Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

До захисту допущено
Завідувач кафедри
комп'ютерної інженерії
к.т.н., доц. О.М.Березький
_____
"___" _____ 20___ р.

ДИПЛОМНА РОБОТА
освітньо-кваліфікаційного рівня "Магістр"
зі спеціальності 8.05010201 "Комп'ютерні системи та мережі"
на тему:

АЛГОРИТМИ СТИСНЕННЯ ВІДЕОЗОБРАЖЕНЬ БЕЗ ВТРАТ НА ОСНОВІ ГРУПОВОГО КОДУВАННЯ

Студент групи КСМзм - 51
Осіпчук В.О.

_____
підпис

Науковий керівник
к.т.н., доцент Коваль В.С.

_____
підпис

Нормоконтролер
Палій І.О.
_____
Прізвище, ініціали Підпис

Тернопіль – 2012

Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

"Затверджую"
Зав. кафедри
комп'ютерної інженерії
к.т.н., доц. О.М. Березький
_____
"____"_____20__ р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Осіпчука Вадима Олександровича

1. **Тема дипломної роботи** "Алгоритми стиснення відеозображень без втрат на основі групового кодування" затверджена наказом університету №_____ від "____" _____20__ р

2.**Термін здачі** закінченої дипломної роботи_____

3. **Об'єкт дослідження**: система кодування відеозображень.
.

4. **Предмет дослідження**: Алгоритмічні, математичні та програмно-технічні засоби системи обробки відеозображень

5. **Перелік задач, які мають бути вирішені**:
- аналіз відомих методів та алгоритмів кодування відеозображень;
- удосконалення та розроблення нових алгоритмів кодування відеозображень;
- удосконалення та розроблення нових алгоритмів декодування відеозображень;
- налагодження та експериментальні дослідження розроблених програмних засобів.

6. **Перелік ілюстративного матеріалу:**
- узагальнена схема роботи кодера,
- схема роботи процедури виділення групових послідовностей,
- структура декодера,
- часова діаграма процесу надходження й обробки даних,

- схема мережевої взаємодії вузлів системи розподілених обчислень,
- схема роботи процедури кодування послідовностей,
- схема роботи алгоритму декодування зображення.

7.**Консультанти по роботі**

Розділ	Консультант	Підпис
1		
2		
3		

КАЛЕНДАРНИЙ ПЛАН

№	Назва структурних частин ДР	Термін виконання	Примітка
1	Сучасний стан кодування зображень і постановка задачі дослідження	15.09.2011 – 5.11.2011	
2	Запропонований алгоритм стиснення відеозображень	6.11.2011 – 31.01.2012	
3	Програмно-технічна реалізація	1.02.2012 – 23.04.2012	

Завдання прийняв до виконання _____
(підпис)

Керівник дипломної роботи _____
(підпис)

РЕФЕРАТ

Дипломна робота на тему “Алгоритми стиснення відеозображень без втрат на основі групового кодування” на здобуття освітньо-кваліфікаційного рівня “Магістр” зі спеціальності “Комп’ютерні системи та мережі” написана обсягом 90 сторінок і містить 30 ілюстрацій, 4 таблиць, 4 додатків та 27 джерел за переліком посилань.

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

Методи досліджень базуються на використанні методів теорій кодування та програмування, прогнозовані оцінки довжин кодів блоків обчислено з використанням положень теорії інформації та теорії ймовірності, оцінювання параметрів виконано методами математичної статистики, оцінку алгоритмів проведено на основі використання положень теорії алгоритмів.

Розроблено алгоритми стиснення відеозображень на основі групового кодування, які за рахунок використання процесу групування за двома напрямами дозволяє стискати зображення із меншим коефіцієнтом стиснення ніж відомий алгоритм RLE без втрат інформації.

Результати роботи програмно реалізовані і дозволяють автоматизувати процеси обробки у системах відообробки.

Можливими напрямками подальших досліджень є продовження робіт по підвищенню рівня стиснення відеозображень без втрат інформації на основі групового кодування.

Ключові слова: КОДУВАННЯ ЗОБРАЖЕНЬ, ДЕКОДУВАННЯ ЗОБРАЖЕНЬ, СТИСНЕННЯ ЗОБРАЖЕННЯ, ГРУПОВЕ КОДУВАННЯ, МЕТОД RLE.

ANNOTATION

The master thesis entitled “Lossless algorithms of image compression using group coding” for master degree at the specialty of “computer systems and networks” includes 90 pages of the text 30 images, 4 tables, 4 pages of appendixes and 27 points of references.

The purpose of this thesis consist into improving of the existing algorithms of image coding for providing of image compression at the image processing systems and designing of the software routines and their experimental researches.

Exploration methods are basing on using of the methods of coding theory and programming, predicting of the code length provided by using of the theory of information and probability theory, estimating of the parameters of proposed algorithms provided by using of the statistic, estimating of the algorithms provided by using the main backgrounds of the algorithmic theory.

It is developed the algorithms for image compression basing on the group coding that allow compress the images with less compression coefficients than existed algorithm taking into account two-dimensional grouping of the similarity at the images.

The main results realized as the software and allow to provide automatization of image processing in the videoprocessing systems.

The future researches direction include continuing this work for increasing compression level by using the lossless approach basing on the group coding .

Keywords: IMAGE CODING, IMAGE DECODING, IMAGE COMPRESSION, GROUP CODING, RLE METHOD

ЗМІСТ

ВСТУП

1. Актуальність теми. Особливості розвитку сучасних інформаційно-телекомунікаційних систем полягають в організації дистанційного збору і обробки відеоінформації та її передачі з використанням бездротових технологій. Зображення є невід'ємною складовою мультимедійної інформації, що найчастіше створюється, накопичується і зберігається на цифрових носіях та передається каналами зв'язку. Компресія відповідних файлів дає змогу пропорційно підвищити швидкість обміну інформацією по мережі та зменшити обсяги використання дискового простору. Тому проблема підвищення ефективності стиснення зображень не втрачає своєї актуальності останні десятиліття і, ймовірно, не втратить у найближчому майбутньому.

На сьогодні існує ряд застосувань, де не допускаються будь-які перетворення відеозображень, що приводять до втрати інформації, наприклад, для представлення фотореалістичних зображень чи синтезованих ілюстрацій та емблем, космічних знімків. у медицині, картографії чи інших галузях діяльності людини. До таких застосувать висуваються вимоги відносно своєчасності обробки та передачі інформації, її високої якості. Зокрема з одного боку існує потреба доставки високоякісних зображень, що мають великі об’єми, в реальному часі з мінімізацією втрат інформації. Але з іншого боку бортові комплекси мають обмежені енергетичні, обчислювальні характеристики ІТ технологій та сеанси зв’язку. Саме в цьому полягає протиріччя. Ефективний напрямок вирішення протиріччя знаходиться у застосуванні методів компактного представлення відеоданих.

Значний вклад в розвиток теорії та розробку методів стиснення внесло багато вчених. Серед них Корольов А.В., Поляков П.Ф., Д. Ватолін, О. Ратушняк, М. Смірнов та ін. З іноземних дослідників великий вклад внесли Зів Дж., Претт У.К., Шеннон К., Хартлі Р.Л. та ін.

Аналіз існуючих технологій, на базі яких реалізується підхід стиснення відеозображень без втрат іннформації, дозволив зробити висновок стосовно того,

що методи кодування без втрати інформації забезпечують низькі коефіцієнти стиску. Такий недолік обумовлено зменшенням якості виявлення закономірностей у масивах відеоданих. Це визвано тим, що існуючі методи без втрат інформації ґрунтуються на усунені статистичної надмірності. Тому існує важлива науково-прикладна задача, яка полягає у підвищенні рівня стиснення відеозображень без втрати інформації.

2. Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконувалась згідно з індивідуальним планом виконання магістерської роботи.

3. Мета і завдання досліджень. Метою магістерської роботи є вдосконалення існуючих алгоритмів кодування відеозображень для автоматизації процесу стиснення у системах обробки відеоінформації, їх програмна реалізація та експериментальне дослідження.

Для досягнення мети необхідно вирішити наступні завдання:

- аналіз відомих методів та алгоритмів кодування відеозображень;
- удосконалення та розроблення нових алгоритмів кодування відеозображень;
- удосконалення та розроблення нових алгоритмів декодування відеозображень;
- налагодження та експериментальні дослідження розроблених програмних засобів.

Об'єкт дослідження система кодування відеозображень.

Предмет дослідження. Алгоритмічні, математичні та програмно-технічні засоби системи обробки відеозображень.

4. Наукова новизна одержаних результатів:

- проведено порівняльний аналіз відомих методів стиснення відеозображень, який показав, що дослідження, які виконуються в даній предметній області орієнтовані в першу чергу на фотореалістичні зображення, оскільки їхній стиск вимагає найбільших витрат і при його

виконанні виникають найбільші труднощі, що дозволило сформувати вимоги до алгоритмів кодування та зробити постановку задачі, що полягає у вдосконаленні методів кодування зображень за рахунок використання двовимірномих напрямів виділення надлишковості;

- розроблено покращений метод стиснення відеозображень на основі групового кодування, який за рахунок використання процесу групування за двома напрямами дозволяє стискати зображення із меншим коефіцієнтом стиснення ніж відомий алгоритм RLE.
- розроблено алгоритм кодера зображення, який на основі вхідного зображення забезпечує його стиснення, в результаті чого формується одновимірний масив меншої розмірності ніж розмір вхідного зображення.
- розроблено алгоритмічне забезпечення функціонування декодера, яке дозволяє реалізовувати алгоритм декодування послідовності чисел, закодованих розробленим кодером і без втрат відновити початкове зображення та бути апаратно чи програмно незалежним з точки зору реалізації.

5. Практичне значення одержаних результатів роботи полягає у тому, що розроблені алгоритми кодування відеозображень програмно реалізовані і дозволяють автоматизувати процеси обробки у системах відообробки.

Зокрема, розроблені алгоритми, аналітичні вирази та програмне забезпечення, що реалізують методи стиснення на основі групового кодування інформації та їх експериментальні дослідження підтверджують високу якість як практичну так і теоретичну у порівнянні з відомими.

6. Публікації та апробація ДР. Основні результати магістерської роботи отримані автором самостійно, на основі власних розробок.

1 СУЧАСНИЙ СТАН КОДУВАННЯ ЗОБРАЖЕНЬ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Інтерес до проблеми стиснення зображень виник більше 35 років тому. Первинна увага дослідників була звернена до питань розробки аналогових методів скорочення смуги частот відеосигналу — підхід, що отримав назву стиснення смуги пропускання. Поява обчислювальної техніки і подальші розробки в області інтегральних мікросхем привели до зміщення інтересу від аналогових методів до цифрових алгоритмів стиснення. Ухвалення відносне недавно декількох ключових міжнародних стандартів стиснення зображень наочно продемонструвало значне зростання даної області — від теоретичних розробок, початих в 1940-х роках К. Шенноном [1] (та іншими вченими), які першими сформулювали імовірнісний підхід до інформації, її уявлення, передачі і стиснення, до практичного застосування цих теоретичних результатів.

В даний час стиснення зображень може розглядатися як «технологія розширення можливостей». На додаток до вищезазначених областей застосування, стиснення зображень є природним способом підтримки сучасних пристроїв введення зображень, кількість яких збільшується, а також все зростаючій складності широкомовних телевізійних стандартів. Більш того, стиснення зображень відіграє істотну роль в багатьох різноманітних і важливих застосуваннях, таких як відеоконференції, дистанційне зондування (використання зображень, що отримуються з супутників, для прогнозу погоди і вивчення земних ресурсів), формування зображень документів, медичні зображення, факсимільна передача (факс), управління безпілотними літальними апаратами у військових, космічних, або інших небезпечних областях.

У даному розділі розглядаються теоретичні і практичні аспекти стиснення зображень, що складають теоретичні основи кодування та теорії інформації.

1.1 Теоретичні основи стиснення інформації

Технологія обробки цифрових зображень досягла в останні роки великих успіхів завдяки тому, що цифрові методи обробки мають ряд переваг перед аналоговими [2-5]. Однак, значною проблемою є великі розміри файлів, які підлягають передачі та обробці. Наприклад, швидкість передачі символів об'єднаного цифрового потоку при передачі кольорових телевізійних зображень цифровими методами в форматі 4:2:2 [5] складе (1.1):

$$\begin{cases} Q_Я = 13{,}5\ Мгц * 8\ біт = 108\ Мбіт/с \\ Q_К = 6{,}75\ МГц * 8\ біт = 54\ Мбіт/с, \\ Q_\Sigma = Q_Я + 2Q_К = 108\ Мбіт/с + 108\ Мбіт/с = 216\ Мбіт/с, \end{cases} \quad (1.1)$$

де $Q_Я$ - швидкість передачі символів цифрового сигнала яскравості;

$Q_К$ - швидкість передачі символів різницевих кольорових сигналів.

Передати такий цифровий потік по існуючих лініях зв'язку вкрай важко. Для зберігання тільки одного телевізійного кадра в цифровій формі необхідно близько 0,5 Мбайта пам'яті, а для однохвилинного репортажа 750 Мбайтів.

Вирішенням проблеми є зменшення об'єму файлів шляхом стиснення даних, або іншими словами кодування зображень.

Актуальність стиснення зображень особливо зросла за останні десять років в зв’язку з появою розвинутих комп’ютерних засобів відображення зображень, представлених в цифровій формі. Це призвело до широкого застосування зображень в комп’ютерних системах та мережах.

Класи зображень

Виділення кодування зображень в окремий клас пов’язано з тим, що зображення – це особливий вид даних, орієнтований на зорове сприйняття людиною, що дозволяє створити спеціальні алгоритми стиснення, які призначені тільки для зображень. Крім того, зображення має надлишковість у двох вимірах, тобто це двовимірний сигнал, що також дає додаткові можливості для стиснення.

Завдяки цьому алгоритми стиснення зображень можуть мати дуже високі характеристики.

Формати графічних файлів можна розділити на два загальні класи: векторні і растрові. Векторні графічні формати для представлення зображення використовують послідовності команд рисування [2-5]. Широко поширеним векторним графічним форматом є метафайл Windows. На рисунку 1.1а показано просте векторне зображення, створене за допомогою команд рисування дуг і прямокутника.

а) растровий символ б) векторний символ

Рисунок 1.1 – Векторне представлення зображення

Векторна графіка представляє зображення як набір геометричних примітивів. Зазвичай вони вибираються як крапки, прямі, кола, прямокутники, а також як сплайни деякого порядку. Об'єктам присвоюються деякі атрибути, наприклад, товщина ліній, колір заповнення. Малюнок зберігається як набір координат, векторів і інших чисел, що характеризують набір примітивів. При відтворенні об'єктів, що перекриваються, має значення їх порядок.

Зображення у векторному форматі може без втрат масштабуватися, повертатися, деформуватися, також імітація тривимірності у векторній графіці простіша, ніж в растровій.

Разом з тим, не будь-яке зображення можна представити як набір з примітивів. Такий спосіб є придатним для схем, використовується для масштабованих шрифтів, ділової графіки, дуже широко застосовується для створення мультфільмів і просто відеороликів різного змісту.

Векторній графіці властиві два основні недоліки. По-перше, вони не підходять для відтворення фотографій або живопису. Наприклад, для відображення картини "Мати" автора Whistler [6], було б потрібно десятки тисяч команд рисування - навіть просто визначення набору команд, необхідних для промальовування картини, було б важким завданням. По-друге, відтворення складних зображень вимагає багато часу. У більшості систем відображення кожен векторний об'єкт повинен перетворюватися в піксельне зображення. Тому, у даній дипломній роботі не розглядатимуться векторні зображення при застосуванні алгоритмів кодування.

Іншим видом відеозображень є растрові зображення, які представляють собою двовимірний масив чисел. Елементи цього масиву називають пікселями (від англійського pixel – елемент зображенн). Кожен елемент масиву визначає колір, який повинен відображуватися в заданому місці розташування. Якщо в пристрої відображення пікселі розташовані досить щільно, людському оку важко помітити структуру масиву точок, який складає зображення [2-5].

Найбільша перевага растрових зображень полягає в їх якості [5]. Основний недолік растрових зображень у великому об'ємі даних, необхідних для їх відтворення. Розмір зображення в байтах (без урахування надлишкових бітів) складає (1.2):

$$\frac{\text{ширина} * \text{висота} * \text{число бітів на піксел} + 7}{8} \qquad (1.2)$$

Таким чином, для відображення або зберігання зображення розміром 600x800, з 24 бітами на піксел, буде потрібно 1440000 байтів пам'яті або дискового простору. Із зростанням об'єму пам'яті компьютерів збільшуються також число і розміри зображень, які можуть відображатися. Для зменшення розміру файлу зображення на диску, зазвичай застосовується процедура стиснення, що дозволяє зберігати більше число зображень.

Іншим недоліком растрових зображень є те, що вони залежні від розміру і непридатні для глобального редагування. При роботі ж із векторними форматами програмі рисування неважко додавати, видаляти і модифікувати окремі елементи. Крім того, над векторним зображенням нескладно виконувати перетворення, такі, як побудова перспективи, укрупнення і масштабування.

На відміну від векторних зображень, для растрових навіть зміна розміру є проблемною. При зменшенні розмірів частина інформації втрачається; при укрупненні зображення виникають ефекти блочності.

У таблиці 1.1 узагальнено переваги векторної і растрової графіки [7]. Важливо відмітити, що жодному з методів не можна віддати повної переваги - для кожного з них існують свої сфери застосування. Деякі програми використовують комбінацію двох технологій.

Таблиця 1.1 – Співставлення растрової і векторної графіки

Критерій порівняння	Растрова графіка	Векторна графіка
Швидкість відображення	+	
Якість зображення	+	
Використання пам’яті		+
Легкість редагування		+
Апаратна незалежність		+

Усі растрові зображення можна поділити на дві групи - з палітрою та без неї [8, 9]. В зображеннях з палітрою у пікселях зберігається число - індекс в деякому одновимірному векторі кольорів, що називається палітрою. Найчастіше за все зустрічаються палітри з 16 та 256 кольорів.

Зображення без палітри бувають в деякій системі представлення кольорів та в градаціях сірого (grayscale). Для останніх значення кожного пікселя інтерпретується як яскравість відповідної точки. Зустрічаються зображення з 2, 16 та 256 рівнями градації сірого кольору. При використанні системи представлення

кольорів кожний піксель представляє собою деяку структуру, полями якої є компоненти кольору. Найбільш розповсюдженою є сисмема RGB, в якій колір представлено значеннями інтенсивності червоної (R), зеленої (G) та синьої (B) компонент. Існують і інші системи представлення кольорів, такі як CMYK, CIE XYZccir60-1 і т.п.

Для того, щоб коректно оцінювати степінь стиснення введемо поняття класу зображень. Під класом будемо розуміти деяку сукупність зображень, для якої застосування алгоритму архівації дає якісно однакові результати. Наприклад, для одного класу алгоритм дає дуже високу степінь стиснення, а для іншого - майже не стискає чи навіть збільшує розмір файлу.

Розглянемо наступні приклади неформального визначення класів зображень [10]:

1. Клас 1. Зображення з невеликою кількістю кольорів (4-16) і великими областями, заповненими одним кольором. Плавні переходи кольорів відсутні. Приклади: ділова графіка - гістограми, діаграми, графіки і т.п.

2. Клас 2. Зображення з плавними переходами кольорів, побудовані на комп'ютері. Приклади: графіка презентацій, ескізні моделі в САПР, зображення, побудовані по методу Гуро.

