальних ресурсів, що може уповільнити процес на слабких пристроях.

Для автоматизації процесу стиснення в системах обробки відеоінформації важлива оптимізація алгоритмів під конкретне апаратне забезпечення. Використання паралельних обчислень або спеціалізованих чипів (наприклад, GPU) може значно підвищити швидкість, але потребує відповідної адаптації програмного забезпечення [18].

Обчислювальна складність відображає кількість ресурсів, необхідних для реалізації алгоритму. Цей критерій включає потребу в оперативній пам’яті, потужності процесора та енергоємності. Алгоритми з високою складністю, такі як AV1, використовують розширені методи прогнозування й трансформації, що вимагає сучасного обладнання. Натомість простіші методи, як MPEG-2, підходять для застарілого апаратного забезпечення, але забезпечують нижчу ефективність.

У контексті автоматизованих систем обробки відео важливим є вибір алгоритму, який відповідає доступним ресурсам. Наприклад, для мобільних пристроїв з обмеженою енергією доцільно використовувати оптимізовані версії кодеків, які зменшують навантаження на батарею [19].

Сумісність із існуючими платформами та стандартами є ключовим критерієм для широкого впровадження. Алгоритми кодування, такі як H.264 і H.265, підтримуються більшістю сучасних пристроїв і програмного забезпечення, що забезпечує їхню універсальність. Натомість новіші

стандарти, такі як VVC (Versatile Video Coding), хоча й обіцяють кращу продуктивність, ще не отримали повної підтримки на ринку.

Для автоматизованих систем обробки відеоінформації важлива інтеграція з популярними медіаплеєрами, потоковими сервісами та апаратними декодерами. Розробка програмного забезпечення має враховувати поточні стандарти, такі як MPEG або ITU-T, щоб гарантувати сумісність із різними екосистемами.

Стійкість до помилок визначає, як алгоритм поводиться в умовах втрати даних або перешкод під час передачі. У відеоінформаційних системах, де передача відбувається через нестабільні мережі, важливим є використання механізмів захисту, таких як вставка службових даних чи повторне кодування. Наприклад, H.264 підтримує інструменти для відновлення кадрів, що дозволяє мінімізувати вплив втрат [20].

Цей критерій особливо актуальний для автоматизованих систем, які працюють у реальному часі, де перерви в трансляції можуть бути неприпустимими. Алгоритми з високою стійкістю до помилок потребують додаткових обчислень, що впливає на інші показники, такі як швидкість.

Енергоефективність стає дедалі важливішою з поширенням портативних пристроїв. Алгоритми кодування оцінюються за споживанням енергії під час стиснення та декодування. Наприклад, кодеки, оптимізовані для мобільних платформ, такі як VP8, намагаються зменшити енерговитрати за рахунок спрощення обчислень. У автоматизованих системах обробки відеоінформації цей критерій впливає на тривалість роботи пристрою без підзарядки.

Декілька величин використовуються для обчислення ефективності алгоритмів стиснення [17].

1) Коефіцієнт стиснення визначається за формулою (3.1):

$$Kcompr = \frac{S(I_{out})}{S(I_{\text{in}})}, \qquad (3.1)$$

де *Kcompr* - коефіцієнт стиснення;

S() – функція визначення розміру зображення;

I_{out} - вихідне зображення, що отримане до кодування;

I_{in} - вхідне зображення до кодування.

Наприклад, коефіцієнт *Kcompr*=0.6 означає, що стиснені дані займають 60% від початкового розміру. Значення більші 1 говорять про те, що вихідний файл більше вхідного (негативне стиснення). Коефіцієнт стиснення прийнято вимірювати в bpb (bit per bit, біт на біт), оскільки він показує, скільки в середньому знадобиться біт стислого файлу для представлення одного біта файлу на вході.

Тут слід згадати ще два поняття, пов'язані з коефіцієнтом стиснення. Термін бітова швидкість (bitrate) є більш загальною, ніж bpb. Метою компресії інформації є представлення даних з найменшою бітовою швидкістю.

2) Величина, зворотна коефіцієнту стиснення, називається фактором стиснення (3.2):

$$Fcompr = \frac{S(I_{in})}{S(I_{out})}, \qquad (3.2)$$

де *Fcompr* – фактор стиснення;

S() – функція визначення розміру зображення;

I_{out} - вихідне зображення, що отримане до кодування; I_{in} - вхідне зображення до кодування.

В даному випадку значення *Fcompr*>1 означають стиснення, а *Fcompr*<1 - розширення. Цей множник представляється більшості людей природнішим показником: чим більше значення фактору, тим краща компресія.

3) Якість стиснення (3.3)

$$Quality = 100 * (Fcompr - 1) \quad (3.3)$$

де *Fcompr* - коефіцієнт стиснення. Його значення, наприклад, *Quality*= 60 означає, що в результаті стиснення зображення займає на 60% менший розмір, ніж початковий файл.

3.3 Експериментальні дослідження

Для проведення порівняння алгоритмів стиснення було спеціально застосовано різні категорії відеозображень, що дає змогу оцінити їхні потенційні можливості. З цією метою обрано зображення з типової колекції CIPR Still Images [8,27], яка наразі слугує стандартним тестовим набором для аналізу методів стиснення.

За результатами тестування зображень із зазначеної колекції за допомогою створеного програмного забезпечення (додаток В) отримано наступні показники (таблиця 3.1 – 3.2).

