РЕФЕРАТ

Пояснювальна записка: 56с., 32 рис., 2 табл., 5 додат., 17 джерел.

Об’єкт дослідження: алгоритм фрактального кодування

Предмет дослідження: фрактальне стиснення зображення

Мета дипломної роботи: дослідження фрактального алгоритму кодування, вивчення можливості модифікації класичного фрактального алгоритму для поліпшення якості декодованого зображення і порівняння результатів з вихідним зображенням.

В першому розділі проаналізовано поняття фракталу, його ідея та математичні основи, складено алгоритм фрактального стиснення та розглянуто одне з головних властивостей фракталасамоподоба.

В спеціальній частині розроблено в середовищі Matlab алгоритм фрактального кодування зоражень за допомогою розкладання quadtree, розглянуто переваги та недоліки данного методу та досліджено фрактальний алгоритм стиснення.

В економічній частині проведений розрахунок трудомісткості та капітальні витрати, до яких входять розрахунок витрат на придбання основних засобів та на оплату праці інженера по телекомунікаціям при дослідженні та розробці програми фрактального алгоритму кодування.

ФРАКТАЛ, САМОПОДОБА, КВАДРОДЕРЕВО, ДВІЙКОВИЙ СИМЕТРИЧНИЙ КАНАЛ, КОДУВАННЯ ХАФМАНА

РЕФЕРАТ

Пояснительная записка: 56 с, 32 рис., 2 табл., 5 прил., 17 источников.

Объект исследования: алгоритм фрактального кодирования

Предмет исследования: фрактальное сжатие изображения

Цель дипломной работы: исследование фрактального алгоритма кодирования, изучение возможности модификации классического фрактального алгоритма для улучшения качества декодированного изображения и сравнение результатов с исходным изображением.

В первом разделе проанализированы понятие фрактала, его идея и математические основы, составлен алгоритм фрактального сжатия и рассмотрен один из главных свойств фрактала самоподобие.

В специальной части разработаны в среде Matlab алгоритм фрактального кодирования изображений с помощью разложения quadtree, рассмотрены преимущества и недостатки данного метода и исследованы фрактальный алгоритм сжатия.

В экономической части произведен расчет трудоемкости и капитальные расходы, включающие расчет затрат на приобретение основных средств и на оплату труда инженера по телекоммуникациям при исследовании и разработке программы фрактального алгоритма кодирования.

ФРАКТАЛ, САМОПОДОБИЕ, КВАДРОДЕРЕВО, ДВОИЧНЫЙ СИММЕТРИЧНЫЙ КАНАЛ, КОДИРОВКА ХАФФМАНА

ABSTRACT

Explanatory note: 56 pages., 32 fig., 2 tab., 5 applications., 17 sources.

Object of study: fractal coding algorithm

Subject of research: fractal image compression

The aim of the thesis is to study the fractal coding algorithm, study the possibility of modifying the classical fractal algorithm to improve the quality of the decoded image and compare the results with the original image.

In the first section, the concept of a fractal, its idea and mathematical foundations are analyzed, the algorithm of fractal compression is compiled and one of the main properties of the fractal self-similarity is considered.

In the special part, in the Matlab environment, the algorithm for fractal image coding using the quadtree decomposition was developed, the advantages and disadvantages of this method were considered, and the fractal compression algorithm was investigated.

In the economic part, labor intensity and capital expenditures were calculated, including the calculation of the costs of acquiring fixed assets and the remuneration of a telecommunications engineer in the study and development of the program of the fractal coding algorithm.

FRACTAL, SELF-LIKE, QUADRODEREVO, BINARY SYMMETRIC CHANNEL, HAFMAN CODING

Список умовних скорочень

ФМК - фрактальний метод кодування;

QT - квадродерево (також дерево квадрантів, англ. quadtree);

ДСК - двійковий симетричний канал.

ЗМІСТ

ВСТУП

Завдання ефективного стиснення даних завжди була однією з найактуальніших в інформатиці, а з розвитком передачі зображень по мережі Інтернет вона набула ще більшого значення. Так, з початку 1980-х рр. було розроблено безліч алгоритмів, спрямованих на ефективне стиснення цифрових зображень для їх передачі та зберігання в компенсованому вигляді.

Розвиток теорії фракталів та відповідного математичного апарату в кінці XX ст. спричинило розробку принципово нового алгоритму стиснення зображень - фрактального, який дає високі коефіцієнти стиснення при прийнятній якості. Ідея фрактального методу стиснення стосовно до зображень була сформульована Майклом Барнслі в 1990 році, і досить швидко отримала резонанс в наукових колах. Більшість наукових статей, що видаються на цю тему, мають на меті усунути головний недолік алгоритму - великий час стиснення, шляхом класифікації та оптимізації величезного числа перебираються подобластей зображення.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

Метою даної роботи є дослідження фрактального алгоритму кодування, вивчення можливості модифікації класичного фрактального алгоритму для поліпшення якості декодованого зображення і порівняння результатів з вихідним зображенням.

Основними завданнями, поставленими для досягнення мети, є розробка і дослідження оптимального способу розбиття зображення на доменні і рангові блоки, з'ясування ідеї даного методу стиснення та побудувати типову схему фрактального алгоритму кодування

1.1 Поняття фракталу

В кінці 70-х з'явилися поняття фрактал і фрактальна геометрія, що міцно увійшли в ужиток математиків і програмістів. У 1975 році БенуаМандельброт запропонував позначати нерегулярні, але самоподібні структури терміном фрактал, що в перекладі з латинської (fractus) означає що складається з фрагментів. У 1977 році була видана книга Мандельброта "TheFractalGeometryofNature", і саме з її виходом пов'язують народження фрактальної геометрії.

Сьогодні роль фракталів в машинній графіці є досить велика. З їх допомогою відбувається побудова ліній і поверхонь складної форми. Фрактальна геометрія, з точки зору машинної графіки, є незамінною при створенні штучних гір, хмар, а також поверхні моря. Можна говорити про те, що знайдений легкий спосіб представлення складних неевклідових об'єктів, чиї образи дійсно дуже схожі на природні.

Одним з головних властивостей фрактала вважається самоподоба - інваріантність щодо переносів і скейлінга (зміни масштабу). Це означає, що фрактал в будь-якій мірі одноманітно влаштований великому діапазоні масштабів. Наприклад, якщо збільшувати маленькі елементи фрактала, то вони

вийдуть схожими на великі. В ідеальному випадку фрактальному об'єкту властива незмінність при зміні масштабу найбільш істотних геометричних особливостей фрактала. У більшості природних об'єктів спостерігається самоподоба, яке є основним організуючим принципом фракталів. Тому не залежно від використовуваної шкали, фрактали будуть нагадувати один одного. Саме тому, визначення фрактала, яке дав Мандельброт, звучить так: «фракталом називається структура, що складається з частин, які в якомусь сенсі подібні цілому» [2].

1.2 Класифікації фракталів

Зазвичай фрактали ділять на алгебраїчні, геометричні та стохастичні. Стохастичні фрактали при певних умовах можуть називатися мультіфрактали. Також існують й інші класифікації:

- Рукотворні і природні. Рукотворні - фрактали, при будь-якому масштабі володіють фрактальними властивостями. Природні - фрактали з обмеженням області існування - максимальний і мінімальний розмір, при яких фрактальні властивості спостерігаються у об'єкта[14].

- Детерміновані (геометричні та алгебраїчні) і недетерміновані (стохастичні).

1) Алгебраїчні фрактали.

Для побудови фракталів даного типу застосовуються ітерації нелінійних відображень, задані алгебраїчними формулами. Побудова фракталів здійснюється двома способами: перший - використання L-систем (від імені Lindenmayer), другий - використання системи IFS (iterated functions ystems). Приклад: побудова дерева з використанням L-системи (рис. 1.1) [6].

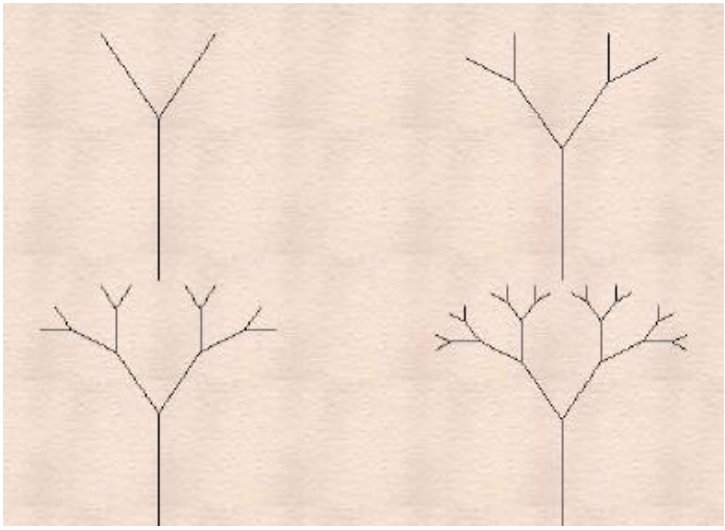


Рисунок 1.1 - Побудова дерева

Побудова IFS - фракталів здійснюється за наступним алгоритмом: вибирається деякий безліч правил здійснення переходу з поточної точки в наступну, з імовірністю p застосування кожного правила. Починаючи з точки (0; 0), правило вибирається випадковим чином і визначається наступна точка, з неї знаходиться наступна, і так далі[11].

Приклад: лист папороті з використанням системи IFS (рис. 1.2).

Рисунок 1.2 - Лист папороті

2) Геометричні фрактали.

В даних фракталах відразу видно самоподібність, тому вони самі наочні. У двовимірному випадку задають деяку ламану, яка називається генератором, і за один крок алгоритму кожен відрізок ламаної змінюється на ламану-генератор, причому у відповідному масштабі. Після багаторазового повторення даної процедури утворюється фрактальная крива[8].

Приклад: крива Коха (рис. 1.3).