3. Клас 3. Фотореалістичні зображення. Приклад: відсканованої фотографії.

4. Клас 4. Фотореалістичні зображення з накладанням ділової графіки. Приклад: реклама.

Розвиваючи дану класифікацію, в якості окремих класів можуть бути запропоновані неякісно відскановані в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт. Формально будучи 8- або 24-бітовими, вони несуть навіть не растрову, а чисто векторну інформацію. Окремі класи можуть формувати і зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і факс з електронного досьє [10].

Дослідження, які виконуються в області стиснення зображень орієнтовані в першу чергу на зображення класу 3 – фотореалістичні зображення, оскільки їхній

стиск вимагає найбільших витрат і при його виконанні виникають найбільші труднощі. Дійсно, стиск, наприклад, комп'ютерної графіки може бути легко виконано передачею файлу, що використовується для формування графічного зображення, стиск зображень з малим числом градацій яскравості також не представляє значних труднощів, оскільки при його стиску можуть бути використані алгоритми з малою обчислювальною складністю.

1.2 Кодування зображень

Для управління об'єктами технічних систем обробки відеозображень необхідно здійснювати управління сигналами. По своїй фізичній природі сигнали можуть бути електричними, тепловими, звуковими, світловими, механічними [11-13]. Умовне позначення сигналу для запису і передачі деяких заздалегідь визначених понять називається кодом сигналу [11-13]. Тобто кожен сигнал, з яких складається повідомлення, має свій код (своє позначення). Процес перетворення повідомлення в коди називається кодуванням [13]. Зворотний процес, тобто відтворення закодованої інформації, називається декодуванням [13].

Основна проблема кодування інформації була сформульована ще Клодом Шенноном в 1948 році [1] при побудові системи комунікації, рішення якої по сьогоднішній день застосовується в теорії інформації.

На рисунку 1.2 приведена загальна схема передачі цифровій інформації. Відмітимо, що будь-який фізичний канал передачі сигналів не може бути абсолютно надійним. На рисунку 1.2 це проілюстровано шумом, який погіршує канал і вносить помилки до передаваної цифрової інформації. Шенон показав, що при виконанні деяких достатньо загальних умов є принципова можливість використовувати ненадійний канал для передачі інформації із довільно великим ступенем надійності. Тому немає необхідності намагатися очистити канал від шумів, наприклад, підвищуючи потужність сигналів (це дорого і часто

неможливе). Замість цього слід розробляти ефективні схеми кодування і декодування цифрових сигналів.

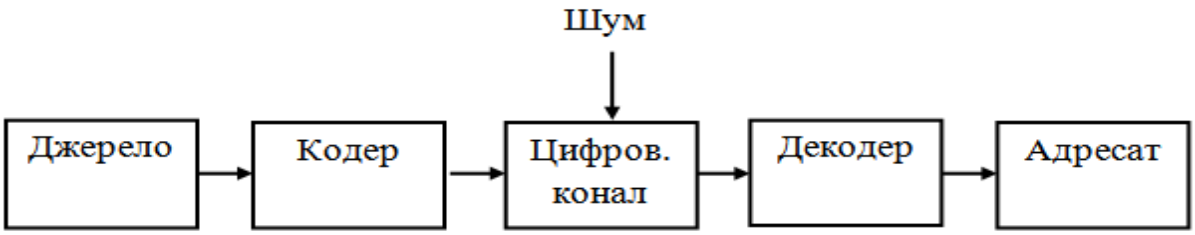


Рисунок 1.2 – Блок-схема системи зв'язку

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 1.3).

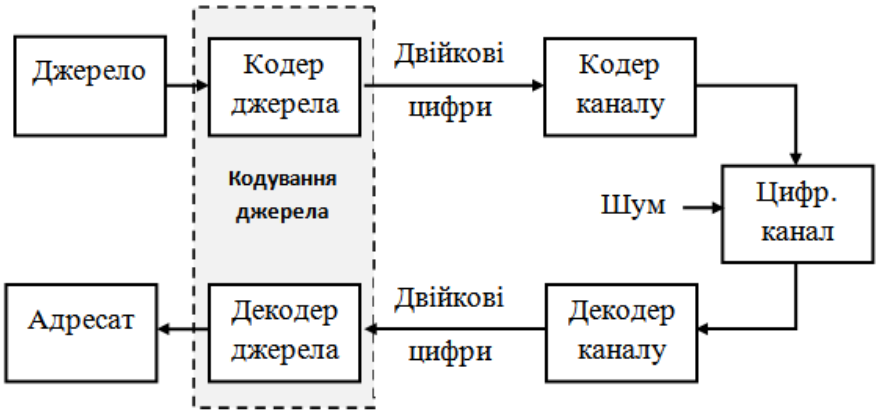


Рисунок 1.3 – Системи зв'язку з роздільним кодуванням.

Завдання кодування каналу полягає в побудові на основі відомих характеристик каналу кодера, що посилає в канал вхідні символи, які будуть декодовані приймачем з максимальним ступенем надійності. Це досягається за допомогою додавання в передавану цифрову інформацію деяких додаткових перевірочних символів. На практиці каналом може служити телефонний кабель, супутникова антена, оптичний диск, пам'ять комп'ютера тощо.

Завданням кодування джерела є створення кодера джерела, що проводить компактний (стиснений) опис початкового сигналу, який необхідно передати

адресатові. Джерелом сигналів може служити текстовий файл, цифрове зображення, оцифрована музика або телепередача. Цей стиснений опис сигналів джерела може бути неточним, тоді слід говорити про розбіжність між відновленим після прийому і декодування сигналом і його оригіналом. Це зазвичай відбувається при перетворенні (квантуванні) аналогового сигналу в цифрову форму.

В даній дипломній роботі у подальших розділах розглядаються алгоритми рішення задачі кодування джерел. Оскільки метою кодування джерел є створення компактного, стислого опису цифрової інформації, цю техніку також прийнято називати стисненням або компресією цифрових даних.

Головна причина використання стиснення даних в комунікаціях полягає в бажанні передавати або зберігати інформацію з найбільшою ефективністю (наприклад використання азбуки Морзе). Як і в звичайному тексті в різних джерелах цифрових даних є надмірні дані, тобто, деякі ділянки, які, "містяться" в інших частинах даних джерела. Тому виникає питання: що є найбільш компактне представлення даних джерела? На це питання відповідь була дана Шеноном [1]. У своїй роботі по теорії інформації він ввів числову характеристику джерела, яка називається ентропією [1]. Фундаментальне значення цієї величини полягає в тому, що вона задає нижню межу можливого стиснення. Крім того, є теорема, яка стверджує, що до цієї межі можна наблизитися скільки завгодно щільно за допомогою відповідного методу кодування джерела. Наприклад, відомо, що ентропія усередненого англійського тексту рівна приблизно 3.2 біт/букву. У комп'ютерному уявленні одна буква займає 8 біт (стандартний код ASCII). Значить, при стисненні типового текстового файлу англійською мовою досягається стиснення приблизно в 2.5 разу.

Ентропія стислих даних співпадає з ентропією початкового джерела. При цьому передбачається, що за стислими даними можна повністю відновити початкову інформацію. Такий підхід прийнято називати стисненням без втрат. Це стиснення, наприклад, застосовується в дистрибутивах програмних продуктів. Це

найбільш вивчена область стиснення даних. Вона є вельми важливою, оскільки більшість методів компресії самих різних типів цифрової інформації часто використовують на певних стадіях алгоритми стиснення без втрат. Таке стиснення ще називається ентропійним стисненням. Надмірністю даних можна назвати різницю між їх об'ємом і ентропією. Тому, компресія без втрат є екстремальним випадком стиснення, при якому ентропія даних залишається незмінною.

1.2.1 Предикативне кодування

У використовуваних в даний час алгоритмах кодування також працює за загальною схемою, що називається предикативним кодуванням (рисунок 1.4).

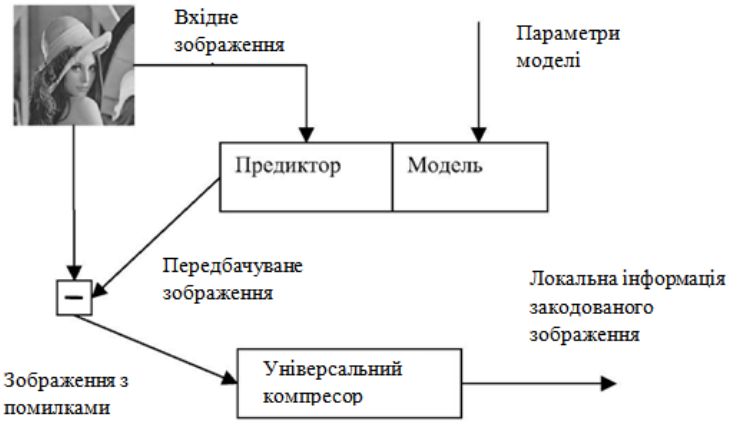


Рисунок 1.4 – Загальна схема предиктивного кодування

Предикативне кодування - це метод кодування, при якому модель зображення використовується для прогнозу значення пікселя зображення на підставі значень ряду відомих пікселів (як правило, сусідніх).

Модель зображення в даному методі кодування - це функція виду $V=f(x, y, n, p)$, яка обчислює (передбачає) значення пікселя з координатами (x, y) залежно від вектора значень n сусідніх пікселів, значення яких вже відомі, і вектора параметрів зображення p.

У загальному випадку процес предиктивного кодування відбувається таким чином [15]:

1. Значення першого пікселя v_1 передається на вхід універсального компресора без змін (або ж відразу передається в локальну область закодованого зображення). Піксель позначається як закодований;

2. По деякому алгоритму (алгоритмом обходу площини), вибирається наступний кодований піксель. Наприклад, часто площина пікселів обходиться зліва-направо і зверху-вниз;

3. Використовуючи модель зображення, обчислюється передбачене значення поточного пікселя $p(i)=f(x_i, y_i, n, p)$, де у вектор n включаються всі пікселі, помічені як закодовані;

4. Обчислюється помилка прогнозу $e(i)=v(i)-p(i)$,которая подається на вхід універсального компресора. Поточний піксель позначається як закодований;

5. Якщо зображення закодоване повністю, то процес кодування завершений, інакше перехід до кроку 2.

З приведеного алгоритму видно, що описана частина предикативного кодування не здійснює безпосередньо стиснення даних, а тільки деяким чином змінює їх. Насправді, безпосереднім стисненням займається остання частина кодувальника, універсальний компресор, який є звичайним компресором, здатним стискати будь-які дані. Описаний же алгоритм використовується для того, щоб підвищити ефективність його роботи. Це відбувається таким чином.

Мінімальний очікуваний об'єм даних на виході універсального компресора в більшості випадків можна оцінити наступною формулою (для найчастіше вживаних статистичних компресорів) (1.3) [14, 15]:

$$l = \sum p(i) \log_2 P(i), \qquad (1.3)$$

де $P(i)$ - відносна частота появи на вході значення і.

Користуючись цією формулою, можна довести, що довжина коди зменшуватиметься при збільшенні «нерівності» серед значень вхідного потоку, тобто збільшення частоти одних значень і зменшення частоти інших.

Розглянемо гістограму відносних частот появи значень для деякого тестового зображення (рисунок 1.5).

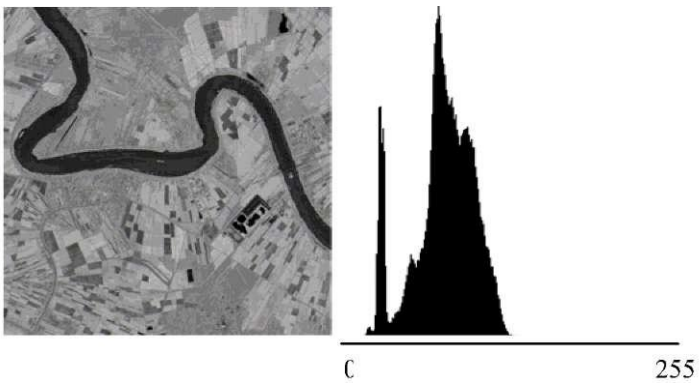


Рисунок 1.5 – Тестова зображення і його гістограма частот

Розглянемо тепер те ж саме зображення, подане на вхід універсального компресора після проходження перших етапів деякого предикативного кодування і аналогічну гістограму для нього (рисунок 1.6).

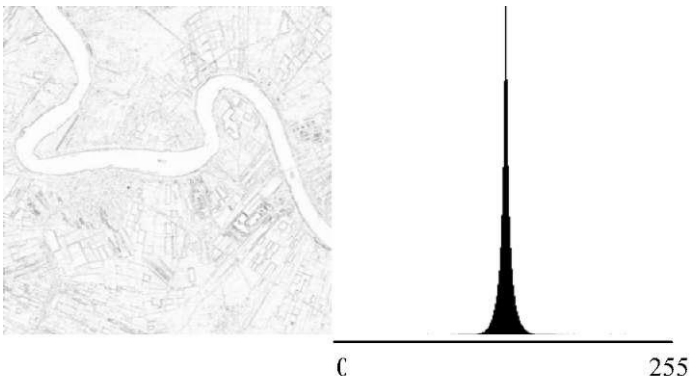


Рисунок 1.6 – Закодоване зображення і його гістограма відносних частот

Можна відмітити, що після кодування розподіл відносних частот значень пікселів став більш нерівномірним, ніж у початкового зображення, що, як було сказано вище, в більшості випадків сприяє зменшенню довжини коду на виході універсального компресора.

Різні варіанти алгоритмів предиктивного кодування розрізняються між собою, перш за все, видом використовуваного предиктора (моделі),

використовуваним універсальним компресором, а також включенням в схему додаткових етапів, службовців для поліпшення коефіцієнта стиснення.

Предиктори, що використовуються в кодувальниках, діляться по статичності моделі на:

- неадаптивні - моделі без параметрів, що використовують один і той же предиктор для різних зображень;

- напівадаптивні - моделі з параметрами, значення яких підбираються індивідуально для кожного оброблюваного зображення, але не міняються в процесі кодування;

- блоково-адаптивні - моделі, параметри яких розраховуються не для всього зображення цілком, а для окремих його частин;

- адаптивні - параметри моделі змінюються по ходу кодування зображення.

Також предиктори діляться по вигляду моделі:

- лінійні, такі, що виражають поточний піксель через лінійну комбінацію закодованих пікселів;

- нелінійні, такі, що використовують для цього виразу деяку нелінійну функцію;

Також можна виділити два підвиди нелінійних предикторів. Це нейромережеві, що використовують для прогнозу нейронну мережу, що навчається, і генетичні, що використовують для прогнозу методи генетичного програмування.

1.2.2 Лінійне предиктивне кодування

Лінійне предиктивное кодування (LP-кодування) - це вид предиктивного кодування, при якому значення поточного символу передбачається через лінійну комбінацію вже закодованих пікселів, тобто (1.4)

$$P(i) = \sum_{j=i-1}^{i-n} k_j v(j) \qquad (1.4)$$

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.

Основне завдання при LP-кодевання полягає у визначенні коефіцієнтів k_j, що забезпечують якомога точніший прогноз [15]. Більш простий варіант реалізації полягає в заданні фіксованих коефіцієнтів k_j,однакових для кожного зображення. Отриманий таким чином предиктор буде статичним.

Для розрахунку оптимальних значень коефіцієнтів предиктора можна відмітити, що кожен елемент мультимедійної інформації, як правило, відхиляється від значення сусідніх елементів по двох причинах: із-за «сильних» змін, обумовлених характером самих даних, - тренда, і «слабких» фонових коливань - шуму [15]. Тому можливі два протилежні типи моделей:

- внесок шуму невеликий у порівнянні з внеском тренда;
- внесок тренда невеликий у порівнянні з внеском шуму.

У першому випадку розумно передбачати значення поточного елементу на підставі тенденції, що склалася, в другому - як рівне середньому арифметичному яких-небудь попередніх елементів.

Найчастіше при реалізації LP-кодування обхід площини відбувається зліва-направо і зверху-вниз, тому надалі припускатимемо саме цей варіант обходу.

Розглянемо найбільш відомі із статичних лінійних предикторов. Позначимо пікселі як показано на рисунку 1.7.

		NN	NNE
	NW	N	NE
WW	W	p_i	

Рисунок 1.7 – Позначення сусідніх пікселів зображення

Прості із статичних лінійних предикторов передбачають значення поточного пікселя як значення одного з сусідніх пікселів, або не передбачають зовсім (значення пікселя відразу подається на вхід універсального компресора). Формули для цих предикторов [15]:

– *p(i)=0* - прогноз не використовується;

– *p(i)=N* - значення поточного пікселя передбачається рівним значенню верхнього пікселя;

– *p(i)= W*- значення поточного пікселя передбачається рівним значенню лівого пікселя;

– *p(i)=NW* - значення поточного пікселя передбачається рівним значенню лівого верхнього пікселя.

Все ці предикторы відповідають шумовій моделі. У сучасних варіантах алгоритму в чистому вигляді вони не використовуються, оскільки забезпечують досить поганий прогноз. В даний час вони використовуються як складові частини адаптивних предикторов.

Наступна група предикторов використовує складніші формули, що враховують значення декількох сусідніх пікселів і трендові або шумові зміни між ними [15]:

– *p(i)=N+W-NW* - вважаємо, що різниця значень поточного пікселя і верхнього рівна різниці між лівим і лівим верхнім;

– *p(i)=N+(W-NW)/2* - вважаємо, що різниця значень поточного пікселя і верхнього рівна половині різниці між лівим і лівим верхнім;

– *p(i)=W+(N-NW) /2* - вважаємо, що різниця значень поточного пікселя і лівого рівна половині різниці між верхнім і лівим верхнім;

– *p(i)=(N+W)/2* - значення поточного пікселя передбачається рівним середньому арифметичному значень лівого і верхнього пікселя.

Перші три предиктора відповідають трендовой моделі, четвертий - шумовий. Предіктори цієї групи дають кращий прогноз, проте, теж найчастіше використовуються як складові частини адаптивних предикторов.

1.2.3 Нелінійні предиктори

Нелінійні статичні предиктори є складнішими у порівнянні з лінійними, вони вже використовуються при кодуванні зображень самостійно, а не у складі біліш складних предикторів. Більшість з них базується на аналізі вже закодованих символів і виділенні яких-небудь особливостей зображення (виділенні ребер, градієнтів і т.п.) в поточній області. Ці предиктори також можна задати у вигляді функцій, проте прийнятий наочніший їх опис - у вигляді алгоритму розрахунку p_i на псевдомові.

Предиктор Paeth. Даний пре диктор розроблений Аланом В. Паєфа [15] і описується наступним алгоритмом (рисунок 1.8):

IF $|N - NW| \le |W - NW|$ AND $|N - NW| \le |N + W - 2NW|$

$p(i) = W$

ELSE IF $|W - NW| \le |N + W - 2NW|$

$p(i) = N$

ELSE

$p(i) = NW$

Рисунок 1.8 – Алгоритм пре диктора *Paeth*

Предиктор DARK, запропонований фірмою Kodak, адаптується до горизонтальних і вертикальних ребер [15]. Властивості предиктора представлено формулою (1.5):

$$\Delta_v = |W - NW|, \Delta_h = |N - NW| \quad (1.5)$$

де, Δ_v - величина вертикального градієнта;

Δ_h - величина горизонтального градієнта.