Таблиця 3.1 – Показники стиснення зображень за алгоритмом RLE

Зображ.	**Симет-ричність**	**Втрата якості**	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
LENA	+	–	3,64	0,24	-73,13
ZELDA	+	–	2,46	0,33	-62,38
BOARD	+	–	0,82	1,22	25,31
BOATS	+	–	3,52	0,34	-64,31
GOLD	+	–	2,92	0,35	-63,2
HOTEL	+	–	0,91	1,10	20,18
BALOON1	+	–	0,95	1,04	9,67
BALOON2	+	–	1,42	0,75	-19,00

Таблиця 3.2 – Показники стиснення зображень розробленим алгоритмом

Зображ.	**Симет-ричність**	**Втрата якості**	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
LENA	–	–	1,22	0,75	-21,32
ZELDA	–	–	1,13	0,82	-12,94
BOARD	–	–	0,57	1,61	63,23
BOATS	–	–	1,33	0,69	-27,14
GOLD	–	–	1,11	0,83	-10,31
HOTEL	–	–	0,69	1,37	40,55
BALOON1	–	–	0,65	1,42	46,64
BALOON2	–	–	1,09	0,83	-11,20

Розглядаючи принципи функціонування двох алгоритмів стиснення зображень, описаних у попередніх розділах, можна зробити висновок, що вони вирішують одну категорію завдань компресії даних, базуючись на груповому кодуванні. Як видно з таблиць 3.1 і 3.2, алгоритм RLE характеризується симетричністю, на відміну від запропонованого підходу. У розробленому методі процес кодування значно триваліший порівняно з декодуванням. Тривалість кодування зумовлена додатковим етапом сегментації однорідних груп повторюваних пікселів на зображенні перед їх стисненням, тоді як декодування цієї процедури не потребує. Натомість у традиційному методі RLE операції кодування та декодування мають приблизно однакову тривалість, якщо не враховувати незначні розбіжності, викликані специфікою реалізації.

Огляд даних про компресію, наведених у таблицях 3.1 і 3.2, свідчить, що не всі зображення стискуються з однаковим коефіцієнтом, а деякі після обробки мають більший розмір, ніж до стиснення. Чітко ілюструє ефективність стиснення показник якості, виражений у відсотках. У таблиці 3.3 представлено відхилення результатів стиснення, отриманих за

запропонованим алгоритмом, порівняно з аналогічними показниками методу RLE.

Таблиця 3.3 – Абсолютні відхилення показників стиснення алгоритмів

Зображення	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
LENA	-2,38	0,51	51,32
ZELDA	-1,43	0,49	50,19
BOARD	-0,15	0,34	35,23
BOATS	-1,77	0,41	41,19
GOLD	-1,67	0,54	54,28
HOTEL	-0,11	0,18	19,76
BALOON1	-0,21	0,33	36,77
BALOON2	-0,11	0,07	8,20

Дані, подані в зазначених таблицях, вказують на категорію зображень, для яких розроблений алгоритм є ефективним: ті, що містять численні повтори (тексти, однорідні ділянки тощо), як-от зображення BALOON1, BOARD, HOTEL. Загалом, аналізуючи показник якості стиснення, можна констатувати, що запропонований метод у 1,28 разів перевищує за якістю компресію RLE, що відкриває перспективи для його впровадження.

3.4 Висновки до розділу 3

Вибір MATLAB як мови програмування для розробки програмного забезпечення кодування відеозображень виправдав себе завдяки її

спеціалізації на обробці мультимедійних даних, наявності вбудованих бібліотек, таких як Image Processing Toolbox і Computer Vision Toolbox, та гнучкості у створенні прототипів. Ця мова забезпечила високу продуктивність при роботі з матрицями та масивами, що є ключовим для реалізації алгоритмів стиснення, таких як H.264 чи HEVC, а також дозволила проводити тестування та візуалізацію результатів у реальному часі. Інтеграція з іншими мовами, як-от C++, та підтримка популярних відеоформатів (MP4, AVI, MOV) зробили MATLAB ідеальним інструментом для автоматизації процесу стиснення, особливо в контексті хмарних і потокових систем, де важлива економія ресурсів і швидкість обробки.

Експериментальні дослідження підтвердили, що запропонований алгоритм стиснення перевершує традиційний метод RLE за показником якості компресії в 1,28 раза, особливо для зображень із численними повторюваними ділянками, такими як BALOON1, BOARD і HOTEL. Хоча коефіцієнт стиснення варіюється залежно від вмісту зображення, а декодування є швидшим за кодування через сегментацію однорідних груп, загальна ефективність алгоритму демонструє його потенціал для впровадження в автоматизовані системи обробки відеоінформації. Критерії порівняння, такі як збереження якості (PSNR, SSIM), ступінь стиснення, швидкість обробки та енергоефективність, підкреслили необхідність адаптації алгоритмів до конкретних умов, що робить подальшу оптимізацію та інтеграцію з апаратним забезпеченням перспективним напрямком розвитку.

ВИСНОВКИ

У роботі було здійснено аналіз методів кодування зображень у сучасних системах відеообробки, а також класифікацію типів відеозображень та програмного забезпечення, що дозволило визначити вимоги до алгоритмів кодування. Розглянуто основні підходи до стиснення відеоданих, зокрема внутрішньокадрове та міжкадрове кодування, з акцентом на їх переваги та недоліки в контексті ефективності обробки і передачі відеоінформації. Особлива увага була приділена характеристикам відеоформатів і стандартів, що впливають на вибір методів кодування. Це дало змогу сформулювати основи для подальшого розроблення оптимізованих алгоритмів, що поєднують високу якість зображення з ефективним використанням обчислювальних ресурсів.

Представлено новий підхід до кодування цифрових відеозображень, який значно перевершує існуючі методи групового кодування за рахунок удосконаленого механізму обробки груп пікселів. Запропоновано нові стратегії, які дозволяють зменшити надлишковість у зображеннях, аналізуючи групи пікселів з однаковими кольоровими значеннями в двох перпендикулярних напрямках. Розроблено алгоритмічні рішення для реалізації як кодувальника, так і декодувальника, які гарантують відсутність втрат даних і забезпечують незалежність від конкретного апаратного або програмного середовища.

Загалом, аналізуючи показник якості стиснення, можна констатувати, що запропонований метод у 1,28 разів перевищує за якістю компресію RLE, що відкриває перспективи для його впровадження.