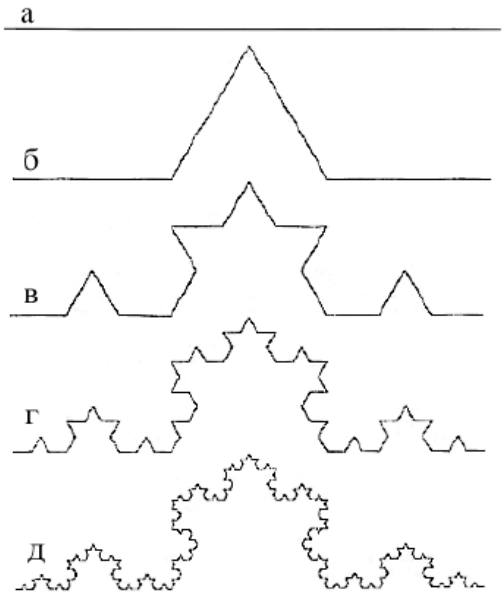


Рисунок 1.3 - Крива Коха

3) Стохастичні фрактали

Фрактали, при побудові яких змінюються різні параметри абсолютно випадковим чином, називаються стохастичними. Вони знаходять своє відображення в фізичних процесах, тому вони найбільш цікаві для фізиків. Двовимірні стохастичні фрактали застосовуються при моделюванні поверхні моря або рельєфу місцевості.

Приклад: стохастичний фрактал на основі безлічі Жюліа (рис. 1.4).

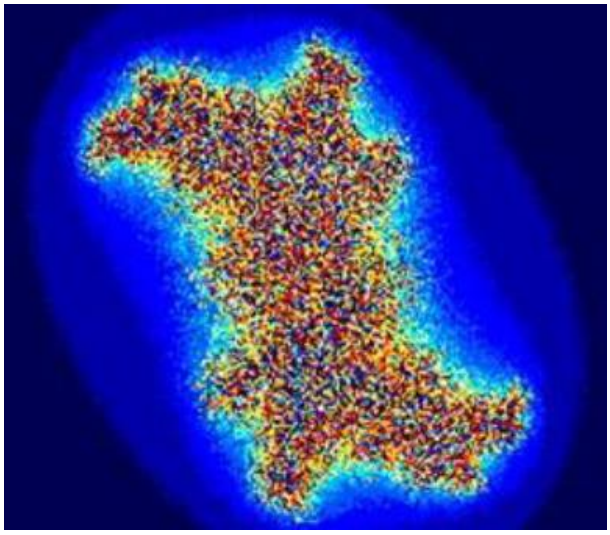


Рисунок 1.4 - Стохастичний фрактал на основі безлічі Жюліа

1.3 Ідея методу фрактального стиснення

Основа ФМК - це виявлення самоподібних ділянок в зображенні, на основі яких в стислий файл заносяться параметри геометричних перетворень, що дозволяють відновити великі ділянки зображення з подібних ділянок меншого розміру. Для кожного рангового блоку алгоритм кодування знаходить найбільш підходящий доменний блок і афінне перетворення, яке переводить цей доменний блок в ранговий блок. Іншими словами, структура зображення відображається в систему рангових блоків, доменних блоків і перетворень[15].

Така система перетворень називається системою ітеративних функцій (IteratedFunctionSystem - IFS). Метод IFS стосовно побудови фрактальнихсамоподібних зображень був винайдений Майклом Барнслі (MichaelBarnsley) в кінці 80-х років минулого століття. Однак у нього були відсутні будь-які відомості про рішення оберненої задачі: як по заданому зображенню знайти аффінне перетворення. У статті Барнслі було показано кілька реалістичних фрактальних зображень, але всі вони були створені вручну[16].

У загальному випадку, необхідно вміти знаходити для будь-якого зображення систему аффінних перетворень (IFSM), яка б відтворювала зображення із заданою точністю:

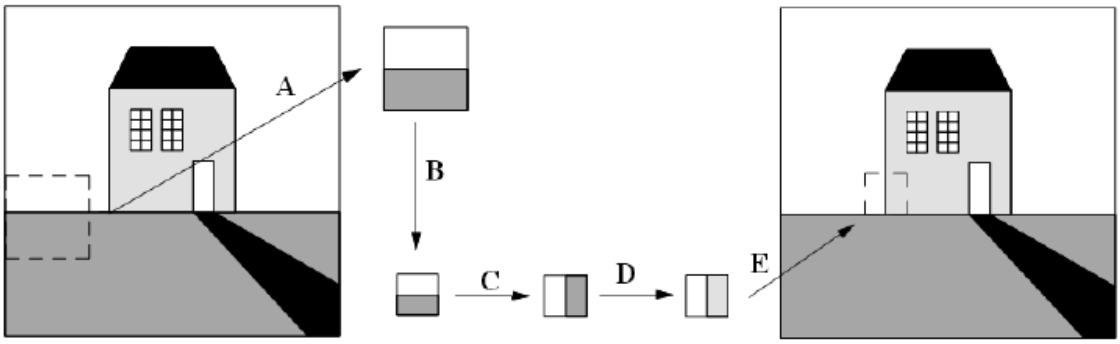


Рисунок 1.5 - Схема фрактального кодування за допомогою IFS

На рис. 1.5 буквами позначені наступні дії: A - беремо черговий доменний блок; B - стискаємо до розміру рангового блоку; З - застосовуємо афінне перетворення; D - підбираємо оптимальні коефіцієнти яскравості і контрастності; E - порівнюємо з ранговим блоком.

Алгоритм фрактального стиснення істотно ассиметричен за часом. Час кодування зображення може перевищувати аналогічний час затрачуване алгоритмом JPEG в 1000 разів, при цьому декодування відбувається набагато швидше. Проте, ступінь стиснення може теоретично досягати 1: 2000, тому фрактальний алгоритм кодування має специфічні сфери застосування. Наприклад, доступ до бібліотек готових зображень через Internet, коли більш важливий обсяг переданого файлу та час розархівації. Також особливістю фрактального алгоритму є можливість збільшення розміру зображення в 2 - 4 рази без появи блочного ефекту.

1.4 Математичні основи фрактального стиснення

Отже, розглянемо математичне обґрунтування можливості фрактального стиснення.

Є відображення $W : \Delta \to \Delta$, де Δ – множина всіх можливих зображень. *W* є об'єднанням відображень w_i:

$$W(R) = \bigcup_i w_i(d_i) \tag{1.1}$$

де *R* – зображеня, а d_i – якісь області зображення *D*. Кожне перетворення w_i переводить d_i в r_i. Таким чином:

$$W(R) = \bigcup_i r_i \tag{1.2}$$

Буде логічно уявити зображення у вигляді функції двох змінних *f (x, y)*. На безлічі всіх таких функцій введемо метрику (відстань між зображеннями), наприклад, таким чином:

$$\delta(f, g) = \max_{x,y} \left(\left| f(x,y) - g(x,y) \right| \right) \tag{1.3}$$

Згідно з теоремою Банаха, існує певний клас відображень, для яких існує константа $c < 1$ така, що для будь-яких зображень *f* і *g* виконується нерівність

$$\delta(W(f), W(g)) \le c \cdot \delta(f, g) \tag{1.4}$$

Такі відображення називаються стискають, і для них справедливо наступне твердження:

Якщо до якогось зображення F_0 ми почнемо багаторазово застосовувати відображення *W* таким чином, що

$$\begin{aligned} F_1 &= W(F_0) \\ F_i &= W(F_{i-1}) \end{aligned} \tag{1.5}$$

то в межі, при *i,* яка прагне до нескінченності, ми отримаємо одне і те ж зображення незалежно від того, яке зображення ми взяли в якостіF_0:

$$\lim_{i\to\infty} F_i = F \qquad (1.6)$$

Це кінцеве зображення *F* називають аттрактором, або нерухомою точкою відображення *W*. Також відомо, що якщо перетворення *wi* є стискають, то їх об'єднання *W* теж є стискає[4].

1.5 Типова схема фрактального стиснення

З урахуванням вищесказаного, схема компресії виглядає так: зображення *R* розбивається на шматочки r_i, звані ранговими областями. Далі для кожної області r_i знаходят область d_i і перетворення w_i такі, що виконуються наступні умови:

1. d_i за розмірами більше r_i.
2. w_i (r_i) має ту ж форму, розміри і положення, що і r_i.
3. Коефіцієнт *u* перетворення w_i повинен бути менше одиниці.
4. Значення має бути якомога менше.

Перші три умови означають, що відображення wi буде стискаючим. А в силу четвертої умови кодоване зображення *R* і його образ *W (R)* будуть схожі один на одного. В ідеалі *R = W (R).* А це означає, що наше зображення *R* і буде нерухомою точкою *W*[17]. Саме тут використовується подобу різних частин зображення (звідси і назва - «фрактальна компресія»). Як виявилося, практично всі реальні зображення містять такі схожі один на одного, з точністю до афінного перетворення, частини.

Таким чином, для компресії зображення *W* потрібно:

1. Розбити зображення на рангові області r_i (непересічні області, що покривають все зображення).

2. Для кожної рангової області r_i знайти область d_i (звану доменної), і відображення w_i, з зазначеними вище властивостями.
3. Запам'ятати коефіцієнти афінних перетворень W, положення доменних областей d_i, а також розбиття зображення на домени.

Відповідно, для декомпресії зображення потрібно буде:

1. Створити якийсь (будь-яке) початкове зображення R_0.

2. Багаторазово застосувати до нього відображення W (об'єднання w_i).

3. Так як відображення W стискуюче, то в результаті, після достатньої кількості ітерацій, зображення прийде до аттрактору і перестане змінюватися. Аттрактор і є нашим вихідним зображенням. Декомпресія завершена.