Коефіцієнт, що характеризує переважання вертикального градієнта над горизонтальним представлено формулою (1.6).

$$a = \frac{\Delta_v}{\Delta_h - \Delta_v} \tag{1.6}$$

Чим більший даний коефіцієнт, тим з більшою вагою необхідно брати для прогнозу лівий піксель і з меншою - верхній (1.7)

$$p(i) = aW + (1 - a)N. \tag{1.7}$$

Предиктор *MED* (median edge detection - визначення середини ребра). Алгоритм даного пре диктора запропонований HP Labs [15] і представлено на рисунку 1.9.

IF $NW \geq \max(N,W)$ THEN

$p(i) = \min(N,W)$

ELSE IF $NW \leq \min(N,W)$

$p(i) = \max(N,W)$

ELSE

$p(i) = N + W - NW$

Рисунок 1.9 – Алгоритм предиктора *MED*

Принцип роботи предиктора полягає в адаптації до наявності локальних горизонтальних або вертикальних ребер. Піксель *N* використовується для прогнозу в тому випадку, якщо виявлено вертикальне ребро. Піксель *W* - якщо виявлено горизонтальне ребро. Якщо ребро не було виявлене, то використовується один із статичних лінійних предикторів.

Предіктор *GAP* (gradient-adjusted predictor, предиктор з настроюванням по градієнту) запропонований в алгоритмі CALIC [15]. Цей предиктор адаптує передбачене значення відповідно до локального градієнта. Використовується наступний алгоритм (рисунок 1.10) із розрахунком величин (1,8):

$$d_h = |W- WW| + |N-NW| + |N- NE|,\ d_v = |W - NW| + |N- NN| + |NE - NNE| \quad (1.8)$$

де d_h - величина горизонтального градієнта;

d_v -величина вертикального градієнта.

```
IF (dv-dh>80)  //четкое горизонтальное ребро
    p(i) = W
ELSE IF (dv-dh<-80)  //четкое вертикальное ребро
    p(i) = N
ELSE {
    p(i) = (N + W) / 2 + (NE - NW) / 4;
    IF (dv-dh>32)  //горизонтальное ребро
        p(i) = (p(i) + W) / 2
    ELSE IF (dv-dh>8)  //слабовыраженное горизонтальное ребро
        p(i) = (3p(i) + W) / 4
    ELSE IF (dv-dh<-32)  //вертикальное ребро
        p(i) = (p(i) + N) / 2
    ELSE IF (dv-dh<-8)  //слабовыраженное вертикальное ребро
        p(i) = (3p(i) + N) / 4
}
```

Рисунок 1.10 – Алгоритм предиктора *GAP*

Даний алгоритм працює за наступним принципом. Якщо величина вертикального градієнта більша горизонтального на деяке порогове значення (в даному випадку 80), то вважається, що в даній ділянці зображення знаходиться чітко виражене горизонтальне ребро, і тому передбачається значення поточного пікселя рівним значенню лівого пікселя. Аналогічним чином, якщо величина

горизонтального градієнта більша вертикального на 80, то передбачається значення поточного пікселя рівне значенню верхнього пікселя.

Інакше передбачається значення поточного пікселя по формулі (1.9) (звичайний лінійний предиктор):

$$p(i)= (N + W)/2 + (NE - NW)/4 \qquad (1.9)$$

Якщо різниця градієнтів досягла будь-якого іншого порогового значення (32 або 8 в даному випадку), то за передбачене значення приймаєnmcz середнє зважене вже знайденого значення і відповідного пікселя з різними вагами.

1.2.4 Адаптивне кодування

Всі описані вище предиктори відносилися до класу статичних, вони застосовували одні і ті ж перетворення над сусідніми пікселями для прогнозу значення поточного, тобто коефіцієнти і структура моделі не змінювалися в процесі кодування. У зв'язку з цим вони не дозволяли врахувати особливості локальних областей зображення. Розглянемо тепер алгоритми з адаптивними предикторами, параметри яких змінюються залежно від цих особливостей [15].

Алгоритми, що належать до цієї групи можна розбити на два великі класи. До першого класу можна віднести предиктори, що комбінують результати прогнозу статичних предикторів з вагами, що адаптивно змінюються, тобто які діють по формулі (1.10)

$$p(i) = \sum_{j=1}^{N} a_j p_j(i), \qquad (1.10)$$

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 (1.11):

$$G_j = \sum_{k=i-1}^{i-L} \left| v(k) - p_j(k) \right|, \tag{1.11}$$

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 по формулі (1.12):

$$a_j = \frac{1/G_j}{\sum_{i=1}^{N} 1/G_i}, \tag{1.12}$$

До другого класу відносяться предиктори, в яких для прогнозу використовується формула звичайного LP-кодування (1.13):

$$p(i) = \sum_{j=1}^{N} k_j v(i-j), \tag{1.13}$$

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

Для динамічного визначення значень цих коефіцієнтів, що зменшують помилку прогнозу, необхідно вирішити оптимізаційну задачу. Найбільш популярний метод рішення цієї задачі - метод найменших квадратів.

Суть цього методу полягає в тому, що якнайкращими значеннями коефіцієнтів k_j вважаються ті, які мінімізують суму квадратів помилок прогнозу для m попередніх пікселів. Величина E, яку необхідно мінімізувати, розраховується по формулі (1.14):

$$E = \sum_{j=i-1}^{i-m} e^2(j) = \sum_{j=i-1}^{i-m} (p(j) - v(j))^2 = \sum_{j=i-1}^{i-m} \left(\sum_{t=1}^{n} k_t v(j-t) - v(j) \right)^2 . \quad (1.14)$$

Для того, щоби дана величина була мінімальною, необхідне виконання наступної умови (1.15):

$$\frac{\partial E}{\partial k_i} = 0, t = 1..n. \quad (1.15)$$

Підставляючи значення *E* в (1.15), отримаємо систему *n* лінійних рівнянь з *n* невідомими, розв'язуючи яку і отримаєм шукані значення коефіцієнтів k_t для кожного пікселя зображення.

Розв'язок даної системи буде наступним (1.16):

$$K = Y * X^{-1}, \quad (1.16)$$

де $K=[k_t]$ - вектор шуканих коефіцієнтів, $X = \begin{pmatrix} v(i-2) & \dots & v(i-n-1) \\ \dots & \dots & \dots \\ v(i-m-1) & \dots & v(i-m-n) \end{pmatrix}$ - матриця відомих значень, $Y=[v(j)]$ - вектор прогнозних значень.

Отриманий за даною формулою вектор коефіцієнтів *K* міститиме оптимальні по методу найменших квадратів коефіцієнти для даного пікселя.

1.3. Способи стиснення зображень, вимоги до алгоритмів та постановка задачі дослідження

1.3.1 Аналіз основних методів кодування зображень

Базовим методом цифрового кодування джерел зображень є імпульсно-кодова модуляція (ІКМ). Вона характеризується тим, що кожному закодованому в цифрову форму слову відповідає квантований в часі і по амплітуді відлік відеоінформації. При цьому повинні виконуватись вимоги теореми дискретизації

$f_д > 2W_0$, де W_0 максимальна частота, яка міститься в сигналі. Щоб запобігти появі фальшивих контурів на однокольоровому зображенні необхідно більше 50 рівнів квантування, що відповідає 6-8 розрядному кодовому слову на кожний елемент (піксель) зображення. Через великі об'єми інформації ІКМ застосовується лише при внутрістудійній передачі телевізійних зображень паралельним кодом і в якості базового канонічного подання зображення у цифровій формі [16-18].

Серед методів кодування з передбаченням найбільш досліджена диференційно-імпульсна кодова модуляція (ДІКМ). Суть ДІКМ така. Організується передбачення значення яскравості кожного наступного елемента зображення на основі лінійної комбінації значень яскравості попередніх елементів. Оцінка отримана в результаті передбачення віднімається від істинного значення яскравості елемента зображення і різницевий сигнал квантується невеликим числом рівнів. За рахунок цього і досягається скорочення об'єму даних[16,18,19].

Методи кодування зображень з передбаченнями дозволяють стиснути зображення в 2-2,5 рази при простій технічній реалізації. Серед недоліків цих методів необхідно відзначити такі:

- похибки в місцях різких перепадів яскравості;
- низька завадостійкість.

Інтерполяційні методи основані на числових методах апроксимації, за допомогою яких послідовність або двомірний масив відліків яскравості наближено подаються через неперервні функції [2,10]. При цьому кодуються лише окремі відліки зображення, а сусідні з ними отримують в результаті інтерполяції поліномами, звичайно, не більше чим третього ступеня. Коефіцієнт стиснення, який досягається при використанні цих методів дорівнює 5-6.

Основним недоліком інтерполяційних методів є великий об'єм обчислень при інтерполяції поліномами високих ступеней, а також необхідність зберігання координат базових відліків зображення.

Важливий клас методів кодування зображень - це методи кодування на основі перетворень. Кодування на основі перетворень непрямий метод.

Зображення піддаються унітарному математичному перетворенню, отримані в результаті коефіцієнти перетворення квантуються і кодуються для передачі по каналу зв'язку або запису в файл. Для більшості зображень значення багатьох коефіцієнтів перетворення порівняно мале. Такі коефіцієнти часто можливо відкинути або відвести для їх кодування невелике число двійкових розрядів. Найбільш часто використовують двомірні ортогональні перетворення. Це такі як перетворення Карунена-Лоева, дискретне перетворення Фур'є(ДПФ), дикретне косинусне перетворення (ДКП), перетворення Уолша-Адамара [2,4,5,20] та інші. Коефіцієнти стиснення при застосуванні цих методів можуть досягати 6-8, завдяки чому деякі з цих методів(ДКП) знайшли широке застосування як основа промислових стандартів по кодуванню зображень. Спільним недоліком цих методів є достатньо висока обчислювальна складність, а також неможливість розв'язання ряду задач обробки зображень на скороченому об'ємі даних.

Серед статистичних методів найбільш широке застосування знаходять блочні методи кодування зображень. Блоки розміром MxN елементів кодуються у відповідності з імовірністю їх появи. Для найбільш ймовірних конфігурацій використовуються короткі кодові слова, а для менш ймовірних довгі кодові слова(алгоритм Хаффмена), в результаті чого досягається стиснення даних[21]. Коефіцієнти стиснення при використанні цих методів можуть досягати 4-5.

Перспективним для кодування як рухомих так і нерухомих зображень є метод покомпонентного кодування [22-24]. Особливістю цього методу є формування декількох двомірних сигналів, що несуть інформацію про деталі зображення різного розміру. Наявність декількох каналів роздільної обробки деталей зображення різного розміру дозволяє ефективно кодувати зображення як внутрікадровими так і міжкадровими методами з урахуванням особливостей сприйняття інформації зоровим аналізатором людини. Важливою перевагою цього методу є те, що формати подання даних після стиснення є компонентами вихідного зображення, тобто кожна компонента візуально представляє

зображення з тим чи іншим ступенем роздільної здатності, а це в свою чергу дозволяє розв'язувати ряд задач обробки зображень на скороченому об'ємі даних.

Хоча практично досягнутий коефіцієнт стиснення при застосуванні цього методу незначно менший в порівнянні з методами кодування з перетвореннями, його технічна реалізація значно простіша і відповідно швидкодія значно більша.

Для стиснення зображень у реальному часі по сукупності таких параметрів як простота технічної реалізації, обчислювальна складність, коефіцієнт стиснення найкращі результати одержують при використанні Wavelet-кодування, що базується на пірамідальних схемах розкладу початкового зображення на компоненти(S-перетворення) [8]. Цей алгоритм орієнтований на стиснення кольорових і чорно-білих зображень з плавними переходами. Ідеальний для картинок типу рентгенівських фотографій. Коефіцієнт стиснення варіюється в межах 5-100. При великих коефіцієнтах стиснення на різких границях, особливо діагональних, можливі спотворення. Важливою перевагою S-перетворення є можливість показати “огрублене” зображення (низької роздільної здатності), використавши тільки початок файлу.

Найбільші коефіцієнти стиснення забезпечує метод фрактального стиснення зображень [25,26,27], відкритий в 1988 році. Процес фрактального стиснення оснований на твердженні, що зображення реального світу мають афінну надлишковість. Коефіцієнти стиснення можуть досягати 50-60 раз. Основним недоліком цього методу є велика обчислювальна складність. Однак, враховуючи великий степінь стиснення, який можна отримати цим методом, а також гігантський прогрес у збільшенні продуктивності мікропроцесорів та інших апаратних засобів слід очікувати самого широкого застосування даного методу в найближчі роки.

1.3.2 Базові стратегії стиснення

Базових стратегій стиснення три [8]:

1. Перетворення потоку ("Ковзаюче вікно-словник"). Опис даних, що поступають, через вже оброблені. Сюди входять LZ-методи для потоків "слів",

тобто коли комбінації елементів, що поступають, передбачені по вже оброблених комбінаціях, наприклад алгоритмами RLE, LPC, DC, MTF, VQ, SEM, Subband Coding, Discrete Wavelet Transform.

Ніякої імовірності, на відміну від другої стратегії, не обчислюється. В результаті перетворення може бути сформовано декілька потоків. Навіть якщо сумарний об'єм потоків збільшується, їх структура поліпшується і подальше стиснення можна здійснити простіше, швидше і краще.

2. Статистична стратегія.

а) Адаптивна (потокова). Обчислення імовірності для вхідних даних на підставі статистики за вже обробленими даними. Для кодування з використанням цієї обчисленої імовірності використовують сімейство PPM-методів - для потоків "слів", адаптивні варіанти методів Хаффмана і Шенона - Фано, арифметичного кодування - для потоків "елементів" [8]. На відміну від першого випадку, давно зібрана статистика має ту ж вагу, що і недавня, якщо метод не бореться з цим спеціально, що набагато складніше, ніж у разі LZ. Крім того, вважаються імовірними всі комбінації, навіть ті, які ще не зустрічалися в потоці і швидше за все ніколи не зустрінуться.

б) Блокова. Окремо кодується і додається до стислого блоку його статистика. Статичні варіанти методів Хаффмана, Шенона - Фано і арифметичного кодування - для потоків "елементів".

3. Перетворення блоку. Вхідні дані розбиваються на блоки, які потім повністю трансформуються, а у разі блоку однорідних даних краще брати весь блок, який потрібно стиснути. Це методи сортування блоків ("BlockSorting"-методы: ST, BWT, PBS), а також Fourier Transform, Discrete Cosine Transform, фрактальні перетворення, Enumerative Coding.

Як і при першій стратегії, в результаті можуть формуватися декілька блоків, а не один. Знову ж таки, навіть якщо сумарна довжина блоків не зменшується, їх структура значно поліпшується і подальше стиснення відбувається простіше, швидше і краще.

Резюмуючи однією пропозицією: метод стиснення може бути або статистичним, чи таким, що трансформує і обробляти дані або потоково, або блоками, причому

– чим однорідніші дані та пам’ять, тим ефективніші блокові методи;

– чим менші і неоднорідніші дані та пам'ять, тим ефективніші потокові методи;

– чим складніше джерело, тим сильніше поліпшить стиснення оптимальні перетворення;

– чим простіше джерело, тим ефективніше прямолінійне статистичне рішення (математичні моделі "джерело Бернуллі" і "джерело Марков").

1.3.3 Вимоги додатків до алгоритмів компресії

У попередній частині даного параграфу визначено, які застосування є споживачами алгоритмів кодування зображень. В той же час додаток визначає характер використання зображень (велика кількість зображень зберігається і використовується, або зображення копіюються по мережі, або зображення великі по розмірах і потрібна можливість отримання лише його частини). Характер використання зображень задає міру важливості наступних нижче суперечливих вимог до алгоритму:

1. Висока міра компресії. Помітимо, що далеко не для усіх застосувань актуальна висока міра компресії. Крім того, деякі алгоритми дають краще співвідношення якості до розміру файлу при високих мірах компресії, проте програє іншим алгоритмам при низьких мірах.

2. Висока якість зображень. Виконання цієї вимоги безпосередньо суперечить виконанню попередньої.

3. Висока швидкість компресії. Це вимога для деяких алгоритмів з втратою інформації є взаємовиключною з першими двома. Інтуїтивно зрозуміло, що чим більше часу аналізується зображення, намагаючись отримати найвищу міру компресії, тим кращим буде результат. І, відповідно, чим менше часу

витрачається на компресію (аналіз), тим нижчою буде якість зображення і більшим його розмір.

4. Висока швидкість декомпресії. Досить універсальна вимога, актуальна для багатьох застосувань. Проте можна привести приклади додатків, де час декомпресії далеко не критичний.

5. Масштабування зображень. Ця вимога має на увазі легкість зміни розмірів зображення до розмірів вікна активного застосування. Річ у тому, що одні алгоритми дозволяють легко масштабувати зображення прямо під час декомпресії, тоді як інші не лише не дозволяють легко масштабувати, але і збільшують імовірність появи неприємних артефактів після застосування стандартних алгоритмів масштабування до декомпресованого зображення. Наприклад, можна навести "погане" зображення алгоритму JPEG - це зображення з досить дрібним регулярним малюнком (піджак в дрібну клітину). Характер JPEG спотворень, що вносяться алгоритмом такий, що зменшення або збільшення зображення може дати неприємні ефекти.

6. Можливість показати огрублене зображення (низької роздільної здатності), використавши тільки ядро файлу. Ця можливість актуальна для різного роду мережевих застосувань, де перекачування зображень може зайняти досить великий час і , отримавши ядро файлу, коректно показати preview. Помітимо, що примітивна реалізація вказаної вимоги шляхом записування зображення його зменшеною копією помітно погіршить міру компресії.

7. Стійкість до помилок. Ця вимога означає локальність порушень в зображенні при псуванні або втраті фрагмента файлу. Ця можливість використовується при широкомовленні (broadcasting) зображень по мережі, тобто в тих випадках, коли неможливо використовувати протокол передачі, що повторно запитує дані з сервера при помилках. Наприклад, якщо передається відеоряд, то було б неправильно використовувати алгоритм, у якого збій призводив би до припинення правильного показу усіх наступних кадрів. Ця вимога суперечить високій мірі архівації, оскільки інтуїтивно зрозуміло, що

необхідно вводити в потік надлишкову інформацію. Проте для різних алгоритмів об'єм цієї надлишковою інформації може істотно відрізнятися.

8. Врахування специфіки зображення. Вища міра архівації для класу зображень, які статистично частіше будуть застосовуватися в застосуванні.

9. Можливість редагування. Тут розуміється мінімальна міра погіршення якості зображення при його повторному збереженні після редагування. Багато алгоритмів з втратою інформації можуть істотно зіпсувати зображення за декілька ітерацій редагування.

10. Вартість апаратної реалізації. Вимоги до алгоритму реально пред'являють не лише виробники ігрових приставок, але і виробники багатьох інформаційних систем. Так, декомпресор фрактального алгоритму дуже ефективно і коротко реалізується з використанням технології MMX і розпаралелювання обчислень, а стискування за стандартом CCITT Group 3 легко реалізується апаратно.

Очевидно, що для конкретного завдання будуть дуже важливі одні вимоги і менш важливі (і навіть абсолютно не важливі) інші.

Отже на практиці для кожного завдання можна сформулювати набір пріоритетів з вимог, викладених вище, який і визначить найбільш відповідний в заданих умовах алгоритм (або набір алгоритмів).

Використовуючи ці критерії, нижче розглядаються алгоритми архівації зображень. При цьому слід зауважити, що один і той же алгоритм часто можна реалізувати різними способами. Багато відомих алгоритмів, такі як RLE, LZW або JPEG, мають десятки реалізацій, що відрізняються. Крім того, у алгоритмів буває декілька явних параметрів, варіюючи якими можна змінювати характеристики процесів архівації і розархівування. При конкретній реалізації ці параметри фіксуються, виходячи з найбільш вірогідних характеристик вхідних зображень, вимог на економію пам'яті, вимог на час архівації і т. д. Тому у алгоритмів одного сімейства коефіцієнти можуть відрізнятися але зображення не зміниться.