Таким чином, проведене дослідження дозволяє зробити висновок, що запропоновані методи кодування відеозображень сприяють значному підвищенню ефективності стиснення без втрат якості, забезпечуючи при

цьому широку гнучкість у використанні на різних платформах та мовах програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. Бутузов, В. П. Методи та алгоритми стиснення відеоданих. Вісник Вінницького національного технічного університету. 2018. № 3. С. 12–17. URL: https://visnyk.vntu.edu.ua/ (дата звернення: 21.05.2025).

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

4. Васильєв, О. М. Розробка програмного модуля для кодування відеозображень за стандартом H.264. Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація. 2019. № 2 (47). С. 83–88. URL: http://science.donntu.edu.ua/ (дата звернення: 21.05.2025).

5. Дзюба, І. А. Оптимізація параметрів кодування відео у системах відеонагляду. Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні науки та інформаційні технології. 2021. № 915. С. 78–84. URL: https://lp.edu.ua/ (дата звернення: 21.05.2025).

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

7. Ковальчук, Р. М. Ефективність сучасних алгоритмів стиснення відео для потокової передачі даних. Інформаційні технології та системи управління. 2022. № 1 (25). С. 45–52. URL: https://itms.nure.ua/ (дата звернення: 21.05.2025).

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

9. Гармаш В.В. Метод стиснення цифрових зображень на основі усічення простору вейвлет-коефіцієнтів / В. В. Гармаш, Н. М. Ольшанська: XLVIII Науково-технічна конференція факультету комп'ютерних систем і автоматики (2019). URL : https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/6617/5488.

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

11. Мороз, Т. В. Методики покращення якості стисненого відеозображення. Вісник Харківського національного університету радіоелектроніки. Серія: Інформатика та комп'ютерні технології. 2019. № 2 (78). С. 90–97. URL: http://nure.ua/ (дата звернення: 21.05.2025).

12. Олійник, Г. Ф. Архітектура програмних систем для інтелектуального стиснення відео. Системи обробки інформації. 2020. № 3 (161). С. 134–141. URL: https://soi.nuou.org.ua/ (дата звернення: 21.05.2025).

13. Сорокін, В. І. Алгоритми попередньої обробки відео для підвищення ефективності стиснення. Наукові праці Харківського національного університету Повітряних Сил імені Івана Кожедуба. Серія: Військові науки. 2023. № 2 (74). С. 100–107. URL: http://hups.mil.gov.ua/ (дата звернення: 21.05.2025).

14. Jain R. Fundamentals of Digital Image Processing / R. Jain. – Englewood Cliffs, NJ : Prentice Hall, 2014. – 348 с.

15. Wu Y. Compression Algorithms for Real-Time Video Processing / Y. Wu, B. Chen. – London : Springer, 2015. – 389 с.

16. Jiang X. Video Compression and Communications: Coding and Streaming / X. Jiang, Y. Zhang. – Amsterdam : Elsevier, 2017. – 276 с.

17. Katz J. Advanced Video Coding for Multimedia Communications and Interactive Services / J. Katz, J. Watson. – Boca Raton : CRC Press, 2019. – 422 с.

18. González L. Compression Techniques for Multimedia Content / L. González, S. Blanch. – Berlin : Springer, 2016. – 318 с.

19. Лисицький О. П. Методи оптимізації стиснення відеоданих для систем реального часу / О. П. Лисицький // Журнал комп’ютерних наук і технологій. – 2023. – № 4 (29). – С. 56–63. – Режим доступу: https://cstjournal.org/ (дата звернення: 21.05.2025).

20. Smith T. Real-Time Video Compression Techniques for Embedded Systems / T. Smith, H. Lee. – New York : Wiley, 2020. – 412 с.

21. Пономаренко В. С. Алгоритми адаптивного стиснення відеозображень для автоматизованих систем обробки інформації / В. С. Пономаренко // Вісник Київського національного університету імені Тараса Шевченка. Серія: Фізико-математичні науки. – 2022. – № 3 (88). – С. 101–108. – Режим доступу: https://knu.ua/ (дата звернення: 21.05.2025).

22. Chen L. Advanced Algorithms for Video Compression in Multimedia Systems / L. Chen, Q. Wang. – Singapore : Springer Nature, 2021. – 355 с.

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

ДОДАТКИ

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ

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

_____
(підпис)

«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації»
08-31.БКР.003.03.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н, доцент каф. АІІТ Володимир ГАРМАШ

_____
(підпис)

«_23_» _травня__ 2025 р.
Розробила студентка гр. АКІТ-21бз
Поліна КОСТРИЦЯ

_____
(підпис)

«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

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

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації» є наказ № 97 від 20.03.2025 р.

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

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є вдосконалення існуючих алгоритмів кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації, їх програмна реалізація та експериментальне дослідження.

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

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студентка групи АКІТ-21бз Кострицяя Поліна Русланівна факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз методів кодування зображень	07.03 – 19.03
Е2	Алгоритм кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації	20.04 – 26.04
Е3	Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформаці	27.04 – 01.06
Е4	Аналіз результатів роботи та підготовка роботи до захисту	02.06 – 20.06

4 Призначення і галузь застосування

Розроблене програмне забезпечення призначене для автоматизованого кодування та стиснення відеозображень у системах обробки відеоінформації, що забезпечує оптимізацію обробки великих обсягів відеоданих. Воно дозволяє зменшувати розмір відеофайлів без значних втрат якості, сприяючи ефективному зберіганню, передачі та аналізу відео в автоматизованих системах. Програмне забезпечення може використовуватися в автоматизованих системах управління, промислових комплексах, медіа-платформах та IoT-пристроях для забезпечення надійної передачі та зберігання відеоінформації.

