Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ковальов Д.В.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ковальову Данилу Вячеславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи перетворення зображень на основі методів диференційного аналізу*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Ковальов Д.В.
(прізвище та ініціали)

АНОТАЦІЯ

Ковальов Д.В. Програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи перетворення зображень на основі методів диференційного аналізу.

Метою розробки є програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу.

Результат роботи – програмна реалізація системи перетворення зображень на основі методів диференційного аналізу.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерна інженерія, перетворення зображень, диференційний аналіз

ABSTRACT

Kovalov D.V. Image conversion system software based on differential analysis methods. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the image transformation system based on differential analysis methods.

The purpose of the development is the software of the image conversion system based on differential analysis methods.

The result of the work is the software implementation of the image transformation system based on differential analysis methods.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: computer engineering, image transformation, differential analysis

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЧХ	–	амплітудно-частотна характеристика
ВП	–	вейвлет-перетворення
ВЧ	–	високі частоти
ВФ	–	вейвлет-функції
ГА	–	генетичний алгоритм
ДПФ	–	дискретне перетворення Фур'є
НЧ	–	низькі частоти
УБК	–	метод усіченого блокового кодування
ФЧХ	–	фазочастотна характеристика

ВСТУП

Актуальність теми. У сучасному світі відбувається генерація контенту в значних обсягах, найбільшу частину займають різні медіа-дані. При безперервному рості обсягу даних потребуючих передачі, смуга пропущення стандартних радіоканалів зв'язку залишається незмінною, що у свою чергу породжує істотні проблеми. Одним зі шляхів рішення є компресія даних і передача за допомогою відповідних протоколів зв'язку. Компресія даних так само дозволяє знизити використовувані для зберігання ресурси пам'яті обчислювальних машин.

Найбільший інтерес для розробки способів компактного зберігання представляють сигнали, що відносяться до графічних, тобто котрі передають різні види зображень, наприклад: фотографії, відеофільми, сигнали цифрового телебачення та інші багатомірні сигнали, сполучені з передачею графічної інформації. У більшості випадків передача й зберігання графічних даних споживає істотні ресурси при значній кількості надлишкової інформації. Застосування тільки статистичних способів стиску як алгоритм Хаффмана, арифметичне кодування й подібних, у більшості випадків не дає істотного ефекту. Тому для ефективного стиску до цифрового сигналу застосовують кілька послідовних перетворень. Частина перетворень заснована на розгляді сигналу як поля, до якого застосовні: спектральний аналіз, аналіз геометричної структури, кореляційний аналіз, диференціальний аналіз. Таким чином, зображення розглядаються як польові структури, а цифрові зображення – дискретні польові структури, відповідно.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем перетворення зображень на основі методів диференційного аналізу.

– Дослідження системи перетворення зображень на основі методів диференційного аналізу.

– Програмна реалізація системи перетворення зображень на основі методів диференційного аналізу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі перетворення зображень на основі методів диференційного аналізу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Різна природа цифрових сигналів визначає їхні особливості. Сигнали прийнятий розділяти на групи по переважному зосередженню інформації в частотній, або в просторовій області. При такому поділі для виключення втрати значимої інформації необхідно враховувати специфіку конкретного сигналу або деякої групи сигналів.

Частотні характеристики сигналу можуть бути отримані за допомогою способів частотної декомпозиції (Фур'є, вейвлет і ін.). Просторові характеристики сигналу можуть бути отримані за допомогою геометричної інтерпретації сигналу, визначення його похідних, вейвлет аналізу, віконного перетворення Фур'є. Вейвлет аналіз, як і віконне перетворення Фур'є, цікаво тим, що дозволяють одержати не тільки частотну характеристику сигналу, але просторову локалізацію спектра.

Використовуючи особливості концентрації інформації, можна виключати деяку «надлишкову» частину із частотного або просторового опису сигналу, що приводить до припустимої втрати інформації, тобто припустимим перекручуванням.

Методи одержання частотних і просторових характеристик сигналу дозволяють синтезувати компактну форму інформаційної складової сигналу. Безліч різних способів перекладу інформації в компактну форму не мають універсальності для різних видів цифрових сигналів. Таким чином, конкретний метод одержання компактної форми, добре застосовний до одного сигналу або групи сигналів, може бути малоефективний для іншого сигналу або групи сигналів. Не випадково в сучасному світі цифрової обробки сигналів присутній така розмаїтість форматів даних.

У даній роботі буде розглянутий новий метод, що дозволяє здійснити стиск зображень (із втратами або без втрат) на базі їхнього диференціального аналізу як польових структур.

1.2 Область застосування

Областю застосування є системи перетворення зображень. Таким видом систем опрацювання графічних зображень є програми, які призначені, в основному, для перегляду графічних зображень. Розвитку даного виду програмного забезпечення сприяло активне використання останнім часом широким колом користувачів цифрових фотоапаратів і відеокамер. Основні можливості цих програм:

– перегляд графічних зображень різних форматів;

– зміна масштабу перегляду зображень;

– конвертація файлів – зберігання зображень у файлах різних форматів;

– демонстрація зображень у режимі слайд-шоу;

– елементарне редагування зображень – змінення розмірів та обертання зображення, збільшення/зменшення яскравості та контрастності кольорів, стиснення даних у файлі, усунення ефекту червоних очей тощо;

– відображення даних про значення властивостей самого зображення і файлу зображення;

– друкування графічного зображення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Gimp: редактор для майстрів

Платформи: Windows, Mac, Linux.

Перша версія цієї програми вийшла ще в 1996 році. Це повністю безкоштовний і відкритий графічний редактор, яким можна вільно користуватися навіть у комерційних цілях. Ще недавно критикуємий за незручний інтерфейс, Gimp зараз перетворився до невпізнанності. Зрозуміло, у кращу сторону. Він дозволяє в дуже широких рамках управляти робітничим середовищем: від повної зміни інтерфейсу й перепризначення гарячих клавіш до додавання наборів кистей, візерунків і градієнтів.

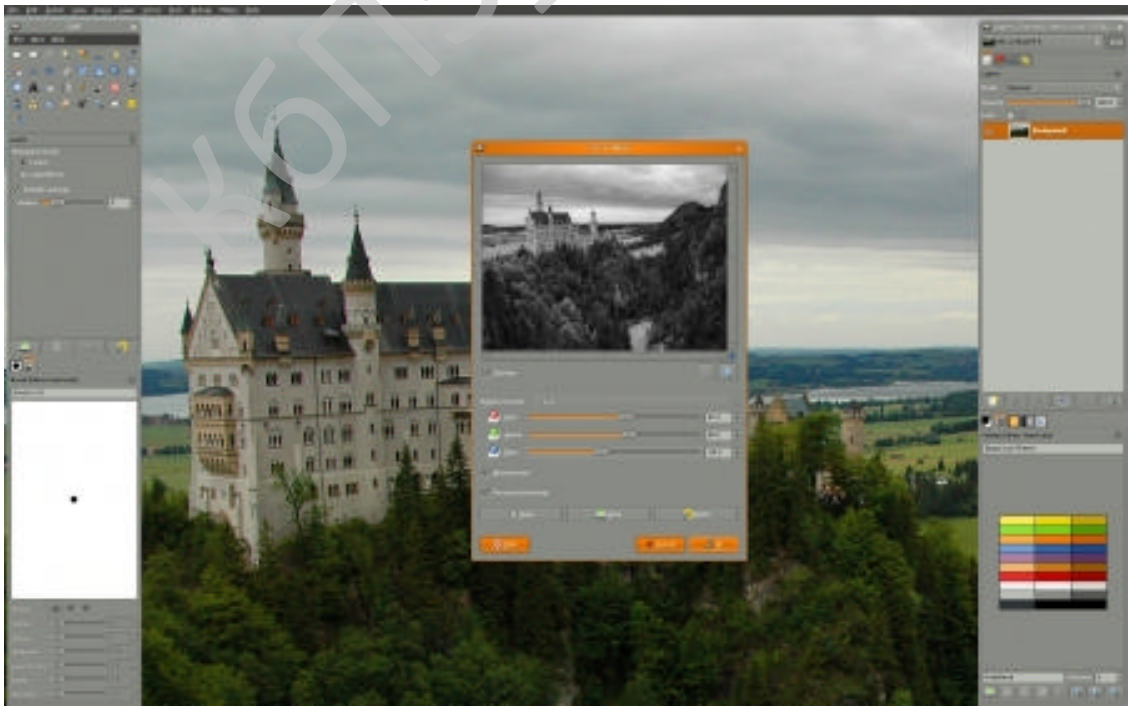

Рисунок 2.1 – Інтерфейс користувача Gimp

Функціональність програми легко розширюється. Сторонні розроблювачі створили для Gimp масу корисних інструментів і доповнень і продовжують створювати нові. Втім, навіть у стандартній версії Gimp є все необхідне для професійної роботи з рисунками, фотографіями й навіть дизайнерськими макетами.

Програма підтримує роботу із шарами й масками, має великий набір інструментів для кольорокоррекції, ретуші й малювання, дозволяє використовувати масштабовані кисті й сумісна із графічними планшетами. Gimp уміє працювати з усіма основними типами графічних файлів і містить більшу базу плагінів для відкриття рідких і маловикористовуємих форматів.

Pixlr Editor: ретуш у режимі онлайн

Платформа: браузер.

Pixlr, розробкою якого займається компанія Autodesk, являє собою набір графічних додатків для настільних комп'ютерів, мобільних пристроїв і браузерів. Сама широка функціональність – в онлайн-версії редактора, що до того ж безкоштовна для використання. Вона цілком підійде навіть професіоналам, у той час як окремі додатки більше призначені для звичайних користувачів.

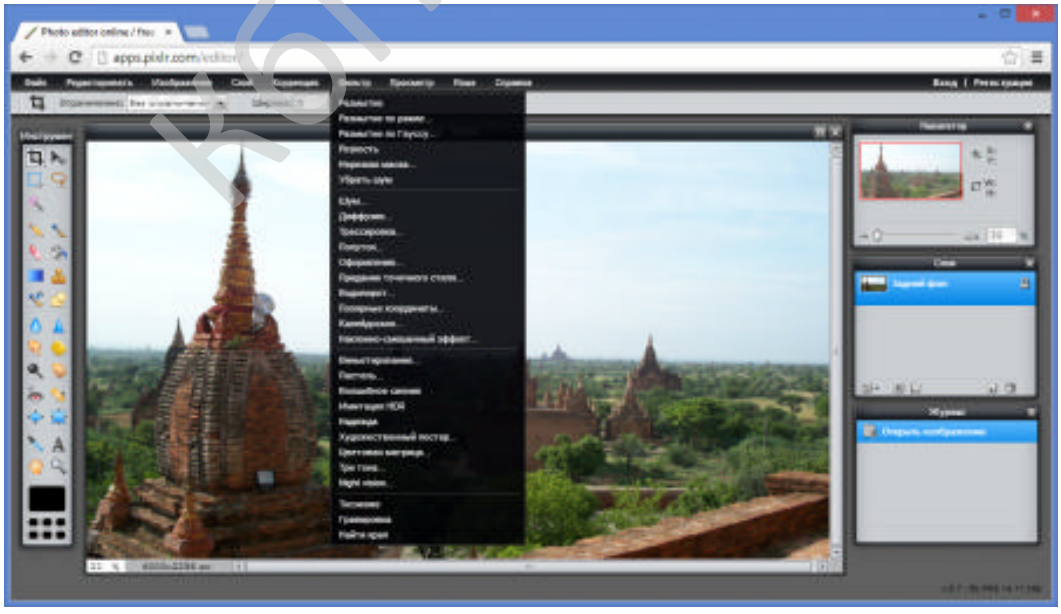

Рисунок 2.2 – Інтерфейс користувача Pixlr

Можливості редактора досить широкі: тут є всі необхідні функції для кольорокоррекції, малювання й обробки фотографій. Pixlr Editor підтримує роботу із шарами й має великий набір різних ефектів і фільтрів. Єдиний недолік онлайн-системи полягає в тому, що її не можна розширити за допомогою сторонніх доповнень.

Після редагування зображення його можна зберегти на локальному комп'ютері або у власної онлайн-бібліотеці Pixlr. Крім того, готове зображення можна відразу завантажити в Facebook, Flickr, Picasa або Google Drive.

Paint.Net

Платформа: Windows.

Paint.Net був створений студентами Університету штату Вашингтон в 2004 році як заміна стандартному редактору Paint в операційній системі Windows. Згодом зі студентського проекту виріс самостійний і досить функціональний графічний редактор. Його користувачам доступні як базові інструменти роботи із зображенням, включаючи керування кольором, контрастністю, різкістю й іншими параметрами, так і більше складні функції – робота із шарами, керування перспективою й «розумне» виділення частин зображення.

Рисунок 2.3 – Інтерфейс користувача Paint.Net

Редактор оптимізований для роботи з найсучаснішими багатоядерними процесорами й використовує апаратне прискорення відеокарти для обробки графіки, що дозволяє запускати його навіть на слабких комп'ютерах. Великий плюс Paint.Net – це необмежена історія змін, що дозволяє скасувати будь-яку послідовність дій над зображенням.

XnView

Платформа: Windows.

На відміну від більшості графічних редакторів, у програмі XnView немає інструментів малювання. Ця програма призначена тільки для роботи з готовими зображеннями. Акцент у ній зроблений на каталогізацію й зручне зберігання великої кількості знімків. Таким чином, це насамперед зручний інструмент для фотографів.

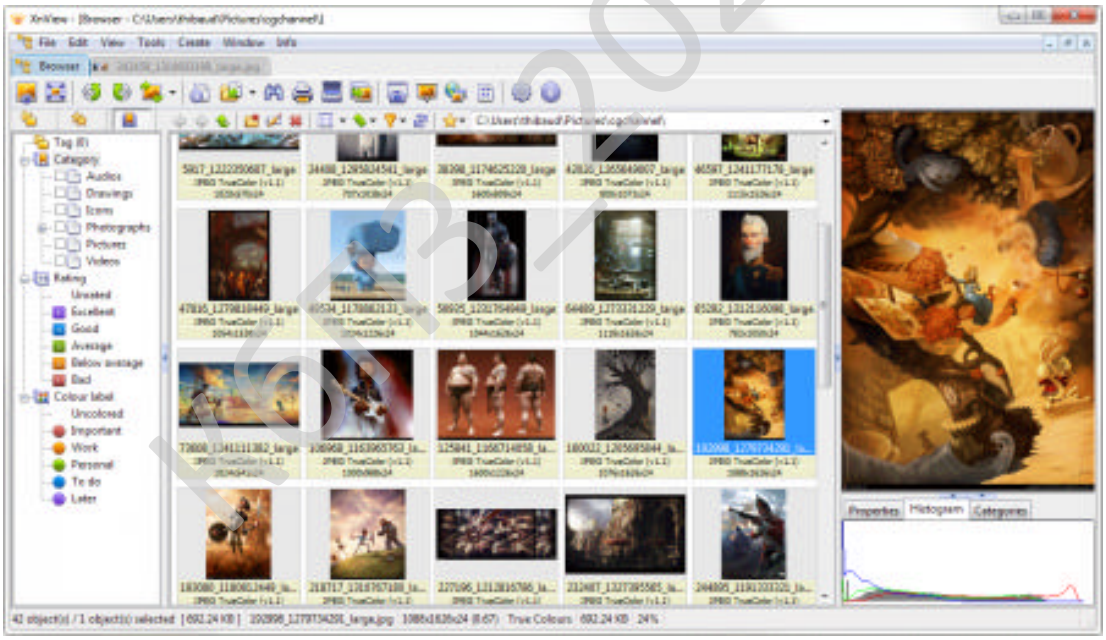

Рисунок 2.4 – Інтерфейс користувача XnView

В XnView є безліч функцій для обробки фотографій без застосування додаткових додатків. Програма дозволяє коректувати рівні, контрастність, гаму й колірний баланс, міняти колірні моделі й працювати з окремими каналами. Є й

непоганий набір різних ефектів і фільтрів, якому можна розширити за допомогою сторонніх плагінів. Передбачено пакетну обробку файлів: потрібні зміи можна вносити відразу у велику кількість знімків.

Для тих, у кого нагромадилося багато безсистемно знятих фотографій, XnView пропонує кілька інструментів для організації фотоархіву. Кожний знімок можна позначити колірною міткою, оцінити й віднести до певних категорій.

XnView уміє працювати з усіма основними типами графічних файлів, а за допомогою готових модулів, що підключаються, досягається сумісність програми з п'ятьома сотнями різних форматів. Редактор безкоштовний для приватного використання, але для комерційного застосування буде потрібно придбати ліцензії по кількості необхідних копій програми.

PicMonkey

Платформа: браузер.

У графічному онлайн-редакторі PicMonkey користувачі знайдуть повний набір інструментів для серйозної обробки зображень. Отут є як базові функції кадрування, повороту картинки, керування різкістю й роботи з кольором, так і просунуті інструменти: рівні, криві, клонування й багато чого іншого, що потрібно для одержання ефектних результатів.

Унікальна особливість редактора – величезний вибір фільтрів, які можна застосувати до зображення. Серед них є як художні ефекти начебто ефекту зернистості фотоплівки або імітації ломографії, так і жартівні. На знімки можна додавати смайлики, квіти, блискітки, ґудзики, феєрверки, бороди й сотні інших несерйозних речей.

Окремий розділ присвячений винятково портретної ретуші. У ньому є спеціалізовані інструменти для коректування губ, брів, очей, додавання засмаги й малювання густої щетини. Втім, веселощі добирається й сюди – людини на портреті можна швидко перетворити в похмурого зомбі або вампіра із блискаючими очами.

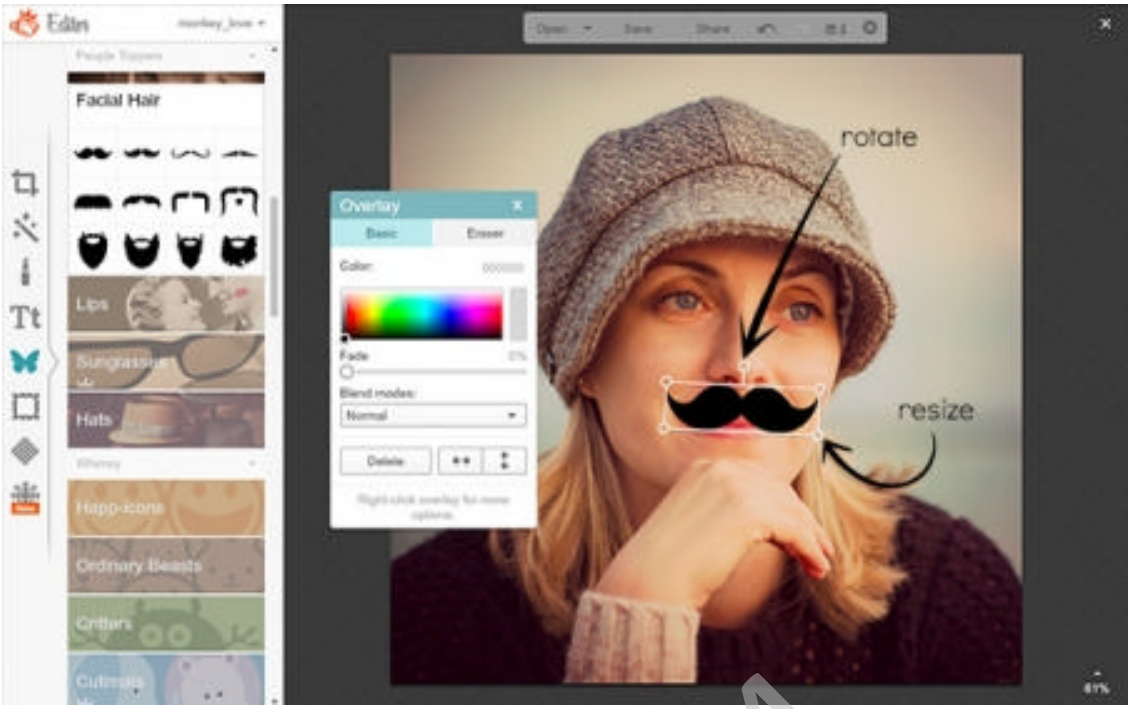

Рисунок 2.5 – Інтерфейс користувача PicMonkey

Більшістю інструментів можна користуватися без обмежень, але деякі просунуті функції або позначать зображення невеликим значком PicMonkey, або будуть закриті до придбання платної підписки.