Таким чином у першому розділі розглянуто способи кодування зображень у сучасних системах відеообробки, а також класифікацію відеозображень та програмного забезпечення, що дозволило сформувати вимоги до алгоритмів кодування особливості яких розглянуто в наступному розділі дипломної роботи.

2. ЗАПРОПОНОВАНИЙ АЛГОРИТМ СТИСНЕННЯ ВІДЕОЗОБРАЖЕНЬ

Як представлено у попередньому розділі дипломної роботи, одним із видів алгоритмів стиснення відеозображень є алгоритми із незмінною ентропією інформації. Такі алгоритми після виконання процесів кодування та наступного за ним декодування не змінюють вхідних даних, тобто без втрат забезпечують відтворення зображень. Достатньо великою групою алгоритмів стиснення відеозображень без втрати інформації виконують на основі групового кодування. У даному розділі розглядаються сучасні алгоритми стиснення відеозображень із використанням групового кодування та способи їх покращення.

2.1 Покращений підхід групового кодування

Типовим представником алгоритмів групового кодування є RLE [27]. У груповому кодування як правило використовується процес впорядкування усіх стрічок зображення в одну (рисунок 2.1) після чого відбувається різні способи усунення надлишковості повторюваних символів або чисел за рахунок її заміни на деяку послідовність.

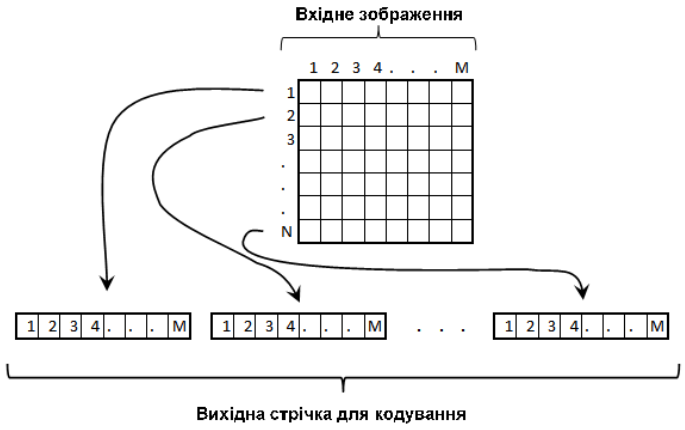


Рисунок 2.1 – Формування стрічки для групового кодування

В алгоритмі RLE такою послідовністю є два байти інформації, перший з яких містить один біт для позначення признаку повторюваності наступних

значень у стрічці кодування та решта 7 біт для позначення кількості повторювань (максимум 128 повторень). Наступним байтом представляється значення кольору пікселя зображення, що повторюється у стрічці кодування. Окрім цього, для відтворення двовимірної структури зображення під час декодування (рисунок 2.2) необхідно зберігати додаткові байти, що визначають розмірність зображення *M*x*N*.

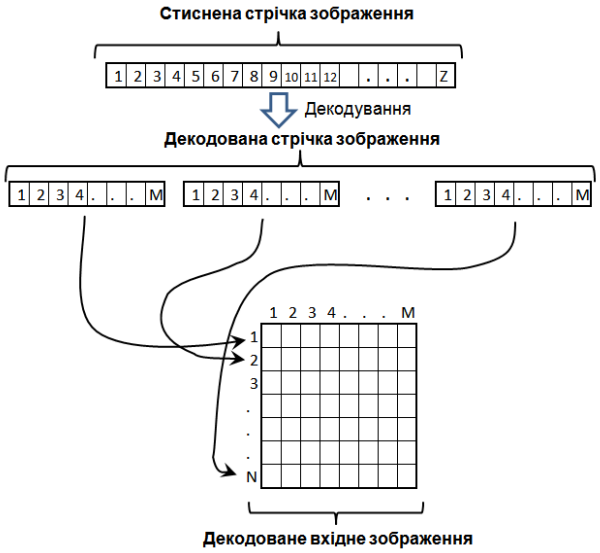


Рисунок 2.2 – Процес декодування зображення

Таким чином у груповому кодуванні за алгоритмом RLE, на кожну повторюючу і неповторюючу послідовність відводиться по одному додатковому байту плюс байти на розмірність зображення (2.1):

$$redudancy = \sum_{i=1}^{K} R_i + \sum_{j=1}^{L} \overline{R}_j + S_{NxM}, \tag{2.1}$$

де *redundancy* – надлишковість байт;

R_i – кількість повторюючих послідовностей;

$\overline{R}_j$ - кількість неповторюючих послідовностей;

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

Додатково слід зазначити, що оскільки для позначення повторюючих чи неповторюючих послідовностей відводиться 7 біт, то такі додаткові байти будуть обов'язково присутні щонайрідше на кожні 128 елементів послідовності. Тобто, навіть за умови, коли у зображенні немає повторюваностей, то будуть додаватись додаткові байти, що призводить до надлишковості. Аналогічно і для повторюваних послідовностей, у випадку, коли розмірність таких послідовностей перевищуватиме 128 елементів, будуть вводитись додаткові байти для позначення таких послідовностей. Такий підхід свідчить про той факт, що коли на зображенні немає повторюваностей, то можлива ситуація, коли розмір стисненого зображення може перевищувати розмір вхідного зображення (так звана від'ємна компресія). Подібна ситуація також можлива, коли на зображенні є часті зміни між повторюваними і неповторюваними послідовностями.

З метою покращення роботи алгоритмів групового кодування запропоновано новий, що дозволяє покращити компресійні характеристики методу RLE.

Основна ідея запропонованого алгоритму полягає у збільшенні розміру груп пікселів зображення, що повторюються і зменшенні при цьому надмірності, що присутня на зображенні за рахунок однакових кольорів. На відміну від існуючого алгоритму RLE пропонується не формувати єдину стрічку для стиснення даних (рисунок 2.1) з метою пошуку групових послідовностей, а здійснювати аналіз послідовності у 2D-просторі. Тобто пропонується аналізувати послідовності пікселів зображення за двома напрямами на площині зображення як сукупності рядків і стовбців (рисунок 2.3) з метою виявлення максимальної площі послідовностей, що містять надлишковість (повторюються).

У багатьох відеозображеннях існують ділянки однорідних кольорів (рисунок 2.3), що викликані відображенням фізичних об'єктів реального світу, які містять одноколірні елементи. Тому на відміну від RLE, який аналізуватиме кожен рядок послідовності і вноситиме додаткові байти у компресовану вихідну послідовність, для позначення повторюваних послідовностей запропоновано

використати позначення максимальних прямокутних (квадратних) ділянок зображень із послідовнвностями пікселів зображення, що містять однакові кольори лише один раз. За рахунок використання такої особливості можна отримати вищий коефіцієнт компресії ніж відомий метод.

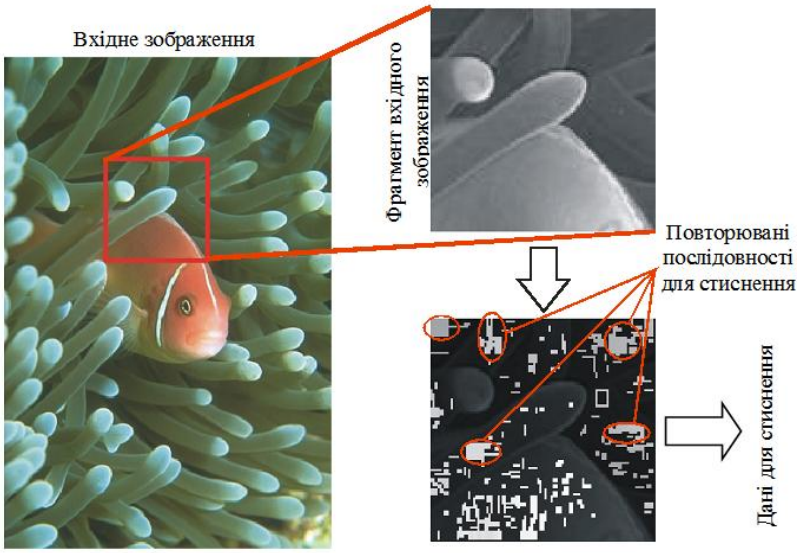


Рисунок 2.3 – Основинй принцип запропонованого алгоритму стиснення відеозображення

Для роботи запропонованого алгоритму групового кодування зображень запропоновано використати наступну структуру признаків, за допомогою якої позначатимуться однакові групові послідовності (рисунок 2.4).

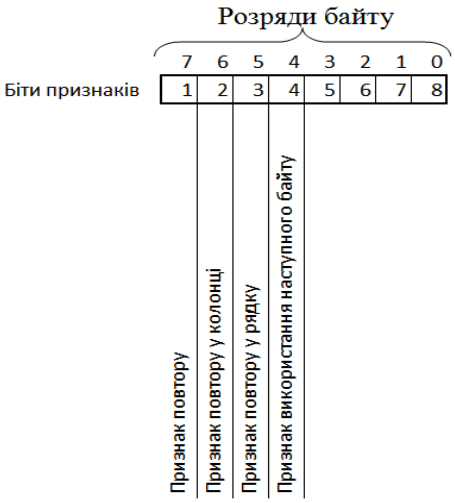


Рисунок 2.4 – Значення розрядів одного байту для позначення признаків повтору

Так, для позначення повторів групових послідовностей використовується один байт, старші розряди якого визначають:

– 1-й біт признаку – визначає наявність/відсутність повторів кольорів. Значення логічної одиниці даного біту визначає наявність повторів кольорів, а значення 0 – відсутність.

– 2-й біт признаку – у випадку, коли 1-й біт признаку рівний логічній одиниці, 2-й біт визначає признак наявності групових повторів у колонці. Його значення, що рівне 1 представляє наявність повторів у колонці, а значення 0 відсутність повторів у колонці.

– 3-й біт признаку – у випадку, коли 1-й біт признаку рівний логічній одиниці, 3-й біт визначає признак наявності групових повторів у рядку. Його значення, що рівне 1 представляє наявність повторів у рядку, а значення 0 відсутність повторів у рядку.

– 4-й біт признаку – визначає признак використання додаткового байту для позначення кількості повторів, у випадку, коли кількість повторів є більшим 8.

– 5-8 біти признаку є інформаційними.

У випадку, коли групове повторення визначається деякою областю зображення, що складається із декількох рядків та колонок, використовується наступна структура признаків (рисунок 2.5).

Рисунок 2.5 – Структура признаків для позначення групового повторення прямокутної форми

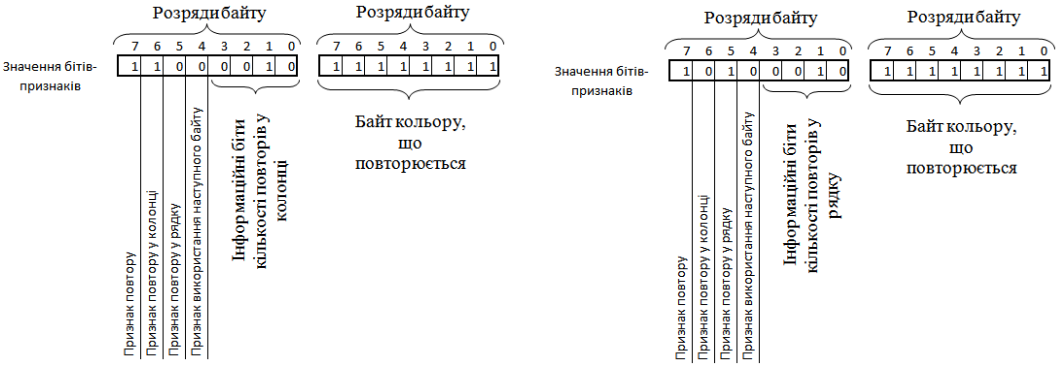
У даному випадку область повторень кольорів зображення є прямокутної форми. У такому випадку для кодування повторів використовуються 3 байти: перший для позначення признаків повтору (старші 4 розряди) та позначення кількості повторів, другий для позначення кількості повторів, третій для позначення кольору, що повторюється. Значення розрядів у даному випадку представлено на рисунку 2.6.

Рисунок 2.6 – Значення розрядів при груповому кодуванні прямокутної області зображення

За таких умов значення першого біту признаку повтору рівний одиниці, оскільки відбувається повтор кольорів; значення другого та третього бітів признаків також рівні логічним одиницям, оскільки кольори повторюються у деякій прямокутній області зображення, що містить декілька колонок та рядків; значення четвертого біту рівна логічній одиниці, оскільки для позначення кількості рядків та колонок прямокутної області необхідно використати додатковий байт. Наступні шість інформаційних бітів-признаків вказують на кількість колонок області, де повторюються кольори (на рисунку 2.6 наведено приклад області із шістьмома колонками). Решта шість інформаційних бітів другого байту вказують на кількість рядків області, де повторюються кольори (на рисунку 2.6 наведено приклад області із п’ятьма рядками). Таким чином, максимальна розмірність області повтору кольорів може містити таку кількість рядків та колонок, що задаються шістьмома розрядами і рівна 2^6x2^6 (64x64) пікселі. У випадку, коли область повтору містить більшу розмірність, така область повинна розбиватись на декілька підобластей.

Розряди останнього байту відображають значення кольору пікселів, що повторюються на зображенні у даній груповій послідовності. Таким чином максимально можлива величина стиснення може досягати 3 байти на 4096 байт, тобто коефіцієнт стиснення $K_{стис}=3/4096=0,0007$. В той же час мінімальна величина області яку доцільно стискати повинна становити не менше як 3 байти (так як потрібно 3 байти для групового кодування). Тобто при використанні області розмірністю 2х2 пікселів коефіцієнт стиснення буде $K_{стис}=3/4=0,75$ і оскільки він менший 1 то в даному випадку є ефект від стиснення такої групової послідовності.

Розглянемо наступний можливий випадок, коли групова послідовність для стиснення може міститись в одному рядку чи стовбці. Зміст розрядів групового кодування у даному випадку є подібним (рисунок 2.7).



а) б)

Рисунок 2.7 – Значення розрядів при груповому кодуванні піксеслів:
а) - колонки (б) - рядка

На відміну від групового кодування прямокутної області зображення, для кодування пікселів рядка чи стовпця може використовувати адаптивну кількість байт: від двох до трьох, в залежності від кількості повторів. Якщо кількість повторів є в межах до 16 елементів групової послідовності, тоді доцільно використовувати 4 молодші розряди, що залишились у першому байті для позначення їх кількості та ще один байт, який відображає колір пікселя, що

повторюється (див. рисунок 2.7) . Слід зауважити, що у даному випадку біт, який відповідає за признак використання наступного байту у груповому кодуванні рівний нулю. В іншому випадку, коли кількість повторів у груповій послідовності більша 15 і менша 4096 (12 біт, відведених для позначення кількості повторів) недоцільно розбивати таку послідовність на декілька частин, а доцільно використати додатковий байт для позначення кількості повторів (рисунок 2.8).

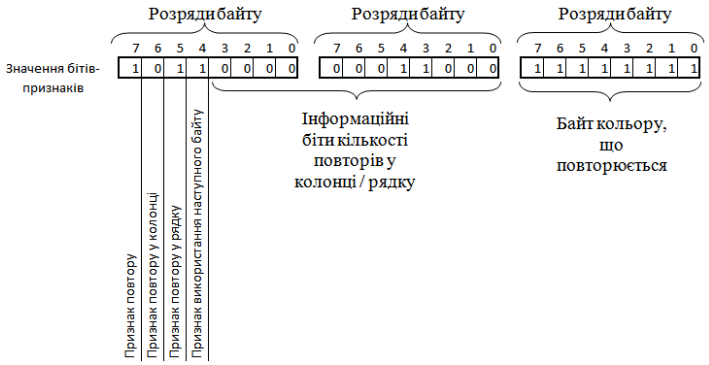
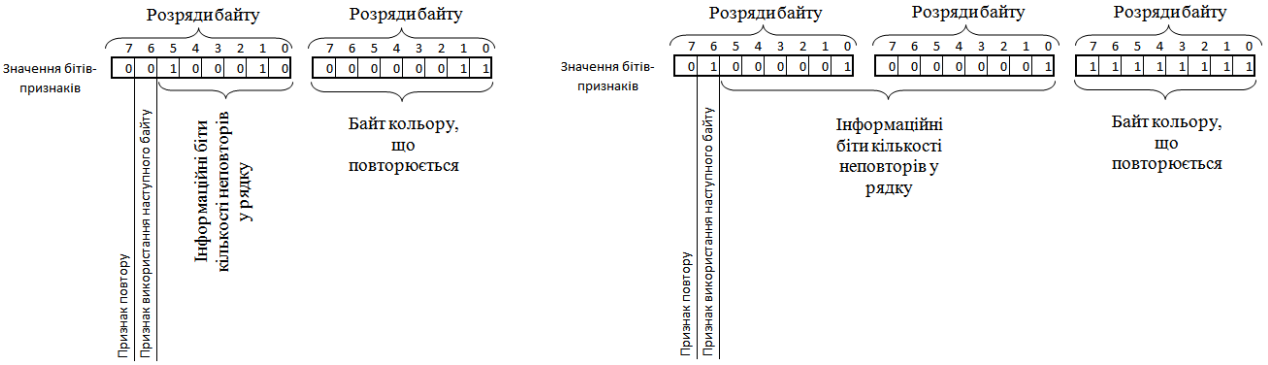


Рисунок 2.8 – Значення розрядів при кодуванні групової послідовності пікселів рядка / колонки

Для кодування такої послідовності використано 3 байти (рисунок 2.8). Причому слід зауважити, що значення біта, що відповідає за признак використання наступного байту у груповому кодуванні рівний одиниці.

Отже відзначимо, що використання такої системи групового кодування дозволить максимально стиснути послідовність із 4096 байт (пікселів) у 3 байти. Тобто коефіцієнт стиснення рівний $K_{стис}=3/4096=0,0007$. Також необхідно звернути увагу на те,що ефективним є стиснення лише послідовностей, що містять більше 2 елементів (пікселів), так як мінімально на групове кодування послідовності використовується 2 байти. Отже при кодуванні 3 пікселів мінімальний ефект від стиснення буде 2/3, тобто коефіцієнт стиснення рівний $K_{стис}=2/3=0,6667$, що є більшим показником компресії ніж при компресії прямокутних областей зображення за рахунок адаптивної кількості байт і аналогічний як і в алгоритмі RLE.

Останнім можливим варіантом використання запропонованого алгоритму групового кодування зображень є застосування наведених принципів для позначення послідовностей, що не повторюються. Такі ділянки зображень можуть займати від одного до найбільшої кількості пікселів серед сегментів зображення. Тому, в даному випадку доцільно використати адаптивну кількість байт для позначення таких послідовностей. Структуру признаків даного кодування представлено на рисунку 2.9.



а) б)

Рисунок 2.9 – Структура признаків для позначення послідовностей, що не повторюються: а) – кількість неповторюваних пікселів менша 64; б) – кількість неповторюваних пікселів більша 64

Отже для представлення послідовностей у яких значення кольорів не повторюються можна використати один або два байти, в залежності від кількості пікселів після яких повинна йти вказана послідовність байт із значеннями кольорів. При цьому, надлишковість становитиме лише один/два байти на усю послідовність.

Особливості запропонованого алгоритму стиснення відеозображень у порівнянні із відомим, пердставлено на рисунку 2.10.