Нехай дано зображення M x N точок (де M і N кратні 8), 256 градацій сірого. Рангові та доменні області братимемо квадратними. Початкове зображення розіб'ємо на рангові області розміром 8 x 8 точок. Доменні області шукатимемо розміром 16 x 16 точок шляхом перебору всіх можливих положень[10]. Існує всього 8 афінних перетворень, що переводять квадрат в квадрат (повороти на 0 °, 90 °, 180 °, 270 °, дзеркальні відображення відносно центральної горизонталі, центральної вертикалі, від головної та побічної діагоналей). Залишилося знайти тільки коефіцієнти для перетворення кольору. Але значення u і v (контрастності і яскравості) можна легко знайти аналітично[9].

Якщо є дві послідовності значень кольору пікселів $a1, a2, ..., aN$ (доменної області) і $b1, b2, ..., bN$ (рангової області), то можна мінімізувати середньоквадратичне відхилення кольору пікселів, що представляє собою варіант метрики відмінності зображень:

$$R = \sum_{i=1}^{N} (u \cdot a_i + v - b_i)^2 \qquad (1.7)$$

Для цього досить прирівняти приватні похідні *R* по *u* і по *v* до нуля, і вирішити рівняння щодо *u* і *v*. Вийдуть такі вирази:

$$u = \frac{N \cdot \sum_{i=1}^{N} a_i b_i - \sum_{i=1}^{N} b_i \sum_{i=1}^{N} a_i}{N \cdot \sum_{i=1}^{N} a_i^2 - \left(\sum_{i=1}^{N} a_i \right)^2}, \quad \frac{\sum_{i=1}^{N} b_i - u \cdot \sum_{i=1}^{N} a_i}{N} \tag{1.8}$$

при цьому, якщо

$$N \cdot \sum_{i=1}^{n} a_i^2 - \left(\sum_{i=1}^{n} a_i \right)^2 = 0$$

то

$$u = 0, \quad v = \frac{\sum_{i=1}^{n} b_i}{N} \tag{1.9}$$

Отже, які ж дані необхідно зберігати в результаті. Сітка розбиття на рангові області постійна для всіх зображень, її зберігати не треба. Залишається положення рангових областей (верхнього лівого кута), номер перетворення і коефіцієнти яскравості і контрастності.

1.6 Постановка завдання

Метою данної роботи є дослідження ФМК зображення. Програма повинна бути представлена в середовищі Matlab. За допомогою деяких функцій необхідно написати програму для демонстрації стиснення зображень фрактальним методом.

Для демонстрації потрібно підготовити робочий простір для вибору зображення, після чого застосувати quadtree розкладання, що є одним з методів фрактального кодування. Після чого використувати кодування Хаффмана для побудови оптимального кодового дерева і відображення код-символ на

основі побудованого дерева, та виконати стиснення зображення. Передати дані вхідного сигналу через двійковий симетричний канал, що має задану ймовірність помилки. Декодувати методом Хаффмана і вивести на екран вихідне зображення та розпаковане.

1.7 Висновки

У першому розділі було розглянуто поняття фракталу, його ідея та математичні основі, після чого зроблено висновки, що одним з головних властивостей фрактала вважається самоподоба - інваріантність щодо переносів і скейлінга. А головним завданням при компресії фрактальним алгоритмом є знаходження відповідних афінних перетворень. Алгоритм фрактального стиснення істотно асиметричен за часом. Час кодування зображення може перевищувати аналогічний час затрачуване алгоритмом JPEG в 1000 разів, при цьому декодування відбувається набагато швидше. Також з'ясовано, що необхідно зберігати в результаті положення рангових областей (верхнього лівого кута), номер перетворення і коефіцієнти яскравості і контрастності.

РОЗДІЛ 2. СПЕЦІАЛЬНА ЧАСТИНА

2.1. Дослідження фрактального алгоритму кодування

Як вже стало відомим з описаної в математичної частини вище теорії, головним завданням при компресії фрактальним алгоритмом є знаходження відповідних афінних перетворень. В теорії можна переводити будь-які за розміром і формою області зображення, проте в цьому випадку виходить величезна кількість варіантів різних фрагментів, які швидко неможливо обробити на поточний момент навіть на потужних комп'ютерах.

Алгоритм методу виглядає наступним чином:

1. Зображення R розбивають на блоки ri, звані ранговими і доменні блокиd, причому сторона домену в два рази більше сторони рангової області. Рангові блоки покривають все зображення цілком, а доменні можуть розташовуватися довільним чином і не покривати всього зображення.

2. Домени зменшуються до розмірів рангу шляхом усереднення яскравості сусідніх пікселів.

3. Кожному рангу підбирається найбільш схожий на нього домен після перетворення, що зменшує різницю між доменом і рангом, шляхом перебору всіх доменів або їх частини. Перетворення включає в себе зміну контрасту і яскравості домену.

Перетворення домену, зменшеного до розмірів рангу, має вигляд:

$$T(D) = sD + oI,$$

де $|s| < 1$- коефіцієнт контрасту; $o \in [-255; 255]$- коефіцієнт яскравості; I - єдина матриця.

Накладаються обмеження потрібні для гарантії збіжності при відновленні зображення.

4. Коефіцієнти перетворення контрасту s і яскравості o обчислюють за формулами

$$s = \frac{n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j} - \sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j} \sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j}}{n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j}^2 - \left(\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}\right)^2}, \tag{2.1}$$

якщо

$$n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} D^2{}_{i,j} - (\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j})^2 = 0, \tag{2.2}$$

то s=0

$$o = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} R_{i,j} - s\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}}{n^2}, \tag{2.3}$$

де *D* і *R*- матриці, що представляють собою відповідно домен, зменшений до розміру рангу, і ранг; n- розмір сторони рангового блоку.

5. На основі розрахованих коефіцієнтів перетворення оцінюється відповідність домену та рангу за формулою

$$r = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} R^2{}_{i,j} + s\left(s\sum_{i=1}^{n}\sum_{j=1}^{n} D^2{}_{i,j} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}R_{i,j} + 2o\sum_{i=1}^{n}\sum_{j=1}^{n} D_{i,j}\right) + o\left(on^2 - 2\sum_{i=1}^{n}\sum_{i=1}^{n} R_{i,j}\right)}{n^4}. \tag{2.4}$$

Оптимальний домен для даного рангу визначається шляхом перебору всіх доменних областей. Наведений вище алгоритм застосовується для всіх рангових блоків зображення. В результаті для кожного рангу визначаються координати найбільш схожого на нього домену, разом з коефіцієнтами перетворення доменного блоку в рангові, що мінімізують різницю між ними[13].

Блок схема алгоритму представлена на рис 2.1

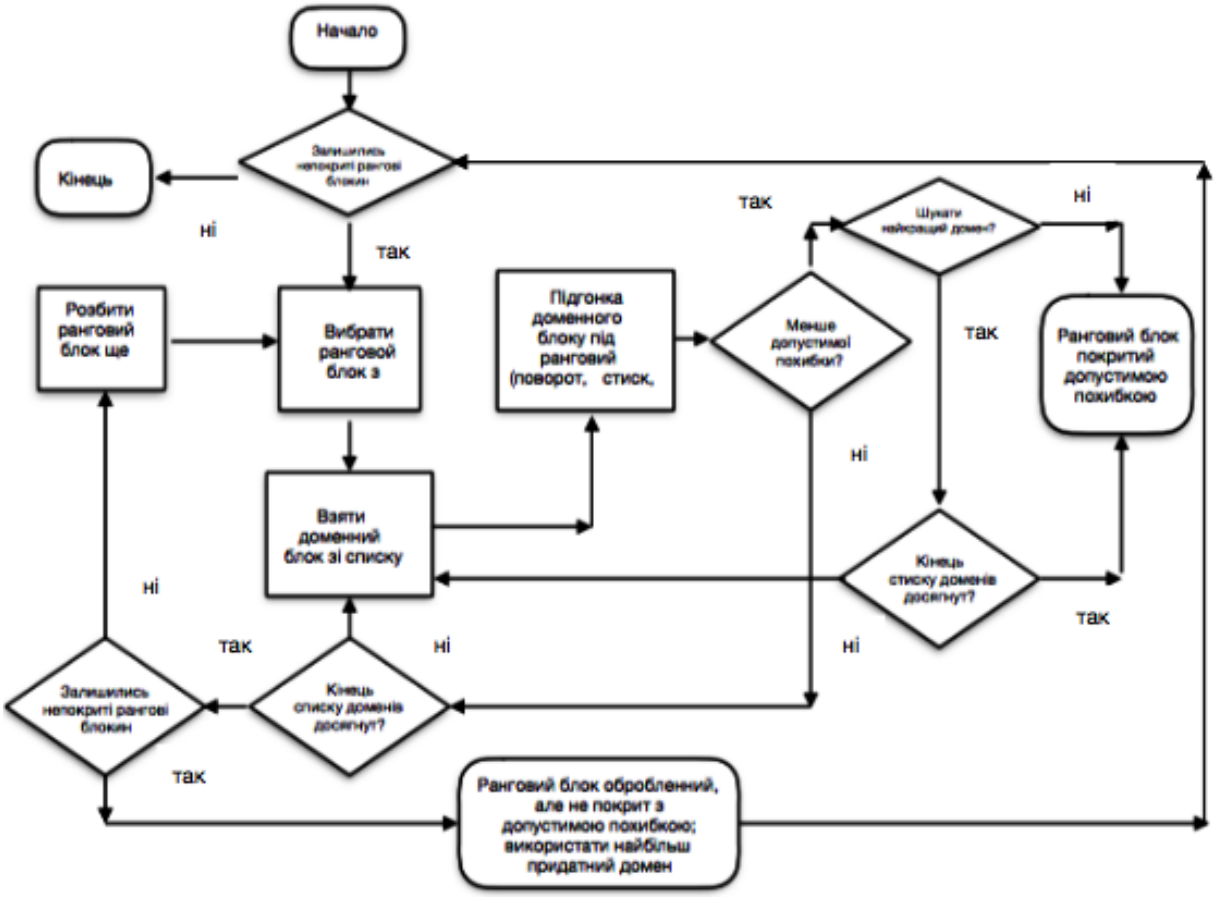


Рисунок 2.1 - Блок схема основних кроків кодування

Для відновлення стисненого зображення необхідно виконати наступні кроки:

1. Створюються два зображення однакового розміру А і Б, при цьому розмір не обов'язково дорівнює розміру початкового зображення. Початковий малюнок областей А і Б не має значення.