Інтерфейс PicMonkey простий і наочний: кожна функція в меню супроводжується картинкою, що наочно демонструє ефект її застосування. Після обробки зображення можна зберегти на локальному комп'ютері або завантажити в хмарне сховище Dropbox.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об’єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних застосунків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна

розробляти звичайні клієнтські застосунки Windows, веб-служби XML, розподілені компоненти, застосунки типу “сервер-клієнт”, застосунки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку застосунків мовою Visual C# і .NET Framework.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об’єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – крапку входу застосунка – інкапсулюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки. При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

Крім служб часу виконання, в .NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керуванням рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному застосунку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи перетворення зображень на основі методів диференційного аналізу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розгляд цифрових сигналів як польових структур дозволяє використовувати чисельні методи розрахунку полів до аналізу й перетворення цифрових сигналів. Найцікавіші методи частотного й диференціального аналізу. Основи методів частотного аналізу цифрових сигналів, у тому числі з метою наступного стиску, добре розібрані у вже класичній літературі, що є, по цифровій обробці сигналів [1-5], тому сконцентруємо увагу на розгляді просторового аналізу сигналів. Існує ряд способів формування компактного виду цифрових сигналів шляхом аналізу просторової структури. Зокрема для одержання компактного виду зображень застосовують векторизацію, тобто в ході морфологічного аналізу зображення виявляють графічні примітиви (прямі, окружності, прямокутники й т.д.), сукупність опису яких дозволяє в деяких випадках одержати компактну форму. Даний спосіб не є універсальним, тобто не всі зображення можна, з метою стиску, ефективно розділити на графічні примітиви. Поділом на графічні примітиви найбільше ефективно можна стискати технічну графіку (креслення, ескізи й ін.) і деякі види зображень, об'єкти яких мають чіткі границі й відносно прості форми. Морфологічний аналіз зображень у більшості випадків є досить складним обчислювальним завданням.

У такий спосіб є фрактальний стиск, заснований на тому, що зображення можна представити більш компактно, за допомогою коефіцієнтів системи ітеруємих функцій (фракталів) [6]. З погляду обчислень, фрактальний стиск є дуже витратним способом, але на деяких зразках зображень показує дуже гарні результати. Відомий спосіб, у якому із зображення, застосовуючи метод «brush fire» (лісова пожежа), формують кістяк багатокутника, товщиною 1-2 пікселя, що є діаграмою Вороного [7]. Даний спосіб так само досить витратний з погляду

обчислень і не одержав широкого поширення. Існують і інші способи просторового аналізу цифрових сигналів з метою одержання компактної форми.

Спосіб дельта-кодування. У найпростішому випадку дельта кодування будується в такий спосіб: для елементів цифрового сигналу $f[x],\ x \in X$, обчислюється ряд дельта-коду $d[x] = f[x+1] - f[x],\ x \in (X-1)$, у силу того, що сусідні елементи вихідного сигналу $f[x]$ звичайно відрізняються друг від друга значно менше діапазону припустимих значень $f[x]$, отриманий ряд дельта-коду $d[x]$ може бути ефективно стислий [8]. Фактично значення ряду дельта-коду $d[x]$ є похідна $f[x]$. Для відновлення вихідного сигналу, крім самого ряду дельта-коду, необхідно знати початкова умова. Розглянемо приклад одержання й деякі особливості дельта-коду: діаграма (рисунок 3.1) і ряди чисельних значень (таблиця 3.1), відповідно. Очевидно, що значення дельта-коду мають меншу амплітуду й для їхнього кодування буде потрібно менше число розрядів. Значення дельта-коду будуть частіше повторюватися й, отже, до них може бути ефективно застосоване статистичне кодування.

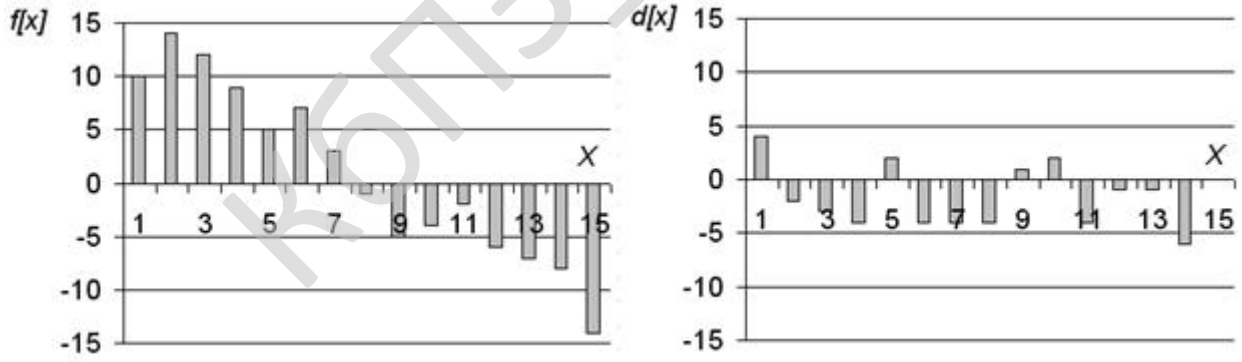

Рисунок 3.1 – Вихідний сигнал $f[x]$ і дельта-код $d[x]$

Розвиваючи ідею дельта-кодування, можна помітити, що більшість сигналів, отриманих у результаті реєстрації реальних фізичних процесів, наприклад, фотографування об'єктів реального миру, можуть бути описані з тим або іншим ступенем точності рівняннями. Диференціальні рівняння – найпоширеніший тип рівнянь, що описує фізичні процеси оточуючого нас миру.

Отже, диференціальні рівняння можуть бути застосовані для опису цифрових сигналів, що є образами реальних фізичних явищ.

Таблиця 3.1 – Значення функції $f[x]$ і її дельта-кода $d[x]$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f[x]$	10	14	12	9	5	7	3	-1	-5	-4	-2	-6	-7	-8	-14
$d[x]=f[x+1]-f[x]$	4	-2	-3	-4	2	-4	-4	-4	1	2	-4	-1	-1	-6	

Розглянемо основні типи рівнянь математичної фізики із частками похідними другого порядку для двох- і трьох мірних просторів (x,y – координати простору, t – координата часу) [9].

Хвильові рівняння:

$$\frac{\partial^2 f(x,t)}{\partial t^2} = a^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \tag{3.1}$$

$$\frac{\partial^2 f(x,y,t)}{\partial t^2} = a^2\left(\frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2}\right). \tag{3.2}$$

До рівнянь даного типу приводить розгляд процесів поперечних коливань струни, поздовжніх коливань стрижнів, електричних коливань у проводах, крутильних коливань валів, коливань газу й т.д. Відзначимо, що не всі процеси, кваліфікуємі сучасною фізикою як хвильові, описуються наведеними хвильовими рівняннями, прикладом може бути поширення хвиль у нелінійному середовищі, солітони.

Рівняння теплопровідності, або рівняння Фур'є:

$$\frac{\partial f(x,t)}{\partial t} = a^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \tag{3.3}$$

$$\frac{\partial f(x,y,t)}{\partial t} = a^2\left(\frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2}\right). \tag{3.4}$$

До цих рівнянь приводить розгляд процесів теплопровідності, фільтрації рідини й газів у пористому середовищі, деякі завдання теорії ймовірностей і т.д.

Рівняння Лапласа й Пуассона:

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = \varphi(x,y), \tag{3.5}$$

$$\frac{\partial^2 f(x,y,z)}{\partial x^2} + \frac{\partial^2 f(x,y,z)}{\partial y^2} + \frac{\partial^2 f(x,y,z)}{\partial z^2} = \varphi(x,y,z). \tag{3.6}$$

Рівняння цього типу дозволяє робити моделювання електромагнітних полів, стаціонарних теплових станів, завдань гідродинаміки, дифузії й т.д. Рівняння Пуассона (3.5) і (3.6) при рівності правої частини нулю ($\varphi(x,y)=0$ і $\varphi(x,y,z)=0$ для всіх значень x,y,z) звертаються в рівняння Лапласа.

Таким чином, більшість процесів (стаціонарних і динамічних), які є первинними джерелами сигналів, мають аналітичний опис у вигляді диференціальних рівнянь. Важливо помітити, що диференціал і похідна функції пов'язані з поруч Фур'є, відповідно до теореми про почленне диференціювання тригонометричного ряду Фур'є [10].

Можна записати вираження, що зв'язує похідну порядку p (де p кожне із чисел $p=1,2,\ldots,m$) і члени ряду Фур'є функції $f(x)$:

$$f^{(p)}(x) = \sum_{k=1}^{\infty} k^p \left[a_k \cos\left(kx - \frac{\pi p}{2}\right) + b_k \sin\left(kx - \frac{\pi p}{2}\right) \right], \tag{3.7}$$

де: a_n, b_n – коефіцієнти ряду Фур'є функції $f(x)$. З рівняння (3.7) треба, що похідні більше високого порядку більше залежні від високочастотних складового ряду Фур'є.

Ґрунтуючись на наведених фактах, перейдемо до розгляду способу стиску, в основу якого покладений аналіз диференціальної структури цифрового сигналу. Під диференціальною структурою мається на увазі похідна, звичайно до третього порядку, даного цифрового сигналу, а для багатомірних сигналів частки похідні, відповідно.

Аналіз диференціальної структури й формування паттерна крайових умов цифрового сигналу

Визначимо деякі поняття використовувані далі. Сукупність умов, що описують стан системи в просторі на границях і деяких внутрішніх областях, називають граничними умовами. Сукупність, що описує стан системи в деякі моменти часу (звичайно початкові $t=0$), називають початковими умовами. Сукупність граничних і початкових умов називається крайовими умовами.

Нехай в області $\mathbf{W}$ заданий сигнал $f(R^n)$, припустимо, що в деякій частині заданої області $\mathbf{U} \subset \mathbf{W}$ сигнал задовольняє диференціальному рівнянню виду:

$$Af(R^n) = u(R^n), \tag{3.8}$$

де: A – деякий диференціальний оператор, $u(R^n)$ – права частина. Крайові умови $p(R^n)$, задані в області $\mathbf{B} \subset \mathbf{W}$, таким чином, що: $\mathbf{U} \cup \mathbf{B} = \mathbf{W}$.Функцію $p(R^n)$, задану в області $\mathbf{B}$, будемо називати паттерном вихідного сигналу $f(R^n)$. Знаючи рівняння (3.8) і паттерн сигналу $p(R^n)$, можна відновити вихідний сигнал $f(R^n)$ в області $\mathbf{U}$. Таким чином, суть аналізу диференціальної структури сигналу $f(R^n)$, складається у виборі рівняння, що аналізує (або рівнянь) (3.8) і відшуканні паттерна $p(R^n)$.

Визначення крайових умов, шляхом аналізу диференціальної структури заданого сигналу, є зворотним завданням обчислювальної математики.

Розглянемо формування паттерна на прикладі одномірних сигналів $f[x]$. Покладемо, що $f(x)$ є безперервний прообраз дискретного образа деякого сигналу $f[x]$, при цьому $f(x)$ описується у всіх областях $\mathbf{U}$ рівнянням, що аналізує, виду:

$$\frac{d^2 f(x)}{dx^2} = 0 \tag{3.9}$$

Переходячи до дискретного образа $f[x_i]$ й думаючи, що він заданий на сітці $x_0, x_1, \ldots, x_{n-1} \in \mathbf{W}$, з рівномірним кроком $h = x_{i+1} - x_i$, де $i \in 0..n-1$, для всіх $x_i \in \mathbf{U}$, можна записати рівняння (3.9) у вигляді:

$$\frac{d^2 f(x)}{dx^2} \approx \frac{\frac{f[x_{i+1}] - f[x_i]}{h} - \frac{f[x_i] - f[x_{i-1}]}{h}}{h} = \frac{f[x_{i+1}] - 2f[x_i] + f[x_{i-1}]}{h^2} = 0 \quad . \quad (3.10)$$

Виражаючи з (3.10) $f[x_i]$, одержимо:

$$f[x_i] = \frac{f[x_{i+1}] + f[x_{i-1}]}{2} \quad . \quad (3.11)$$

Обчислювати паттерн $p[x_i]$ будемо за наступною схемою.

1. На початку покладемо, що у всій області $\mathbf{W}$ всі значення паттерна дорівнюють значенням сигналу тобто: $p[x_i] = f[x_i]$.

2. Далі виберемо деяке числове значення Δ-критерію, по якому будемо робити виключення елементів з паттерна.

3. Наступним кроком будемо обчислювати для всіх значень $f[x_i]$ в області $\mathbf{W}$ значення відповідно до вираження, отриманим з обліком (3.11):

$$s = \left| f[x_i] - \frac{f[x_{i+1}] + f[x_{i-1}]}{2} \right|, \quad (3.12)$$

якщо умова $s < \Delta$ істинна, то значення $p[x_i]$ (для відповідні x_i) виключається з паттерна, у противному випадку, елемент залишається в паттерні.

У результаті виконання зазначеної послідовності дій, ми одержимо паттерн сигналу $f[x_i]$.

На зображенні (рисунок 3.2) показаний приклад вихідного сигналу (data) і паттерна (patt).

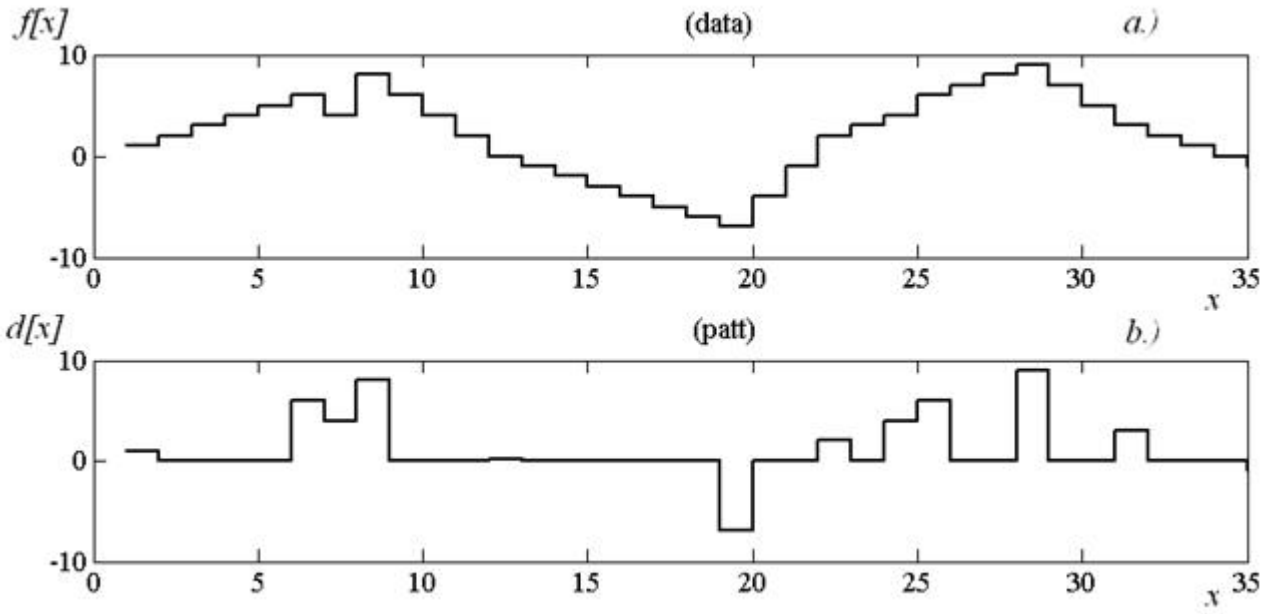

Рисунок 3.2 – Вихідний сигнал (data) і паттерн (patt)

Очевидно, що паттерн (рисунок 3.2) містить значно більше число рівнозначних елементів, чим вихідний сигнал, що дозволяє його ефективніше стиснути. Як ознака виключення елемента з паттерна відповідне значення можливо, наприклад, дорівнювати нулю. Такий принцип може викликати неоднозначні («спірні») ситуації, наприклад, якщо у вихідному сигналі присутні нульові значення, тому що для відновлення сигналу необхідно точно знати належить конкретний елемент паттерна крайовим умовам чи ні. Уникнути такої ситуації допоможе зміна «спірних» значень у вихідному сигналі на деяку певну величину залежно від формату й припустимого діапазону цифрового сигналу. Як ознака того, що елемент не є частиною крайових умов, можна використовувати величини вихідні за діапазон значень сигналу, але припустимі форматом даних.

Змінюючи значення величини Δ-критерію, можна управляти числом елементів увійшли в паттерн, від цього буде залежати кількість інформації, яка виключається, і ступінь стиску, відповідно. Для точного відновлення сигналу буде потрібно прийняти значення Δ-критерію менше кроку квантування сигналу й у деяких випадках зберігати, наприклад, у вигляді бітової маски, дані що дозволяють відновити нульові (або інші «спірні») значення в сигналі. На

зображенні (рисунок 3.3) показаний вихідний сигнал (рисунок 3.3, a.) і його поле паттернів (рисунок 3.3, b.). Поле паттернів отримане в результаті об'єднання серії паттернів (для різних значень Δ-критерію) в один масив. Чорний колір відповідає нульовим елементам паттерна, що не є крайовими умовами, інші (відтінки сірого) є крайовими умовами. При збільшенні значення Δ-критерію відбувається зменшення числа елементів паттерна, що належать безлічі крайових умов.

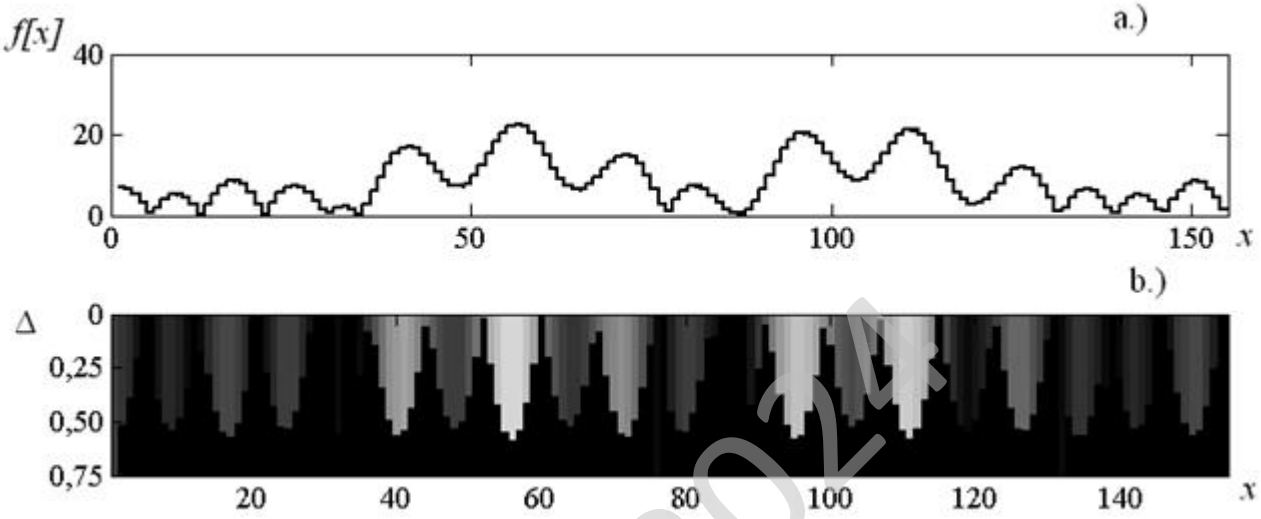

Рисунок 3.3 – Сигнал (a) і його поле паттернів (b) залежно від Δ-критерію

Паттерн, що містить відносно невелике число елементів, що є крайовими умовами, що дозволяють досить точно відновити вихідний сигнал, може бути отриманий тільки у випадку добре підібраного рівняння для формування паттерна, що в більшості випадків можливо лише для гладких функцій. Отже, застосування даного способу для стиску шумоподібних сигналів (з високою ентропією) неефективно.

Змінюючи порядок похідних диференціального оператора A (3.8) і вибираючи деякі функції правої частини $u(R^n)$, можна впливати на якість відновлення сигналу й число виключених елементів з паттерна.

Відновлення вихідного сигналу здійснюється за допомогою рішення обраного раніше в якості диференціального рівняння, що аналізує, і крайових умов, що втримуються в паттерні. Чисельне рішення можна здійснити, наприклад, за допомогою методу кінцевих різниць (МКР).

3.2 Розробка структурної схеми

Найбільш затребуваний стиск для даних називаних медіа-контент, до них відносяться статичні й динамічні (відео) зображення, аудіо дані й ін. Останнім часом активно розвиваються різні формати відео даних [5,11], на базі яких синтезуються об'ємні відео-сцени зображення високої чіткості й ін. Цифрові зображення це один з видів сигналів, до яких можна ефективно застосовувати стиск за рахунок аналізу диференціальної структури. Багато зображень містять досить значні поля градієнтних переходів (кольору, яскравості); контрастні границі об'єктів у просторі для статичних і в часі й просторі для динамічних зображень. Границі об'єктів, що втримуються в зображеннях, можна розглядати як крайові умови, а поля досить плавних переходів, як області, що задовольняють обраному диференціальному рівнянню, що аналізує.