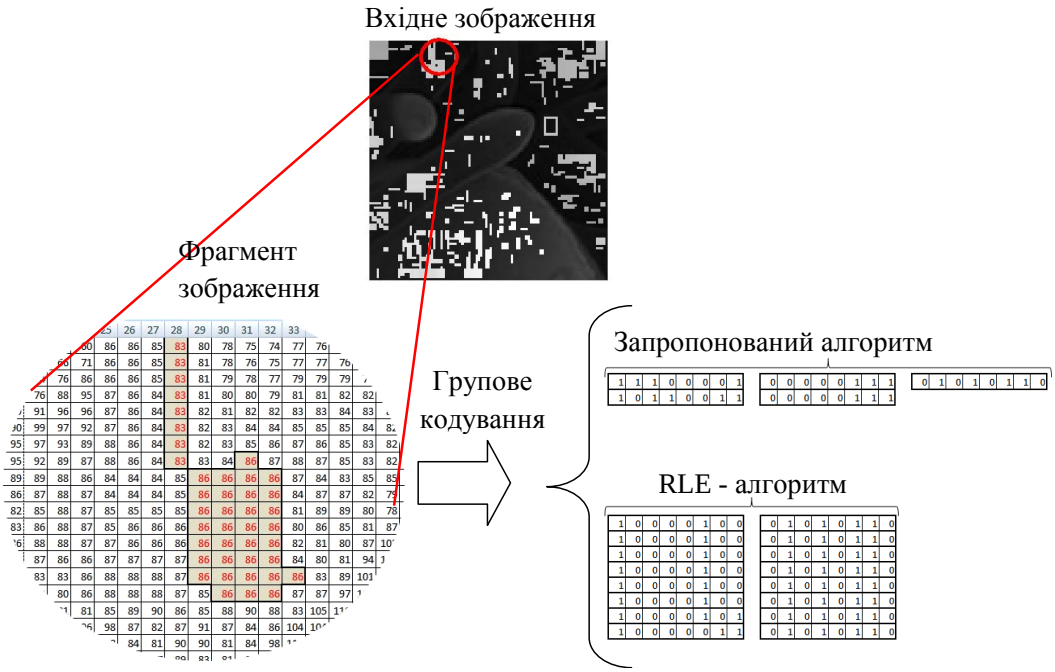


Рисунок 2.10 – Особливості запропонованого алгоритму стиснення відеозображень

Як наведено на фрагменті відеозображення (рисунок 2.10) існує можливість для компресії надлишкових даних розміщених у стовпці (28-колонка на фрагменті відеозоображення із кольором пікселя рівним 83). Для відомого алгоритму RLE така колонка однакових значень кольорів пікселів не є предметом для розгляду, оскільки у випадку формування стрічки для групового кодування (див. рисунок 2.1) таке поєднання пікселів буде розформоване. В результаті (за методом RLE), після того як буде сформована стрічка для групового кодування, значення кожного із пікселів 28 колонки зображення розміщуватимуться незалежно і не підлягатимуть компресії.

На противагу відомому алгоритму, у запропонованому, кожен стовбець (рівнозначно як і колонка) є предметом аналізу на належність пікселів до групи надлишкової інформації з метою компресії. Тому, така колонка буде закодована двома байтами. Тобто 8 байт пікселів зображення, що знаходяться у колонці будуть замінені 2 байтами (отримаємо виграш в 4 рази), коли в існуючому алгоритмі ніякого стиснення не відбудеться. У випадку наявності двох альтернатив компресії у колонці чи рядку, починаючи від деякої позиції пікселя

на зображенні, запропонованим алгоритмом буде вибрано той напрям, площа вибраної ділянки якої буде найбільшою.

Більш наглядно особливість роботи запропонованого алгоритму представлено на рисунку 2.10 для фрагменту зображення між 29 та 33 колонками (виділено товстою лінією). Наведений на рисунку діапазон містить групову послідовність пікселів однакового кольору (значення 86). При застосуванні відомого алгоритму RLE, кожна із дев'яти стрічок (окрім першої) області зображення кодується додатковим байтом, який вказує на кількість повторів у стрічці. Тобто із 37 байт (пікселів) групової послідовності (див. рисунок 2.10) для кодування за алгоритмом RLE буде використано лише 36 (байт, що у першому рядку залишається без кодування, так як розмірність стрічки з груповою послідовністю повинна бути більшою двох послідовних байт) і в результаті буде отримано 16 скомпресованих байтів. В цьому випадку коефіцієнт стиснення $K_{стис\_RLE}=16/37=0,4324$.

При застосуванні запропонованого алгоритму групового кодування, спочатку буде знайдена максимальна площа прямокутника вписаного в область зображення з однорідним кольором. Після чого, така область розділиться на дві частини. В результаті без компресії залишаться два байти (піксель що у першому рядку групової послідовності, та крайній правий). В результаті, решта 35 пікселів стиснеться в два етапи до 5 байт. В цьому випадку коефіцієнт стиснення $K_{стис\_new}=5/35=0,1429$, що майже в 3 рази ефективніше по відношенні до величини стиснення відносно відомого алгоритму.

Запропонований процес групового кодування полягає у формуванні стрічки кодів, позначаючих групові повтори чи неповтори кольорів зображень, починаючи із першого пікселя і закінчуючи останнім (рисунок 2.11).

Тобто процес кодування можна розглядати як циклічний процес, в результаті якого формується масив байтів, розмір якого менший від розміру вхідного зображення.

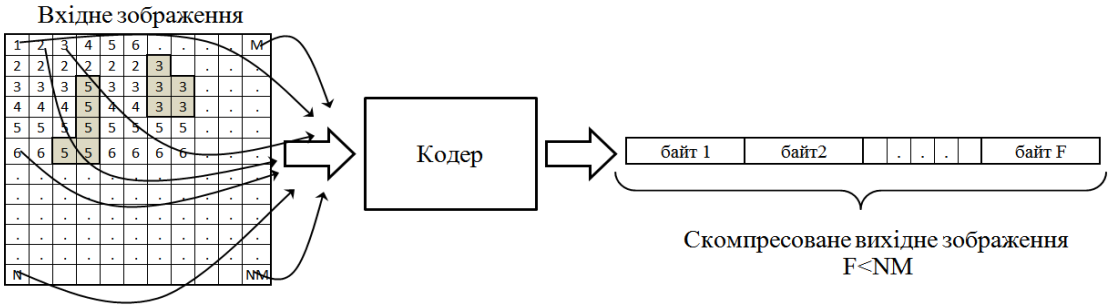


Рисунок 2.11 – Узагальнена схема запропонованого алгоритму групового кодування

Таким чином в даному параграфі запропоновано покращений метод стиснення відеозображень на основі групового кодування, який за рахунок використання процесу групування за двома напрямами дозволяє стискати зображення із меншим коефіцієнтом стиснення ніж відомий алгоритм RLE.

У наступному параграфі представлено схему алгоритму виконання групового кодування запропонованого підходу.

2.2 Алгоритм кодування групових послідовностей запропонованого вдосконаленого підходу

Як представлено у попередньому параграфі дипломної роботи, запропоновано новий підхід у кодуванні групових послідовностей, що дозволяє здійснювати стиснення відеозображень. Для використання запропонованого алгоритму у системі обробки відеозображень обов’язково повинні бути присутніми елементи кодування та декодування (відтворення) зображень (рисунок 1.2). У даному параграфі розглядається схема роботи алгоритму кодування відеозображень згідно принципів представлених у попередньому параграфі.

Як представлено на рисунку 2.11 кодер забезпечує перетворення вхідного (не компресованого зображення) у масив байтів, що кодують вхідне зображення. Причому якщо розмірність вихідного масиву менша розміру початкового

зображення, то тоді можна стверджувати про стиснення вхідної інформації, в іншому про розширення. Більш детальний аналіз величин кодування зображень запропонованим методом представлено у третьому розділі дипломної роботи. Також важливим моментом є можливість відтворення початкового зображення без спотворень використовуючи стиснену стрічку як результату групового кодування. Основні операції, що виконуються кодером представлено на рисунку 2.12.

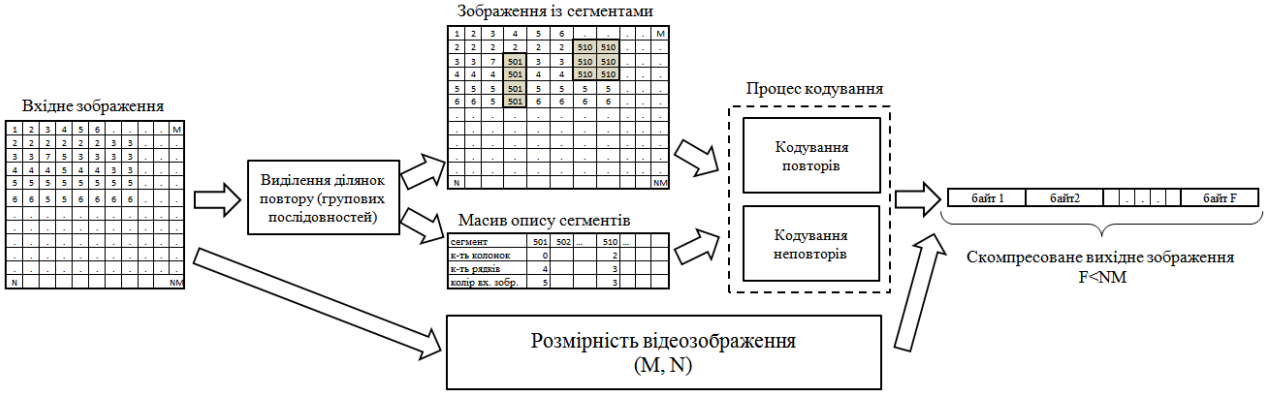


Рисунок 2.12 – Узагальнена структура кодера

На вхід кодера поступає початкове зображення у формі матриці значень кольорів у градаціях сірого кольору. Причому вважатимемо, що на кожен колір відводиться один байт. Тобто діапазон можливих варіацій кольору становить від 0…255. Якщо виникає потреба у компресії кольорових відеозображень, скажімо формату RGB, тоді виділення таблиці окремих складових червоного, зеленого і синього відтінків повинна бути виділена до початку роботи кодера. У даному випадку кодер забезпечуватиме кодування кожної складової таблиці RGB формату, що в цілому може призвести до ще вищого ступеня стиснення вхідного зображення.

Першим етапом роботи кодера зображення є процес виділення ділянок на зображенні, які містять групові повтори однакових кольорів (рисунок 2.12). Причому виділенню підлягають лише ті ділянки, що відповідають певній розмірності (від 4 елементів до максимум 64 х 64 елементи при прямокутній ділянці, чи від 3 до 4096 при одновимірній розмірності ділянки по стовбцю чи

рядку зображення). В результаті роботи даного процесу формується масив опису групових послідовностей, в якому вказується номер сегменту починаючи від 501, кількість колонок та рядків і номер кольору зображення. Значення 501 вибрано з метою маркування, яке не перекриває діапазон кольорів пікселів зображення (максимум 256: 8 біт на колір). При цьому на самому зображенні значення кольорів сегментів, що відповідають елементам групових послідовностей замінюються на значення групових послідовностей. Таким чином, для подальшої роботи кодера використовуються масив опису сегментів та модифіковане зображення із відміченими сегментами групових послідовностей.

На наступному етапі роботи кодера забезпечується попіксельний аналіз модифікованого зображення, в результаті якого виконується кодування зображення (рисунок 2.12). Тобто, якщо поточний піксель відображає маркер деякої групової послідовності, то відбувається кодування повторів групової послідовності, а самі елементи модифікованого зображення при цьому помічаються деяким маркером (наприклад, максимально-можливим цілим числом) для того, щоби повторно не враховувати дані пікселі, оскільки вони вже закодовані кодером. В іншому випадку, коли значення кольору пікселя не відображає маркер групової послідовності, він є признаком відсутності повтору і тому відбувається кодування неповторів. В результаті роботи даного процесу формується вихідна стрічка як масив байтів закодованого зображення.

На останньому етапі, до закодованої послідовності в кінець додаються додаткові байти, що відображають розмірність вхідного зображення (рисунок 2.12). Такі додаткові байти потрібні з метою забезпечення коректної декомпресії стисненого зображення. Для цього резервуються 4 байти (два байти для відображення кількості рядків і два байти для відображення кількості стовпців). Два байти, що містять 16 розрядів дозволяють адресувати 2^{16}=65536 рядків та 65536 колонок, що є цілком достатньо для роздільних здатностей сучасних відеозображень.

По завершенні роботи кодера формується одна стрічка набору байтів, яка може передаватись каналами зв’язку чи використовуватись для подальших маніпуляцій із відеозображеннями.

Більш детальніше схему роботи кодера можна представити наступною послідовністю дій (рисунок 2.13).

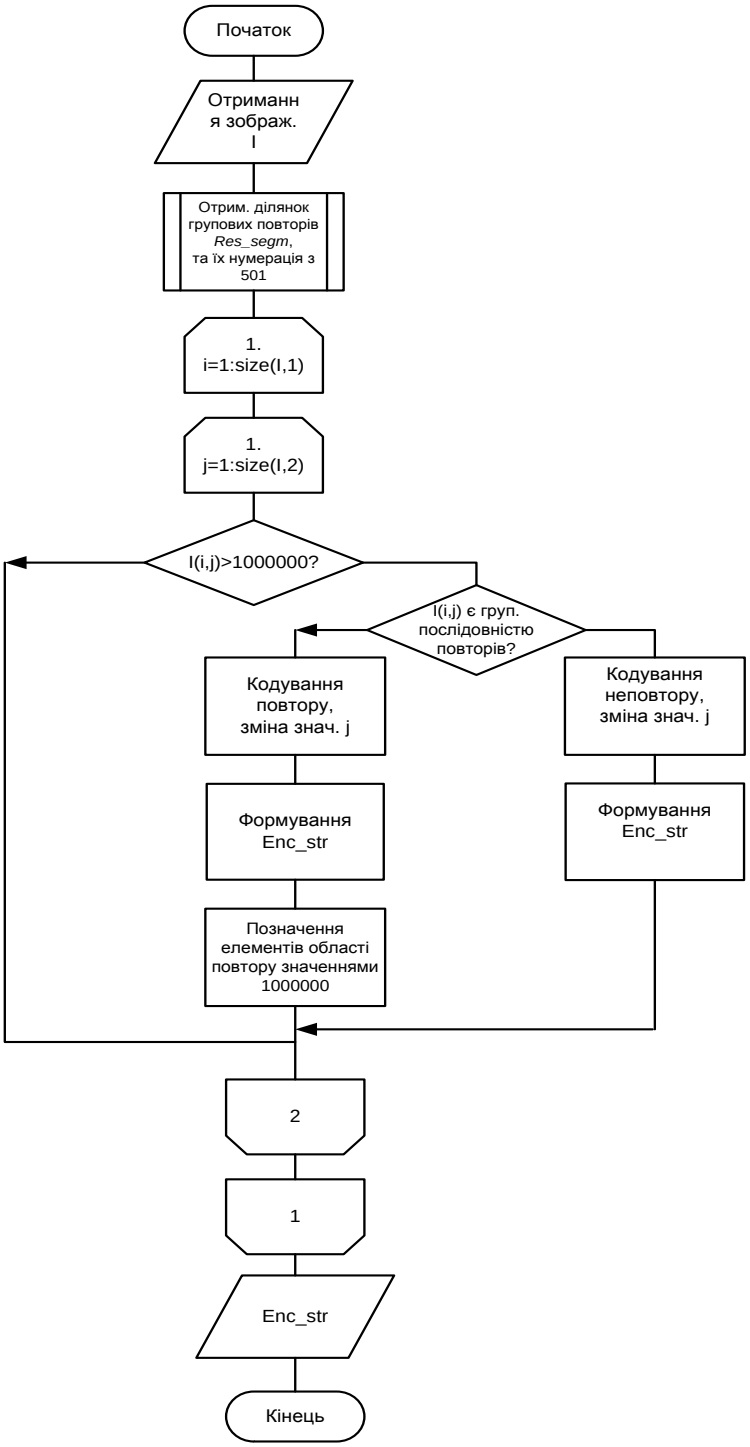


Рисунок 2.13 – Узагальнена схема роботи кодера

Одним із процесів виконання кодера є визначення ділянок зображення із однаковими послідовностями кольорів (групових повторів). Схема роботи даної процедури представлено на рисунку 2.14.

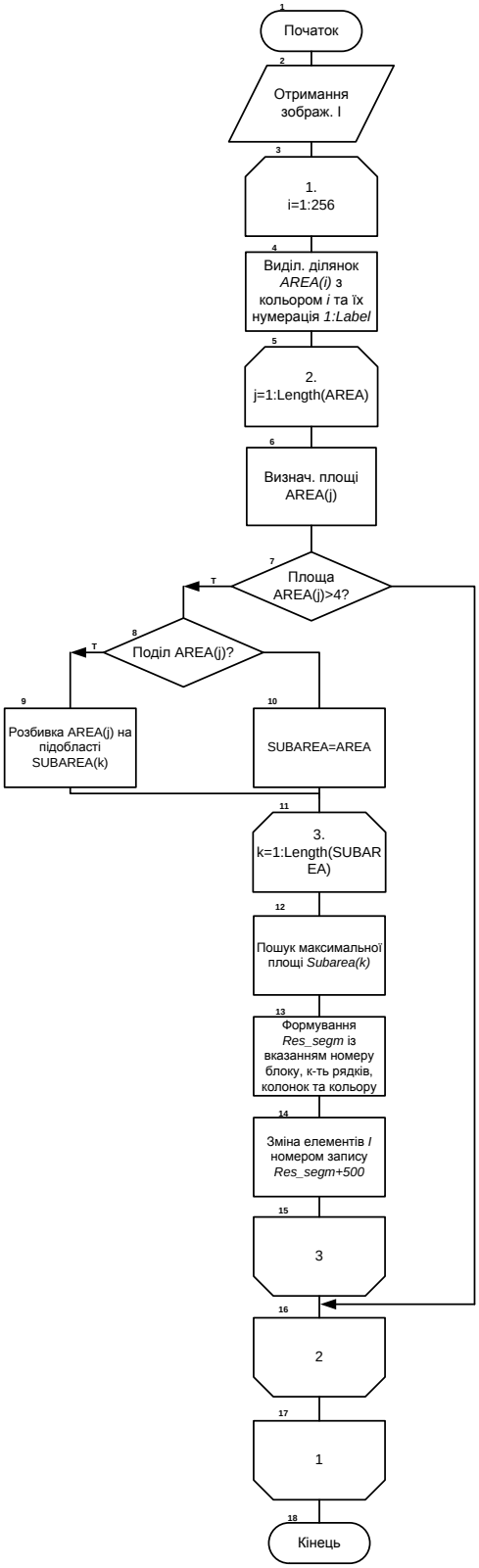


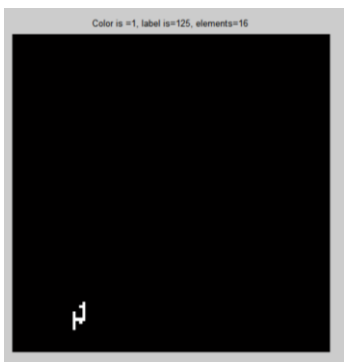
Рисунок 2.14 - Схема роботи процедури виділення групових послідовностей

Основна ідея процесу виділення групових послідовностей полягає у виділенні таких ділянок зображення, які відображають 4-зв’язні послідовності із пікселів з однаковими кольорами. Чотири-зв’язні сусіди означають зв'язок пікселів за напрямами по горизонталі чи вертикалі (рисунок 2.15).