5 Технічні дані

5.1 тип цифрового зображення – півтонові та кольорові;
5.2 мінімальна роздільна здатність зображення 128 х 128 пікселів,
5.3 максимальна роздільна здатність 2048 х 2048 пікселів;
5.4 коефіцієнт стиснення від 1.1 до 3.
5.5 виграш у стисненні від 0,5% до 10% у порівнянні з класичними алгоритмом RLE.

6 Джерела розробки

6.1 Бутузов, В. П. Методи та алгоритми стиснення відеоданих. Вісник Вінницького національного технічного університету. 2018. № 3. С. 12–17. URL: https://visnyk.vntu.edu.ua/ (дата звернення: 21.05.2025).
6.2 Гармаш В.В. Метод стиснення цифрових зображень на основі усічення простору вейвлет-коефіцієнтів / В. В. Гармаш, Н. М. Ольшанська: XLVIII Науково-технічна конференція факультету комп'ютерних систем і автоматики (2019). URL : https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/6617/5488.
6.3 González L. Compression Techniques for Multimedia Content / L. González, S. Blanch. – Berlin : Springer, 2016. – 318 c.
6.4 Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проектів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с. URL: https://iq.vntu.edu.ua/fm/fdb/940/bdr/bdrmetod.pdf.

Здобувач ст. гр. АКІТ-21бз _____ Поліна КОСТРИЦЯ

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОДУВАННЯ ВІДЕОЗОБРАЖЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ СТИСНЕННЯ У СИСТЕМАХ ОБРОБКИ ВІДЕОІНФОРМАЦІЇ