Як приклад розглянемо послідовно всі етапи стиску й відновлення статичного зображення за допомогою аналізу диференціальної структури. Далі наведене короткий опис послідовності дій для стиску й відновлення зображення.

Стиск зображення

Стиск зображення відбувається в такий спосіб:

– вибір диференціального рівняння, що аналізує (або системи) і колірного простору для формування паттерна;

– якщо необхідно, перетворення колірного простору;

– формування паттерна (граничних умов);

– компресія паттерна й додаткової інформації для відновлення сигналу.

Відновлення зображення

Відновлення зображення відбувається в такий спосіб:

– декомпресія паттерна й додаткової інформації;

– відновлення зображення за допомогою рішення диференціального рівняння (або системи) з урахуванням граничних умов;

– якщо необхідно, перетворення колірного простору.

Докладно розглянемо кожний з перерахованих вище пунктів. Перше завдання це вибір рівняння, що аналізує (або системи) для формування паттерна й наступного відновлення двовимірного сигналу $f[x,y]$зображення , заданого на $X0Y$ площині . Зупинимося на рівнянні Лапласа, тому що воно добре описує плавні переходи (градієнт кольору і яскравості) в областях між граничними умовами:

$$\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2}=0 \quad . \tag{3.13}$$

Вибравши шаблон диференціювання для функції (рисунок 3.4), переходимо до дискретної форми $f[x,y]$. Уважаючи що крок сітки, з урахуванням індексації пікселів зображення, $\Delta x=\Delta y=1$, рівняння (3.13) можна записати у вигляді:

$$\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2}\approx$$
$$\approx \frac{f[x+\Delta x,y]-2f[x,y]+f[x-\Delta x,y]}{\Delta x^2}+\frac{f[x,y+\Delta y]-2f[x,y]-f[x,y-\Delta y]}{\Delta y^2}=$$
$$=f[x+\Delta x,y]+f[x-\Delta x,y]+f[x,y+\Delta y]+f[x,y-\Delta y]-4f[x,y]=0 \quad . \tag{3.14}$$

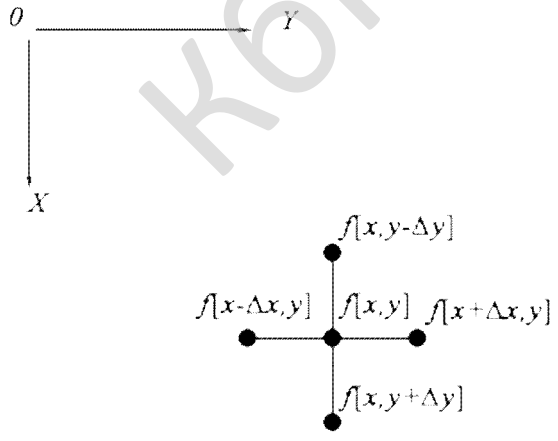

Рисунок 3.4 – Шаблон диференціювання функції $f[x,y]$

Вибір шаблона може вплинути на кінцевий результат, у цьому випадку (рисунок 3.4) обраний шаблон, що показав досить високу ефективність у рамках розглянутого методу.

Виражаючи з (3.14) $f[x,y]$, одержимо:

$$f[x,y]=\frac{1}{4}\left(f[x+\Delta x,y]+f[x-\Delta x,y]+f[x,y+\Delta y]+f[x,y-\Delta y]\right). \quad (3.15)$$

Наступне завдання – вибір колірного простору. Практичні дослідження показали, що найбільш компактний паттерн, отже, найбільш високий стиск, можна одержати в колірному просторі RGB і YCbCr (варіація Y_{709}CbCr), залежно від специфіки зображення. Колірний простір RGB асоційовано з особливостями сприйняття зорової системи людини й у ньому добре передається специфіка колірного градієнта, зберігаються кольори вихідного зображення. Простір YCbCr може більш ефективно забезпечити збереження границь об'єктів зображення при деякої втрати колірної інформації.

Формування паттерна є ключовим етапом стиску зображення. Від якості формування паттерна буде залежати можлива якість відновленого сигналу й ефективність стиску. Відзначимо деякі особливості, замічені при експериментах з різними підходами до формування паттерна. Найбільш висока якість відновленого сигналу вдається одержати при генерації паттерна, у якому кожному значущому елементу ставиться у відповідність усе компоненти колірного простору. При цьому конфігурація паттернів, з погляду положення елементів граничних умов, всіх колірних шарів однакова, а колір елементів граничних умов залишається в повній відповідності з вихідним зображенням. У результаті того, що паттерни всіх колірних компонентів (шарів) збігаються, можливо, стискати послідовність серій, що не значать елементів для одного шару й використовувати його як шаблон для відновлення інших, що дозволяє істотно підвищити фактора стиску.

У паттерн обов'язково включаються граничні піксели зображення (два стовпці – праворуч і ліворуч, і два рядки – зверху й знизу). Для виключення

елементів з паттерна застосовують Δ-критерій на базі порівняння абсолютної величини різниці значення обчисленого по (3.15) і фактичного в точці (x,y), з деяким заданим коефіцієнтом. Причому для кожного компонента колірного простору, можливо своє значення Δ-критерію й, отже, різна кількість виключеної інформації про диференціальну структуру сигналу.

Маскування (коректування) споконвічно нульових значень, що втримуються в зображенні й скопійованих у паттерн, необхідна для їхньої відмінності від виключених з паттерна й виключення «спірних» ситуацій. Відзначимо, що розрахунок виробляється по трохи видозміненому вираженню (3.15), до виду:

$$\left(f[x+\Delta x,y]+f[x-\Delta x,y]+f[x,y+\Delta y]+f[x,y-\Delta y]\right)-4\cdot f[x,y], \quad (3.16)$$

причому, множення на чотири в програмі виробляється за допомогою порозрядного зрушення вліво (<<2), такий підхід дозволяє виконувати всі обчислення в цілочисельній арифметиці зі збереженням точності розрахунків. Виконання цілочисельних операцій процесор робить істотно швидше, ніж операції над числами із плаваючою точкою. Від величини Δ-критеріїв залежить число виключених з паттерна елементів, а в остаточному підсумку, і ступінь стиску зображення. Чим більше величина Δ-критерію для колірного каналу, тим більше значень буде виключено.

У тому випадку, якщо отриманий, у результаті розглянутого вище перетворення пат терн, містить значне число повторюваних елементів і граничних умов досить для відновлення вихідного сигналу із заданою точністю, можна говорити про ефективний (з метою стиску) виключенні надлишкової інформації.

Для стиску паттерна пропонуються наступні перетворення. Перше перетворення орієнтовано на стиск значного числа повторюваних елементів (кодування довжин серій), у першу чергу елементів виключених з паттерна й відзначених у даній реалізації нульовим значенням. Як уже було сказано, найкраща якість відновлення зображення виходить при однаковій структурі (по

положенню виключених елементів) для всіх колірних компонентів, а число послідовних серій, виключених елементів кратно трьом (по числу елементів колірного простору), що можна врахувати при стиску серій. Отримані на виході, після кодування довжин серій, дані можуть бути стислі статистичним методом на основі перетворення Хаффмана або арифметичного кодування.

Відновлення зображення виробляється у зворотній послідовності, починаючи з декомпресії паттерна. Після одержання масиву паттерна його використовують як граничні умови для рішення МКР диференціального рівняння, що аналізує (3.13) (або іншого обраного при стиску), відповідно до різницевої схеми (3.15) і шаблоном (рисунок 3.4). Ітераційну схему рішення можна виразити рівнянням:

$$f^t[x,y] = \frac{1}{4}\left(f^{t-1}[x+\Delta x,y] + f^{t-1}[x-\Delta x,y] + f^{t-1}[x,y+\Delta y] + f^{t-1}[x,y-\Delta y]\right), \quad (3.17)$$

де t – крок ітерації. Якщо відносити початок ітерації з початком циклів обходу масиву відновлюваного зображення, а завершення ітерації із завершенням циклів обходу, то вибравши в циклах позитивне збільшення значень лічильників, і використовуючи для наступних розрахунків раніше обчислені значення ($f[x-\Delta x,y], f[x,y-\Delta y]$), рівняння (3.17) можна записати у вигляді:

$$f^t[x,y] = \frac{1}{4}\left(f^{t-1}[x+\Delta x,y] + f^{t}[x-\Delta x,y] + f^{t-1}[x,y+\Delta y] + f^{t}[x,y-\Delta y]\right). \quad 3.18)$$

При розпаралелюванні обчислювального процесу МКР по елементах сітки буде працювати ітераційна схема відповідно до вираження (3.17), при реалізації кожної ітерації циклічними переборами (по елементах сітки) буде працювати ітераційна схема (3.18).

Завершення ітераційного процесу можна робити за критерієм нев'язання, що обчислюється як сума різниць всіх значень елементів (без граничних умов) для сусідніх ітерацій t і $t-1$:

$$E = \sum_{x=1}^{Xsize-2} \sum_{y=1}^{Ysize-2} (f^t[x,y] - f^{t-1}[x,y]), \quad (3.19)$$

де: $Xsize = 0...(Xsize-1)$ і $Ysize = 0...(Ysize-1)$ – відповідно розміри зображення.

Підвищення ефективності стиску з урахуванням специфіки цифрового сигналу

Як уже було відзначено, різноманіття цифрових сигналів дуже велико, як велика й розмаїтість методів стиску. При стиску із втратами різні вимоги, пропоновані до припустимої втрати інформації. Значним фактором успішного стиску цифрових сигналів є вибір найбільш підходящого способу. Стиск на базі аналізу диференціальної структури може бути найбільше ефективно застосоване до сигналів (зокрема до зображень), що містять значні поля градієнтних переходів амплітуди й досить контрастні (з погляду зміни амплітуди) границі таких полів. До сигналів подібного виду можна віднести зображення із чіткими границями об'єктів і значних областей градієнтного або однотонного наповнення. Прикладом зображень подібного виду можуть бути:

– висококонтрастні фотографії технічних або інших об'єктів, що містять контрастні контури й однотонні або градієнтні поля;

– деякі види висококонтрастних космічних фотографій, у тому числі фотографій поверхні планет;

– картографічні зображення;

– мальовані зображення з контрастними границями об'єктів;

– текстурні зображення;

– деякі види зображень, синтезовані за допомогою машинної графіки й ін.

Дослідження стиску зображень за допомогою аналізу диференціальної структури дозволили визначити додаткові перетворення, застосування яких сприяє ефективності стиску.

При виборі як аналізуючого рівняння – рівняння Пуассона (3.5), (3.6), що може сприяти формуванню більше компактного паттерна в порівнянні з рівнянням Лапласа, необхідно визначити його праву частину. Як права частина можна використовувати матрицю певних значень, сформовану по вихідному зображенню, розділеному на області, наприклад, 8×8 пікселів. Значення матриці можуть бути отримані вибором мінімальної амплітуди з поля (8×8), такий підхід

у більшості випадків, дозволить коректувати щільність розподілу значень амплітуд у паттерні таким чином, що до нього можна більш ефективно застосовувати статистичний стиск.

У випадку стиску фотозображень, можливо, застосовувати спеціальну обробку для видалення з вихідного зображення шуму виникаючого, наприклад, у результаті дефектів ПЗЗ-матриці (скорочення від прилад із зарядовим зв'язком).

Формування копії зображення в результаті операції згортка з функцією розсіювання точки (ФРТ) і наступне використання цієї копії для аналізу диференціальної структури, при цьому в паттерн зберігаються як граничні умови елементи вихідного зображення. Використання згортки при стиску в деяких випадках дозволяє підвищити візуальну якість відновленого сигналу. Низькочастотна фільтрація зображення дозволяє виключити з паттерна випадковий шум, наприклад, властивий цифровим фотографіям. Негативною стороною застосування низькочастотних фільтрів, є можлива втрата корисної інформації з області високих частот, наприклад, дрібних деталей зображення. У деяких випадках може бути корисним застосування високочастотних фільтрів границі, що підкреслює, об'єктів на зображенні. Практичні дослідження показали, що для реальних цілей підвищення ефективності стиску за допомогою згортки необхідно використовувати функції розсіювання точки розміром $[3\times3]$ або $[5\times5]$.

Використання, для аналізу диференціальної структури, похідних вищих порядків і різних шаблонів диференціювання так само дозволяє в деяких випадках підвищити ефективність стиску. Вибір виду диференціального рівняння значною мірою залежить від особливостей зображення. Експерименти показали, що зображення, отримані в результаті фотографування, відображення реальних і багатьох синтезованих об'єктів добре аналізуються за допомогою похідних другого порядку, для них добре працюють рівняння Лапласа й Пуассона.

На зображенні (рисунок 3.5) наведений приклад втрати корисної інформації про форму сигналу в області, де мале значення другої похідної. Найбільш уразливі з погляду втрати інформації області, де відбувається плавна

зміна функції сигналу, особливо критичним це може бути для областей, де міняється знак першої похідної. Протидіяти втрати такої інформації може включення в паттерн локальних мінімумів і максимумів сигналу.

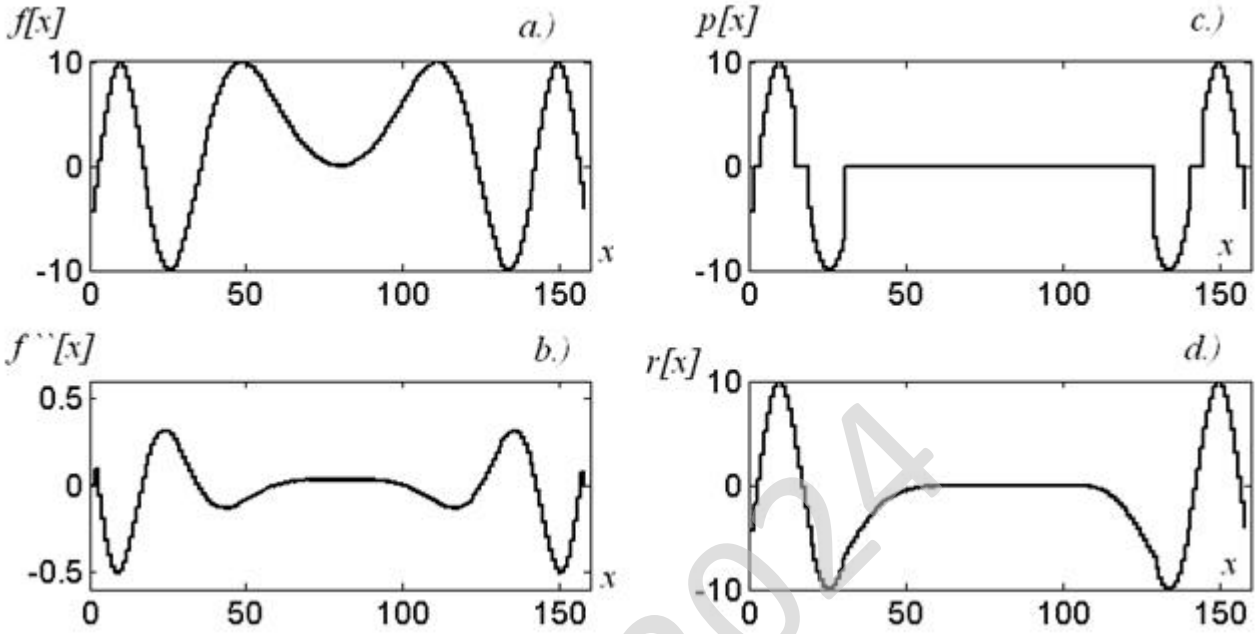

Рисунок 3.5 – Втрата інформації про сигнал при малих значеннях другої похідної

На зображенні позначені: a.) – вихідний сигнал $f[x]$; b.) – друга похідна $f''[x]$; c.) – паттерн $p[x]$; d.) – відновлений сигнал $r[x]$.

Для формування паттерна додана згортка, як приклад задана ФРТ називана «ковзного середнього» гнітючі високі частоти сигналу, у тому числі випадкові шуми й посилюючий сигнал до 9 разів. Розмір квадратної матриці ФРТ у даному прикладі зберігається в константі psfxy. При практичній реалізації для більшої функціональності ФРТ із розміром psfxy можна динамічно змінювати (наприклад, передавати як параметр), що дозволить задавати для різних типів сигналів різні ФРТ і формувати найкращий паттерн із погляду мінімізації розміру стислого паттерна при максимумі якості відновленого сигналу. Для цифрових фотографій утримуючі контрастні об'єкти й однотонні поля, із присутністю

випадкових шумів, досить гарні результати формування паттерна й наступного стиску показала наведена в програмі ФРТ «ковзного середнього». Зверніть увагу, що в паттерн копіюються не результати згортки зображення, а елементи вихідного зображення, що дозволяє залишити більше вихідної інформації й одержати більш якісний результат при відновленні. Згортка із заданої ФРТ робить розмиття полів і границь, що у свою чергу дозволяє врахувати в паттерні не тільки елементи самих границь, але й більше віддалені від границь елементи. На цифрових фотографіях і подібних їм цифрових сигналах, значення елементів зображення, що утворять границі об'єктів, можуть мати значні відмінності від значень елементів усередині або зовні границі, тому як граничні елементи необхідно вибирати елементи поблизу границь, близькі за значеннями до елементів обмеженої області. У прикладі в результаті згортки значення елементів результату згортки можуть істотно перевищувати (до дев'яти разів) значення елементів вихідного сигналу. Нормування динамічного діапазону або використання не цілочисельних коефіцієнтів ФРТ буде збільшувати час обчислення, більше ефективним, з погляду швидкості обчислень, є зміна значень Δ-критеріїв (delLo, delAv, delHi) до відповідної величини з урахуванням коефіцієнтів матриці ФРТ.

Обчислювальна оптимізація МКР на базі відшукання проміжного рішення

При відновленні стислих даних немаловажним фактором є швидкість. МКР досить повільний ітераційний спосіб рішення диференціальних рівнянь, отже, значний час може бути витрачене для рішення багатомірних завдань. Тому істотним питанням є оптимізація обчислень.

Відомий спосіб скорочення часу обчислень МКР для м'яких диференціальних рівнянь [12] у деякій області $\mathbf{U}$ із крайовими (граничними для фізичних областей і початковими для часу) умовами в областях $\mathbf{B}$, сутність якого складається в укрупненні кроку h сітки $\mathbf{U}$ в області $\mathbf{U} \cup \mathbf{B} = \mathbf{W}$. При цьому відбувається втрата точності рішення аж до того, що воно стає непридатним.

Наприклад, в основному завданні електростатики можуть бути розглянуті електричні заряди, описані єдиною точкою. При укрупненні сітки такі граничні умови можуть бути загублені, що принципово міняє сутність проміжного рішення. При формуванні паттерна можуть виникати крайові умови, наприклад, у вигляді відокремленої точки. Таким чином, для наших завдань втрата точності в результаті укрупнення сітки так само актуальна.

Із цієї причини звичайним є рішення, що містить декілька послідовно застосовуваних сіток. Спочатку застосовується сама велика сітка $\mathbf{U_1}$, що дозволяє одержати наближення рішення. Потім рішення уточнюють, послідовно застосовуючи сітки з меншим кроком $\mathbf{U_k}, k>1$. Загальне число різних сіток звичайно становить 2–3. Крок сітки h може бути різним у різних напрямках і областях сітки $\mathbf{U}$. Таким чином, прискорення процедури збіжності до рішення завдання із заданою точністю p відбувається за рахунок більше швидкого формування деякого проміжного рішення в області $\mathbf{U}$, що є прямим наслідком теореми про збіжність різницевого рішення [13]. При цьому кожне уточнення рішення є ітераційним, що має обчислювальну складність

$$O\left(k\prod_{i=1}^{n}N_i\right), \tag{3.20}$$

де: n – розмірність розглянутого завдання; N_i – число вузлів сітки в кожному напрямку; k – число ітерацій, що забезпечує задану точність на даному етапі. У багатомірних завданнях величина (3.20) може бути дуже велика навіть при розрідженій сітці.

Зазначене завдання знаходження проміжного (наближеного) рішення в області $\mathbf{U}$ можна вирішити шляхом апроксимації значень $f(R^n)$ рівняння (3.8). Апроксимацію $f(R^n)$ можна робити різними функціями, залежно від фізичної сутності завдання, наприклад, багаточленом виду:

$$a_0 + a_1 z + a_2 z^2 + ... + a_p z^p, \tag{3.21}$$

де $a_0, a_1, a_2, \ldots, a_p$ – коефіцієнти багаточлена; z – змінна, уздовж якого відбувається апроксимація.