2. Дані з області А перетворюються в область Б. Для цього спочатку зображення Б ділиться на домени, як на стадії процесу стиснення (в заголовку файлу описано розташування доменів). Для кожного домена області Б проводиться відповідне афінське перетворення рангових областей зображення А, яке описано коефіцієнтами з стисненого файлу. Результат записується в область Б. На цій стадії обробки з'являється нове зображення.

3. Дані з області Б перетворюються в область А. Дія подібно до попереднього, тільки зображення А і Б міняються місцями, тобто область А ділиться на блоки і рангові області зображення Б записуються на ці блоки.

4. Дії 2 і 3 треба повторювати до тих пір, поки зображення А і Б не стануть непомітними.

Алгоритм фрактального стиснення – несиметричний алгоритм. Це виражається тим, що пряме перетворення (стиснення) вимагає часу значно довше, ніж зворотне перетворення (відновлення).

До незаперечних переваг ФМК можна віднести такі як: високі коефіцієнти стиснення, високу швидкість відновлення. До недоліків можна віднести: тимчасові витрати на стиск, залежність результату від принципів відбору базових елементів і доменів.

2.2. Розкладання QuadTree

Однією з найбільш зарекомендованних структур даних для представлення двовимірних зображень є квадродерево (QT). Завдяки природній ієрархічній структурі і способу організації QT поєднують в собі значну економію обсягів пам'яті з ефективністю доступу до елементів зображення. Ідеологія QT застосовується не тільки для подання растрових зображень, але і використовується для ефективної організації великих баз будь-яких просторових даних, що складаються як з растрових, так і векторних зображень[3].

Спосіб просторової індексації точок площині - квадродерево - було застосовано М. Бранслі в його перших статтях по фрактальному стиску. При побудові QT спочатку проводиться грубе розбиття зображення, наприклад на чотири прямокутники. Для кожного отриманого рангового блоку алгоритм намагається підібрати відповідний доменний блок і коефіцієнти відображення, які найкращим чином покривають рангові блок. Якщо вийшло покриття в межах допустимої похибки, то цей ранговий блок відзначається як оброблений і береться наступний. У разі, якщо покриття не знайдено, блок аналогіч-

но розбивається на 4 частини. Процедура розбиття проводиться до тих пір, поки не буде досягнута максимальна задана глибина вкладеності.

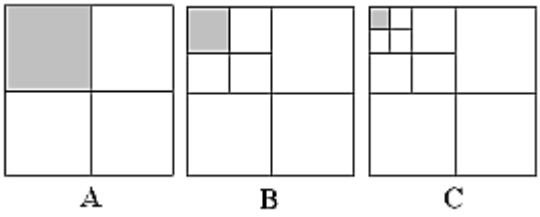


Рисунок 2.2 - Індексація областей в QT: A-100, B-110, C-111

Найпростіша нумерація областей в QT наступна: кожна область представляється у вигляді списку з 4х векторів, довжина кожного вектора дорівнює максимальному рівню вкладеності. Значення першого вектора показує номер прямокутника, відлічуваний за годинниковою стрілкою і отриманий на першому кроці розбиття. Решта вектора визначаються аналогічно. Наприклад, для зафарбованих областей на рис. 2.2 індексація буде наступна: для A-100, для B-110 і для C-111.

У пропонованому варіанті реалізовано інше уявлення індексу. Все QT представляється у вигляді звичайного дерева:

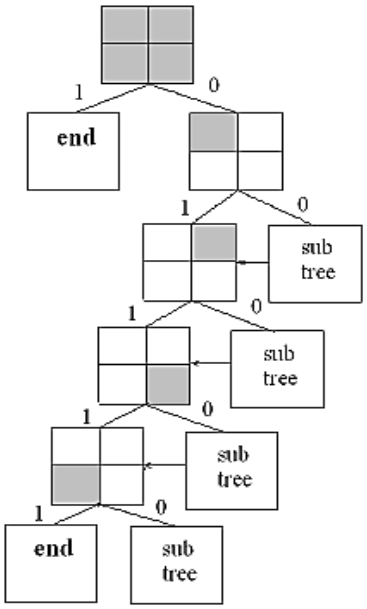


Рисунок. 2.3 - Альтернативна індексація QT

Після увімкнення алгоритму покажчик встановлений на корінь дерева, індекс корня- 000, тобто рангові блок спочатку дорівнює всьому зображенню. Залежно від результату покриття (порівняння за допомогою будь-якої метрики візуальної схожості блоків), повертається значення 1 - покриття вдалося, або 0 - покриття не вдалося. Таким чином, ми переходимо по одній з гілок дерева. Блок «subtree» встановлює покажчик на знов сгенероване дерево. Це дерево перебирає підобласті в межах того блоку, який його викликав. У міру проходження піддерева, блок, який викликав піддерево, виявиться покритим (або буде здійснено прохід до максимального рівня вкладеності), а покажчик встановлюється по гілці «1» викликаного блоку[1].

Витрати на зберігання індексу у другому способі індексації істотно менше. Наприклад, для всіх блоків позначених на рис 2.2 С розмір індексу першого способу S = 10 (блоків) * 2 (біта на 1 вектор) * 2 * 2 = 80 біт, для другого випадку індекс в який входять результати переходу по гілках дерева - 000111111111111 - 15 біт. Таким чином, другий спосіб індексації істотно економніше і дозволяє передавати послідовність рангових областей окремо від коефіцієнтів перетворень кожної області.

При стандартному підході розбиття зображення способом QT в разі, якщо для рангового блоку не знайдено покриття доменним блоком, то він розбивається на 4 підблока. При цьому виникає безліч випадків, коли рангові блок досить було б розбити всього лише на 2 підблока для здійснення шуканого покриття. Щоб позбавиться від надмірності рангових блоків, Ю.Фішер запропонував використовувати так зване HV-розбиття. При цьому розбитті рангові блоки діляться на 2 підблока, і лінія розбиття блоку проводиться з урахуванням меж зображення. Ф Девойн (F. Davoine) для зменшення числа рангових блоків і поліпшення апроксимації структури зображення запропонував використовувати змішане триангулярне-квадратичне розбиття.

Пропонується інше розширення ідеї QT, де врахована можливість поділення непокритого рангового блоку не тільки на 2 підблока, але і можливо на 3, при цьому зберігається простота індексації QT, описана вище.

При розбитті рангового блоку з метою порівняння спочатку видаються підблоки A, B, C, D - містять всілякі поєднання чвертей. Якщо жодне з них не вдалося покрити доменним блоком, то тоді алгоритм перебору запускається вже для першої чверті (рис. 2.4 E).

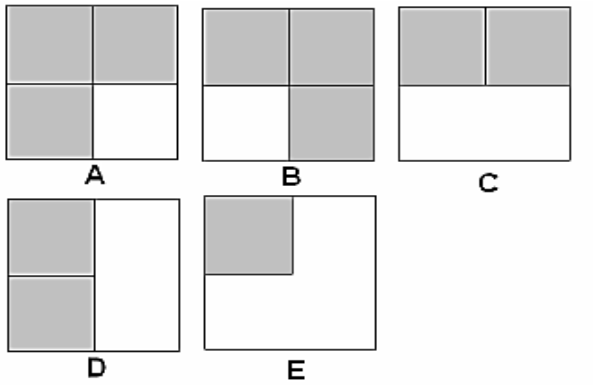


Рисунок.2.4 - Можливі поєднання підблоков

У разі, якщо вдалося знайти прийнятне покриття для одного з поєднань підблоків, то решта непокрита частина дробиться аналогічно, і перебираються все підрозбиття, починаючи з частин з максимальною площею.

У порівнянні зі звичайним QT, розширений спосіб в більшості випадків дає виграш в стисненні близько 10-15%, ефективність від його застосування також зростає при збільшенні припустимої помилки візуального подібності блоків.

2.3 Візуалізація квадротомірованного зображення, його переваги і недоліки

Візуалізація зображення, представленого QT, являє собою просту рекурсивну процедуру. Починаючи з кореня дерева проглядається кожен вузол. Якщо вузол не є листовим, то він пропускається і проглядається його перший дочірній вузол. Якщо цей дочірній вузол не листовий, то він пропускається і відбувається перехід до його першого дочірнього вузла і т.д. При досягненні листового вузла його колірне значення відображається у відповідній позиції.

Після цього відбувається рекурсивне повернення до пропущеного батьківського вузла і аналогічним чином проглядається його другий дочірній вузол. Цей процес триває до тих пір, поки не будуть відвідані всі листя дерева. Візуалізація повного зображення відбувається в міру просування по QT.

Більшість додатків QT до даних було зроблено для зображень (Klinger, Dyer, 1976), але були проведені також сучасні алгоритмічні розробки, які дали результати, подібні до тих, що використовуються при обробці географічних даних. Сюди входять розрахунки площ, розпізнавання образів, класифікація зображень, визначення сусідства, перетворення відстаней, поділ зображень, згладжування даних і посилення крайових ефектів. Внаслідок цих переваг окремі дослідники запропонували використовувати QT для зберігання географічних даних. Основна перевага QT складається в компактному поданні зображення. Компактність QT цілком залежить від зображення. Зображення з великими областями, пофарбованими в один колір представляються дуже компактно, в той час як зображення, в якому всі пікселі відображаються різними кольорами зводить нанівець всі переваги QT.