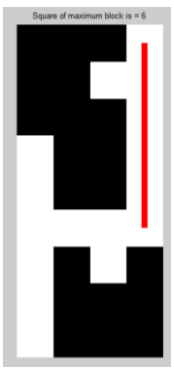
	1	
2	*	3
	4	

Рисунок 2.15 – Чотири-зв’язні сусіди пікселя зображення

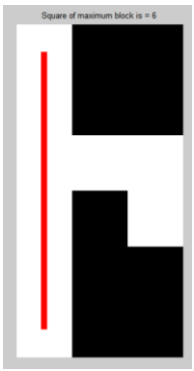
Приклад результату пошуку групових послідовностей представлено на рисунку 2.3. Причому пошук є ітераційним процесом в якому сегментується зображення за кольором 0..255. В результаті кожен сегмент підлягає аналізу на 4-зв’язність пікселів та розмірність. Окрім цього, кожен сегмент одного кольору може містити достатньо складну форму, а тому підлягає розбиванню на частини, кожна з яких включається в огрему групову послідовність. Наприклад, на рисунку 2.16а,г представлено розміщення різних сегментів зображення із значенням кольору рівним 1. Форма таких сегментів є достатньо складною, а тому забезпечується поділ кожного на частини. З цією метод шукається така ділянка сегменту, яка мітить набільшу площу вписаного прямокутника (рисунок 2.16б,д для відповідних сегментів, що на рисунку 2.16а,г).



а)

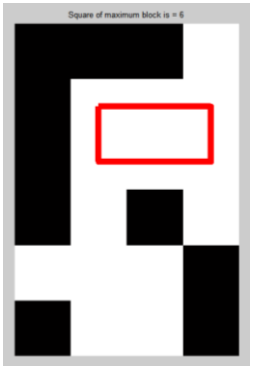


б)

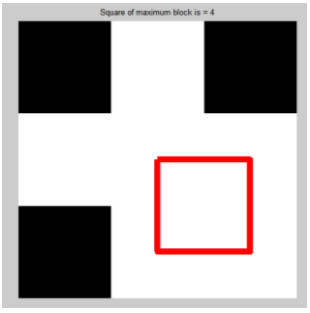


в)

г)



д)



е)

Рисунок 2.16 – Формування групових послідовностей із використанням 4-зв'язних сусідніх пікселів: а),г) – послідовності на зображенні; б),в) – розбиття на частини послідовності (а); д), е) – розбиття наа частини послідовності (г)

Після даного етапу, ділянка із найбільшою площею позначається як групова послідовність, і продовжується аналіз поточного сегменту на предмет наявності інших групових послідовностей у ному, відсікаючи вже вибрану групову послідовність (рисунки 2.16в,е для відповідних сегментів, що на рисунку 2.16а,г). Такий процес є ітераційним для усіх сегментів зображення.

Однією із найбільш очислювальних процедур роботи кодера є виконання процесу кодування групових послідовностей чи послідовностей в яких відсутні повтори. Схему роботи даного блоку кодера представлено у додатку А.

Вхідними змінними для роботи процедури кодування послідовностей є (блок 1 на додатку А): *I(N,M)* – модифіковане зображення із відміченими груповими послідовностями, що містять значення більші 500; *N,M* – розмірність зображення; *res_segm* – масив опису групових послідовностей на зображенні *I(N,M)*. Після отримання наведених змінних, забезпечується формування змінної *Enc_str*, яка є вихідним масивом, що містить значення байтів закодованої послідовності. Для цього їй присвоюється значення пустої множини, що пізніше буде наповнюватись кодовими послідовностями (блок 2 на додатку А). На наступному етапі забезпечується перебір усіх значень пікселів зображення *I(N,M)*, внаслідок чого організовуються два цикли з індексами *i,j*, які відображають номер

рядка та стовпця зображення відповідно (блоки 3-4 на додатку А). При цьому діапазон значень $i = \overline{1..N}$, $j = \overline{1..M}$.

У тілі вкладеного другого циклу перевіряється умова належності пікселя *I(i,j)* до вже опрацьованих даних (маркер опрацьованих даних рівний максимально-можливому цілому числу, або у даному прикладі 1000000). Якщо групова послідовність належить до опрацьованих даних *I(i,j)>=1000000*, то відбувається перехід до наступної операції циклу (блок 5 на додатку А). Використання даних позначень необхідна в основному при маркуванні елементів вертикальних чи прямокутних ділянок групових повторів зображення. Оскільки опрацювання пікселів зображення відбувається у циклі пострічково, то можлива ситуація, коли елементи вже закодованих послідовностей (наприклад по стовбцях) будуть предметом аналізу на наступних ітерація циклу. Тому, з метою обходу (уникнення повторного аналізу) даних ділянок, такі елементи позначаються відповідним максимальним числом. При відсутності позначень *I(i,j)>=1000000* необхідно перевірити умову належності значення кольору пікселя *I(i,j)* до позначення признаку повторюваних елементів групової послідовності (блок 6 на додатку А).

Елементи повторюваних групових послідовностей позначають значенням кольору пікселя більшим 500. Тобто, якщо значення *I(i,j)<=500,* то це означає, що послідовність пікселів зображення не повторюються, тобто не належать до групової послідовності. У цьому випадку необхідно визначити кількість неповторюваних пікселів *Num* аналізуючи значення кольорів пікселів у стрічці за стрічкою (блок 19 на додатку А) доки значення пікселів зображення не буде належати до групової послідовності (*I(i,j)>500)*. Якщо кількість неповторюючих пікселів менша 64 то це означатиме, що для кодування такої послідовності достатньо одного байту (блоки 20,21 на додатку А). У цьому випадку у вихідну послідовність *Enc_str* вноситься байт даних із значеннями старших бітів "00" плюс кількість повторів, що представляються 6 молодшими бітами (блок 21 на додатку А). В іншому випадку, коли кількість неповторюючих пікселів більша

63, їх неможливо адресувати 6 молодшими бітами і в цьому випадку доцільно використати додатковий байт. Отже, в даному випадку у вихідну послідовність *Enc_str* вноситься байт даних із значеннями старших бітів "01" плюс кількість повторів, що представляються 14 наступними бітами (блок 22 на додатку А). Після кодування даної послідовності необхідно внести саму послідовність кольорів у вихідну послідовність *Enc_str* (блоки 23-25). Після даної операції відбувається визначення (позиціонування) координати наступного пікселя для аналізу *I(i,j)* (блок 27), та продовження циклу.

У випадку, коли значення пікселя *I(i,j)>500*, воно вказує на наявність групової послідовності повторюваних кольорів (блок 6 на додатку А). При цьому можливі три варіанти повторів (блок 7): повтори у колонці, повтори у рядку, повтори у прямокутній ділянці зображення. Якщо повтори у колонці чи рядку (перша та друга вітки блоку 7), то визначається кількість повторів *Num*. Кількість повторів у колонці фіксується другим виміром масиву *Num=res_segm(I(i,j)-500,2),* а кількість повторів у рядку фіксується третім виміром у масиві *Num=res_segm(I(i,j)-500,2).* Якщо *Num<16*, це означає, що для кодування таких послідовностей достатньо одного байту. Тому у блоках 10,16 на додатку А у вихідну змінну *Enc_str* вносяться байти значень *Enc_str=Enc_str+'1100'+4 біти* значення змінної *Num* для повторів у колонці, та *Enc_str=Enc_str+'1010'+4 біти* значення змінної *Num* у випадку повторів у радку. Інакше, коли кількість повторів *Num>15* для кодування послідовностей потрібно запозичити додатковий байт. У цьому випадку у відповідних блоках 11,17, що на додатку А для кодування повторів у колонці *Enc_str=Enc_str+'1101'+12 бітів* (відображають значення змінної *Num*), а у випадку кодування повторів у рядку *Enc_str=Enc_str+'1011'+12 бітів* (відображають значення змінної *Num*). Для групової послідовності прямокутної форми (коли значення масиву *res_segm(I(i,j)-500,2)<>0 & res_segm(I(i,j)-500,3)<>0*), в даному випадку *Enc_str=Enc_str+'1111'+6 бітів (res_segm(I(i,j)-500,2))+6 бітів (res_segm(I(i,j)-500,3))*, що представлено блоком 18 на додатку А.

Після того як закодовано групові послідовності, до *Enc_str* необхідно додати байт із значенням кольору, який повторюється у груповій послідовності і визначається третім виміром масиву *res_segm* (блок 12 додатку А): *Enc_str=Enc_str+res_segm(I(i,j)-500,3).*

По завершенні кодування групової послідовності необхідно позначити пікселі зображення, що відповідають елементап послідовності признаком використання (значенням 1000000), що представлено у блоці 13 на додатку А. Даною процедурою завершується процес кодування групової послідовності і здійснюється перехід до наступного кроку циклу (блоки 27, 28 на додатку А).

Після виконання кодування всього зображення до вихідного масиву *Enc_str* додаються 4 байти розмірності зображення *Enc_str=Enc_str+2 байти (кількість колонок)+ 2 байти (кількість рядків).*

Таким чином у даному параграфі представлено алгоритм роботи кодера зображення, який на основі вхідного зображення забезпечує його стиснення, в результаті чого формується одновимірний масив меншої розмірності ніж розмір вхідного зображення. Такий вихідний масив є предметом аналізу декодером що представлено у наступному параграфі дипломної роботи.

2.3 Алгоритм декодування групових послідовностей вдосконаленого підходу

Представлений у попередніх параграфах алгоритм групового кодування відеозображень повинен володіти якісними засобами відтворення закодованої послідовності без втрат інформації. У даному параграфі представлено алгоритм роботи декодера, який відтворює початкове зображення без втрат інформації.

Принцип роботи декодера може бути представлений набором процедур, що дозволяють перетворити вхідну для декодера послідовність бітів у відеозображення (рисунок 2.17).

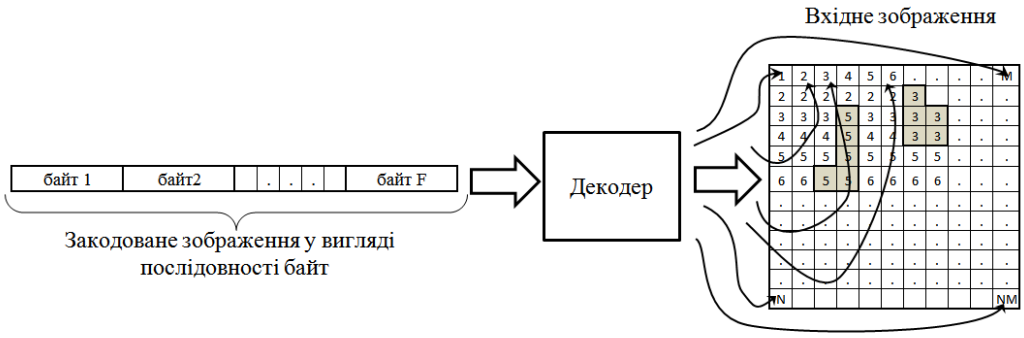


Рисунок 2.17 – Основний принцип роботи декодера

Вхідними даними для роботи декодера є послідовність байтів, яка була сформована кодером внаслідок групового кодування, що представлено у попередньому параграфі. Причому, чотири останні байти послідовності містять дані про кількість стовпців *M* та рядків *N* розмірності початкового зображення, що було піддане процесу кодування. Тому перед початком декодування вхідної послідовності, здійснюється процедура її попередньої обробки у якій із послідовності видаляються останні 4 байти розмірності та на їх основі виконується процес створення макету вхідного зображення. Початковим значенням кольорів пікселів макету зображення присвоюються значення -1. Використання даного підходу дозволить контролювати наповненість кольорів початкового зображення при декодуванні послідовності.

В результаті таких маніпуляцій на вхід декодера поступають послідовність байт закодованого зображення (від 1 до *K*) та макет матриці (див. структуру декодера, що на рисунку 2.18).

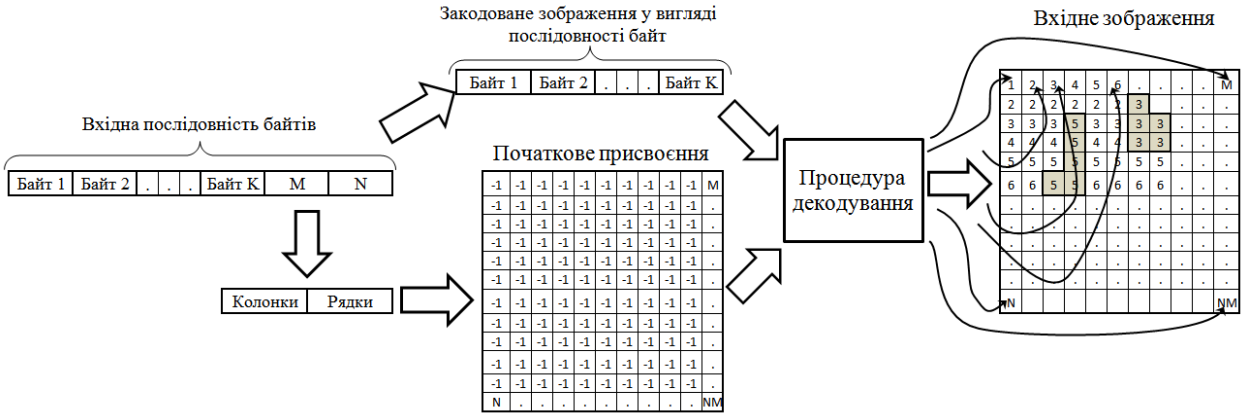


Рисунок 2.18 – Структура декодера

Схема роботи процедури декодування представлена у додатку Б. Розглянемо більш детально роботу декодера при груповому кодуванні відеозображень.

Отримавши з каналу зв'язку чи інших джерел файл із послідовністю байтів, першочерговими етапами є виділення розмірності *NxM* початкового зображення, присвоєння початкових значень змінним рядка *i=1* та колонки *j=1* координати пікселя колір якого декодуватиметься, а також формування макету матриці зображення *I(N,M)*, що відображено блоками 0-2 на схемі кодера (див. додаток А). Після даної процедури із вхідної послідовності видаляються останні чотири байти для подальшої роботи алгоритму.

На наступному етапі забезпечується загальний цикл процесу декодування у якому перевіряється умова можливості зчитування (наявності) наступного байту даних у закодованій послідовності для опрацювання (блок 3 алгоритму декодування). Якщо наступний байт зчитати неможливо (внаслідок його відсутності) - це означає, що всі байти у закодованій послідовності опрацьовані, зображення декодоване. В такому випадку завершується виконання процедури декодування (блок 5 алгоритму).

В іншому випадку зчитується байт даних *Byte1* із послідовності (блок 4 алгоритму) для подальшого опрацювання. У закодованій послідовності страші розряди першого байту завжди є признаками кодування групової послідовності

повторів чи неповторів зображення. Як представлено у параграфі 2.1 дипломної роботи, старші розряди першого байту можуть містити наступні признаки:

- "00….." – признак відсутності на початковому зображенні групових повторів пікселів з однаковими кольорами, починаючи з поточної *I(i,j)* позиції зображення, причому кількість пікселів, що не повторюються задається наступними 6 розрядами (максимум 2^6=32);
- "01……" – признак відсутності на початковому зображенні групових повторів пікселів з однаковими кольорами, починаючи з поточної *I(i,j)* позиції зображення, причому кількість пікселів, що не повторюються задається наступними 6 розрядами поточного байту та 8 розрядами наступного байту (максимум 2^{14}=16384);
- "110….." – признак наявності групового повтору пікселів початкового зображення з однаковими значеннями кольорів у колонці, починаючи із пікселя розміщеного у позиції *I(i,j);*
- "101….." – признак наявності групового повтору пікселів початкового зображення з однаковими значеннями кольорів у рядку, починаючи із пікселя розміщеного у позиції *I(i,j);*
- "1111...." – признак наявності групового повтору пікселів початкового зображення з однаковими значеннями кольорів прямокутної форми починаючи із пікселя розміщеного у позиції *I(i,j),* як верхнього лівого кута форми.

Аналізуючи значення представлених розрядів *Byte1*, по кожному із них виконується процедура декодування початкового зображення (заповнення пікселів макету *I(i,j)* значеннями із закодованої послідовності. Такий підхід може реалізуватись case-умовою алгоритму (блок 6 алгоритму), які розглянемо більш детально.

У першому випадку, коли значення старших розрядів *Byte1="00……",* визначається кількість наступних байт, які потрібно зчитати із закодованої послідовності і розмістити послідовно у макеті зображення (блок 7 алгоритму). Така кількість *Num* визначається 6 молодшими розрядами *Byte1* (2.2):

$$Num=Bin2Dec[Bytes(Byte1,[1\text{-}6])], \tag{2.2}$$

де *Num* – кількість байтів;

Bin2Dec() – функція, що здійснює перетворення з двійкової системи числення у десяткову;

Bytes() – функція, що виділяє розряди [1-6] із байту *Byte1;*

Byte1 – двійкове значення байту закодованої послідовності.

Після даної процедури організовується визначений цикл із *Num*-повторів (блоки 8,15 алгоритму), у якому зчитується наступний байт *Byte2* із закодованої послідовності (блок 8 алгоритму) та його значення присвоюється пікселю макета початкового зображення *I(i,j)=Byte2*. Причому, присвоєння відбувається лише у випадку, коли значення пікселя макету початкового зображення ще не зайняте груповою послідовністю, тобто (2.3)

$$I(i,j)=\begin{cases} Byte2,\ j=j+1, & if\ \ I(i,j)=-1; \\ I(i,j),\ j=j+1, & else. \end{cases} \tag{2.3}$$

Крім того, якщо значення змінної що відображає індекс колонки $j>M$, то відбувається перехід вказівника координати пікселя на зображенні на наступний рядок (2.4).

$$(i,\ j)=\begin{cases} (i=i+1,\ j=1), & if\ \ j>M; \\ (i=i,\ j=j), & else. \end{cases} \tag{2.4}$$

Процес зміни значень кольорів пікселів макету початкового зораження *I(i,j)* представлено блоками 10-14 на схемі алгоритму.

У випадку, коли значення старших розрядів *Byte1="01....."*, значення логічної одиниці означає потребу в додатковому байті для визначення кількості неповторів. Тому, здійснюється зчитування наступного байту *Byte2* із закодованої

послідовності (блок 16 алгоритму). На наступному кроці визначається кількість байт, які потрібно зчитати із закодованої послідовності і розмістити послідовно у макеті зображення (блок 17 алгоритму). Така кількість *Num* визначається 6 молодшими розрядами *Byte1* та 8 розрядами *Byte2* (2.5):

$$Num=Bin2Dec[Bytes(Byte1,[1\text{-}6])+Byte2], \tag{2.5}$$

де *Num* – кількість байтів;

Bin2Dec() – функція, що здійснює перетворення з двійкової системи числення у десяткову;

Bytes() – функція, що виділяє розряди [1-6] із байту *Byte1;*

Byte1, Byte2 – двійкові значення байтів закодованої послідовності.

Після даної процедури, як і в попередньому випадку, організовується визначений цикл із *Num*-повторів (блоки 18,25 алгоритму), у якому зчитується наступний байт *Byte3* із закодованої послідовності (блок 19 алгоритму) та його значення присвоюється пікселю макета початкового зображення *I(i,j)=Byte3*. Причому, присвоєння відбувається лише у випадку, коли значення пікселя макету початкового зображення ще не зайняте груповою послідовністю, тобто (2.6)

$$I(i,j)=\begin{cases} Byte3,\ j=j+1, & if\ I(i,j)=-1; \\ I(i,j),\ j=j+1, & else. \end{cases} \tag{2.6}$$

Крім того, якщо значення змінної що відображає індекс колонки $j>M$, то відбувається перехід вказівника координати пікселя на зображенні на наступний рядок (2.4).