У загальному випадку, якщо лінії сітки не паралельні координатним осям, що утворять простір $\mathbf{W}$, то z може не збігатися з набором змінних вихідної завдання. У цьому випадку необхідно враховувати поворот системи координат, у яких розглянутий аргумент z, щодо вихідної системи координат. У випадку n-мірної ($n > 1$) завдання, апроксимовано необхідно робити послідовно для всіх ліній, що утворять сітку в кожному напрямку, з наступною оцінкою середніх значень для кожного вузла сітки. Середнє значення вузла сітки обчислюється як середнє значення апроксимації в точках ліній сітки, що належать даному вузлу. У певному змісті можна говорити, що для знаходження деякого проміжного рішення завдання МКР розбивається на безліч одномірних завдань МКЕ, число яких дорівнює числу ліній сітки МКР.

Можливо спільне застосування способу укрупнення сітки й пропонованого способу, адже навіть для укрупненої сітки процес прискорення збіжності залишається актуальним. У загальному випадку застосування запропонованого способу прискорення збіжності повинне вироблятися з урахуванням фізичної сутності процесів і умов кожного конкретного завдання. При укрупненні сітки використовуються не всі граничні умови, а тільки їхня частина, тому можна також використовувати деякі усереднені граничні умови. У цьому випадку, фактично для одержання нових граничних умов, буде застосований запропонований спосіб, але в локальній області.

Проміжне рішення, отримане пропонованим способом [14,15], у деяких випадках (залежно від особливостей граничних умов), може бути «не дуже» гладким. На практиці можливе застосування деяких додаткових дій, що підвищують гладкість. Наприклад, при апроксимації можливий облік (з деякими ваговими коефіцієнтами) значень у сусідніх вузлах, що не ввійшли як значення або аргументу у функцію апроксимації, застосовувану до даної лінії сітки.

Пропонований спосіб знаходження попереднього рішення базується не на послідовних ітераційних наближеннях, а на апроксимації проміжного рішення в області $\mathbf{U}$ за наявним даними про граничні умови $\mathbf{B}$, з урахуванням правої частини (3.8). При цьому обчислювальна складність може бути оцінена як:

$$O\left(n\prod_{i=1}^{n}N_i\right), \tag{3.22}$$

де, як правило, $n<k$, причому n й k мають той же зміст, що й в (3.20), що дозволяє говорити про зниження обчислювальної складності пропонованого способу стосовно послідовного здрібнювання сітки.

У загальному випадку зазначений підхід підвищення швидкості МКР застосуємо не тільки для відновлення вихідного сигналу по паттерну, але й для інших завдань, рішення яких можливо за допомогою МКР. Додатково спосіб тестувався на різних завданнях електротехніки, описуваних рівнянням Лапласа й Пуассона, у просторах із числом вимірів простору $n=2$ й $n=3$.

У всіх випадках отримане проміжне рішення дозволяло істотно скоротити число ітерацій МКР для одержання рішення із заданою точністю. Особливий виграш у скороченні числа ітерацій спостерігається при гладких крайових умовах, аж до того, що при деякій достатній гладкості може бути отримане рішення тільки із застосуванням зазначеного способу без ітераційного процесу МКР. Такі крайові умови цілком актуальні для деяких типів зображень.

Істотне прискорення збіжності, у результаті оптимізації розглянутим способом, на прикладі нев'язань у рішенні двовимірної (плоско-паралельної) завдання електростатики, наведено на зображенні (рисунок 3.6). Для розбивки області $\mathbf{W}$ вибиралася прямокутна сітка, утворена лініями, паралельними координатним осям, з рівним у всіх напрямках кроком h. Як багаточлен (3.21) було використане рівняння прямій a_0+a_1z, коефіцієнти якої (3.a_0,a_1) розраховувалися по двох елементах крайових умов з $\mathbf{B}$.

Відзначимо, що застосування розглянутого способу в деяких двовимірних завданнях, для досягнення рішення із заданої по нев'язанню точністю, дозволило зменшити число ітерацій в 10-100 разів, що говорить про істотне підвищення ефективності МКР.

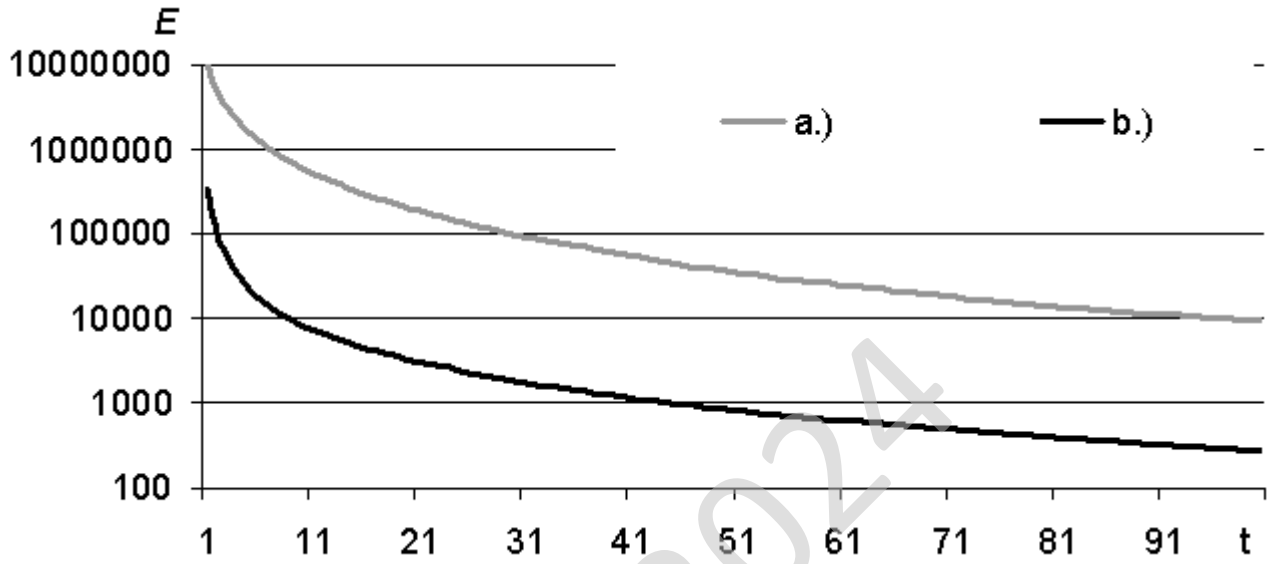

Рисунок 3.6 – Значення логарифмів сумарних нев'язань $\ln[E]$ по всіх вузлах сітки залежно від номера ітерації t у двовимірному просторі XOY. а.) – без пошуку попереднього рішення; b.) – з пошуком попереднього рішення

Ще одним додатковим способом підвищення швидкості обчислень МКР є розпаралелювальні ітераційного процесу, наприклад, по елементах сітки. Досить просто таке рішення реалізується на базі сучасних графічних процесорів, наприклад, за допомогою технології CUDA, що дозволяє багаторазово (у сотні разів) підвищити швидкість обчислень [16].

Порівняння й деякі результати стиску цифрових сигналів за допомогою аналізу диференціальної структури

Розглянемо деякі результати практичного застосування викладеного вище принципу стиску на прикладі компресії зображень. Як аналізуюче рівняння було обране рівняння Лапласа (3.13). Зображення для тестування вибиралися з бази даних стандартних тестових зображень [17], з урахуванням фактора, що стиск на

підставі аналізу диференціальної структури, найбільше ефективно може бути застосоване до графічних об'єктів, що містять контрастні границі й значні поля градієнта або одного тону.

Одне з обраних для тестування зображень (рисунок 3.7) має наступні властивості: є цифровим сигналом, отриманим у результаті фотографування; містить контрастні границі об'єктів; містить значні поля одного тону. Розглянемо залежності від значення Δ-критерію наступних величин:

– число виключених елементів з паттерна;

– коефіцієнт стиску:

$$k = \frac{D_{out}}{D_{in}}, \tag{3.23}$$

де: D_{out} – обсяг стислих (вихідних) даних (біт); D_{in} – обсяг вихідних (вхідних) даних (біт);

– середньоквадратична помилка (MSE – mean square error):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f[i] - r[i])^2, \tag{3.24}$$

де: $f[i]$ – значення вихідного й $r[i]$ – відновленого сигналів;

– пікове відношення сигнал/шум (PSNR – peak signal to noise ratio):

$$PSNR = 20\log_{10}\frac{\max_i(|f[i]|)}{\sqrt{MSE}}; \tag{3.25}$$

– відношення сигнал/шум (SNR – signal to noise ratio):

$$SNR = 20\log_{10}\frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} f[i]^2}}{RMSE}, \tag{3.26}$$

де: RMSE – корінь середньоквадратичної помилки тобто: $RMSE = \sqrt{MSE}$.

Як відзначалося вище, від величини Δ-критерій залежить число виключених з паттерна елементів, у кінцевому результаті, стиск. При цьому,

можна чекати наступну закономірність: чим більше величина Δ-критерію, тим більше виключено елементів з паттерна й тем менш точно можна відновити вихідний сигнал. Коли величина Δ-критерію буде мати значення більше або рівне деякому граничному, паттерн, буде містити тільки елементи приналежній границі зображення. Отже при граничній величині Δ-критерію можна одержати максимальний стиск. При цьому питання про якість відновленого зображення, отриманого з паттерна сформованого по Δ-критерії більш або рівного граничному, залишається відкритим.

Рисунок 3.7. Стандартне тестове зображення «example_kropivnitskii.bmp»

Розмір вихідного файлу зображення 196662 байт, розміри 256×256 пікселів, глибина кольору 24 біта, розмір цифрового сигналу зображення $256 \times 256 \times 3 = 196608$ байт.

У таблиці (таблиця 3.2) наведені чисельні результати експериментів. Для всіх колірних каналів величина Δ-критерію була обрана однакова, формування паттерна вироблялося в колірному просторі RGB. Після формування паттерна вироблявся стиск довжин серій, а потім кодування Хафманна. Для результуючих даних був розроблений спеціальний формат що дозволяє зберегти у файл не тільки самі дані але й деяку службову інформацію: дату, розміри зображення, використовуване колірний простір, деякі інші дані записані користувачем. Службова інформація трохи збільшує розмір готового файлу в порівнянні

розміром стислого сигналу. Довжина службових даних у файлах, створюваних стиском за допомогою аналізу диференціальної структури, склала 52 байта, а у вихідному файлі «example_kropivnitskii.bmp» 54 байта. При загальному розмірі вихідного файлу «example_kropivnitskii.bmp» 196662 байт. Для відновлення зображення застосовувався МКР (1000 ітерацій) з попереднім відшуканням проміжного рішення. Розпаралелювальні обчислювального потоку не застосовувалося. Розрахунок коефіцієнта стиску й фактора стиску вироблявся шляхом співвідношення розмірів кінцевих файлів вихідного зображення «example_kropivnitskii.bmp» (3.196662) і файлу отриманого в результаті стиску за допомогою аналізу диференціальної структури.

Таблиця 3.2 – Деякі оцінки стиску зображення «example_kropivnitskii.bmp»

Δ-критерій	Число виключених елементів		Коефіцієнт стиску	Фактор стиску	*MSE*	*PSNR*	*SNR*
	байт	%	*k*	*c*			
0	60	0,030518	0,889	1,125	0,001302	76,98	72,37041
5	23535	11,97052	0,840	1,191	0,121638	57,28	52,66611
10	57828	29,41284	0,699	1,431	0,800832	49,10	44,48139
15	84129	42,79022	0,582	1,719	2,204839	44,70	40,08303
20	107364	54,60815	0,481	2,080	5,200922	40,97	36,356
25	127779	64,99176	0,388	2,578	11,78631	37,42	32,80302
30	144492	73,49243	0,304	3,290	22,84199	34,54	29,92946
35	157290	80,00183	0,234	4,275	36,16757	32,55	27,93361
40	166086	84,47571	0,184	5,436	56,36549	30,62	26,00667
45	171954	87,46033	0,148	6,759	85,77358	28,80	24,18327
50	176058	89,54773	0,123	8,132	127,981	27,06	22,44535
55	179193	91,14227	0,104	9,618	192,4735	25,29	20,67309
60	181608	92,37061	0,089	11,239	268,6562	23,84	19,22483
65	183483	93,32428	0,077	12,991	384,8277	22,28	17,66414
70	184998	94,09485	0,068	14,710	503,2751	21,11	16,49875

На рисунку 3.8 наведені деякі зображення, що дозволяють одержати візуальне подання про якість відновленого зображення й відповідному паттерні.

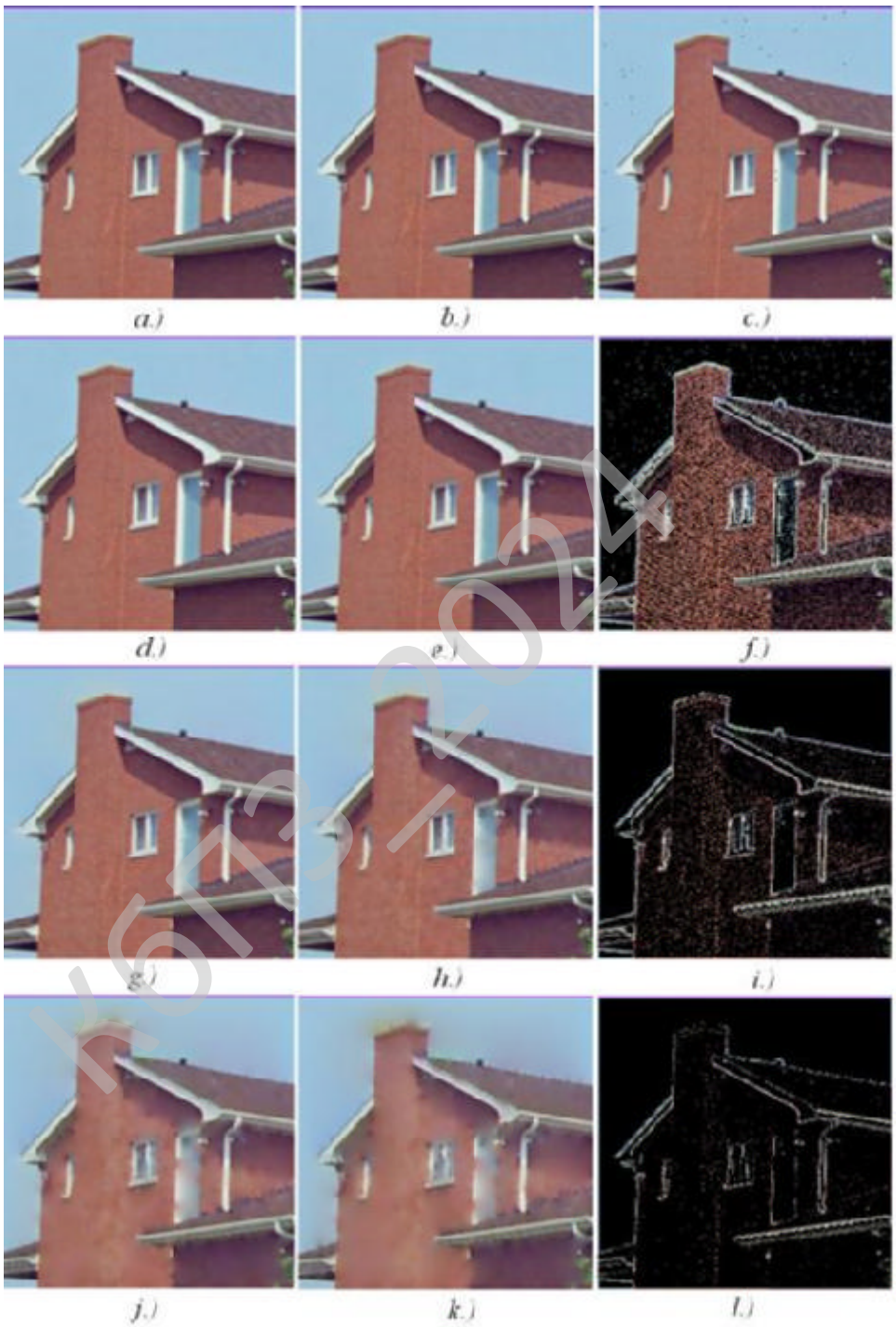

Рисунок 3.8 – Вихідне зображення (a), далі відновлені (*b., d., e., g., h., j., k.*) 1000 ітерацій МКР із використанням проміжного рішення. Колонка праворуч (*c., f., i., l.*) відповідає паттернам. Значення Δ-критерію наступні: *b.*, *c.* – 0; *d.* – 10; e., *f.* – 20; *g.* – 30; *h.*, *i.* – 40; *j.* – 50; *k.*, *l.* – 60

Звичайне значення коефіцієнта $PSNR$ для зображень прийнятної якості становить величину порядку $20...40$ одиниць [8]. Зіставляючи результати (таблиця 3.2) і (рисунок 3.8) можна помітити, що в області значень Δ-критерію порядку $40...50$ спостерігається істотне, для візуального сприйняття, погіршення якості зображення, виникають області розмиття поблизу границь об'єктів, при цьому значення $PSNR = 30,62$ (для $\Delta = 40$) і $PSNR = 27,06$ (для $\Delta = 50$), що відповідає очікуваній області обмеження прийнятної якості. Коефіцієнт стиску $k = 0,184$ (для $\Delta = 40$) і $k = 0,123$ (для $\Delta = 50$). При подальшому росту величини Δ-критерію спостерігається подальше погіршення якості,, що проявляється особливо на границях графічних об'єктів, вивчення структури паттерна показує, що знижується і якість відображення текстуро-подібної однотонної області цегельних стін, при порівнянні паттерна (рисунок 3.8, *h*.) з (рисунок 3.8, *j*.) помітно, що пропадають граничні елементи, що розділяють окремі малоконтрастні елементи зображення. При подальшому збільшення величини Δ-критерію до $70...80$ одиниць уміст зображення, незважаючи на значні перекручування, залишалося пізнаваним. Далі розглянемо зіставлення отриманих результатів з деякими іншими, найпоширенішими графічними форматами (таблиці 3.3, 3.4, 3.5).

Серед зображень отриманих різними способами стиску виберемо по одному, що має припустиме для візуального сприйняття якість при мінімальному значенні коефіцієнта стиску. Незважаючи на те, що даний вибір носить суб'єктивний характер, він цілком припустимий серед інших порівнянь, тому що дозволяє врахувати фактор специфіки сприйняття зображення людиною. Зображення (рисунок 3.9) і чисельна оцінка стиску різними методами (таблиця 3.6) дозволяють зробити зіставлення результатів.