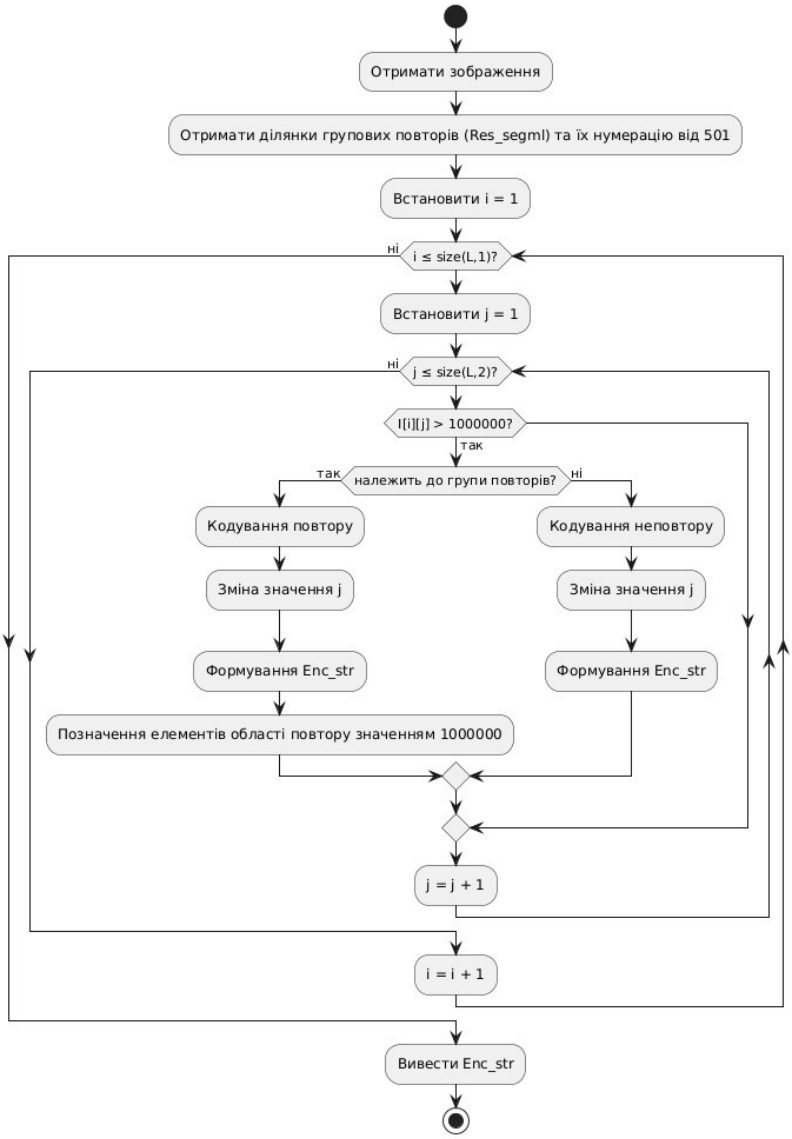


Рисунок Б.1 – Діаграма активності роботи кодера

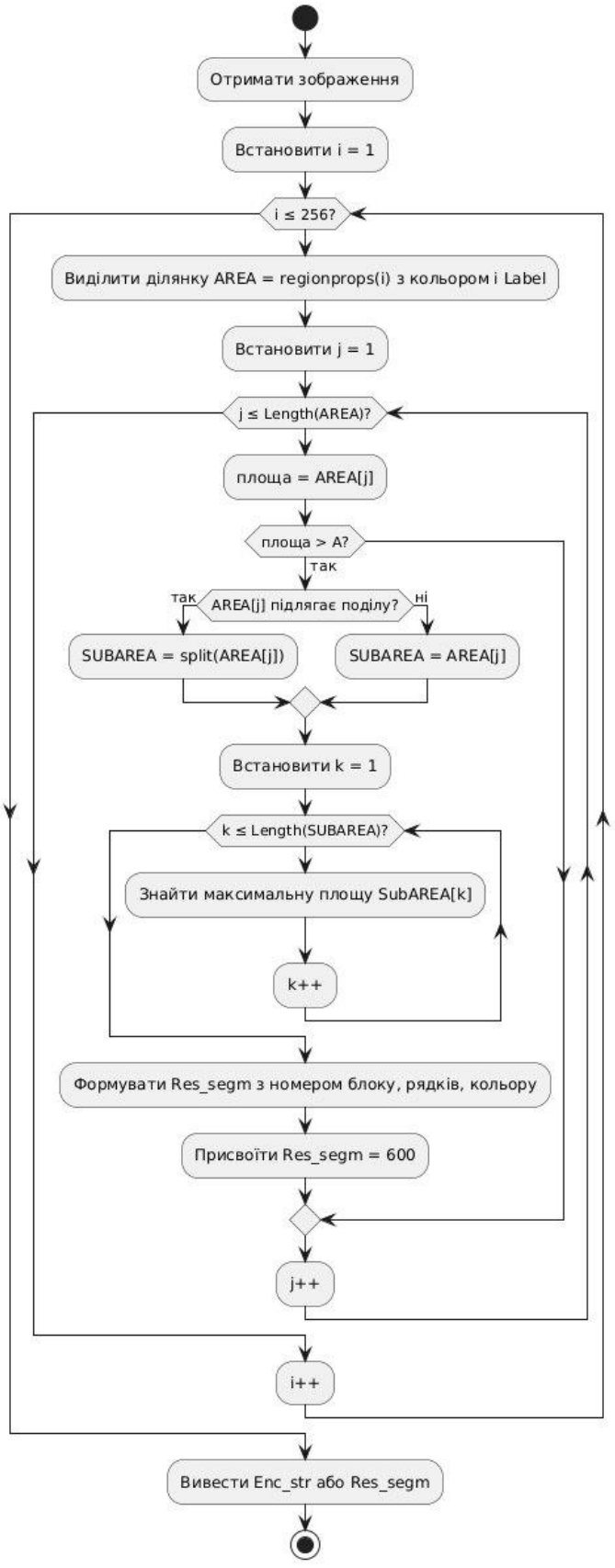


Рисунок Б.2 – Діаграма активності процедури виділення групових послідовностей

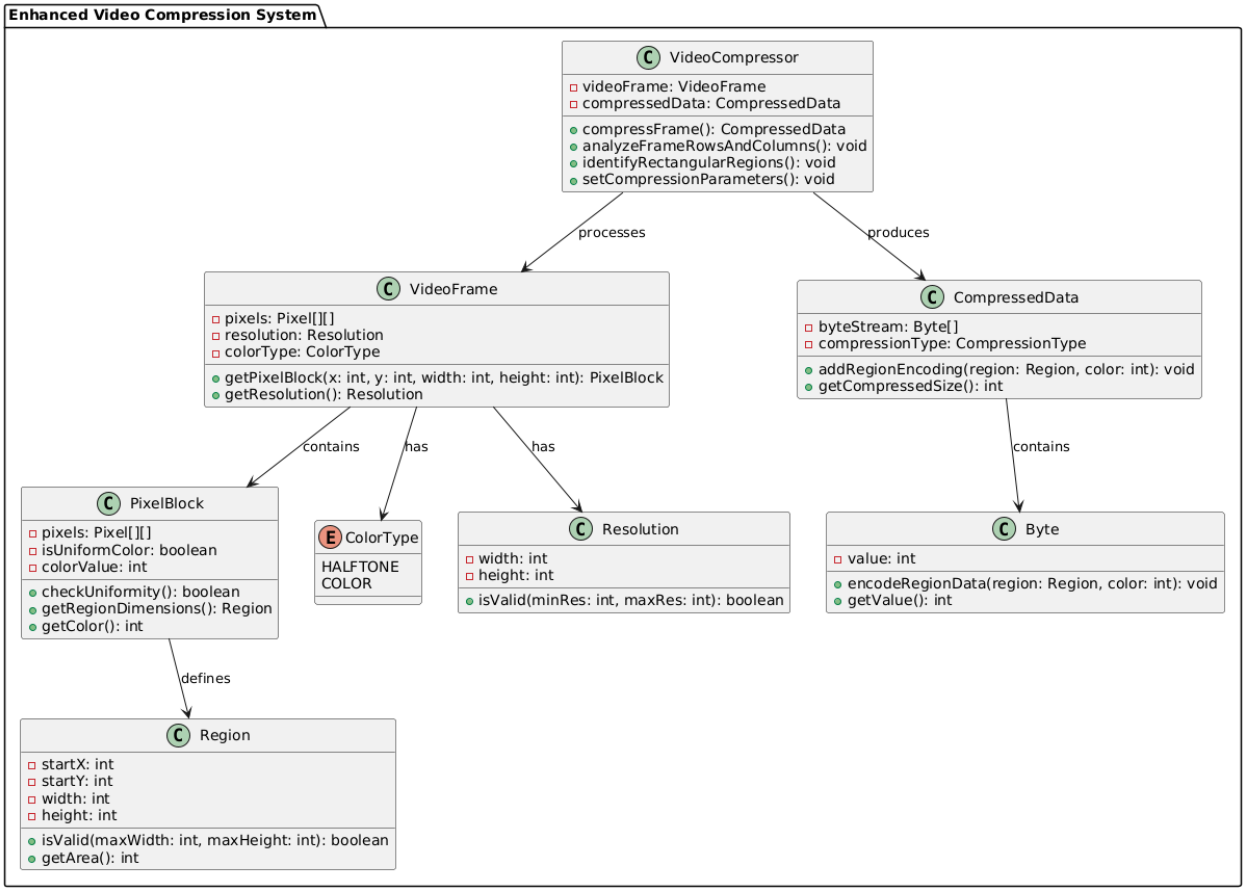


Рисунок Б.3 – Діаграма класів системи стиснення зображень

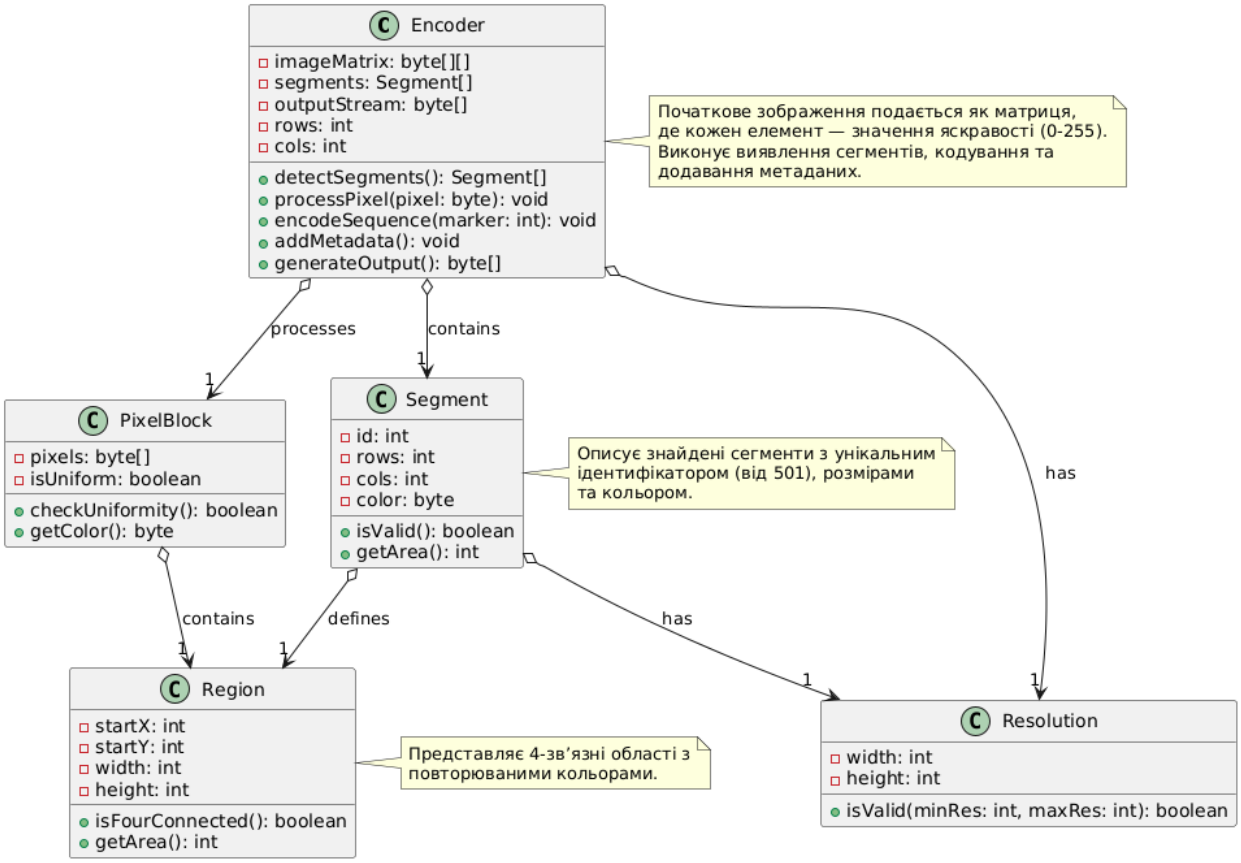


Рисунок Б.4 – Діаграма класів процесу стискання

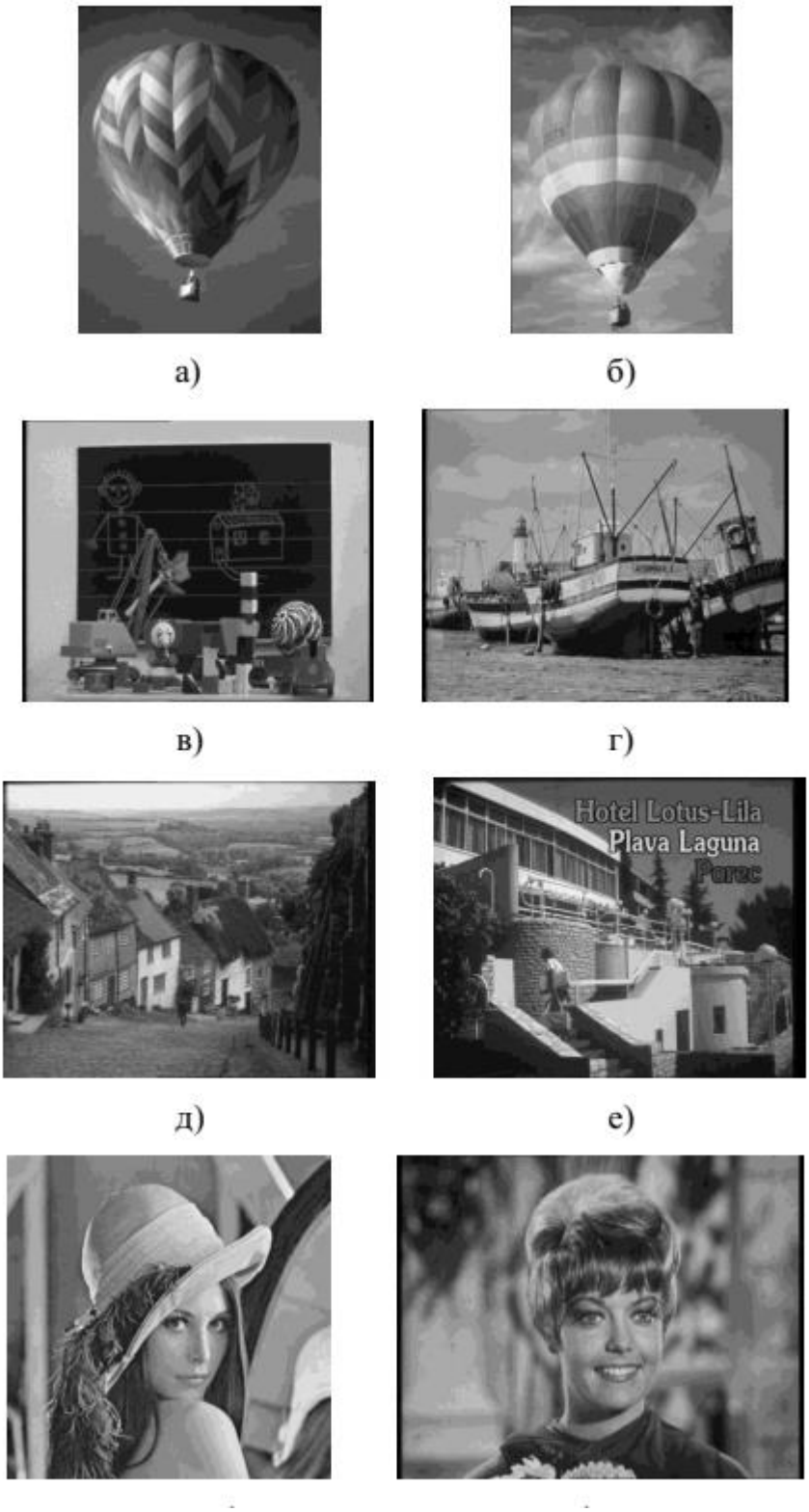


Рисунок Б.5 – Тестові зображення

Додаток В
(обов'язковий)
Лістинг модуля програми

```
%---------- RLE Compresssion --- function Output=rle_vko(Input) pict=Input;

clear all; clc;

%I = imread('concord.jpg');
%I = imread('image1.jpg');
%I = imread('board.jpg');
%I = imread('lena.jpg');
%I = imread('zelda.jpg'); I = imread('baloon2.jpg');
%I = imread('test1.jpg'); pict=double(I(:,:,:)); pict=pict(:,:,1);
%pict=[3 2 2 2 7 6 6 5]; %     Set for testing
% --------- Line creation --------
h = waitbar(0,'Line Creation, please wait       ');
tmp_str=[]; tic
for i=1:size(pict,1) tmp_str=[tmp_str pict(i,:)];
waitbar(i/size(pict,1),h,sprintf('i=%d',i));
%        drawnow; end
toc

%---------- RLE Compresssion ---
%h = waitbar(0,'Compression, please wait     ');
tic comp_char=500; Out_code=''; i=1;
while i<=length(tmp_str)
if comp_char==tmp_str(i) out_cycle=1;
Num=1;
while comp_char==tmp_str(i) & Num<128 & out_cycle==1 Num=Num+1;
i=i+1;
if i>length(tmp_str) out_cycle=0;
end end
povtor=['1' dec2bin(Num-1,7)]; Out_code=[Out_code povtor dec2bin(comp_char,8)];
else
Num=1; k=i; out_cycle=1;
while comp_char~=tmp_str(k) & Num<128 & out_cycle==1 comp_char=tmp_str(k);
Num=Num+1; k=k+1;

if k>length(tmp_str) out_cycle=0; k=k-1;
end end
povtor=['0' dec2bin(Num-2,7)]; Out_code=[Out_code povtor]; while i<=k-2
Out_code=[Out_code dec2bin(tmp_str(i),8)]; i=i+1;
end end
if i==length(tmp_str)-1 disp(i);
end
```

```
waitbar(i/length(tmp_str),h,sprintf('i=%d of %d',i,length(tmp_str)));

end size(Out_code) toc



%       Програма кодера
clear all; clc;
sum_square=0; sum_number=0;

%I = imread('concord.jpg');
%I = imread('image1.jpg'); I = imread('test1.jpg'); pict=double(I(:,:,:)); pict=pict(:,:,1);
pict=pict(:,1:size(pict,2)-1)-pict(:,2:size(pict,2));

my_fig=figure; hold on; set(my_fig,'DoubleBuffer','on')