Як і для будь-якої іншої "деревної" структури даних, ієрархічну структуру QT дозволяє забезпечувати високоефективний доступ до елементів дерева. Існує також безліч алгоритмів обходу і маніпулювання деревами в їх різних формах і всі ці напрацювання можуть бути поширені на QT.

Головний недолік QT полягає в тому, що майже неможливо порівняти два зображення, які відрізняються, наприклад, лише поворотом. Це обумовлено тим, що QT, що представляють такі зображення є абсолютно різними. Алгоритми повороту квадротомірованного зображення обмежуються лише поворотами на кути, кратні 90 градусам. Повороти на будь-які інші кути неможливі[2].

Хоча QT мають масу плюсів в Геоінформаційних-додатках, їх поширення в інших областях стримується їх недоліками. Більшість проблем пов'язано саме з тим, що при повороті зображення для нього доводиться заново перебудовувати QT. При цьому зіставлення QT вихідного і повернутого зо-

бражень стає дуже складним завданням. Внаслідок цього застосування QT, наприклад, в геометричному аналізі форм і розпізнаванні образів залишається досить вузьким.

2.4 Двійковий симетричний канал

Двійковий симетричний канал (BinarySymmetricChannel) визначається діаграмою ймовірностей переходу, зображеної на рис. 2.5 На вхід каналу надходять виконавчі сигнали, наприклад 0 і 1. Для кожного з цих вхідних сигналів є ймовірність $q_0 > \frac{1}{2}$ того, що цей сигнал отриманий правильно, і ймовірність $p_0 = 1 - q_0 < \frac{1}{2}$ того, що він отримано неправильно.

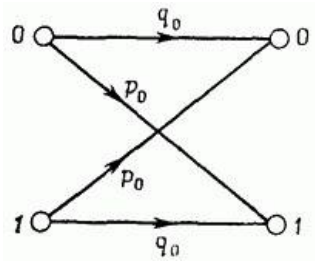


Рисунок 2.5 - Двійковий симетричний канал

Злий жартівник, який вводить помилки в передачу, дуже простодушний: у нього немає пам'яті і він "переписує" символи випадково і незалежно один від одного. Його дії руйнівні, але в ньому немає свідомої шкідливості і діяльність його стійка, принаймні, в статистичному сенсі[5].

Абстрагована схема передачі інформації, з якою ми будемо, таким чином, мати справу, зображень на рис 2.6 На вхід кодуючого пристрою надходить деяка довга двійкова послідовність $\bar{x}$, що складається з символів 0 і 1, яку ми будемо називати інформаційною послідовністю. Ця послідовність може бути абсолютно довільною. Вона повинна бути точно відтворена на виході декодуючого пристрою з імовірністю, близькою до одиниці. Кодуючи й і декодуючий пристрій пов'язані тільки двійковим симетричним каналом, для якого відома ймовірність переходу p_0.

У цій ситуації кодуючий пристрій явно обмежений тим, які операції він може виробляти. Природа ДСК така, що він пропускає тільки виконавчі послідовності.

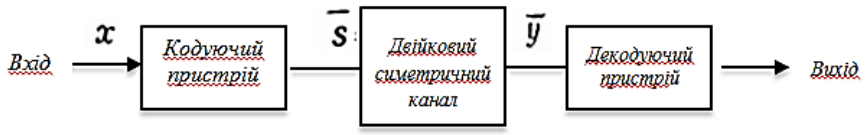


Рисунок 2.6 - Передача інформації за допомогою двійковий симетричного каналу.

Але кодуючий пристрій може перетворювати послідовність $\bar{x}$ на його вході в більш довгу послідовність $\bar{s}$ на його виході. У канал потрапляє, таким чином, послідовність $\bar{s}$ а на вхід декодуючого пристрою надходить спотворений її варіант $\bar{y}$. Завдання декодуючого пристрою при відомих перехідною ймовірності каналу p_0 отриманої на вході цього пристрою спотвореної послідовності y_0 і способі кодування, що задає перетворення -прийняти рішення щодо надійшовшої на вхід кодуючого пристрою інформаційної послідовності.

Для заданого ДСК завдання кодування полягає в тому, щоб визначити сукупність правил, за допомогою яких будь-яка інформаційна послідовність $\bar{x}$ кодується в деяку послідовність $\bar{s}$ так щоб декодер міг єдиним чином і з довільно малою вірогідністю помилки відновити $\bar{x}$, незважаючи на спотворення, що виникають в каналі. Є зацікавлення не тільки в тому, щоб вказати, як кодуючий пристрій $\bar{x}$ породжує $\bar{s}$ (проблема кодування), але так само і в тому, щоб вказати, як декодер отримує $\bar{x}$ з $\bar{y}$ (проблема декодування).

Існує, принаймні, одне просте і очевидне рішення цього завдання: кожен символ послідовності $\bar{x}$ повторити 2j + 1 раз. Наприклад, інформаційної послідовності

$$\bar{x} = 01101\ldots \tag{2.5}$$

при j=2 буде відповідати передана послідовність

$$\bar{s} = 000001111111111000001111 \ldots \quad (2.6)$$

Будемо декодувати $\bar{y}$ за правилом більшості. Якщо j+1 або більше символів в кожному блоці з 2j + 1 символів рівні 1, то декодер буде друкувати символ 1, в іншому випадку - символ 0. Якщо $p_0 < \frac{1}{2}$ то ясно, що при j → ∞ ймовірність помилки Pr (ε) → 0. Але, на жаль, і число символів, які можуть бути вручені одержувачу на виході декодирующего пристрою, буде при цьому прагнути до 0[12].

Класичний спосіб зменшення ймовірності помилки при передачі чисельної інформації в перекладі на мову ДСК полягає в тому, що, по-перше, слід зменшити перехідну ймовірність p_0, побудувати кращий канал. Якщо на якомусь етапі подальше поліпшення каналу виявляється неекономічним або технічно неможливим, то передача повторюється стільки раз, скільки виявиться потрібним для того, щоб результуюча ймовірність помилки стала нижче деякої задовольняючого проектувальника кордону. Труднощі, пов'язана з класичним підходом, полягає в тому, що коли ця межа ймовірності помилки прагне до нуля, то чи канал стає непропорційно дорогим, або дохід від його використання виявляється непропорційно низьким. Іншими словами, досконаліть зазвичай обходиться дорого[7].

В основній роботі Шеннона з теорії інформації доведені дві загальні теореми, які знаходяться в явному протиріччі з нашими очікуваннями.

1. Для заданого каналу можливо за допомогою відповідним чином підібраного кодування вести передачу з імовірністю помилки, меншою будь-якого наперед заданого значення, якщо швидкість передачі інформації не перевищує певної межі, відомого під назвою пропускної здатності каналу C.

2. Назад, для швидкостей передачі інформації, великих C, неможливо вести передачу із завгодно малою ймовірністю помилки.

У разі довічного симетричного каналу зручно відносити швидкість передачі інформації до одного переданому символу, а не до одиниці часу.

Коли всі можливі послідовності $\bar{x}$ на вході різновірогідні, швидкість передачі інформації R_t визначається відношенням

$$R_t = \frac{\text{Число символов в } \bar{x}}{\text{Число символов в } \bar{s}}. \tag{2.7}$$

Ми бачимо, що R_t вимірює щільність інформації на переданий символ.

Пропускна здатність повністю визначається самим каналом, і її також можна віднести до одного переданому символу. Можна показати, що для двійкового симетричного каналу з перехідною ймовірністю p_0

$$C = 1 - H(p_0), \tag{2.8}$$

де

$$H(p_0) = -p_0 \log p_0 - q_0 \log q_0. \tag{2.9}$$

Нам буде зручно всюди (окрім тих місць, де це спеціально обмовляється) використовувати логарифми за основою 2. Функція H (p) відома під назвою ентропії.

З теорем Шеннона видно, що не потрібно (як могло б здатися з розгляду наведеного вище прикладу з правилом більшості) у безперервний спосіб зменшувати R_t або p_0 для того, щоб зменшити ймовірність помилки. Якщо виконана умова

$$R_t < C, \tag{2.10}$$

то можна за допомогою відповідного способу кодування досягти бажаного результату, залишаючи R_t постійною. Якщо ж умова не виконується, то надійна передача інформації виявляється неможливою. Здається розумним тому вимірювати ефективність передачі інформації, або ступінь використовуваного каналу, ставленням $\frac{R_t}{C}$

Достовірність передачі інформації в системах цифрового зв'язку характеризується статистичною величиною - ймовірністю помилки на біт (BER). BER - це ймовірність помилкового прийому при передачі одного біта інформації, усереднена для статистично великого обсягу. Одним з методів розрахунку є теоретичний, графіки будуються за аналітичними формулами.

У MATLAB є інструмент BERTool (BitErrorRateTool), що дозволяє автоматизувати розрахунки залежності ймовірності помилок від відношення сигналу на шум для обраного модему і каналу зв’язку.

Виконується порівняльний аналіз модемів PAM, PSK, FSK, QAM. Діапазон зміни ставлення сигналу на шум 0 ... 18, число кодових комбінацій M = 16. Командою BERTool викликаємо інструмент. Для кожного типу модему виконуємо команду Plot, яка малює графік залежності ймовірності помилок від відношення сигналу на шум. На рис 2.7 для кожної залежності відображається легенда з ім'ям графіка.

Змінюється імена графіків на імена модемів. Для цього використовуємо список графіків, що відображається у верхній частині вікна інструменту. Подвійне клацання по імені за замовчуванням виділяє це ім'я, яке можна редагувати.