Таблиця 3.3 – Деякі оцінки JPEG-стиску зображення «example_kropivnitskii.bmp»

Якість	Коефіцієнт стиску k	Фактор стиску c	*MSE*	*PSNR*	*SNR*
12	0,494	2,023	1,56	46,17	41,56
11	0,359	2,778	5,63	40,62	36,01
10	0,282	3,538	9,35	38,41	33,80
9	0,245	4,069	12,97	37,00	32,39
8	0,222	4,497	17,13	35,79	31,18
7	0,201	4,970	23,31	34,45	29,84
6	0,199	5,006	40,10	32,10	27,49
5	0,187	5,333	46,79	31,43	26,82
4	0,179	5,556	51,48	31,01	26,40
3	0,174	5,746	58,72	30,44	25,83
2	0,164	6,064	75,88	29,33	24,72
1	0,160	6,243	96,81	28,27	23,66
0	0,158	6,316	105,61	27,89	23,28

Таблиця 3.4 – Деякі оцінки PNG-стиску зображення «example_kropivnitskii.bmp»

Число розрядів	Число кольорів	Коефіцієнт стиску k	Фактор стиску c	*MSE*	*PSNR*	*SNR*
24	16777216	0,609136	1,641668	0	inf	inf
8	256	0,191079	5,233434	10,80395	37,79498	33,18098
8	128	0,15489	6,45619	16,1552	36,04768	31,43368
8	64	0,111638	8,957504	24,60652	34,22031	29,6063
8	32	0,091278	10,95549	34,69823	32,72773	28,11373
8	16	0,040465	24,71249	50,41061	31,10559	26,49158
8	8	0,026284	38,04643	140,4574	26,65536	22,04136
8	4	0,01436	69,63952	252,7536	24,10383	19,48983
8	2	0,007144	139,973	569,2647	20,57766	15,96366

Таблиця 3.5 – Деякі оцінки GIF-стиску зображення «example_kropivnitskii.bmp»

Число кольорів	Коефіцієнт стиску k	Фактор стиску c	*MSE*	*PSNR*	*SNR*
128	0,171243	5,839653	16,91316	35,84856	31,23455
64	0,123583	8,091754	26,3118	33,9293	29,3153
32	0,100335	9,966653	37,71211	32,36599	27,75199

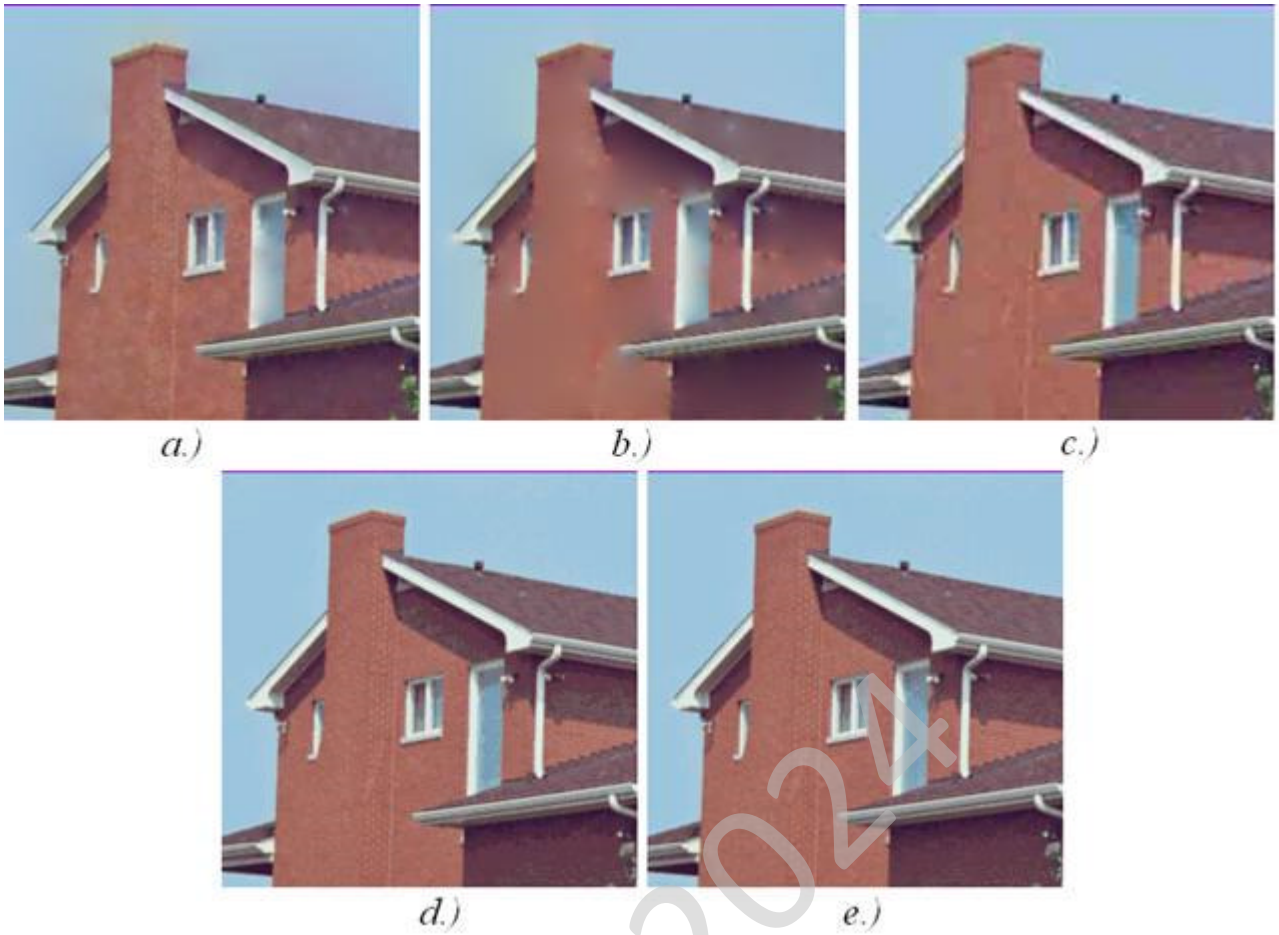

Рисунок 3.9 – Зображення, отримані різними способами стиску (таблиця 3.6):

a.) – стиск на основі аналізу диференціальної структури ($\Delta = 39$ таблиця 3.2);

b.) – стиск на основі аналізу диференціальної структури з попередньої зверткою;

c.) – JPEG-стиск (якість 0 таблиця 3.3);

d.) – PNG-стиск (розрядність 8, число кольорів 32 таблиця 3.4);

e.) – GIF-стиск (32 кольори таблиця 3.5)

Таблиця 3.6 – Порівняння деяких результатів стиску зображення «example_kropivnitskii.bmp», різними способами

Зображення (рисунок 3.9)	Коефіцієнт стиску k	Фактор стиску c	*MSE*	*PSNR*	*SNR*
a.	0,193	5,183	52,1964	30,95	26,34
b.	0,156	6,391	76,36	29,30	24,69
c.	0,158	6,316	105,61	27,89	23,28
d.	0,091	10,955	34,69	32,72	28,11
e.	0,100	9,967	37,712	32,37	27,75

Візуальне (рисунок 3.9) і чисельне (таблиця 3.6) порівняння отриманих результатів показують, що стиск на основі аналізу диференціальної структури дозволяє домогтися стиску порівнянного з JPEG при порівнянній якості. При цьому, якість відновленого зображення, стисливого за допомогою аналізу диференціальної структури, має краще чисельні оцінки (3.MSE, PSNR, SNR), що може бути особливо корисно для подальших машинних перетворень, наприклад з метою розпізнавання образів.

Цікаві результати застосування згортки (рисунок 3.10), (таблиця 3.7) із ФРТ:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

яка є низькочастотним однорідним фільтром «ковзного середнього» з посиленням. Застосування згортки дозволило більш ефективно виділити границі великих об'єктів і практично виключити шумову складову. Виключення високочастотних складових і шуму дозволяє одержати більше протяжні ланцюжки виключених з паттерна елементів, що у свою чергу додатково сприяє стиску. Недоліком застосування згортки є можлива втрата високочастотних елементів, наприклад, невеликих об'єктів, тонких ліній і т.д.

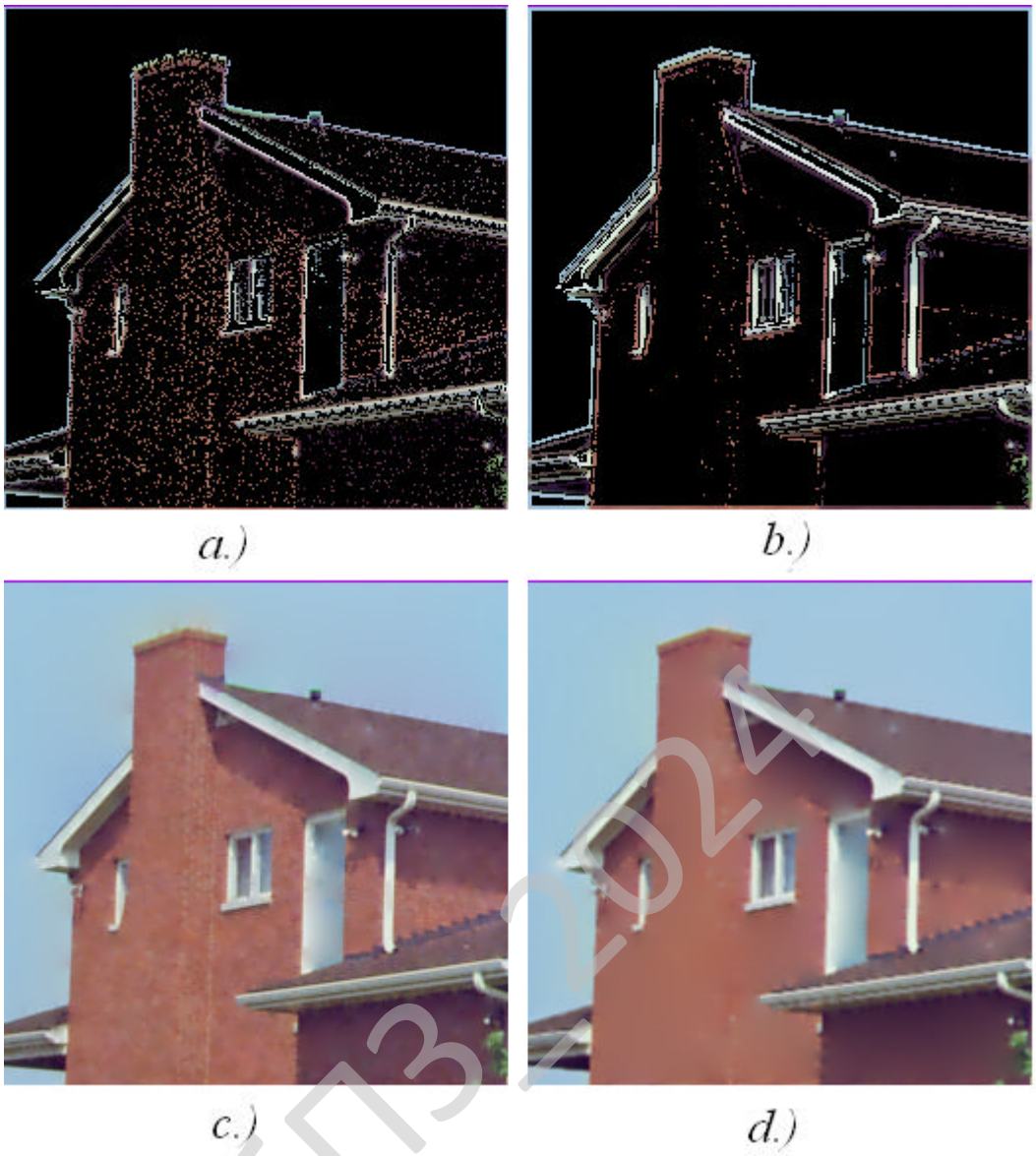

Рисунок 3.10 – Паттерни й відновлені зображення: *a*. – паттерн отриманий без застосування згортки; *b*. – паттерн отриманий із застосуванням згортки; *c*. – зображення відновлене по паттерну (*a*.); *d*. – зображення відновлене по паттерну (*b*.)

Таблиця 3.7 – Результати оцінки впливу згортки на стиск за допомогою аналізу диференціальної структури на прикладі зображення «example_kropivnitskii.bmp»

Застосування згортки	Число виключених елементів з паттерна	Коефіцієнт стиску k	Фактор стиску c	MSE	$PSNR$	SNR
немає	166086	0,184	5,434	56,114	30,640	26,026
так	166119	0,172	5,829	66,427	29,907	25,293

Результати порівняння (таблиця 3.7) і (рисунок 3.10) показують, що застосування згортки дозволяє одержати більше компактний вид стислого файлу при кращій візуальній якості відновленого зображення.

Додаткові дослідження на значному числі тестових зображень показали, що розроблений метод стиску в більшості випадків за чисельними оцінками (MSE, $PSNR$, SNR) показує результати переважаючі (при рівних коефіцієнтах стиску) отримані розповсюдженим методом стиску JPEG і JPEG2000. При цьому якість візуального сприйняття (при рівному ступені стиску) виходить порівнянне (з JPEG иJPEG2000) для контрастних зображень, із чітко виділеними границями об'єктів і значними градієнтними або однотонними просторами. Для малоконтрастних зображень, з маловираженими границями об'єктів візуальна якість виходить нижче. Проявляється зниження якості в характерному розмитті границь, і втрати малоконтрастних фрагментів зображення.

Отримані результати показують конкурентоспроможність способу стиску за допомогою аналізу диференціальної структури. При цьому необхідно відзначити, що спосіб стиску зображень на основі аналізу диференціальної структури має додатковий потенціал. При стиску паттерна в даних дослідженнях використовувався код Хафмена, відомо, що, наприклад, арифметичне кодування (застосовуване в JPEG2000) дозволяє робити більше ефективний стиск.

Додатково підвищити ефективність стиску можливо, використовуючи, для формування паттерна й відновлення сигналу, замість рівняння Лапласа, рівняння Пуассона. Причому в правій частині рівняння Пуассона можна врахувати спектральні особливості сигналу, за допомогою вейвлет або Фур'є перетворення. Такий підхід дозволить поряд з диференціальною враховувати й частотну структуру сигналу.

На рисунку 3.11 наведено структурну схему системи. Вона складається з наступних блоків:

– Стиснуте зображення.

– Нетиснуте зображення.

– Кодер.

– Декодер.

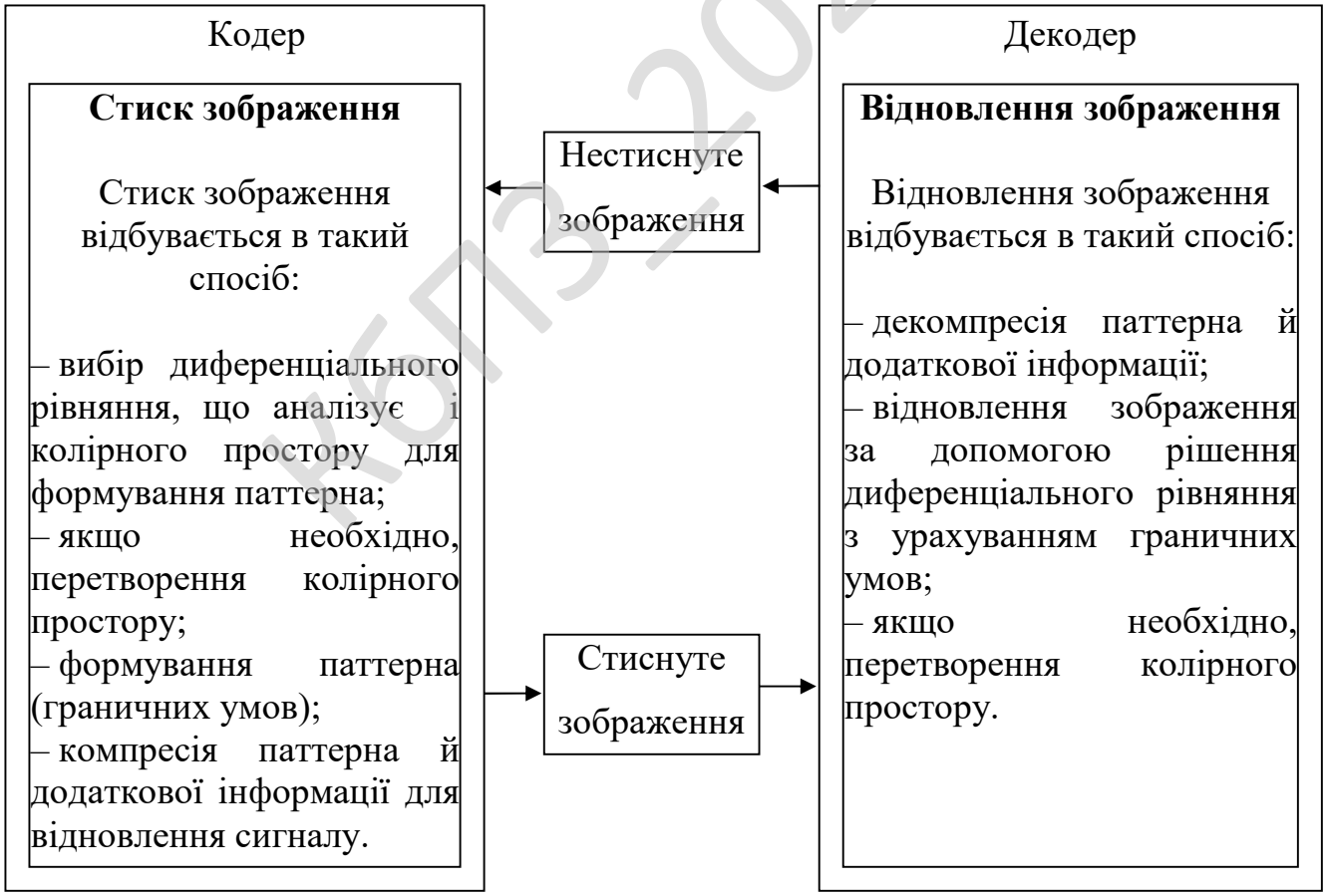

Рисунок 3.11 – Структурна схема системи

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.12. З рисунку видно, що розроблена система складається з наступних частин:

1. Інтерфейс користувача програмного перетворення зображень на основі методів диференційного аналізу.

2. Блок відкриття різних типів зображень.

3. Блок оптимізації, стиску й збереження зображень.

4. Вікно перегляду результатів стиску.

5. Вікно відображень розмірів файлів (вихідного й стислого).

6. Вікно роботи з метаданими зображення.

7. Набір стандартних інструментів (поворот, масштабування, відбиття зображення як горизонтально, так і вертикально).

8. Блок змін яскравості, контрастності, гами, і інвертування.

9. Блок налаштування файлів JPEG.

10. Блок налаштування файлів GIF.

11. Блок налаштування файлів PNG.

12. Блок налаштування маски зображення.

Основні характеристики:

− програма може відкривати велику кількість різних типів зображень, таких як: jpg, jpeg, jpe, jif, png, gif, bmp, tif, tiff, psd, ico, tga, targa, mng, jng, j2k, j2c, jp2, pcd, pcx, wpa, wbmp, wbm, xbm, xpm, dds, g3, koa, iff, lbm, pbm, pgm, ppm, ras, cut, sgi, pct, pict, pic;

− за допомогою програми можна оптимізувати й згодом зберегти вихідне зображення у форматах JPEG, GIF і PNG, причому для цього не прийдеться мати специфічні знання, інтерфейс програми настільки простий що розібратися зможе кожний;

− вікно програми розділене на два вікна: вихідне зображення й стиснуте, багато налаштувань застосовуються в режимі реального часу, без додаткового натискання клавіш – автоматичний перегляд результатів;

− можна виставити заданий розмір файлу, для стиснутого зображення;

− на екрані відразу відображається розмір підсумкового файлу й вихідного;

− у програмі можна працювати як з одним файлом, так і з декількома одночасно;

− є можливість роботи із прозорістю;

− можна поміняти метадані зображення (коментарі, IPTC, Adobe XMP, EXIF профайл, ICC профайл), не підтримувані метадані будуть видалені;

− можлива передача метаданих між зображеннями (але це в тому випадку якщо кінцевий файл підтримує такі типи метаданих);

− зі стандартних інструментів доступні: поворот, масштабування, і їсти можливість відбити зображення як горизонтально так і вертикально;

− можна так само можливо перемінити яскравість, контрастність, гаму, і інвертувати;

− у реальному часі є можливість зменшити кількість кольорів PNG і GIF, для того що б розмір файлу став менший;

− можлива зміна розміру зображення за допомогою відомих фільтрів таких як: Lanczos3, Catmull Rom, Bicubic, і ін.;

− у доповненнях можна знайти підтримку зовнішніх оптимізаторів зображення PNG (optiPNG, PNGOut).

Для файлів JPEG доступні наступні налаштування:

− якість стиску;

− розширені можливості вибору насиченості кольору (відсутній, низький 4:2:2, середній 4:2:0, високий 4:1:1);

− можна зберегти зображення як відтінки сірого 8-біт;

− стандартна оптимізація (оптимальні таблиці Хаффмана) або прогресивне кодування.

Підтримується на вибір наступна глибина кольору: 24 біта, 8 біт відтінки сірого.

Рисунок 3.8 – Функціональна схема системи

Налаштування для файлів GIF:

1. Зміна кольору:

− зміна кольору від 256 до 2 кольорів (без згладжування) з використанням Xiaolin Wu або NeuQuant neural net;

− палітра 8 біт відтінки сірого;

− floyd-steinberg згладжування 1 біт монохромний.

2. Збереження черезрядкового.

На виході підтримується глибина: 4, 8 біт, 8 біт градацій сірого, 1 біт монохромний.

Налаштування для файлів PNG:

1. Підтримуються вихідні кольори (24 біт RGB, 32 біт RGBA).

2. Зміна кольору:

− зміна кольору від 256 до 2 кольорів (без згладжування) з використанням Xiaolin Wu або NeuQuant neural net;

− палітра 8 біт відтінки сірого;

− floyd-steinberg згладжування 1 біт монохромний.

3. Збереження черезрядкового.

4. Домогтися максимальних налаштувань стиску можна шляхом використання інтеграції з популярними оптимізаторами PNG (optiPNG, PNGOut):

− додати / видалити / змінити зовнішній інструмент оптимізації PNG.

На виході підтримується глибина: 4, 8, 24, 32 бітний колір, 8 біт градацій сірого, 1 біт монохромний.