h = waitbar(0,'Please wait...'); numb=1;
tic;
for i=1:256
waitbar(i/256,h,sprintf('Please wait...[%d/256] Spend time (%5.1f)',i,toc)); pict2=pict==i;
pict3=bwlabel(pict2,4); % labeling of the all blocks label_num=max(max(pict3));
j=1;
while j<=label_num
%display(sprintf('Color=%d, label=%d',i,l)); pict_label=pict3==j;
if sum(sum(pict_label))>4

tmp=max(pict_label); t=1;
while tmp(t)==0 t=t+1;

end col_min=t;
while tmp(t)~=0 & t<length(tmp) t=t+1;
end col_max=t-1;
tmp=max(pict_label'); t=1;
while tmp(t)==0 t=t+1;
end row_min=t;
while tmp(t)~=0 & t<length(tmp) t=t+1;
end row_max=t-1;
block_label=pict_label(row_min:row_max,col_min:col_max);
[row,col,wide,high]=rect_find(block_label);

%       Show
figure(my_fig); subplot(1,2,1);
imshow(pict_label,[]); title(sprintf('Color is =%d, label is=%d,
elements=%d',i,j,sum(sum(pict_label))));
figure(my_fig); subplot(1,2,2); imshow(block_label);
line([col col+wide col+wide col col],[row row row+high row+high
row],'LineWidth',8,'Color','red') title(sprintf('Square of maximum block is =
%d',(wide+1)*(high+1)));
drawnow; pause;
%------------ End show ---
```

```
pict_label(row_min+row-1:row_min+row-1+high,col_min+col-1:col_min+col-1+wide)=0;
pict(row_min+row-1:row_min+row-1+high,col_min+col-1:col_min+col-1+wide)=numb+500;
resul_compres(numb,:)=[row_min+row-1,col_min+col-1,wide,high,i];
numb=numb+1; sum_square=sum_square+(wide+1)*(high+1); sum_number=sum_number+1;
pause(0.1); pict2=pict==i;
pict3=bwlabel(pict2,4); % labeling of the all blocks
label_num=max(max(pict3)); j=1;

end j=j+1;
end

%------ Show ----
%       figure(my_fig); subplot(1,1,1);
%               imshow(pict2,[]); title(sprintf('Color is =%d',i)); pause; %pause(0.9);
% %    line([j-2 j-2 j+2 j+2 j-2],[i-2 i+2 i+2 i-2 i-2],'LineWidth',2,'Color','red')
%       drawnow end
disp(sum_square);disp(sum_number); close(h);
figure; imshow(pict,[]);

disp('------ Press any key to continue ');
pause;

encoding_string=''; encoding_string2=''; encoding_string3=''; for i=1:size(pict,1)
disp(i); drawnow; j=1;
while j<=size(pict,2) encoding_byte='0'; if pict(i,j)<1000000
if pict(i,j)>500 encoding_byte(2)='1';
switch resul_compres(pict(i,j)-500,3)+resul_compres(pict(i,j)-500,4) case 0 %--
resul_compres(3)==0 & resul_compres(4)<>0