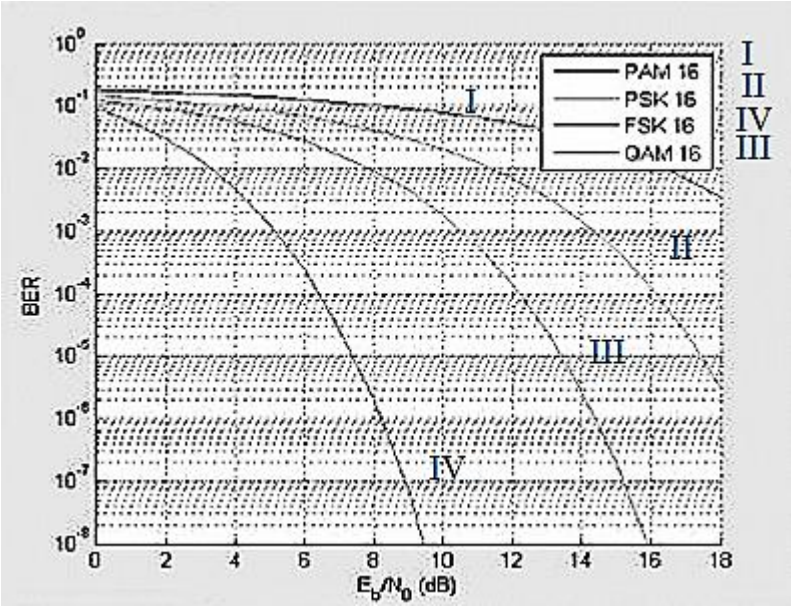


Рисунок 2.7 - Графік залежності ймовірності помилок від відношення сигналу на шум

Можна зробити висновки, що при PAM - помилки найбільші. Однак пропускна здатність каналу краща через самого вузького спектра частот сигналу, що передається. А у FSK - найнижчі помилки. Однак пропускна здатність каналу найгірша з-за самого широкого спектру частот сигналу, що передається.

2.5 Функції Matlab при кодуванні зображення фрактальним методом

В середовищі Matlab було використано в процесі кодування зображення такі функції :

1. fname = uigetfile (filter)визначає розширення файлу, за допомогою якого фільтруються файли, що відображаються у діалоговому вікні. Використовуйте цей синтаксис з будь-якими комбінаціями вихідних аргументів у попередніх синтаксисах. Як правило, відображаються лише файли з відповідним розширенням файлу. На деяких платформах uigetfile відображає файли, які не збігаються з фільтром, але тьмяніть ці імена файлів. Якщо фільтр відсутній або порожній, uigetfile використовує список типів файлів за замовчуванням.

2. I = imread (fname) зчитує зображення з файлу, зазначеного за іменем файлу, виводячи формат файлу з його вмісту. Якщо ім'я файлу є файлом з декількома зображеннями, то імідж читає перше зображення у файлі.

3. I = imresize (A, scale) повертає зображення I, що є масштабом, що перевищує розмір A. Вхідне зображення A може бути зображенням сірого, RGB або двійкового зображення. Якщо A має більше двох вимірів, то imresize змінює лише перші два виміри. Якщо шкала знаходиться в діапазоні [0, 1], B менше A. Якщо масштаб більше 1, B більше, ніж A. За замовчуванням, imresize використовує бікубічну інтерполяцію.

4. figure,imshow(I) створює нове вікно з використанням значень за замовчуванням. imshow (I) відображає зображення у градаціях сірого I на малюнку. imshow використовує діапазон відображення за типом зображення за

замовчуванням і оптимізує властивості малюнка, осей і зображень для відображення зображень.

5. tic запускає таймер секундоміра для вимірювання продуктивності. Функція записує внутрішній час.

6. s = qtdecomp(I,threshold,[mindimmaxdim]) виконує розкладку на квадратичних зображеннях на зображенні у градаціях сірого і повертає структуру квадратиків у розрідженій матриці s. За замовчуванням qtdecomp розбиває блок, якщо всі елементи в блоці не рівні. Не буде створювати блоки менші, ніж mindim або більше, ніж maxdim. Блоки більші, ніж maxdim, розбиваються, навіть якщо вони відповідають умовам порогу.

7. L = length (X) повертає довжину найбільшого виміру масиву в X. Для векторів довжина просто число елементів. Довжина порожнього масиву дорівнює нулю.

8. C = unique(A) повертає ті ж дані, що й у A, але без повторів. C знаходиться в відсортованому порядку. Якщо A є таблицею або розкладом, то unique повертає унікальні рядки в A у відсортоване замовлення. Для розкладу часу унікальне враховує час рядків і значення рядків, враховуючи при цьому, чи є рядки унікальними, і сортує вихідний розклад C по рядках.

9. Y = round (X) округлює кожен елемент X до найближчого цілого числа. У разі зв'язку, де елемент має дробову частину рівно 0,5, кругла функція округляється від нуля до цілого з більшою величиною.

10. dict = huffmandict (symbols,p) генерує словник коду Хаффмана, що відповідає джерелу з відомою вірогідною моделлю. Symbols перераховує різні значення сигналу, які виробляє джерело. Він може мати форму числового вектора, числового масиву клітин або алфавітно-цифрової матриці елементів. Якщо це матриця, вона повинна бути або рядком, або стовпцем. P - це вектор ймовірності, де довжина p повинна дорівнювати довжині символів.

11. comp = huffmanenco (sig, dict) кодує сигнал sig, використовуючи коди Хаффмана, описані кодовим словником dict. Аргумент sig може мати форму числового вектора, числового масиву клітин або алфавітно-цифрового

матриці елементів. Якщо sig - це масив клітин, він повинен бути або рядком, або стовпцем. dict - це масив клітин N-by-2, де N - кількість різних можливих символів, які необхідно закодувати. Перший стовпець dict являє собою окремі символи, а другий - відповідні кодові слова. Кожне кодове слово представлено у вигляді числового вектора рядків, а кодове слово в dict не може бути префіксом будь-якого іншого кодового слова в dict. Ви можете генерувати dict за допомогою функції huffmandict.

12. dsig = huffmandeco (comp, dict) декодує числовий вектор коду Хаффманасomp з використанням кодового словника dict. Аргумент dict - це масив N-by-2, де N - кількість різних можливих символів у вихідному сигналі, кодованих як comp. Перший стовпець dict являє собою окремі символи, а другий - відповідні кодові слова. Кожне кодове слово представлено у вигляді числового вектора рядків, а кодове слово в dict не може бути префіксом будь-якого іншого кодового слова в dict. Можно генерувати dict, використовуючи функцію huffmandict та comp, використовуючи функцію huffmanenco. Якщо всі значення сигналу в dict є числовими, dsig - вектор; якщо будь-яке значення сигналу в dict є алфавітним, dsig - одновимірний масив клітин.

13. toc зчитує минулий час з таймера секундоміра, запущеного функцією tic. Функція зчитує внутрішній час при виконанні команди toc і відображає минулий час з моменту останнього виклику функції tic, яка не мала виходу, в секундах.

14. ndata = bsc (data,probability) передає дані двійкового вхідного сигналу через двійковий симетричний канал, що має задану ймовірність помилки. Канал вводить бітну помилку і обробляє кожен елемент вхідних даних незалежно. Дані повинні бути масивом двійкових чисел, ймовірність має бути скалярною від 0 до 1.

15. dsig = huffmandeco (comp, dict) декодує числовий вектор коду Хаффманасomp з використанням кодового словника dict. Аргумент dict - це масив N-by-2, де N - кількість різних можливих символів у вихідному сигналі, кодованих як comp. Перший стовпець dict являє собою окремі символи, а

другий - відповідні кодові слова. Кожне кодове слово представлено у вигляді числового вектора рядків, а кодове слово в dict не може бути префіксом будь-якого іншого кодового слова в dict. Можемо генерувати dict, використовуючи функцію huffmandict та comp, використовуючи функцію huffmanenco. Якщо всі значення сигналу в dict є числовими, dsig - вектор; якщо будь-яке значення сигналу в dict є алфавітним, dsig - одновимірний масив клітин.

16. B = cast (A, newclass) перетворює A до класу newclass, де newclass - це ім'я вбудованого типу даних, сумісного з A. Функція cast скорочує будь-які значення в A, які занадто великі, щоб відобразити їх у новий клас.

2.6 Стиснення зображення фрактальним методом кодування

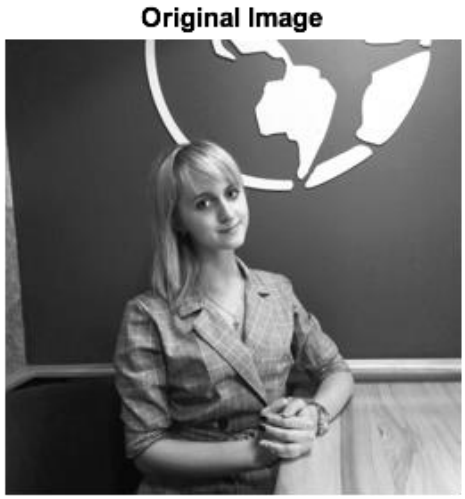
Для демонстрації стиснення зображень фрактальним методом наведені наступні п'ять прикладів, вхідними зображеннями для яких наведено на рисунку 2.8

А

Б

В

Г



Д

Рисунок 2.8 - Вхідні зображення А, Б, В, Г, Д

Алгоритм дій наступний:

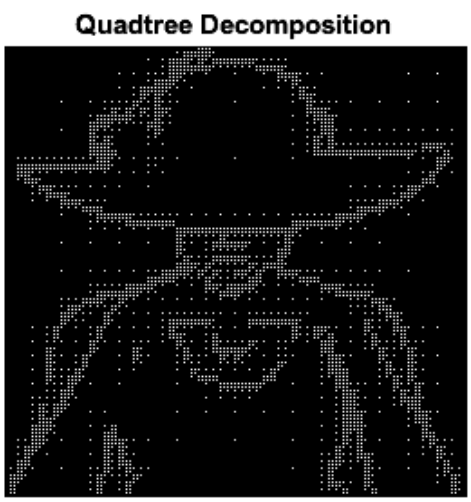
1) готуємо робочий простір для вибору зображення і змінюємо розмір на 256x256;