Налаштування для метаданих

Можна зберегти або видалити наступні метадані:

− коментарі;

− Adobe XMP інформація;

− IPTC інформація;

− профіль EXIF (у тому числі GPS і творець замітки);

− профілі кольору ICC.

Невідомі або не підтримувані метадані автоматично видаляються.

Налаштування маски

У програмі можна вибрати кілька варіантів прозорості:

− Зберігати прозорість (використання порога для переходу від альфа до індексованої прозорості).

− Суміш із чистим тлом (можна вибрати колір для змішування прозорості у фоновому режимі (альфа состав)).

− Можна зробити не прозорим (можливість видалення прозорості інформації, роблячи всі непрозорим).

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.9. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування ПЗ.

– Формування шаблону перетворення зображення.

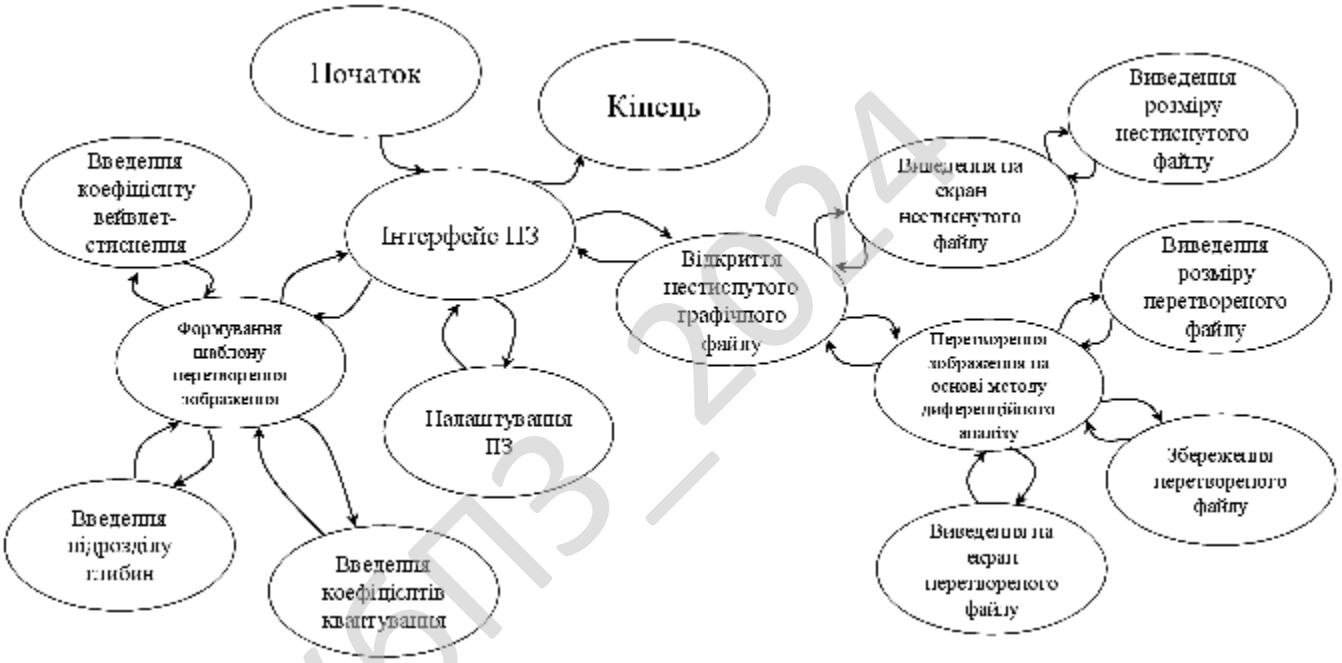

Рисунок 3.9 – Діаграма взаємодії процесів

– Введення коефіцієнту вейвлет-стиснення.

– Введення підрозділу глибин.

– Введення коефіцієнтів квантування.

– Відкриття нестиснутого графічного файлу.

– Виведення на екран нестиснутого файлу.

– Виведення розміру нестиснутого файлу.

– Перетворення зображення на основі методу диференційного аналізу.

– Виведення розміру перетвореного файлу.

– Збереження перетвореного файлу.

– Виведення на екран перетвореного файлу.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

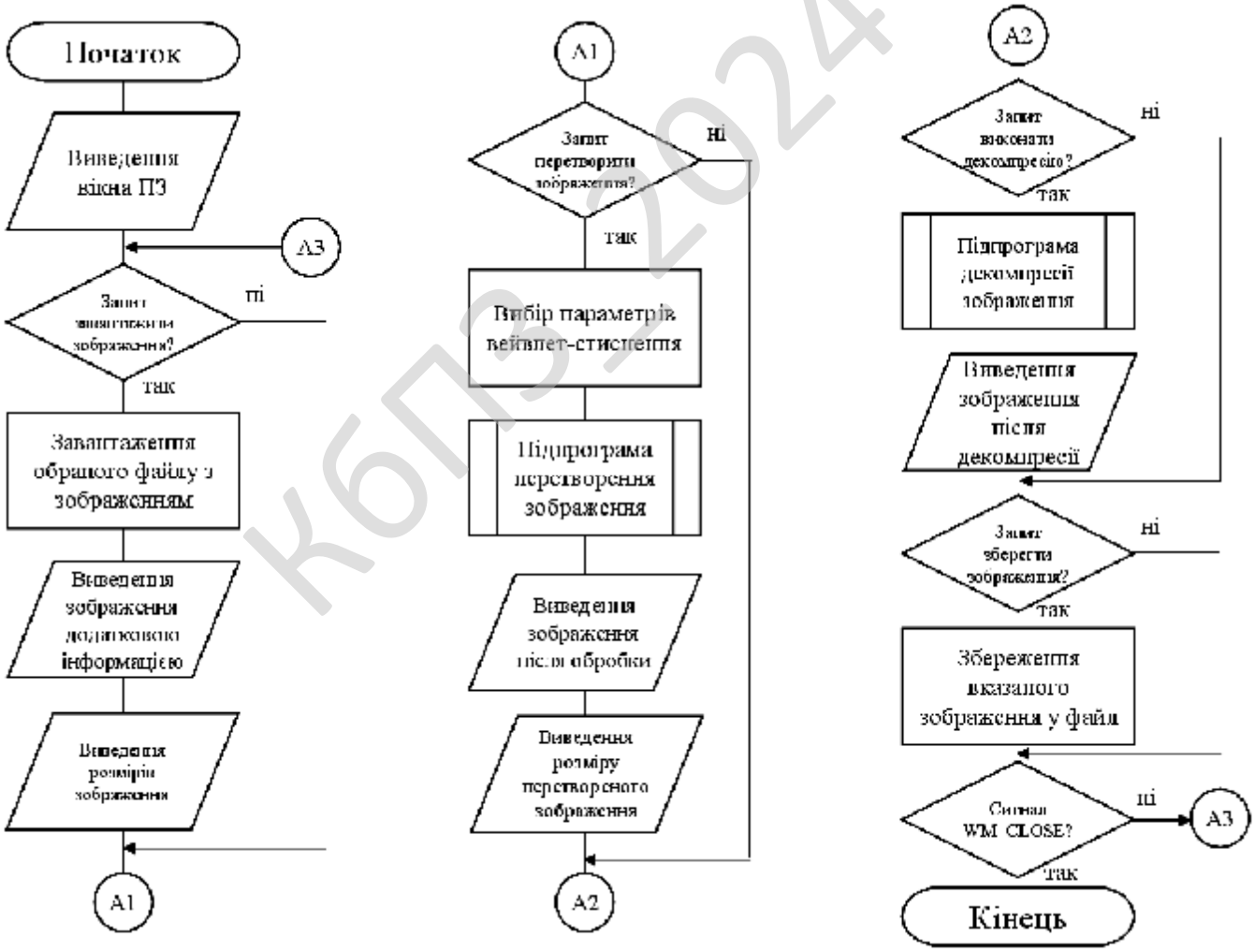

Рисунок 4.1 – Блок схема основної програми

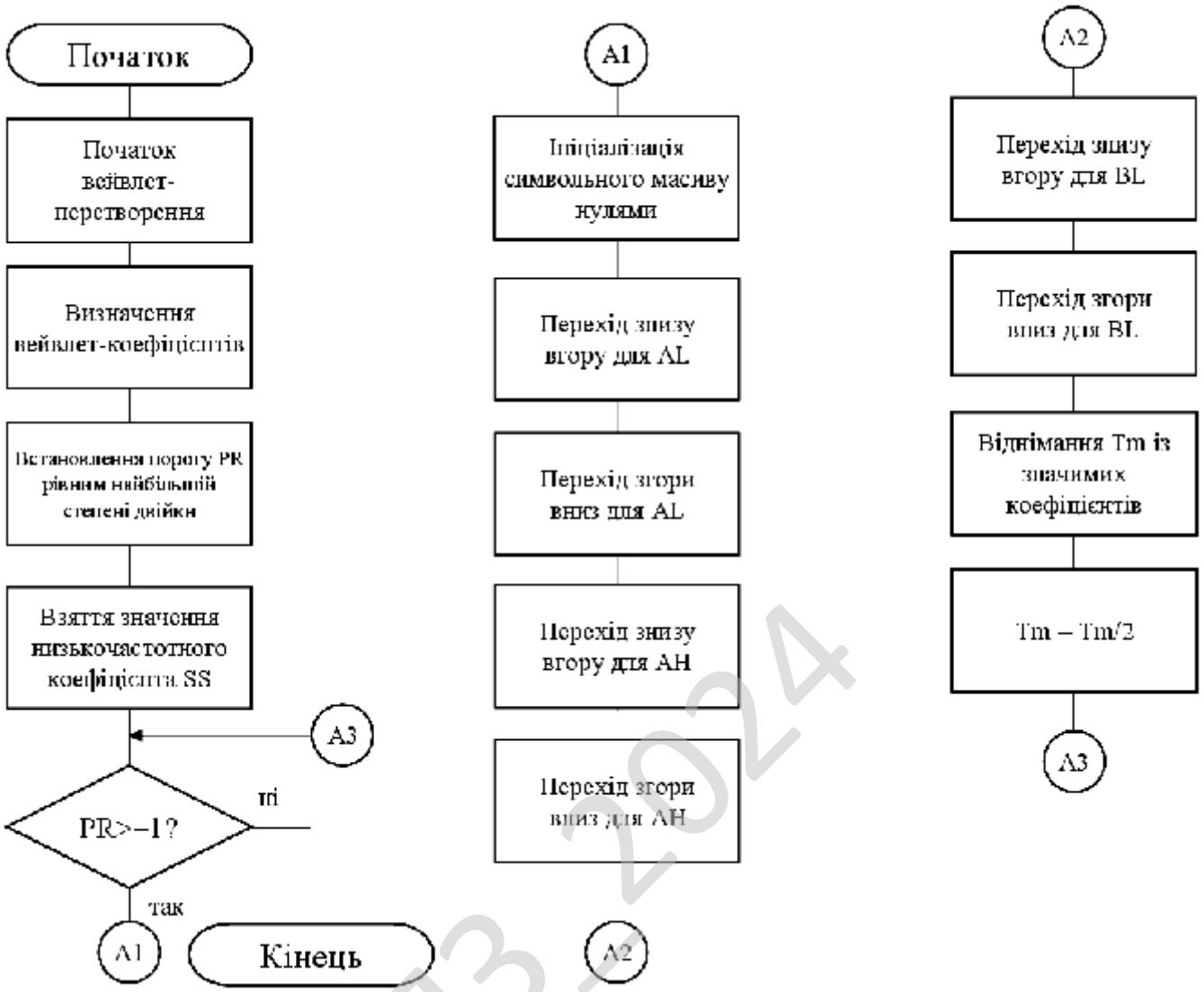

Рисунок 4.2 – Блок схема підпрограми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю перетворення зображень на основі методів диференційного аналізу.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Опис алгоритмів функціонування системи

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Опис системи перетворення зображень

Використовується система Вейвлет перетворення. Вона застосовується в різних галузях прикладної науки, таких як обробка сигналів і зображень. Найбільш широке застосування дискретне вейвлет-перетворення

використовується в кодуванні сигналів, де властивості перетворення використовуються для зменшення надмірності в представленні дискретних сигналів, часто – як перший етап в компресії даних.

Алгоритм, який використовується в роботі для стиснення зображень, заснований на отриманні за допомогою двохетапного дискретного вейвлет-перетворення чотирьох зображень:

а) LL – і в горизонтальному, і у вертикальному напрямі низькочастотні відліки;

б) LH – в горизонтальному низькочастотні, у вертикальному високочастотні;

в) HL – в горизонтальному високочастотні, у вертикальному низькочастотні;

г) HH – в обох напрямах високочастотні.

Потім відліки цих зображень подаються як вхідні дані для алгоритму стиснення GZIP у вигляді одновимірного вектора розмірністю чотири.

Суть методу полягає в розкладанні сигналу по базисних функціях локалізованим як в просторі, так і в часі. Це дозволяє враховувати нестаціонарну поведінку коефіцієнтів вейвлет-перетворення.

Крім того, масштабованість базисних функцій дозволяє аналізувати сигнал зображення з різним ступенем деталізації, що знаходить своє застосування при стисканні зображень.

Стискання сигналів здійснюється через набори банку фільтрів із заданими властивостями здійснюючи ітераційно декомпозицію зображення.

При цьому фільтри підбираються так, щоб забезпечити точне відновлення початкового сигналу по вейвлет-коефіцієнтам.

Ця умова накладає певні обмеження на характеристики використовуваних фільтрів. Так, наприклад, фільтри аналізу і синтезу повністю або частково ортогональні один одному.

Опис системи перетворення формату YCrCb до RGB

Для отримання початкового зображення слід провести зворотне перетворення із YCrCb в RGB:

```
private byte[] YCrCb_to_RGB(double[, ,] yuv, int w, int h, double Ydiv, double
                            Udiv, double Vdiv)
        {
            byte[] bytes_flat = new byte[3 * w * h];
            double vr, vg, vb;
            double v, vCb, vCr;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            for(int j = 0; j < h; j++)
            {
                for (int i = 0; i < w ; i++)
                {
                    vCr = yuv[0, i, j] / Vdiv;
                    vCb = yuv[1, i, j] / Udiv;
                    v = yuv[2, i, j] / Ydiv;
                    vr = v + 1.402f * (vCr - 128f);
                    vg = v - 0.34414f * (vCb - 128f) - 0.71414f * (vCr - 128f);
                    vb = v + 1.722f * (vCb - 128f);
                    if (vr > 255) {vr = 255;}
                    if (vg > 255) {vg = 255;}
                    if (vb > 255) {vb = 255;}
                    if (vr < 0) {vr = 0;}
                    if (vg < 0) {vg = 0;}
                    if (vb < 0) {vb = 0;}
                    if (vb < 0) {vb = 0;}
                    bytes_flat[j * w * 3 + i * 3 + 0] = (byte)vb;
                    bytes_flat[j * w * 3 + i * 3 + 1] = (byte)vg;
                    bytes_flat[j * w * 3 + i * 3 + 2] = (byte)vr;
                }
            }
            return bytes_flat;
        }
```

Опис системи перетворення формату RGB до YCrCb

Зображення яке використовується слід перетворити з формату RGB в YCrCb. Дана операція здійснюється за допомогою наступного методу:

```
private double[, ,] RGB_to_YCrCb(byte[, ,] BytesRGB, int c, int c,
                                 double Ydiv, double Udiv, double Vdiv,int o,int o)
        {
// опис типів даних
double vr, vg, vb;
double kr = 0.299, kg = 0.587, kb = 0.114, kr1 = -0.1687, kg1 = 0.3313, kb1 = 0.5,
      kr2 = 0.5, kg2 = 0.4187, kb2 = 0.0813;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            double[, ,] YCrCb = new double[3, c, c];
// основний цикл
            for (int j = 0; j < o; j++)
            {
                for (int i = 0; i < o; i++)
                {
                 vb = (double)BytesRGB[0, i, j];
                 vg = (double)BytesRGB[1, i, j];
                 vr = (double)BytesRGB[2, i, j];
                 YCrCb[2, i, j] = (kr * vr + kg * vg + kb * vb) * Ydiv;
                 YCrCb[1, i, j] = (kr1 * vr - kg1 * vg + kb1 * vb + 128) * Udiv;
                 YCrCb[0, i, j] = (kr2 * vr - kg2 * vg - kb2 * vb + 128) * Udiv;
                }
            }
            return YCrCb;
        }
```

Кожен канал кодується окремо із різним ступенем стиснення. Для роботи з графікою необхідно обробляти двовимірні масиви даних. Кратномасштабний аналіз повинен будуватися в просторі функцій два змінних. Існує декілька способів узагальнити викладені раніше конструкції на функції декілька змінних (ми говоритимемо тільки про функції два змінних).

Найпростіший і широко поширений шлях – тензорний добуток одновимірних КМА. Як двовимірна скейлінг-функція береться

$$\Phi(t,s) = \varphi(t)\varphi(s).$$

Замість одного вейвлету виникає три:

$$\Psi_{LH}(t,s) = \varphi(t)\psi(s), \Psi_{HL}(t,s) = \psi(t)\varphi(s), \Psi_{HH}(t,s) = \psi(t)\psi(s).$$

(L означає низьку частоту, H – високу частоту). Простори породжуються зсувами скейлінг – функції на одному і тому ж масштабі:

$$V_j = \left\{2^j \Phi(2^j t - k, 2^j s - m)\right\},$$

простори деталей мають вигляд:

$$W^{LH}{}_j = \left\{2^j \Psi_{LH}(2^j t - k, 2^j s - m)\right\}, W^{HL}{}_j = \left\{2^j \Psi_{HL}(2^j t - k, 2^j s - m)\right\},$$
$$W^{HH}{}_j = \left\{2^j \Psi_{HH}(2^j t - k, 2^j s - m)\right\}$$

Відповідні проекції знаходяться застосуванням фільтрів:

$$x \mapsto \left\{H_t \otimes H_s(x), H_t \otimes G_s(x), G_t \otimes H_s(x), G_t \otimes G_s(x)\right\},$$

де x – двовимірний сигнал. Класична схема Малла передбачає рекурсивне застосування тієї ж процедури до низькочастотної складової. Коефіцієнти обираються згідно визначених правил.

Опис стиснення зображень

Найпростіший підхід до стиснення зображень за допомогою вейвлет-перетворення полягає в наступному:

– Виконати вейвлет-перетворення.

– Упорядкувати коефіцієнти .

– Відкинути “хвіст” впорядкованого масиву, енергія якого дорівнює допустимій (за умовами завдання) величині.

– Запам'ятати збережені коефіцієнти і їх положення в масиві початкових коефіцієнтів.

– При відновленні замінювати відкинуті коефіцієнти нулями.

Ця ідея в тій або іншій формі присутня у всіх методах вейвлетного стискування. Дану процедуру можна, наприклад, застосовувати роздільно до кожного з квадрантів, отриманих при розкладанні по оптимально вибраних вейвлет-пакетам.