warning('No defined activity in case loop!!!');
case resul_compres(pict(i,j)-500,3) %-- resul_compres(3)<>0 & resul_compres(4)==0
encoding_byte(3)='0'; encoding_byte(4)='1';
encoding_byte=strcat(encoding_byte,dec2bin(resul_compres(pict(i,j)-500,3)+1,12));

case resul_compres(pict(i,j)-500,4) %-- resul_compres(3)==0 & resul_compres(4)<>0
encoding_byte(3)='1'; encoding_byte(4)='0';
encoding_byte=strcat(encoding_byte,dec2bin(resul_compres(pict(i,j)-500,4)+1,12));

otherwise %-- resul_compres(3)<>0 & resul_compres(4)<>0 encoding_byte(3)='1';
encoding_byte(4)='1';
encoding_byte=strcat(encoding_byte,dec2bin(resul_compres(pict(i,j)-500,3)+1,6));
encoding_byte=strcat(encoding_byte,dec2bin(resul_compres(pict(i,j)-500,4)+1,6));
end
encoding_byte=strcat(encoding_byte,dec2bin(resul_compres(pict(i,j)-500,5),8));
bits

tmp=pict==pict(i,j); tmp=tmp.*1000000; j=j+resul_compres(pict(i,j)-500,3)+1; pict=pict+tmp;
else
encoding_byte(2)='0';
out_cycle=0; col=j; number_of_marks=0; while out_cycle==0
if pict(i,col)>=1000000 col=col+1;
number_of_marks=number_of_marks+1; elseif pict(i,col)<500
```

```
col=col+1; else
out_cycle=1; end
if col>size(pict,2) | (col-j)>16384 %  limit for image size or maximum from 2^14 reserved

out_cycle=1; end
end
encoding_byte=strcat(encoding_byte,dec2bin(col-j-number_of_marks,14)); while j<col
if pict(i,j)<1000000 encoding_byte=strcat(encoding_byte,dec2bin(pict(i,j),8));
end

j=j+1; end

end
encoding_string=strcat(encoding_string, encoding_byte); else
j=j+1; end
%       [row,col,wide,high]=rect_find(pict(i,j));
% %    Show
%       figure(my_fig); subplot(1,2,1);
%       line([col col+wide col+wide col col],[row row row+high row+high
row],'LineWidth',8,'Color','red')
%       title(sprintf('Square of maximum block is = %d',(wide+1)*(high+1)));
%       drawnow;
% %---------- End show ---
end encoding_byte='10000000';
encoding_string=strcat(encoding_string, encoding_byte); end
%       rozpodil po bajtah
numb_enc=1;
for i=1:8:size(encoding_string,2) encoding_string2(numb_enc,:)=encoding_string(1,i:i+7);
numb_enc=numb_enc+1;
%pause End



% ------------------- Функція пошуку прямокутної області ---------------------
function [row,col,wide,high]=rect_find(block) box_num=1;
for i=1:size(block,1) for j=1:size(block,2)
if block(i,j)~=0 row_box=i; col_box=j;