Процес зміни значень кольорів пікселів макету початкового зораження *I(i,j)* у випадку *Byte1="01....."* представлено блоками 20-24 на схемі алгоритму.

У випадку, коли значення старших розрядів *Byte1="101....."*, даний признак означає наявність повторів групової послідовності пікселів з однаковими

значеннями кольорів у рядку початкового зображення. Тому, у блоці 26 алгоритму здійснюється перевірка необхідності запозичення додаткового байту для визначення кількості повторів за значенням 5-го біту *Byte1* і у випадку, коли він рівний логічній одиниці зчитується наступний байт *Byte2* із закодованої послідовності. Розрахунок кількості повторів у колонці визначаємо за (2.7), що представлено блоками 27-29 на схемі алгоритму:

$$Num = \begin{cases} Bin2Dec[Bytes(Byte1,[1-4])], & if\ \ Bytes(Byte1,[5-8]) = "1100"; \\ Bin2Dec[Bytes(Byte1,[1-4]) + Byte2], & if\ \ Bytes(Byte1,[5-8]) = "1101". \end{cases} \quad (2.7)$$

де *Num* – кількість байтів;

Bin2Dec() – функція, що здійснює перетворення з двійкової системи числення у десяткову;

Bytes() – функція, що виділяє розряди [1-4] із байту *Byte1;*

Byte1, Byte2 – двійкові значення байтів закодованої послідовності.

Після даної процедури зчитується наступний байт *Byte3* із закодованої послідовності (блок 30 алгоритму), що відображає значення кольору пікселів групової послідовності, що повторюється на зображенні. Для відображення даного кольору на макеті організовується визначений цикл із *Num*-повторів (блоки 31,37 алгоритму), у якому значення *Byte3* присвоюється пікселям макета початкового зображення *I(i,j)=Byte3*. Причому, присвоєння відбувається лише у випадку, коли значення пікселя макету початкового зображення *I(i,j)* ще не зайняте груповою послідовністю (2.6). Крім того, якщо значення змінної що відображає індекс колонки *j>M*, то відбувається перехід вказівника координати пікселя на зображенні на наступний рядок (2.4).

Процес зміни значень кольорів пікселів макету початкового зораження *I(i,j)* у випадку *Byte1="110....."* представлено блоками 32-36 на схемі алгоритму.

Аналогічним чином змінюються значення пікселів макету початкового зображення у випадку наявності групових повторів у колонці, коли

Byte1="110....." (блоки 38-48) з тією відмінністю, що зміни вносяться у колонку макету зображення.

Останньою можливою комбінацією старших розрядів *Byte1* для декодера є послідовність, коли *Byte1="1111...."*, яка означає наявність групових повторів значень кольорів пікселів у початковому зображенні прямокутної форми. В даному випадку обов'язково використовується додатковий байт *Byte2*, який зчитується із закодованої послідовності (блок 49 алгоритму) для визначення кількості рядків і колонок (блок 50 алгоритму) групової послідовності прямокутної форми (2.8).

$$\begin{cases} Col = Bin2Dec[Bytes(Byte1,[1-4]) + Bytes(Byte2,[7-8])]; \\ Row = Bin2Dec[Bytes(Byte2,[1-6])]. \end{cases} \quad (2.8)$$

де, *Col, Row* – кількість колонок та рядків у груповій послідовності прямокутної форми.

На наступному кроці зчитується байт даних із закодованої послідовності *Byte3* (блок 51 алгоритму), що відображає значення кольору пікселів групової послідовності. Зміна значень кольорів пікселів (2.9) макету зображення здійснюється у подвійному циклі, де перший визначає координати рядків у діапазоні *k1=[i, i+Row]*, а другий (вкладений у перший) визначає координати колонок у діапазоні *k2=[j, j+Col]*.

$$I(k1,k2) = Byte3, \quad k1 = [i, i + Row],\ k2 = [\,j,\ j + Col\,], \quad (2.9)$$

де *I(k1,k2)*, значення кольору пікселя макету початкового зображення, що знаходиться у *k1* – рядку і *k2* – колонці.

Процедуру заміну значень кольорів пікселів макету початкового зображення представлено блоками 52-56 схеми алгоритму декодера. По завершенні декодування пікселів макету зображення груповою послідовністю

прямокутної форми,забезпечується зміна значень координати *(i,j)* пікселя макету зображення (2.10) після якої забезпечується перевірка умови (2.4).

$$j=j+Col+1. \tag{2.10}$$

Алгоритм декодування циклічно зчитує байти даних із закодованої послідовності та здійснює заміну значень кольорів пікселів макету початкового зображення *I(i,j)*, до моменту коли зчитуваний байт буде признаком кінця послідовності або ж пустою множиною, що є признаком завершення алгоритму. В результаті виконання алгоритму усі від'ємні значення кольорів макету початкового зображення заміняться на значення із закодованої послідовності. Тому по завершенні алгоритму декодування макет початкового зображення буде відображати декодоване зображення, яке буде ідентичним (без втрат) зображенню, що отримане до моменту кодування.

Таким чином у даному параграфі представлено розроблений алгоритм декодування зображення, який дозволяє без втрат відновити початкові кольори зображення із закодованої послідовності чисел за рахунок використання признаків групових посторів.

В розділі 2 представлено запропонований новий алгоритм кодування цифрових відеозображень, який за рахунок групового кодування дозволяє забезпечувати більш якісніше стиснення інформація ніж відомі алгоритми групового кодування.

Розроблені нові підходи для кодування відеозображень, що на відміну від існуючих забезпечують зменшення надлишковість у масивах зображень за рахунок аналізу групових послідовностей пікселів із однаковими значеннями кольорів за двома напрямами.

Розроблено алгоритмічне забезпечення функціонування кодера, яке дозволяє реалізовувати алгоритм групового кодування відеозображень без втрат

інформації різними мовами програмування та бути апаратно чи програмно незалежним.

Розроблено алгоритмічне забезпечення функціонування декодера, яке дозволяє реалізовувати алгоритм декодування послідовності чисел, закодованих розробленим кодером і без втрат відновити початкове зображення та бути апаратно чи програмно незалежним з точки зору реалізації.

Основні техніко-економічні показники функціонування розробленого алгоритму стиснення відеозображень на основі групового кодування представлено у наступному розділі дипломної роботи.

3 ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ

3.1 Критерії порівняння алгоритмів

Представлений у попередньому розділі алгоритм групового кодування дозволяє здійснювати стиснення вхідних даних за двома напрямами, що є властиво при обробці відеозображень. В даному параграфі представлено основні критерії, що дозволяють оцінити якість роботи запропонованого алгоритму.

Відмітимо, що характеристики алгоритму щодо деяких вимог застосувань, залежать від конкретних умов, в які буде поставлений алгоритм. Так, степінь компресії залежить від того, на якому класі зображень алгоритм тестується. Аналогічно швидкість компресії нерідко залежить від того, на якій платформі реалізований алгоритм. Перевага одному алгоритму перед іншим може дати, наприклад, можливість використання в обчисленнях алгоритму технологій нижнього рівня, типу MMX, а це можливо далеко не для всіх алгоритмів. Так, наприклад, LZW не використовує технології MMX. Крім того, необхідно враховувати, що деякі алгоритми розпаралелюються легко, а деякі ні.

Таким чином, неможливо скласти універсальний порівняльний опис відомих алгоритмів. Це можна зробити тільки для типових класів застосувань за умови використання типових алгоритмів на типових платформах. Проте такі дані незвичайно швидко застарівають. Наприклад, ще в 1994 р., інтерес до показу огрубленого зображення, використовуючи тільки початок файлу, був чисто абстрактним. Реально ця можливість практично ніде не була потрібна, і клас застосувань, що використовують дану технологію був невеликий. З розповсюдженням Інтернету, який характеризується передачею зображень по порівняно повільних каналах зв'язку, використання Interlaced GIF (алгоритм LZW) і Progressive JPEG (варіант алгоритму JPEG), що реалізовують цю можливість, різко зросло. Те, що новий алгоритм (наприклад, wavelet) підтримує таку можливість, що є істотним плюсом для нього сьогодні.

В той же час можливим залишається розгляд такої рідкісної на сьогодні вимоги, як стійкість до помилок. Можна припустити, що незабаром (через 5-10 років) з розповсюдженням широкомовлення в мережі Інтернет для його забезпечення використовуватимуться саме алгоритми, стійкі до помилок.

Зі всіма зробленими вище обмеженнями, виділимо найбільш важливі критерії порівняння алгоритмів стиснення, які і використовуватимуться надалі.

Гірша, середня і краща степінь стиснення. Тобто частка, на яку зросте зображення, якщо початкові дані будуть якнайгіршими; деяка середньостатистична степінь для того класу зображень, на який орієнтований алгоритм; і краща степінь стиснення. Остання необхідна лише теоретично, оскільки показує степінь стиснення найкращого (як правило, абсолютно чорного) зображення, іноді фіксованого розміру.

Одним із критеріїв аналізу алгоритмів стиснення є клас зображень, на який орієнтований алгоритм. Іноді вказується причина, чому на інших класах зображень виходять гірші результати.

Ще одним критерієм порівняння алгоритмів є симетричність. Відношення характеристики алгоритму кодування до аналогічної характеристики при декодуванні. Характеризує ресурсоємність процесів кодування і декодування. Найбільш важливою є симетричність за часом: відношення часу кодування до часу декодування. Іноді можна аналізувати симетричність по використанню пам'яті.

Наступним критерієм є показник втрати якості у зображенні, і якщо є, то за рахунок чого змінюється ступінь стиснення. Річ у тому, що у більшості алгоритмів стиснення з втратою інформації існує можливість зміни ступеню стиснення. У дипломній роботі розглядаються алгоритми стиснення зображень на основі групового кодування які повинні відтворювати однозначно кодоване зображення. Тому даний показник повинен 100% відтворювати вхідні зображення без втрат.

Додатково потрібно враховувати характерні особливості алгоритму і зображень, до яких його застосовують. Тут можуть указуватися найбільш важливі для алгоритму властивості, які можуть стати такими, що визначають при його виборі.

Використовуючи дані критерії, приступимо до розгляду алгоритмів архівації зображень.

Перш ніж безпосередньо почати розмову про алгоритми, хотілося б зробити обмовку. Один і той же алгоритм часто можна реалізувати різними способами. Багато відомих алгоритмів, таких, як RLE,, LZW або JPEG мають десятки реалізацій, що розрізняються. Крім того, у алгоритмів буває декілька явних параметрів, варіюючи які можна змінювати характеристики процесів архівації і розархівування. При конкретній реалізації ці параметри фіксуються, виходячи з найбільш вірогідних характеристик вхідних зображень, вимог на економію пам'яті, вимог на час архівації і так далі Тому у алгоритмів одного сімейства кращий і гірший ступені стиснення можуть відрізнятися, але якісно картина не зміниться.

Декілька величин використовуються для обчислення ефективності алгоритмів стиснення [27].

1) Коефіцієнт стиснення визначається за формулою (3.1):

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

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):

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

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)

$$Quality = 100 * (Fcompr - 1), \tag{3.3}$$

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

Таким чином у даному параграфі визначень критерії згідно яких можна порівнювати ефективність функціонування розробленого алгоритму кодування у порівнянні з відомими.

3.2 Експериментальні дослідження

У даному параграфі представляються результати досліджень запропонованого алгортиму стиснення відеозображень на основі групового кодування, що представлено у другому розділі дипломної роботи у порівнянні із відомим алгоритмом групового кодування RLE за критеріями, що представлені у попередньому параграфі роботи.

Для виконання порівняння алгоритмів стиснення цілеспрямовано використанємо різні класи відеозображень, що дозволить оцінити можливості алгоритмів. З цією метою використаємо відеозображення із типової бібліотеки CIPR Still Images [8,27], що на сьогоднішній день використовується як типовий тестовий набір при аналізі алгоритмів стиснення (рисунок 3.1).

В результаті тестування зображень із наведеного набору, використовуючи розроблене програмне забезпечення (додаток В) отримані наступні показники:

Таблиця 3.1 – Показники стиснення зображень за алгоритмом RLE

Зображ.	**Симет-ричність**	**Втрата якості**	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
BALOON1	+	–	0,91	1,10	9,89
BALOON2	+	–	1,25	0,80	-20,00
BOARD	+	–	0,78	1,28	28,21
BOATS	+	–	3,25	0,31	-69,23
GOLD	+	–	2,89	0,35	-65,40
HOTEL	+	–	0,83	1,20	20,48
LENA	+	–	3,86	0,26	-74,09
ZELDA	+	–	2,78	0,36	-64,03

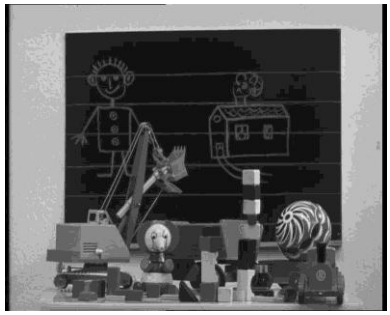
Таблиця 3.2 – Показники стиснення зображень розробленим алгоритмом

Зображ.	**Симет-ричність**	**Втрата якості**	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
BALOON1	–	–	0,68	1,47	47,06
BALOON2	–	–	1,13	0,88	-11,50
BOARD	–	–	0,61	1,64	63,93
BOATS	–	–	1,38	0,72	-27,54
GOLD	–	–	1,12	0,89	-10,71
HOTEL	–	–	0,71	1,41	40,85
LENA	–	–	1,28	0,78	-21,88
ZELDA	–	–	1,15	0,87	-13,04

Аналізуючи принцип роботи двох алгоритмів стиснення зображень, що представлено у попередніх розділах роботи, можна стверджувати, що вони розв’язують один клас задач компресії даних на основі групового кодування. Причому, як видно із наведених таблиць 3.1, 3.2 видно, що алгоритм RLE є симетричним на відміну від запропонованого. У запропоновану, процес кодування є значно довшим за декодування. Час виконання процесу кодування є дещо довшим за рахунок того, що попередньо відбувається етап виділення однорідних групових повторень пікселів (сементація однорідностей) на зображенні, після чого забезпечується їх стиснення чого немає при декодуванні у зв’язку із відсутністю потреби виконання такої процедури. На противагу цьому у відомому методі RLE процедури кодування і декодування є майже симетричними, яещо нехтувати незначними відмінностями у часі виконання, що зумовлені особливостями реалізації алгоритмів.

а)

б)



в)

г)

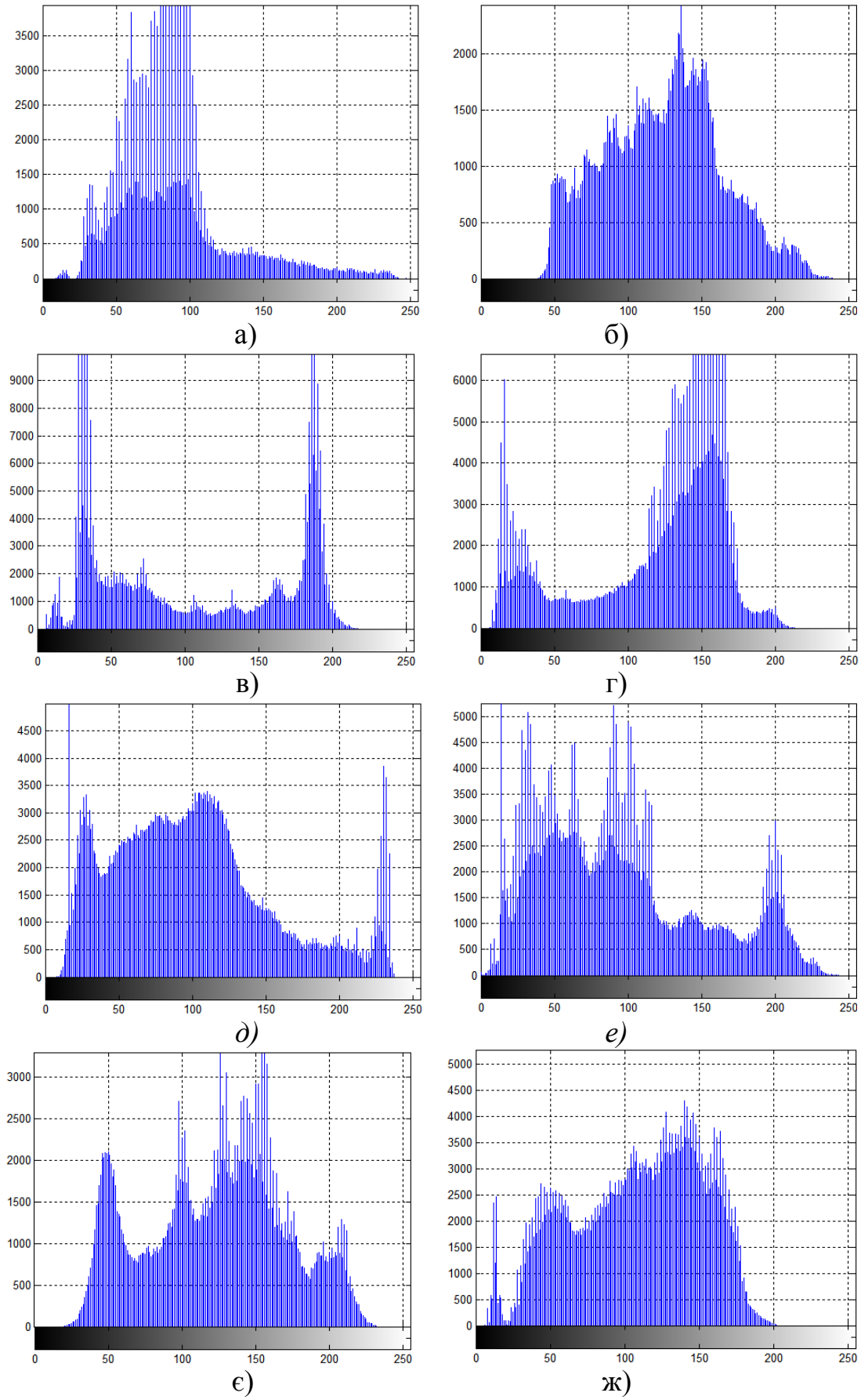
д)

е)

є)

ж)

Рисунок 3.1 – Типовий набір CIRP відеозображень: а) – BALOON1, б) – BALOON2, в) – BOARD, г) – BOATS, д) – GOLD, е) – HOTEL, є) – LENA, ж) - ZELDA



Рисуно 3.2 – Гістограми кольорів відеозображень: а) – BALOON1, б) – BALOON2, в) – BOARD, г) – BOATS, д) – GOLD, е) – HOTEL, є) – LENA, ж) - ZELDA

Аналіз показників компресії, що представлені у таблицях 3.1, 3.2 то очевидним є те, що не усі зображення стискуються з одинаковим коефіцієнтом стиснення і більше того, деякі зображення після стиснення отримали більший розмір ніж до компресії. Надвичайно очевидно представляє величину стиснення показник якості, який у відсотках представляє результат. У таблиці 3.3 наведено відхилення показників стиснення запропонованим алгоритмом від аналогічних показників відомого методу RLE.