2) застосовуємо Quadtree розкладання, результати представлені на рисунку 2.9

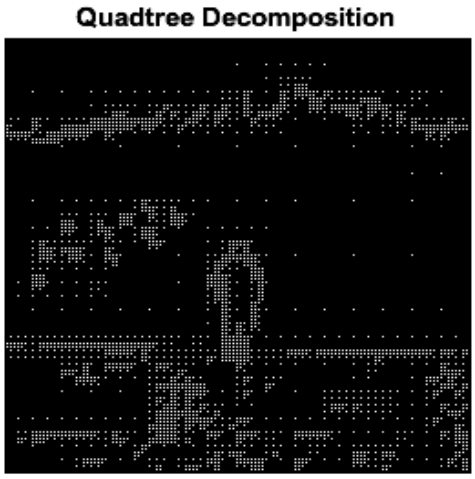
3) використовуємо кодування Хаффмана для побудова оптимального кодового дерева і відображення код-символ на основі побудованого дерева;

4) виконуємо стиснення зображення, розраховуємо необхідний час і коефіцієнт стиснення;

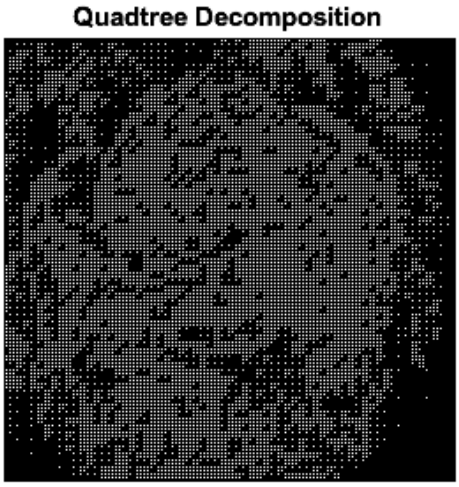
5) декодуємо зображення методом Хаффмана і розпаковане зображення виводимо на екран, результати представлені на рисунку 2.10



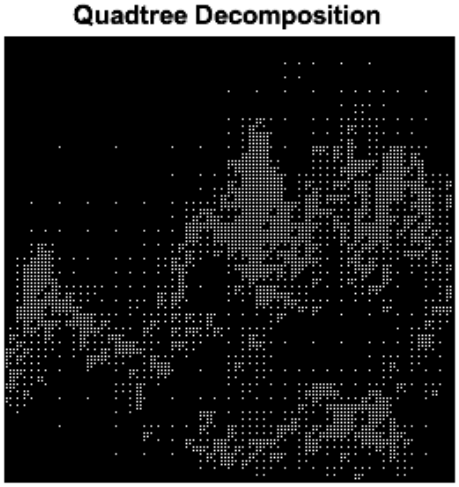
А



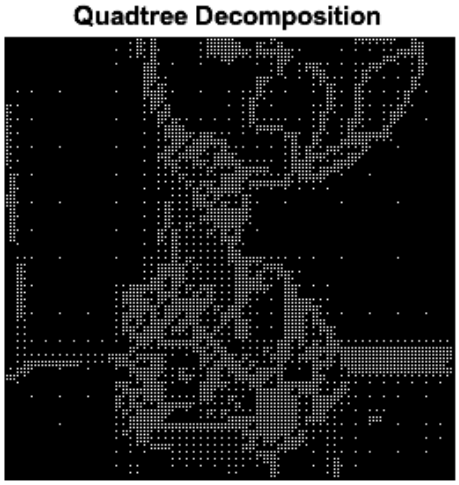
Б



В



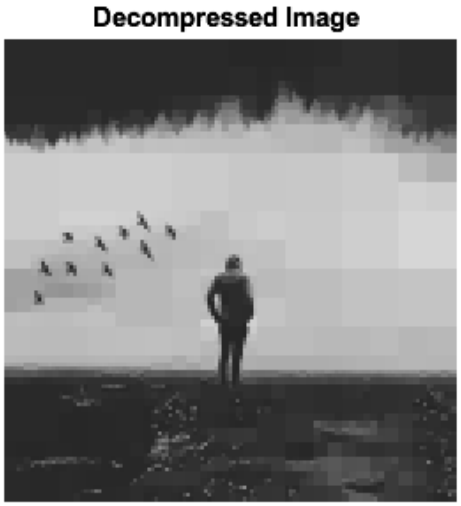
Г



Д

Рисунок 2.9 - QuadTree розкладання А, Б, В, Г, Д

А



Б

В

Г

Д

Рисунок 2.10 - Вихідне зображення А, Б, В, Г, Д

Як видно з порівняння вихідних зображень з відновленими стислими, за допомогою проведеної роботи вдалося відновити вихідні зображення ФМК.

2.7 Висновки

У другому розділі було розглянуто алгоритм ФМК, що застосовується для всіх рангових блоків зображення. В результаті для кожного рангу визначаються координати найбільш схожого на нього домену, разом з коефіцієнтами перетворення доменного блоку в рангові, що мінімізують різницю між ними. Алгоритм фрактального стиснення – несиметричний алгоритм. Це виражається тим, що пряме перетворення (стиснення) вимагає часу значно довше, ніж зворотне перетворення (відновлення). Було з’ясовано, що однією з найбільш зарекомендованних структур даних для представлення двовимірних зображень є квадродерево. Основна перевага QT складається в компактному поданні зображення. Компактність QT цілком залежить від зображення. Зображення з великими областями, пофарбованими в один колір представляються дуже компактно, в той час як зображення, в якому всі пікселі відображаються різними кольорами зводить нанівець всі переваги QT. Було здійснено стиснення зображення ФМК, використовуючи кодування Хаффмана для побудови оптимального кодового дерева і відображення код-символ на основі побудованого дерева. Також були представлені вхідні зображення, quadtree розкладання, і виконано стиснення та виведення на екран розпакованих зображень на екран.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

В даній роботі було досліджено фрактальний метод кодування зображення.

У економічному розділі наведено:

- розрахунок трудомісткості;
- розрахунок капітальних витрат.

3.1 Розрахунок трудомісткості

Трудомісткість - показник, що характеризує витрати робочого часу на виробництво певної споживної вартості або на виконання конкретної технологічної операції. Трудомісткість визначає ефективність використання одного з головних виробничих ресурсів — робочої сили .

$$t = t_{\text{пк}} + t_{\text{им}} + t_{\text{мо}} + t_{\text{тс}} + t_{\text{па}} + t_{\text{рк}} + t_{\text{вк}} + t_{\text{бск}} + t_{\text{ф}} + t_{\text{сз}} + t_{\text{оф}}, \text{ год} \quad (3.1)$$

де $t_{\text{пк}}$ – витрати часу на дослідження поняття фракталу на його класифікація;

$t_{\text{им}}$ – витрати часу на розглядання ідеї ФМК;

$t_{\text{мо}}$ – витрати часу на аналіз математичних основ фрактального стиснення;

$t_{\text{тс}}$ – витрати часу на розглядання типової схеми фрактального стиснення;

$t_{\text{па}}$ – витрати часу на складання алгоритму стиснення і декомпресії;

$t_{\text{рк}}$– витрати часу на дослідження розкладання по QT;

$t_{\text{вк}}$– витрати часу на розглядання візуалізації квадротомірованного зображення, його достоїнства та недоліки;

$t_{\text{бск}}$– витрати часу на розглядання двійкового симетричного каналу;

$t_{\text{ф}}$- витрати часу на опис функцій використовуваних в процесі кодування зображень;

$t_{\text{сз}}$- витрати часу на дослідження стиснення зображення ФМК;

$t_{\text{оф}}$- витрати часу на оформлення дипломної роботи.

У табл.3.1 наведено розрахунок трудомісткості по дослідженню фрактального методу кодування зображення.

Таблиця 3.1 - Розрахунок трудомісткості по дослідженню ФМК

№	Вид робіт	Час, години
1	Дослідження поняття фракталу на його класифікація	5
2	Розглядання ідеї методу фрактального стиснення	6
3	Аналіз математичних основ фрактального стиснення	8
4	Розглядання типової схеми фрактального стиснення	11
5	Складання алгоритму стиснення і декомпресії	5
6	Дослідження розкладання по QT	5
7	Розглядання візуалізації квадротомірованного зображення, його достоїнства та недоліки	24
8	Розглядання двійковий симетричного каналу	10
9	Опис функцій використовуваних в процесі кодування зображень	35
10	Дослідження стиснення зображення ФМК	50
11	Оформлення дипломної роботи	10
ВСЬОГО		169

Отже, трудомісткість складають згідно формули 3.1:

$$t = 5 + 6 + 8 + 11 + 5 + 5 + 24 + 10 + 35 + 50 + 10 = 169 \text{ год}$$

3.2 Розрахунок капітальних витрат на дослідження фрактальних методом кодування

Капітальні витрати - частина інвестицій, спрямована на відтворення основних засобів виробничого і не виробничого призначення, на створення нових, реконструкцію і розвиток наявних основних засобів, включаючи об'єкти соціальної сфери.

Капітальні витрати складаються з двох частин:

– витрати на придбання основних засобів;
– витрати на оплату праці інженера по телекомунікаціям.

У табл.3.2 наведено розрахунок витрат на дослідження ФМК.

Таблиця 3.2 - Розрахунок витрат на дослідження ФМК

№	Основні засоби	Вартість, грн
1	Ноутбук ASUS VivoBook 15	22599
2	Офіс	2500
3	Matlab ліцензований на 6 тижнів	4166
4	Microsoft Office для дому та бізнесу 2019 для 1 ПК	7508
5	Канцелярія	125
Сума		336898

Розрахунок заробітної плати фахівця з телекомунікаційної інженерії

$$ЗП = Т * t, \text{ грн} \quad (3.2)$$

де ЗП - заробітна плата, грн; Т – тарифна ставка за одиницю відпрацьованого часу, грн/год; t – кількість відпрацьованого часу, год.