% square_box(box_num,1)=i;square_box(box_num,2)=j;square_box(box_num,5)=0;
box_wide=size(block,2);
out1=true; %condition for terminating of loop 1 while row_box<=size(block,1) & out1==true
if block(row_box,j)~=0
out2=true; %condition for terminating of loop 2 col_box=j;
while col_box<=size(block,2) & out2==true
if block(row_box,col_box)~=0 & col_box<=box_wide col_box=col_box+1;
else
out2=false; end
end
if (col_box-1)<=box_wide
square_box(box_num,3)=row_box;square_box(box_num,4)=col_box-1; %koord pravy-

nyznij kut
```

```
nyznij kut


box_wide=col_box-1; else
square_box(box_num,3)=row_box;square_box(box_num,4)=box_wide; %koord pravy-

end
square_box(box_num,1)=i;square_box(box_num,2)=j;        %koord livyj-verhnij kut
square_box(box_num,5)=(row_box-i+1)*(box_wide-j+1); %Ploshcha
else
out1=false; end
box_num=box_num+1; row_box=row_box+1;
end end
end end

[maximum,index]=max(square_box);
% square_box(index(5),:)
% figure; subplot(1,1,1); imshow(block); hold on;
% line([square_box(index(5),2) square_box(index(5),4) square_box(index(5),4)
square_box(index(5),2) square_box(index(5),2)],[square_box(index(5),1)
square_box(index(5),1) square_box(index(5),3) square_box(index(5),3)
square_box(index(5),1)])

row=square_box(index(5),1); col=square_box(index(5),2); wide=square_box(index(5),4)-
square_box(index(5),2); high=square_box(index(5),3)-square_box(index(5),1);
%pause; return
```

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмного забезпечення кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації

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

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

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 6,12 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- [x] Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- [ ] У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- [ ] У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

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

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

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

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Гармаш В.В.
(прізвище, ініціали, посада)

Здобувач _____ (підпис) Кострица П.Р.
(прізвище, ініціали)