Вейвлет-перетворення виконується за допомогою наступної функції:

```
private double[, ,] WaveletePack(double[, ,] ImgArray, int Component, int c,
                                 int c, int dwDevider, int dwTop, int dwStep)
{
            short Value;
```

```
            int cw2 = c / 2;
            int c2 = c / 2;
            double dbDiv = 1f / dwDevider;
            ImgArray = Wv(ImgArray, c, c, Component, WV_TOP_TO_BOTTOM);
            ImgArray = Wv(ImgArray, c, c, Component, WV_LEFT_TO_RIGHT);
            // квантування
            for (int j = 0; j < c; j++)
            {
                for (int i = 0; i < c; i++)
                {
                    if ((i >= cw2) || (j >= c2))
                    {
                        Value = (short)Math.Round(ImgArray[Component, i, j]);
                        if (Value != 0)
                        {
                            int value2 = Value;
                            if (value2 < 0) { value2 = -value2; }
                            if (value2 < dwTop)
                            {
                                ImgArray[Component, i, j] = 0;
                            }
                            else
                            {
                                ImgArray[Component, i, j] = Value * dbDiv;
                            }
                        }
                    }
                }
            }
            return ImgArray;
}
```

Процедура швидкого ліфтингу дискретного біортогонального вейвлету, що використовується в вейвлет-перетворенні:

```
private double[, ,] Wv(double[, ,] ImgArray, int n, int dwCh,
                       int Component, int Side)
        {
            double a;
            int i, j, n2 = n / 2;
            double[] xWavelet = new double[n];
            double[] tempbank = new double[n];
            for (int dwPos = 0; dwPos < dwCh; dwPos++)
```

```
{
    if (Side == WV_LEFT_TO_RIGHT)
    {
        for (j = 0; j < n; j++) {
            xWavelet[j] = ImgArray[Component, dwPos, j];
        }
    }
    else if (Side == WV_TOP_TO_BOTTOM)
    {
        for (i = 0; i < n; i++) {
            xWavelet[i] = ImgArray[Component, i, dwPos];
        }
    }
    // Прогноз 1
    a = -1.586134342f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];
    // Оновлення 1
    a = -0.05298011854f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];
    // Прогноз 2
    a = 0.8829110762f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];
    // Оновлення 2
    a = 0.4435068522f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];
    // масштаб
    a = 1f / 1.149604398f;
    j = 0;
    // множимо непарні на коефіцієнт "a"
    // ділимо парні на коефіцієнт "a"
```

```
            if (Side == WV_LEFT_TO_RIGHT)
            {
                for (i = 0; i < n2; i++) {
                    ImgArray[Component, dwPos, i] = xWavelet[j++] / a;
                    ImgArray[Component, dwPos, n2 + i] = xWavelet[j++] * a;
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for (i = 0; i < n2; i++) {
                    ImgArray[Component, i, dwPos] = xWavelet[j++] / a;
                    ImgArray[Component, n2 + i, dwPos] = xWavelet[j++] * a;
                }
            }
        }
        return ImgArray;
    }
```

Опис розгортання вейвлету відбувається за допомогою наступного методу:

```
private void WUnPack(double[,,] ImgArray,
int Component, int c, int c, int dwDevider)
{
        int cw2 = c / 2, ch2 = c / 2;
        double dbDiv = 1f / dwDevider;
// деквантування значень
        for(int i = 0; i < c; i++)
        {
            for(int j = 0; j < c; j++)
            {
                if ((i >= cw2) || (j >= ch2))
                {
                    if (ImgArray[Component, i, j] != 0)
                    {
                        ImgArray[Component, i, j] /= dbDiv;
                    }
                }
            }
        }
        for(int i = 0; i < c; i++)
        {
            RA(ref ImgArray, c, Component, i, WV_LEFT_TO_RIGHT);
        }
```

```
            for(int j = 0; j < c; j++)
            {
                RA(ref ImgArray, c, Component, j, WV_TOP_TO_BOTTOM);
            }
}
```

Процедура зворотного швидкого ліфтингу дискретного біортогонального вейвлету, що застосовується в розгортанні вейвлету:

```
private void RA(ref double[, ,] shorts, int n, int z, int dwPos, int Side)
{
            double a;
            double[] xWavelet = new double[n];
            double[] tempbank = new double[n];
// Основні дії WV_LEFT_TO_RIGHT, WV_TOP_TO_BOTTOM
            if(Side == WV_LEFT_TO_RIGHT)
            {
                for(int j = 0; j < n; j++)
                {
                    xWavelet[j] = shorts[z, dwPos, j];
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for(int  i = 0; i < n; i++)
                {
                    xWavelet[i] = shorts[z, i, dwPos];
                }
            }
            for(int i = 0; i < n / 2; i++)
            {
                tempbank[i * 2] = xWavelet[i];
                tempbank[i * 2 + 1] = xWavelet[i + n / 2];
            }
            for(int i = 0; i < n; i++)
            {
                xWavelet[i] = tempbank[i];
            }
            // відмінити масштабування
            a = 1.149604398f;
            for(int  i = 0; i < n; i++)
            {
                if(i % 2 != 0)
```

```
                {
                    xWavelet[i] = xWavelet[i] * a;
                } else {
                    xWavelet[i] = xWavelet[i] / a;
                }
            }
// Повернути оновлення 2
            a = -0.4435068522f;
            for (int i = 2; i < n; i += 2)
            {
            xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];
// повернути прогнозування 2
            a = -0.8829110762f;
            for (int i = 1; i < n - 1; i += 2)
            {
            xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];
// Повернути оновлення 1
            a = 0.05298011854f;
            for (int i = 2; i < n; i += 2)
            {
             xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];
// повернути прогнозування 1
            a = 1.586134342f;
            for (int i = 1; i < n - 1; i += 2)
            {
            xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];
            if(Side == WV_LEFT_TO_RIGHT)
            {
                for (int j = 0; j < n; j++)
                {
                    shorts[z, dwPos, j] = xWavelet[j];
                }
            }
            else if(Side == WV_TOP_TO_BOTTOM)
```

```
            {
                for(int i = 0; i < n; i++)
                {
                    shorts[z, i, dwPos] = xWavelet[i];
                }
            }
}
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в

дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи, та включає в себе:

– Меню: Файл; Вид; Довідка.

– Функціональні кнопки: Відкрити; Стиснути; Зберегти.

– Розмір вхідного BMP-файлу.

– Розмір стиснутого файлу.

– Коефіцієнти Kicked/Zero та відсоток нульових значень пікселів при стисканні.

– Коефіцієнт стиснення (0-найвища якість).

– Підрозділ глибини. (Max – найбільше стиснення).

– Квантування бітів Y-UV (8, 8 – найвища якість).

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем.

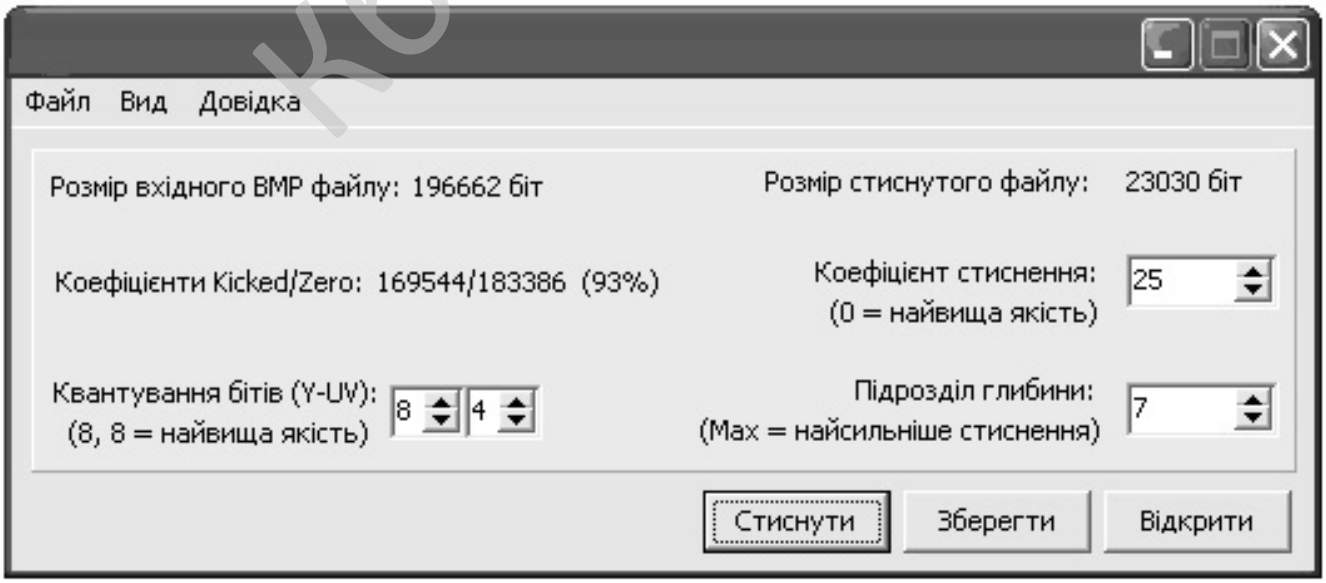

Рисунок 5.1 – Головне вікно ПЗ

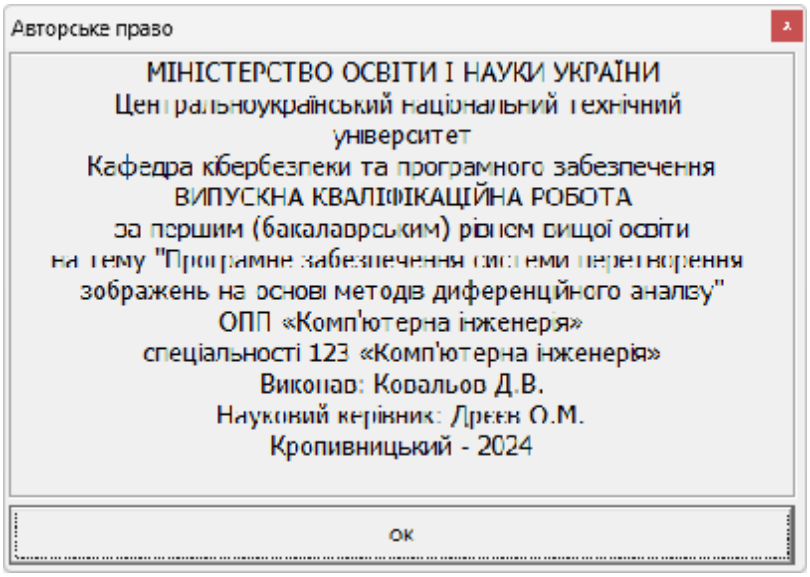

Рисунок 5.2 – Вікно довідки

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи перетворення зображень на основі методів диференційного аналізу.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем перетворення зображень на основі методів диференційного аналізу.

– Досліджена система перетворення зображень на основі методів диференційного аналізу.

– На основі отриманих результатів досліджень створена програмна реалізація системи перетворення зображень на основі методів диференційного аналізу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання перетворення зображень на основі методів диференційного аналізу.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи перетворення зображень на основі методів диференційного аналізу. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

2. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

3. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

4. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

5. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

6. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

7. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

8. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

9. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

11. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

12. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

13. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

14. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

15. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

16. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

17. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

18. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and

cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

19. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

20. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

21. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

22. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

23. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

24. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

25. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

26. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

27. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

28. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

29. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

30. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

31. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

32. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

33. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

34. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

35. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

36. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

37. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

38. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

39. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

40. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

41. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

42. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

47. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". - Випуск 2 (118). т.2. - Х.: ХУПС - 2014. - С. 64-67

51. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку ІТ-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. - 2014. - С. 240.

52. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

53. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи перетворення зображень на основі методів диференційного аналізу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи перетворення зображень на основі методів диференційного аналізу.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи перетворення зображень на основі методів диференційного аналізу;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 81 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення системи перетворення зображень на основі методів диференційного аналізу

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 25

Літера: РП

Кропивницький – 2024 року

```
// wvDecompress.cs - модуль, що реалізує декомпресію

using System;
using System.Collections.Generic;
using System.Text;

namespace WaveleteCompression
{
    class wvDecompress
    {

        // Константи
        public const int WV_LEFT_TO_RIGHT = 0;
        public const int WV_TOP_TO_BOTTOM = 1;

        public byte[] run(byte[] compressed)
        {
            int z;
            int dwDepth = 6; // кількість рівнів згортки вейвлету (чим їх
більше, тим краще стискається)
            // жорстко забиті розміри зображення
            int w = 512;
            int h = 512;
            // по суті розмір зображення й от цих коефіцієнтів повинні братися
із заголовку(header) стисненого файлу
            int[] dwDiv = { 48, 32, 16, 16, 24, 24, 1, 1 }, dwTop = { 24, 32,
24, 24, 24, 24, 32, 32 };
            int SamplerDiv = 2, YPerec = 100, crPerec = 85, cbPerec = 85;

            double[,,] yuv = doUnPack(compressed, w, h, dwDepth);

            // Розгорнення вейвлету
            for(z = 0; z < 2; z++)
            {
                for(int dWave = dwDepth - 1; dWave >= 0; dWave ---i)
                {
                    int w2 = Convert.ToInt32(w / Math.Pow(2, dWave));
                    int h2 = Convert.ToInt32(h / Math.Pow(2, dWave));
                    WaveleteUnPack(yuv, z, w2, h2, dwDiv[dWave] * SamplerDiv);
                }
            }
            z = 2;
            for(int dWave = dwDepth - 1; dWave >= 0; dWave ---i)
            {
                int w2 = Convert.ToInt32(w / Math.Pow(2, dWave));
                int h2 = Convert.ToInt32(h / Math.Pow(2, dWave));
                WaveleteUnPack(yuv, z, w2, h2, dwDiv[dWave]);
            }
            // YCrCb декодування + розкладання зображення в плоский масив
            byte[]  rgb_flatened = this.YCrCbDecode(yuv, w, h, YPerec, crPerec,
cbPerec);
            return rgb_flatened;
        }

        // Дана процедура є зворотньою процедурі DoPack у класі wvCompress.
        // Вона назад переводить його в (short) double-тип з типу byte[]
        private static double[, ,] doUnPack(byte[] Bytes, int c, int c, int
dwDepth)
        {
```

```
            int lPos = 0;
            byte Value;
            int intIndex = 0;
            // розмір підсумкового зображення в байтах
            int size = c * c * 3;
            // тимчасовий масив для результуючих коефіцієнтів згорнутого
вейвлету
            double[,,] ImgData = new double[3, c, c];

            int shortsLength = Bytes.Length - size;
            short[] shorts = new short[shortsLength / 2];
            Buffer.BlockCopy(Bytes, size, shorts, 0, shortsLength);

            for (int d = dwDepth - 1; d >= 0; d-d--)
            {
                int wSize = (int)Math.Pow(2, d);
                int W = c / wSize;
                int H = c / wSize;
                int w2 = W / 2;
                int h2 = H / 2;
                // лівий верхній
                if (d == dwDepth - 1)
                {
                    for (int z = 0; z < 3; z++)
                    {
                        for (int j = 0; j < h2; j++)
                        {
                            for (int i = 0; i < w2; i++)
                            {
                                Value = Bytes[lPos++];
                                if(Value == 255)
                                {
                                    ImgData[z, i, j] = shorts[intIndex++];
                                }
                                else
                                {
                                    ImgData[z, i, j] = Value - 127;
                                }
                            }
                        }
                    }
                }
                // верхній правий + нижній правий
                for (int z = 0; z < 3; z++)
                {
                    for (int j = 0; j < H; j++)
                    {
                        for (int i = w2; i < W; i++)
                        {
                            Value = Bytes[lPos++];
                            if(Value == 255)
                            {
                                ImgData[z, i, j] = shorts[intIndex++];
                            } else {
                                ImgData[z, i, j] = Value - 127;
                            }
                        }
                    }
                }
                // лівий нижній
```

```
                for (int z = 0; z < 3; z++)
                {
                    for (int j = h2; j < H; j++)
                    {
                        for (int i = 0; i < w2; i++)
                        {
                            Value = Bytes[lPos++];
                            if (Value == 255)
                            {
                                ImgData[z, i, j] = shorts[intIndex++];
                            }
                            else
                            {
                                ImgData[z, i, j] = Value - 127;
                            }
                        }
                    }
                }
            }
            // повертаємо результат
            return ImgData;
        }

        // Функція розгорнення вейвлету
        private void WaveleteUnPack(double[,,] ImgArray, int Component, int c,
int c, int dwDevider)
        {
            int cw2 = c / 2, ch2 = c / 2;
            double dbDiv = 1f / dwDevider;
            // деквантування значень
            for(int i = 0; i < c; i++)
            {
                for(int j = 0; j < c; j++)
                {
                    if ((i >= cw2) || (j >= ch2))
                    {
                        if (ImgArray[Component, i, j] != 0)
                        {
                            ImgArray[Component, i, j] /= dbDiv;
                        }
                    }
                }
            }
            // Розгорнення вейвлету
            for(int i = 0; i < c; i++)
            {
                reWv(ref ImgArray, c, Component, i, WV_LEFT_TO_RIGHT);
            }
            for(int j = 0; j < c; j++)
            {
                reWv(ref ImgArray, c, Component, j, WV_TOP_TO_BOTTOM);
            }
        }

        // Процедура зворотного швидкого ліфтингу дискретного біортогонального
CDF 9/7 вейвлету
        private void reWv(ref double[, ,] shorts, int n, int z, int dwPos, int
Side)
        {
```

```
            double a;
            double[] xWavelet = new double[n];
            double[] tempbank = new double[n];

            if(Side == WV_LEFT_TO_RIGHT)
            {
                for(int j = 0; j < n; j++)
                {
                    xWavelet[j] = shorts[z, dwPos, j];
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for(int  i = 0; i < n; i++)
                {
                    xWavelet[i] = shorts[z, i, dwPos];
                }
            }

            for(int i = 0; i < n / 2; i++)
            {
                tempbank[i * 2] = xWavelet[i];
                tempbank[i * 2 + 1] = xWavelet[i + n / 2];
            }
            for(int i = 0; i < n; i++)
            {
                xWavelet[i] = tempbank[i];
            }

            // відмінити масштабування
            a = 1.149604398f;
            for(int  i = 0; i < n; i++)
            {
                if(i % 2 != 0)
                {
                    xWavelet[i] = xWavelet[i] * a;
                } else {
                    xWavelet[i] = xWavelet[i] / a;
                }
            }

            // Повернути оновлення 2
            a = -0.4435068522f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];

            // повернути прогнозування 2
            a = -0.8829110762f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];

            // Повернути оновлення 1
```

```
            a = 0.05298011854f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];

            // повернути прогнозування 1
            a = 1.586134342f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];

            if(Side == WV_LEFT_TO_RIGHT)
            {
                for (int j = 0; j < n; j++)
                {
                    shorts[z, dwPos, j] = xWavelet[j];
                }
            }
            else if(Side == WV_TOP_TO_BOTTOM)
            {
                for(int i = 0; i < n; i++)
                {
                    shorts[z, i, dwPos] = xWavelet[i];
                }
            }
        }



        // Метод перекодування YCrCb в RGB
        private byte[] YCrCbDecode(double[, ,] yuv, int w, int h, double Ydiv,
double Udiv, double Vdiv)
        {
            byte[] bytes_flat = new byte[3 * w * h];
            double vr, vg, vb;
            double v, vCb, vCr;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            for(int j = 0; j < h; j++)
            {
                for (int i = 0; i < w ; i++)
                {
                    vCr = yuv[0, i, j] / Vdiv;
                    vCb = yuv[1, i, j] / Udiv;
                    v = yuv[2, i, j] / Ydiv;
                    vr = v + 1.402f * (vCr - 128f);
                    vg = v - 0.34414f * (vCb - 128f) - 0.71414f * (vCr - 128f);
                    vb = v + 1.722f * (vCb - 128f);
                    if (vr > 255) {vr = 255;}
                    if (vg > 255) {vg = 255;}
                    if (vb > 255) {vb = 255;}
                    if (vr < 0) {vr = 0;}
                    if (vg < 0) {vg = 0;}
```

```
                    if (vb < 0) {vb = 0;}
                    bytes_flat[j * w * 3 + i * 3 + 0] = (byte)vb;
                    bytes_flat[j * w * 3 + i * 3 + 1] = (byte)vg;
                    bytes_flat[j * w * 3 + i * 3 + 2] = (byte)vr;
                }
            }
            return bytes_flat;
        }

    }
}
```

// Program.cs - Основна програма

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Drawing.Imaging;


namespace WaveleteCompression
{
    class Program
    {
        static void Main(string[] args)
        {

            // Файл із зображенням для стиснення (реалізація алгоритму дозволяє
стискати
            // тільки квадратні зображення, розмір сторін у яких дорівнює
ступеню числа 2)
            string path =
"F:\\Projects\\WaveleteCompression\\WaveleteCompression\\bin\\Release\\test.bmp"
;

            // Копресор
            // Засікаємо час
            DateTime startTime = DateTime.Now;
            wvCompress c = new wvCompress();
            byte[] compressed = c.run(path);
            // Розрахунок витраченого часу
            TimeSpan duration = DateTime.Now - startTime;
            Console.Write("Compressed: ");
            Console.WriteLine(duration.Seconds * 1000 + duration.Milliseconds +
" ms");


            // Декомпресор
            // Засікаємо час
            startTime = DateTime.Now;
            wvDecompress d = new wvDecompress();
            byte[] decompressed = d.run(compressed);
            duration = DateTime.Now - startTime;
            Console.Write("Decompressed: ");
            Console.WriteLine(duration.Seconds * 1000 + duration.Milliseconds +
" ms");


            // Розпаковане зображення
            Bitmap bitmap1 = BytesToBitmap(decompressed);
            // Збереження розпакованого зображення
            bitmap1.Save(path + ".bmp", ImageFormat.Bmp);

            // Збереження в RAW без пост-стиску (для того, щоб можна було
поекспериментувати із пост-стиском)
            FileStream f = new System.IO.FileStream(path + ".bmp.raw",
FileMode.Create, FileAccess.Write);
```

```
            f.Write(decompressed, 0, decompressed.Length);
            f.Close();

            string ret = Console.ReadLine();

        }



        public unsafe static Bitmap BytesToBitmap(byte[] data)
        {
            Size size = new System.Drawing.Size(512, 512);
            GCHandle handle = GCHandle.Alloc(data, GCHandleType.Pinned);
            Bitmap bmp = new Bitmap(size.Width, size.Height, size.Width * 3,
PixelFormat.Format24bppRgb, handle.AddrOfPinnedObject());
            handle.Free();
            return bmp;
        }

    }
}
```