Таблиця 3.3 – Абсолютні відхилення показників стиснення алгоритмів

Зображення	**Коефіцієнт стиснення**	**Фактор стиснення**	**Якість стиснення, %**
BALOON1	-0,23	0,37	37,17
BALOON2	-0,12	0,08	8,50
BOARD	-0,17	0,36	35,73
BOATS	-1,87	0,42	41,69
GOLD	-1,77	0,55	54,68
HOTEL	-0,12	0,20	20,36
LENA	-2,58	0,52	52,22
ZELDA	-1,63	0,51	50,99

Дані у вище наведених таблицях свідчать про клас зображень, для яких розрахований алгоритм: ті, що мають значну кількість повторів (тексти, однорідності і т.п.), що представлені у зображеннях BALOON1, BOARD, HOTEL.

В цілому, беручи до уваги показник якості стиснення зображень можна стверджувати, що запропонований в 1,33 рази показує кращу якість компресії ніж RLE, що представляє пожливості до впровадження.

ВИСНОВКИ

У дипломній роботі розглянуто розв’язок задачі розробки програмних засобів і алгоритмів стиснення цифрових відеозображень та зменшення надмірності даних під час процесу кодування у системах відеообробки.

На основі розглянутих у першому розділі дипломної роботи способів кодування зображень у сучасних системах відеообробки, а також класифікації відеозображень та програмного забезпечення, сформувано вимоги до алгоритмів кодування.

Запропоновано новий алгоритм кодування цифрових відеозображень, який за рахунок групового кодування дозволяє забезпечувати більш якісніше стиснення інформація ніж відомі алгоритми групового кодування.

Розроблені нові підходи для кодування відеозображень, що на відміну від існуючих забезпечують зменшення надлишковості у масивах зображень за рахунок аналізу групових послідовностей пікселів із однаковими значеннями кольорів за двома напрямами.

Розроблено алгоритмічне забезпечення функціонування кодера, яке дозволяє реалізовувати алгоритм групового кодування відеозображень без втрат інформації

На основі розглянутих критеріїв згідно яких можна порівнювати ефективність функціонування розробленого алгоритму кодування у порівнянні з відомими та на основі проведених експериментальних досліджень запропонованого підходу до стиснення відеозображень, визначено клас зображень, для яких розрахований запропонований підхід: ті, що мають значну кількість повторів (тексти, однорідності і т.п.).

В цілому, беручи до уваги показник якості стиснення зображень можна стверджувати, що запропонований в 1,33 рази показує кращу якість компресії ніж RLE, що представляє можливості до впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

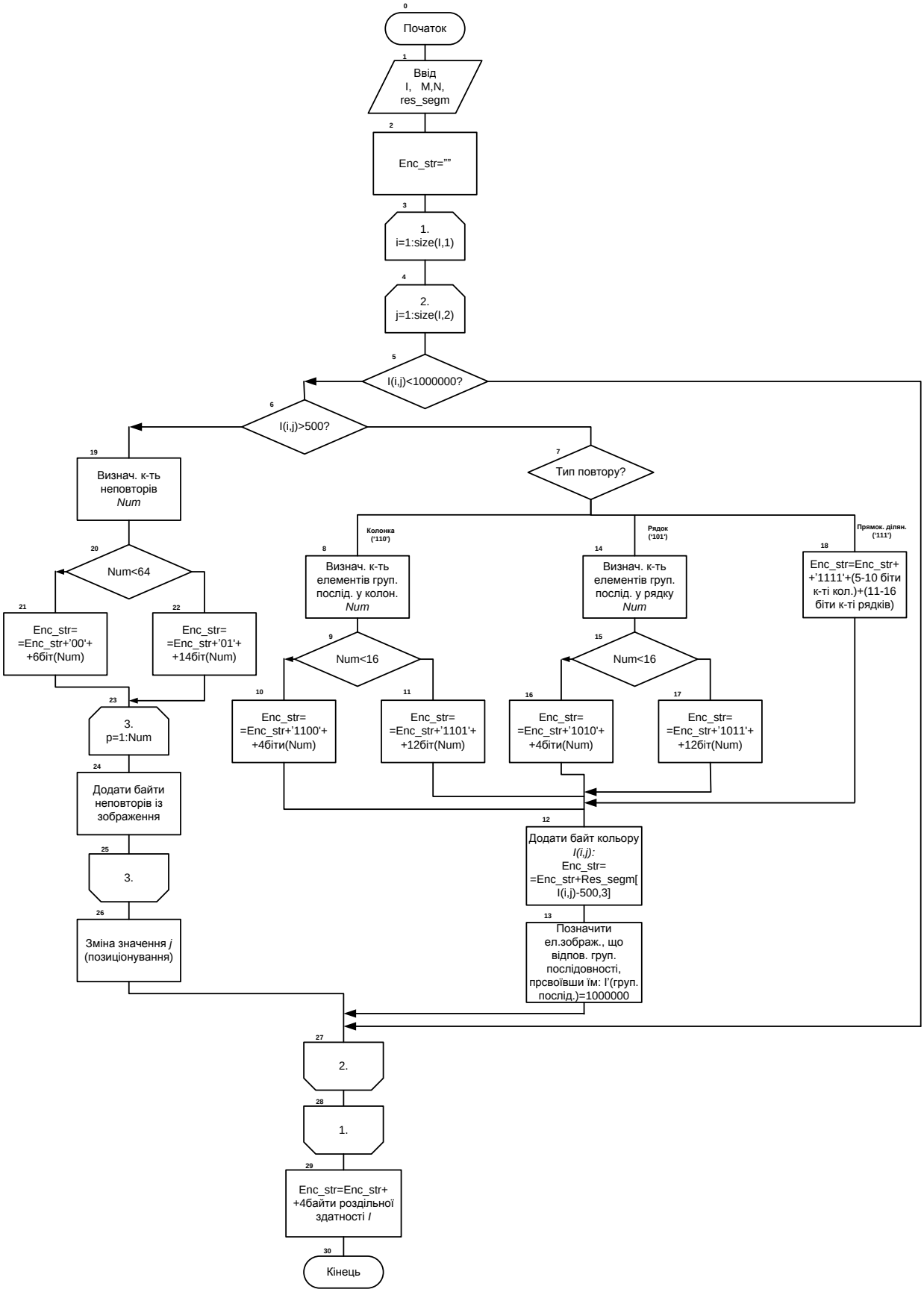
1. Шеннон К. Работы по теории информации и кибернетике. - М.: Мир, 1963. - 830 с.
2. Прэтт У. Цифровая обработка изображений. Кн. 1,2. — М.: Мир, 1982. – 788с.
3. Гузман И.С., Киричук В.С. Цифровая обработка изображений в информационных системах: Учебное пособие. – Новосибирск: Изд-во НГТУ, 2002. –352с.
4. Ярославский Л.П. Введение в цифровую обработку изображений. — М.: Сов. радио, 1979. – 538с.
5. Гонсалес Р., Вудс Р. Цифровая обработка изображений. - М.:Техносфера, 2005. –1072 с.
6. http://nevsepic.com.ua/art-i-risovanaya-grafika/3734-hudozhnik-james-abbott-mcneill-whistler-1834-1903-99-rabot.html
7. Миано Дж. Форматы и алгоритмы сжатия изображений в действии. – М.: Издательство Триумф, 2003. — 336 с.
8. Ватолин Д.С. Алгоритмы Сжатия изображений. – М.: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В.Ломоносова (лицензия ЛР № 040777 от 23.07.96), 1999 г. - 76 с.
9. Мюррей Д.Д. , Райнер У. Ван. Энциклопедия форматов графических файлов. Пер. с англ. - Киев: BHV, 1997, - 535с.
10. Бабак В.П., Хандецький В.С., Шрюфер Е.К. Обробка сигналів. - К.: Либідь, 1996. - 392 с.
11. Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования. – К.: Высш. Школа, 1986. – 430 с.
12. Дмитриев В.И., Прикладная теория информации. –М.: Высш. школа, 1989. - 230с.
13. Сергиенко А.Б. Цифровая обработка сигналов. – Спб.:Питер, 2003. – 608 с.
14. Bell T., Witten I., Clearly J. Modeling For Text Compression // ACM Computing Surveys, Vol.21, No.4, pp.557-591, Dec. 1989.

www.compression.ru/download/articles/rev_univ/bell_1989_modeling/bell_1989_modeling_part1.html

15. Бредихин Д. Ю. Сжатие графики без потерь качества. – 2004. – 21 с. http://www.compression.ru/download/articles/i_lless/bredikhin_2004_lossless_image_compression.pdf
16. Кривошеев М.И.,Виленчик С.Л., Красносельский И.Н. Цифровое телевидение. - М.: Связь, 1980. - 264 с.
17. Цуккерман И.И. Цифровое кодирование телевизионных изображений. - М.: Радио и связь, 1981. -240 с.
18. Птачек М. Цифровое телевидение. Теория и техника: пер. с чешск. - М.: Радио и связь, 1990. - 528 с.
19. Харатишвили Н.Г. Цифровое кодирование с предсказанием непрерывных сигналов. - М.: Радио и связь, 1986. – 25с.
20. Хуанга Т.С. Быстрые алгоритмы в цифровой обработке изображений. - М.: Радио и связь, 1984. - 224с.
21. Кунт М., Джонсон О. Блочное кодирование графических материаловв. Обзор//ТИИЭР. - 1980. - Т.68. - №7. - С.21-40.
22. Брауде-Золотарев Ю.М., Кожемяко В.П., Майданюк В.П. Особенности кодирования цифровых телевизионных сигналов вещательного телевидения // 1-я Всесоюзная конференция "Распознавание образов и анализ изображений: новые информационные технологии": Тез. докл. - Минск, 1991. - Ч.1, С. 31-36.
23. Майданюк В.П. Разработка алгоритмов и аппаратурных средств систем сжатия телевизионных изображений.: Дис. канд.техн.наук. Винница, 1993. - 223с.
24. Майданюк В.П. Кодування зображень в комп'ютерних системах. - К.,1996. - 16с.
25. Барнсли М., Ансон Л.Фрактальное сжатие изображений // Мир ПК.1992. - №10. - С. 52-58.
26. http://www.dip.ee.uct.ac.za/ imageproc/compression/fractal/.
27. Сэломон Д. Сжатие данных, изображения и звука. — М.: Техносфера, 2004. - 368с.

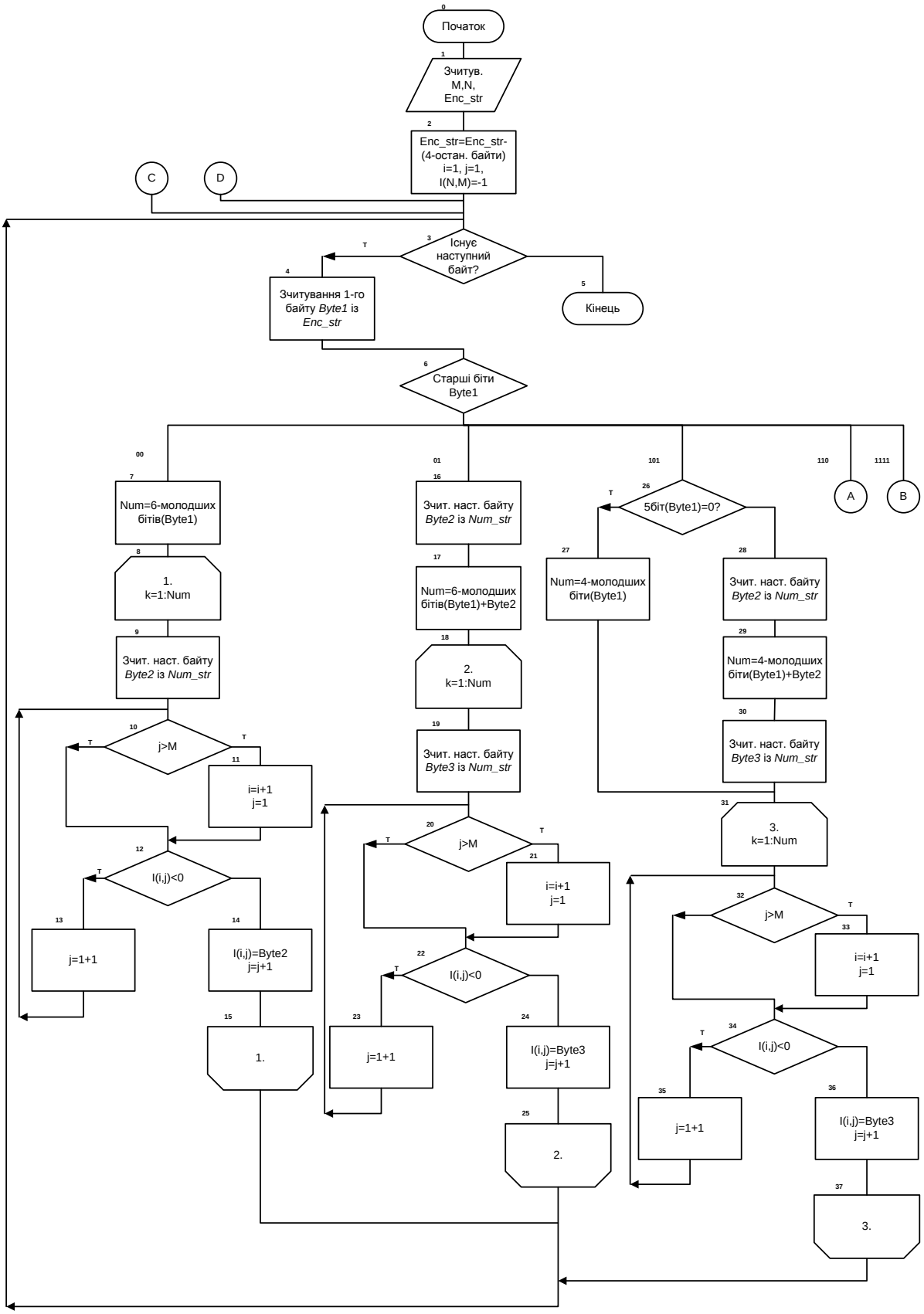
ДОДАТОК А

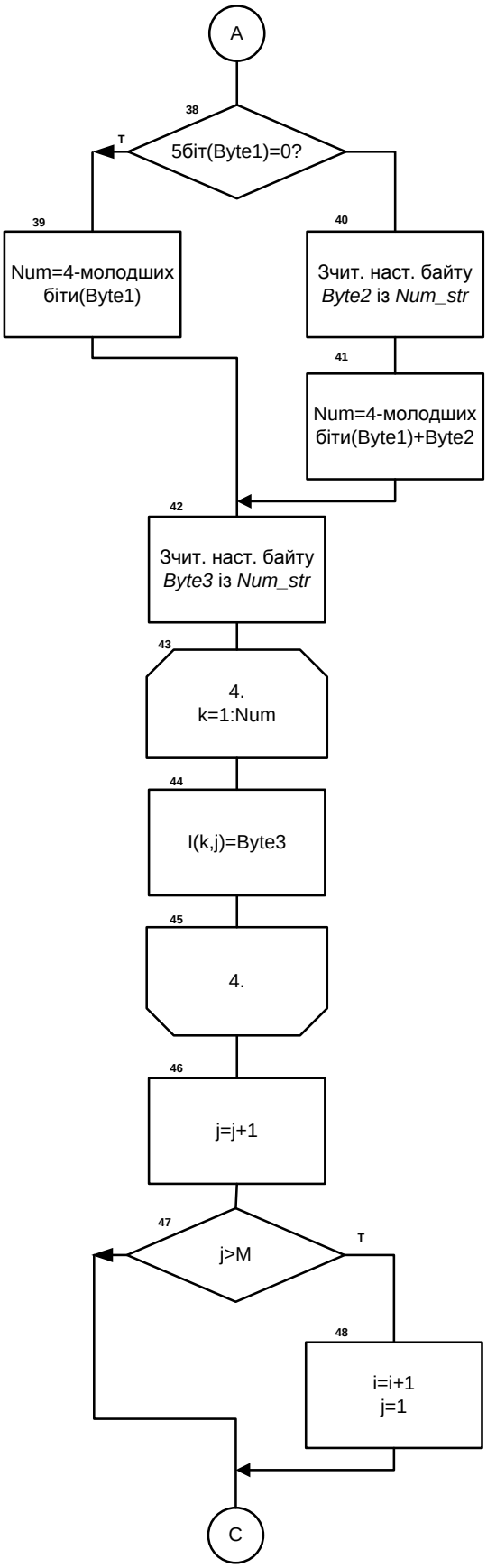
СХЕМА РОБОТИ ПРОЦЕДУРИ КОДУВАННЯ ПОСЛІДОВНОСТЕЙ

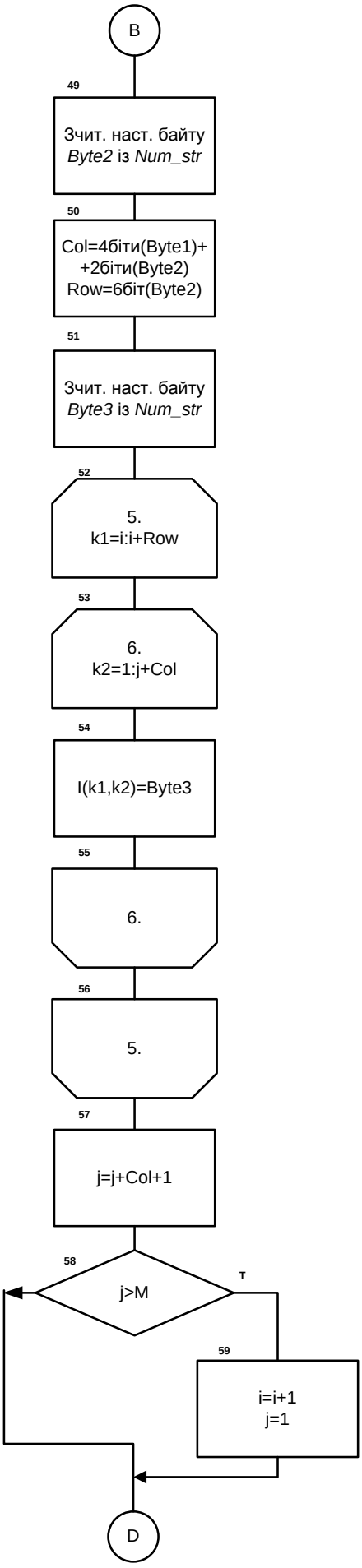


ДОДАТОК Б

СХЕМА РОБОТИ АЛГОРИТМУ ДЕКОДУВАННЯ ЗОБРАЖЕННЯ




A
38
5біт(Byte1)=0?
T
39
Num=4-молодших біти(Byte1)
40
Зчит. наст. байту *Byte2* із *Num_str*
41
Num=4-молодших біти(Byte1)+Byte2
42
Зчит. наст. байту *Byte3* із *Num_str*
43
4.
k=1:Num
44
I(k,j)=Byte3
45
4.
46
j=j+1
47
j>M
T
48
i=i+1
j=1
C


B
49
Зчит. наст. байту *Byte2* із *Num_str*
50
Col=4біти(Byte1)+ +2біти(Byte2)
Row=6біт(Byte2)
51
Зчит. наст. байту *Byte3* із *Num_str*
52
5.
k1=i:i+Row
53
6.
k2=1:j+Col
54
I(k1,k2)=Byte3
55
6.
56
5.
57
j=j+Col+1
58
j>M
T
59
i=i+1
j=1
D

Додаток В

Лістинги програм

```
%--------------- 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));

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

```
                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
ДОВІДКА ПРО ВПРОВАДЖЕННЯ