Т= Оклад / Час роботи = 8000/ (21*8) = 48 грн /год, (3.3)

то при підстановці (3.3) у (3.2) отримаємо:

ЗП = 48 * 169 = 8112 грн

Обчислюється обов'язковий страховий внесок в Україні, збір якого здійснюється в системі загальнообов'язкового державного страхування в обов'язковому порядку та на регулярній основі

ЄСВ = ЗП * 0,36 = 8112 * 0,36 = 2921 грн (3.4)

де ЄСВ -обов'язковий страховий внесок в Україні.

Тоді загальна зарплата та капітальні витрати фа

заг.ЗП = ЗП + ЄСВ = 8112 + 2921 = 11033 грн (3.5)

заг.Витрати = 11033 + 336898 = 347931 грн, (3.6)

Отже, загальна зарплата складає 11033 грн, а капітальні витрати 347931 грн.

3.3 Висновки

Виконана робота дозволила розрахувати трудомісткість по дослідженню фрактального методу кодування, що складає 169 год, також розрахувати капітальні витрати – 347931 грн, до яких входять розрахунок витрат на придбання основних засобів та на оплату праці інженера по телекомунікаціям.

ВИСНОВКИ

В основу даної роботи було покладено фрактальний алгоритм кодування зображення. В результаті було проаналізовано ідея методу фрактального стиснення і його математичні основи. Так само було досліджено фрактальний метод кодування, його побудова та декомпресія. Розглянуто розкладання одного з фрактальних методів QuadTree і продемонстрована можливість застосування даного методу для стиснення зображень.

В ході роботи була написана програма, яка дозволила виявити основний недолік фрактального алгоритму компресії - чимало часу обробки, необхідне для стиснення зображення. Відповідно, це перешкоджає використанню такого алгоритму в додатках, де необхідна швидка обробка зображення в реальному проміжку часу.

На жаль, запропонований метод не повністю вирішують проблему підвищення швидкодії фрактального стиснення зображень. Тому перед дослідниками відкриваються можливості підвищення ефективності і пошуку інших способів оптимізації фрактального кодування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Barnsley M. FractalsEverywhere / M. Barnsley. – London: AcademicPressInc., 1988. – 370 p.
2. Jacquin E. Image Coding Basedon a Fractal Theory of Iterated Contractive ImageTransformation / E. Jacquin // IEEE Transactions on Image Processing. – January 1992. – P. 45-52.
3. Zhang Aihua, 2014. Fractal image coding combined with discretecosine transform complement. Computer Technology and Development, Vol. 24, No. 1, pp. 61– 68.
4. Wang Qiang, 2010. Fast fractal imageen coding based on structural information feature. Computer Engineering, Vol. 36, No. 24, pp. 195–199.
5. Yang Fengxia, 2012. Study on the effective method to improving the oding visual effect of fractal image. Laser Infrared, Vol. 42, No. 9, pp. 1068–1070.
6. Уэлстид С. Фракталы и вейвлеты для сжатия изображений в действии / С. Уэлстид. – М.: Триумф, 2003. – 295 с
7. Мюррей, Д. Энциклопедия форматов графических файлов [Текст] / Д. Мюррей, У. ван Райпер; пер. с англ. – К.: Издатель- скаягруппа BHV, 1997. – 672 с.
8. Бенуа, Б. Мандельброт. Фрактальная геометрия природы [Текст] / Б. Бенуа. – М.: Институт компьютерных исследований, 2002. – 666 с.
9. Ватолин, Д. Методы сжатия данных [Текст] / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М.: Диалог-МИФИ, 2002. – 381 с.
10. Барнсли, М. Фрактальное сжатие изображений [Текст] / М. Барнсли, Л. Ансон // Мир ПК. – 1992. – No 10. – С. 52–58.
11. Красильников, Н. Н. Цифроваяобработка 2D- и 3D-изображений [Текст]: уч. пос. / Н. Н. Красильников. – СПб.: БХВ-Петербург, 2011. – 608 с.

12. Ватолин, Д. С. Использование ДКП для ускорения фрактального сжатия изображений [Текст] / Д. С. Ватолин // Программирование. – 1999. – No 3. – С. 51–57.
13. Карпов, П. М. Быстрый фрактальный алгоритм сжатия изображений [Текст] / П. М. Карпов. – Научная сессия МИФИ, 2006. – Т. 15.
14. Шабаршин, А. А. Метод фрактальногосжатияизображений [Текст] / А. А. Шабаршин // Научныешколы УПИ-УГТУ. –
15. Винокуров, С. В. Методы повышения временной эффективности алгоритмов фрактального сжатия изображений [Текст]: матер. конф. // Фундаментальные и прикладные проблемы приборостроения, информатики и экономики, сборник Информатика, 2005. – С. 53–59.
16. Винокуров, С. В. Эффективный алгоритм фрактального сжатия изображений с использованием пространственно-чувстви- тельного хеширования [Текст] / С. В. Винокуров // Открытоеобразование. – 2006. – Т. 4, No 57. – С. 62–70.
17. Осокин, А. Н. Исследование возможности распараллеливания процесса фрактального сжатия изображений [Текст] : сб. трудов VIII Всеросс. научно-практ. конф. / А. Н. Осокин, М. П. Шарабайко // Молодежь и современныеинформацион- ныетехнологии. – Томск, 2010. – Т. 1, Ч. 2. – С. 212–213.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітки
Документація				
1	А4	Реферат	3	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	1	
5	А4	Стан питання. Постановка задачі	11	
6	А4	Спеціальна частина	22	
7	А4	Економічний розділ	4	
8	А4	Висновки	1	
9	А4	Перелік посилань	2	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
12	А4	Додаток Г	1	
13	А4	Додаток Д	3	

ДОДАТОК Б. Перелік документів на оптичному носії

1. Пояснювальна записка
2. Презентація

ДОДАТОК В. Відгук керівника економічного розділу

Керівник розділу _____ Романюк Н.М.
(підпис) (прізвище, ініціали)

ДОДАТОК Г. Відгук керівника кваліфікаційної роботи

В І Д Г У К

на кваліфікаційну роботу студентки групи ТКіт-15-1Дмитрієвої В.І.

на тему: «Дослідження фрактального алгоритму кодування»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, розташованих на 56 сторінках.

Мета кваліфікаційної роботи є актуальною, оскільки вона направлена на розробку фрактального алгоритму кодування зображення. Автор зумів відобразити вищезгадану специфіку, аргументовано обґрунтував актуальність теми свого дослідження.

Характеризуючи роботу необхідно відзначити, що вибрана автором логіка дослідження, послідовність і зміст розділів дають змогу якісно розкрити тему.

В економічному розділі розраховані трудомісткість по дослідженню фрактального методу кодування, заробітна плата інженера телекомунікацій, капітальні витрати.

Рівень запозичень у кваліфікаційній роботі відповідає вимогам "Положення про систему виявлення та запобігання плагіату".

Як зауваження необхідно відзначити деякі стилістичні неточності та недостатню проробку окремих питань.

В цілому кваліфікаційна робота бакалавра заслуговує оцінки «_____», а її автор присвоєння кваліфікації «Бакалавр з телекомунікації та радіотехніки».

Керівник спеціальної частини,
асистент кафедри БІТ Ю.П. Рибальченко

Керівник роботи,
д.т.н., проф. кафедри БІТ В.І. Корнієнко

ДОДАТОК Д.

Лістинг програми фрактального методу кодування зображення

Дана програма розроблена в програмному середовищі MATLAB для стиснення зображень фрактальним методом з використанням розкладання Quadtree і кодуванням Хаффмана.

```
clc;clear;closeall;

fname=uigetfile('*.jpg');
I=imread(fname);
I=imresize(I,[256 256]);
figure,imshow(I);title('OriginalImage');
drawnow;
tic;

s=qtdecomp(I,0.2,[2 64]);
[i,j,blksz] = find(s);
blkcount=length(i);
avg=zeros(blkcount,1);
for k=1:blkcount
avg(k)=mean2(I(i(k):i(k)+blksz(k)-1,j(k):j(k)+blksz(k)-1));
end
avg=uint8(avg);
figure,imshow((full(s)));title('QuadtreeDecomposition');
drawnow;

i(end+1)=0;j(end+1)=0;blksz(end+1)=0;
data=[i;j;blksz;avg];
data=single(data);
symbols= unique(data);
```

```
counts = hist(data(:), symbols);
p = counts./ sum(counts);
sp=round(p*1000);
dict = huffmandict(symbols,p');
comp = huffmanenco(data,dict);

t=toc;
fprintf('Timetakenforcompression = %f seconds\n',t);
bits_in_original=8*256*256;
bits_in_final=length(comp)+8*length(symbols)+8*length(sp);
CR= bits_in_original/bits_in_final;
fprintf('compressionratio= %f\n',CR);

ndata = bsc(comp, .00005);

tic;
datanew = huffmandeco(comp,dict);
zeroindx=find(data==0);
inew=datanew(1:zeroindx(1)-1);
jnew=datanew(zeroindx(1)+1:zeroindx(2)-1);
blksznew=datanew(zeroindx(2)+1:zeroindx(3)-1);
avgnew=datanew(zeroindx(3)+1:end);


avgnew=uint8(avgnew);
for k=1:blkcount
  outim(inew(k):inew(k)+blksznew(k)-1,jnew(k):jnew(k)+blksznew(k)-
1)=avgnew(k);
end
figure,imshow(outim);title('DecompressedImage');drawnow;
```

```
t=toc;
fprintf('TimetakenforDecompression = %f seconds\n',t);
 [M,N] = size(I);
m = double(0);
I = cast(I, 'double');
outim = cast(outim, 'double');
for i = 1:M
for j=1:N
        m = m + ((I(i,j)-outim(i,j))^2);
end
end
m = m/(M*N);
psnr = 10*log10(255*255/m);
fprintf('PSNR= %f\n', psnr);
```