// wvCompress.cs - модуль, що реалізує компресію

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.IO.Compression;
using System.IO;

namespace WaveleteCompression
{

    class wvCompress
    {

        // Константи
        public const int WV_LEFT_TO_RIGHT = 0;
        public const int WV_TOP_TO_BOTTOM = 1;

        public byte[] run(string path)
        {

            // Завантажуємо зображення з файлу
            Bitmap bmp = new Bitmap(path, true);

            // Конвертуємо завантажене зображення в байтовий масив
            byte[, ,] b = this.BmpToBytes_Unsafe(bmp);

            // Застосування вейвлету
            byte[] o = this.Compress(b, bmp.Width, bmp.Height);

            // Збереження в RAW без пост-стиску
            FileStream f = new System.IO.FileStream(path + ".raw",
FileMode.Create, FileAccess.Write);
            f.Write(o, 0, o.Length);
            f.Close();

            //Попереднє стиснення отриманого масиву звичайним Gzip-ом і
збереження у файл
            // Якщо для стиснення використовувати щось інше замість GZIP, то
можна одержати файл розміром ще в 2 рази менше
            string outGZ = path + ".gz";
            FileStream outfile = new FileStream(outGZ, FileMode.Create);
            GZipStream compressedzipStream = new GZipStream(outfile,
CompressionMode.Compress, true);
            compressedzipStream.Write(o, 0, o.Length);
            compressedzipStream.Close();

            // повертаємо нестиснений GZip-ом масив
            return o;

        }

        private byte[] Compress(byte[, ,] rgb, int c, int c)
```

```
        {
            // Значення, для квантування коефіцієнтів вейвлету
            int[] dwDiv = { 48, 32, 16, 16, 24, 24, 1, 1 };
            int[] dwTop = { 24, 32, 24, 24, 24, 24, 32, 32 };
            int SamplerDiv = 2, SamplerTop = 2;
            // Відсотки квантування Y, cr, cb компонентів кольору
            int YPerec = 100, crPerec = 85, cbPerec = 85;
            int WVCount = 6; // кількість рівнів згортки вейвлету
            // Перекодування RGB в YCrCb
            double[, ,] YCrCb = YCrCbEncode(rgb, c, c, YPerec, crPerec, cbPerec,
c, c);
            // Застосовуємо вейвлет згортку по черзі до кожного каналу кольорів
            for (int z = 0; z < 3; z++)
            {
                // Кожний канал згортаємо вказану кількість разів
                for (int dWave = 0; dWave < WVCount; dWave++)
                {
                    int wave = Convert.ToInt32(c / Math.Pow(2, dWave));
                    int wave = Convert.ToInt32(c / Math.Pow(2, dWave));
                    if (z == 2)
                    {
                        // Канал з компонентом Y квантуємо на менше значення,
                        // так як у ньому лежить структура зображення (складова
яскравості), а в інших каналах дані про кольори
                        YCrCb = WaveletePack(YCrCb, z, wave, wave, dwDiv[dWave],
dwTop[dWave], dWave);
                    }
                    else
                    {
                        YCrCb = WaveletePack(YCrCb, z, wave, wave, dwDiv[dWave]
* SamplerDiv, dwTop[dWave] * SamplerTop, dWave);
                    }
                }
            }
            // конвертація масиву в одномірний
            byte[] flattened = doPack(YCrCb, c, c, WVCount);
            return flattened;

        }

        /* Процедура впаковує масив типу Double у масив типу Byte
        За рахунок наявності в масиві великої кількості значень, що поміщаються
в межі байту.
        На початку всі Double приводяться до типу Short.
        Потім значення, що не вміщаються в тип байт дописуються в кінець
вихідного потоку, а замість них у масив байтів
        записується значення 255 */
        private byte[] doPack(double[, ,] ImgData, int c, int c, int wDepth)
        {
            short Value;
            int lPos = 0;
            int size = c * c * 3;
            // резервування для short значень
            int intCount = 0;
            short[] shorts = new short[size];
            byte[] Ret = new byte[size];
            // прохід масиву поступово по вейвлет-рівнях
            for(int d = wDepth-1; d >= 0; d-d--)
            {
                int wSize = (int)Math.Pow(2f, Convert.ToDouble(d));
```

```
int W = c / wSize;
int H = c / wSize;
int w2 = W / 2;
int h2 = H / 2;
// лівий верхній кут
if (d == wDepth - 1)
{
    for (int z = 0; z < 3; z++)
    {
        for (int j = 0; j < h2; j++)
        {
            for (int i = 0; i < w2; i++)
            {
                Value = (short)Math.Round(ImgData[z, i, j]);
                if ((Value >= -127) && (Value <= 127))
                {
                    Ret[lPos++] = Convert.ToByte(Value + 127);
                }
                else
                {
                    Ret[lPos++] = 255;
                    shorts[intCount++] = Value;
                }
            }
        }
    }
}
// правий верхній + правий нижній
for (int z = 0; z < 3; z++)
{
    for (int j = 0; j < H; j++)
    {
        for (int i = w2; i < W; i++)
        {
            Value = (short)Math.Round(ImgData[z, i, j]);
            if ((Value >= -127) && (Value <= 127))
            {
                Ret[lPos++] = Convert.ToByte(Value + 127);
            }
            else
            {
                Ret[lPos++] = 255;
                shorts[intCount++] = Value;
            }
        }
    }
}
// лівий нижній
for (int z = 0; z < 3; z++)
{
    for (int j = h2; j < H; j++)
    {
        for (int i = 0; i < w2; i++)
        {
            Value = (short)Math.Round(ImgData[z, i, j]);
            if ((Value >= -127) && (Value <= 127))
            {
                Ret[lPos++] = Convert.ToByte(Value + 127);
            }
            else
```

```
                            {
                                Ret[lPos++] = 255;
                                shorts[intCount++] = Value;
                            }
                        }
                    }
                }
            }
            // склеювання двох масивів (byte[] і short[]) в один
            int shortArraySize = intCount * 2;
            Array.Resize(ref Ret, Ret.Length + shortArraySize);
            Buffer.BlockCopy(shorts, 0, Ret, Ret.Length - shortArraySize,
shortArraySize);
            // повертаємо результуючий плоский масив
            return Ret;
        }

        private double[, ,] WaveletePack(double[, ,] ImgArray, int Component,
int c, int c, int dwDevider, int dwTop, int dwStep)
        {
            short Value;
            int cw2 = c / 2;
            int c2 = c / 2;
            // підрахунок коефіцієнта квантування
            double dbDiv = 1f / dwDevider;
            ImgArray = Wv(ImgArray, c, c, Component, WV_TOP_TO_BOTTOM);
            ImgArray = Wv(ImgArray, c, c, Component, WV_LEFT_TO_RIGHT);
            // квантування
            for (int j = 0; j < c; j++)
            {
                for (int i = 0; i < c; i++)
                {
                    if ((i >= cw2) || (j >= c2))
                    {
                        Value = (short)Math.Round(ImgArray[Component, i, j]);
                        if (Value != 0)
                        {
                            int value2 = Value;
                            if (value2 < 0) { value2 = -value2; }
                            if (value2 < dwTop)
                            {
                                ImgArray[Component, i, j] = 0;
                            }
                            else
                            {
                                ImgArray[Component, i, j] = Value * dbDiv;
                            }
                        }
                    }
                }
            }
            return ImgArray;
        }

        // Швидкий ліфтинг дискретного біортогонального CDF 9/7 вейвлету
        private double[, ,] Wv(double[, ,] ImgArray, int n, int dwCh, int
Component, int Side)
        {

            double a;
```

```
int i, j, n2 = n / 2;
double[] xWavelet = new double[n];
double[] tempbank = new double[n];

for (int dwPos = 0; dwPos < dwCh; dwPos++)
{
    if (Side == WV_LEFT_TO_RIGHT)
    {
        for (j = 0; j < n; j++) {
            xWavelet[j] = ImgArray[Component, dwPos, j];
        }
    }
    else if (Side == WV_TOP_TO_BOTTOM)
    {
        for (i = 0; i < n; i++) {
            xWavelet[i] = ImgArray[Component, i, dwPos];
        }
    }

    // Прогноз 1
    a = -1.586134342f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }

    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];

    // Оновлення 1
    a = -0.05298011854f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];

    // Прогноз 2
    a = 0.8829110762f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];

    // Оновлення 2
    a = 0.4435068522f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];

    // масштаб
    a = 1f / 1.149604398f;
    j = 0;

    // множимо непарні на коефіцієнт "a"
    // ділимо парні на коефіцієнт "a"
    if (Side == WV_LEFT_TO_RIGHT)
    {
        for (i = 0; i < n2; i++) {
            ImgArray[Component, dwPos, i] = xWavelet[j++] / a;
            ImgArray[Component, dwPos, n2 + i] = xWavelet[j++] * a;
        }
```

```
                }
                else if (Side == WV_TOP_TO_BOTTOM)
                {
                    for (i = 0; i < n2; i++) {
                        ImgArray[Component, i, dwPos] = xWavelet[j++] / a;
                        ImgArray[Component, n2 + i, dwPos] = xWavelet[j++] * a;
                    }
                }

            }
            return ImgArray;
        }

        // Метод перекодування RGB в YCrCb
        private double[, ,] YCrCbEncode(byte[, ,] BytesRGB, int c, int c, double
Ydiv, double Udiv, double Vdiv, int o, int o)
        {
            double vr, vg, vb;
            double kr = 0.299, kg = 0.587, kb = 0.114, kr1 = -0.1687, kg1 =
0.3313, kb1 = 0.5, kr2 = 0.5, kg2 = 0.4187, kb2 = 0.0813;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            double[, ,] YCrCb = new double[3, c, c];
            for (int j = 0; j < o; j++)
            {
                for (int i = 0; i < o; i++)
                {
                    vb = (double)BytesRGB[0, i, j];
                    vg = (double)BytesRGB[1, i, j];
                    vr = (double)BytesRGB[2, i, j];
                    YCrCb[2, i, j] = (kr * vr + kg * vg + kb * vb) * Ydiv;
                    YCrCb[1, i, j] = (kr1 * vr - kg1 * vg + kb1 * vb + 128) *
Udiv;
                    YCrCb[0, i, j] = (kr2 * vr - kg2 * vg - kb2 * vb + 128) *
Udiv;
                }
            }
            return YCrCb;
        }



        private unsafe byte[, ,] BmpToBytes_Unsafe(Bitmap bmp)
        {
            BitmapData bData = bmp.LockBits(new Rectangle(new Point(),
bmp.Size), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            // кількість байтів в bitmap
            int byteCount = bData.Stride * bmp.Height;
            byte[] bmpBytes = new byte[byteCount];
            Marshal.Copy(bData.Scan0, bmpBytes, 0, byteCount); // Скопіювання
заблокованих байтів з пам'яті

             // Не забудьте відкрити bitmap!!
            bmp.UnlockBits(bData);
            byte[, ,] ret = new byte[3, bmp.Width, bmp.Height];
            for (int z = 0; z < 3; z++)
            {
                for (int i = 0; i < bmp.Width; i++)
```

```
                {
                    for (int j = 0; j < bmp.Height; j++)
                    {
                        ret[z, i, j] = bmpBytes[j * bmp.Width * 3 + i * 3 + z];
                    }
                }
            }
            return ret;
        }

    }
}
```

// Gzip.cs - робота з архівами

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Ionic.Zip;
using System.IO;

namespace archiverZIP
{
    public partial class Gzip : Form
    {
        public ZipFile zip;
        public FolderBrowserDialog saveDialog;
        public OpenFileDialog openFiles;
        public SaveFileDialog saveFile;
        public List<string> files = new List<string>();
        public string archive;

        public Gzip()
        {
            InitializeComponent();
        }

        private void buttonChooseFiles_Click(object sender, EventArgs e)
        {
            try
            {
                openFiles = new OpenFileDialog();
                openFiles.Title = "Виберіть файли, які необхідно заархівувати";

                if (openFiles.ShowDialog() == DialogResult.OK)
                {
                    files.AddRange(openFiles.FileNames);
                }
                else return;
            }
            catch (Exception ex) { MessageBox.Show("Помилка під час вибору
файлів для архівації, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonChooseArchive_Click(object sender, EventArgs e)
        {
            try
            {
                openFiles = new OpenFileDialog();
                openFiles.Title = "Виберіть архів, який необхідно
розархівувати";

                if (openFiles.ShowDialog() == DialogResult.OK)
                {
                    archive = openFiles.FileName;
                }
                else return;
            }
```

```
            catch (Exception ex) { MessageBox.Show("Помилка під час вибору
архіву для розархівація, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonSaveArchive_Click(object sender, EventArgs e)
        {
            try
            {
                string path = "";

                if(radioButton1.Checked)
                    path = Application.StartupPath + "\\Archive (" +
DateTime.Now.ToShortDateString() + ").zip";
                else if(radioButton2.Checked)
                    path = Directory.GetCurrentDirectory() + "\\Archive (" +
DateTime.Now.ToShortDateString() + ").zip";
                else if(radioButton3.Checked)
                {
                    saveFile = new SaveFileDialog();
                    saveFile.Title = "Збереження архіву";
                    saveFile.FileName = "Archive (" +
DateTime.Now.ToShortDateString() + ")";
                    saveFile.Filter = "Файл ZIP|*.zip";

                    if (saveFile.ShowDialog() == DialogResult.OK)
                    {
                        path = saveFile.FileName;
                    }
                }
                else throw new Exception("Не обране місце для збереження
архіву!");

                //Створюємо об'єкт для роботи з архівом
                using (zip = new ZipFile(path, Encoding.UTF8))
                {
                    //Установлюємо рівень стиснення
                    zip.CompressionLevel = Ionic.Zlib.CompressionLevel.Default;
                    //Задаємо системну директорію TEMP для тимчасових файлів
                    zip.TempFileFolder = System.IO.Path.GetTempPath();
                    //Додаємо файл і вказуємо де він буде розташовуватися в
архіві
                    foreach (string f in files)
                    {
                        zip.AddFile(f, "\\");
                    }
                    //Зберігаємо архів
                    zip.Save();
                    zip = null;
                }

                MessageBox.Show("Дані успішно збережені", "Інфо",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (Exception ex) { MessageBox.Show("Помилка при спроби
збереження архіву, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonSaveFiles_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                string path = "";

                if (radioButton6.Checked)
                    path = Application.StartupPath;
                else if (radioButton5.Checked)
                    path = Directory.GetCurrentDirectory();
                else if (radioButton4.Checked)
                {
                    saveDialog = new FolderBrowserDialog();
                    saveDialog.Description = "Виберіть папку для розархівації";

                    if (saveDialog.ShowDialog() == DialogResult.OK)
                    {
                        path = saveDialog.SelectedPath;
                    }
                }
                else throw new Exception("Не обране місце для розархівації
архіву!");

                //Створюємо об'єкт для роботи з архівом
                using (zip = new ZipFile(archive, Encoding.UTF8))
                {
                    //Задаємо системну директорію TEMP для тимчасових файлів
                    zip.TempFileFolder = System.IO.Path.GetTempPath();
                    //Додаємо файл і вказуємо де він буде розташовуватися в
архіві
                    zip.ExtractAll(path,
ExtractExistingFileAction.OverwriteSilently);
                    zip = null;
                }

                MessageBox.Show("Дані успішно витягнуті", "Інфо",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (Exception ex) { MessageBox.Show("Помилка при спроби
збереження архіву, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }
    }
}
```

// Gzip.Designer.cs - робота з архівами

```
namespace archiverZIP
{
    partial class Gzip
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>

        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>

        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>


        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.buttonSaveArchive = new System.Windows.Forms.Button();
            this.radioButton3 = new System.Windows.Forms.RadioButton();
            this.radioButton2 = new System.Windows.Forms.RadioButton();
            this.radioButton1 = new System.Windows.Forms.RadioButton();
            this.buttonChooseFiles = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.buttonSaveFiles = new System.Windows.Forms.Button();
            this.radioButton4 = new System.Windows.Forms.RadioButton();
            this.radioButton5 = new System.Windows.Forms.RadioButton();
            this.radioButton6 = new System.Windows.Forms.RadioButton();
            this.buttonChooseArchive = new System.Windows.Forms.Button();
            this.label2 = new System.Windows.Forms.Label();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.SuspendLayout();
```

```
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.buttonSaveArchive);
            this.groupBox1.Controls.Add(this.radioButton3);
            this.groupBox1.Controls.Add(this.radioButton2);
            this.groupBox1.Controls.Add(this.radioButton1);
            this.groupBox1.Controls.Add(this.buttonChooseFiles);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Location = new System.Drawing.Point(11, 16);
            this.groupBox1.Margin = new System.Windows.Forms.Padding(4);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Padding = new System.Windows.Forms.Padding(4);
            this.groupBox1.Size = new System.Drawing.Size(230, 240);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Архівація";

            //
            // buttonSaveArchive
            //
            this.buttonSaveArchive.Location = new System.Drawing.Point(71, 195);
            this.buttonSaveArchive.Name = "buttonSaveArchive";
            this.buttonSaveArchive.Size = new System.Drawing.Size(89, 29);
            this.buttonSaveArchive.TabIndex = 5;
            this.buttonSaveArchive.Text = "Зберегти";
            this.buttonSaveArchive.UseVisualStyleBackColor = true;
            this.buttonSaveArchive.Click += new
System.EventHandler(this.buttonSaveArchive_Click);

            //
            // radioButton3
            //
            this.radioButton3.AutoSize = true;
            this.radioButton3.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton3.Location = new System.Drawing.Point(10, 158);
            this.radioButton3.Name = "radioButton3";
            this.radioButton3.Size = new System.Drawing.Size(194, 19);
            this.radioButton3.TabIndex = 4;
            this.radioButton3.TabStop = true;
            this.radioButton3.Text = "Зберегти в обрану папку";
            this.radioButton3.UseVisualStyleBackColor = true;

            //
            // radioButton2
            //
            this.radioButton2.AutoSize = true;
            this.radioButton2.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton2.Location = new System.Drawing.Point(10, 132);
            this.radioButton2.Name = "radioButton2";
            this.radioButton2.Size = new System.Drawing.Size(184, 19);
            this.radioButton2.TabIndex = 3;
            this.radioButton2.TabStop = true;
            this.radioButton2.Text = "Зберегти в поточну папку";
            this.radioButton2.UseVisualStyleBackColor = true;
```

```
            //
            // radioButton1
            //
            this.radioButton1.AutoSize = true;
            this.radioButton1.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton1.Location = new System.Drawing.Point(10, 106);
            this.radioButton1.Name = "radioButton1";
            this.radioButton1.Size = new System.Drawing.Size(217, 19);
            this.radioButton1.TabIndex = 2;
            this.radioButton1.TabStop = true;
            this.radioButton1.Text = "Зберегти в папку із програмою";
            this.radioButton1.UseVisualStyleBackColor = true;

            //
            // buttonChooseFiles
            //
            this.buttonChooseFiles.Location = new System.Drawing.Point(127, 59);
            this.buttonChooseFiles.Name = "buttonChooseFiles";
            this.buttonChooseFiles.Size = new System.Drawing.Size(75, 23);
            this.buttonChooseFiles.TabIndex = 1;
            this.buttonChooseFiles.Text = "Огляд";
            this.buttonChooseFiles.UseVisualStyleBackColor = true;
            this.buttonChooseFiles.Click += new
System.EventHandler(this.buttonChooseFiles_Click);

            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label1.Location = new System.Drawing.Point(7, 32);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(210, 15);
            this.label1.TabIndex = 0;
            this.label1.Text = "Виберіть файл(ы) для архівації:";

            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.buttonSaveFiles);
            this.groupBox2.Controls.Add(this.radioButton4);
            this.groupBox2.Controls.Add(this.radioButton5);
            this.groupBox2.Controls.Add(this.radioButton6);
            this.groupBox2.Controls.Add(this.buttonChooseArchive);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Location = new System.Drawing.Point(253, 16);
            this.groupBox2.Margin = new System.Windows.Forms.Padding(4);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Padding = new System.Windows.Forms.Padding(4);
            this.groupBox2.Size = new System.Drawing.Size(230, 240);
            this.groupBox2.TabIndex = 1;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Розархівація";
```

```
            //
            // buttonSaveFiles
            //
            this.buttonSaveFiles.Location = new System.Drawing.Point(71, 195);
            this.buttonSaveFiles.Name = "buttonSaveFiles";
            this.buttonSaveFiles.Size = new System.Drawing.Size(89, 29);
            this.buttonSaveFiles.TabIndex = 9;
            this.buttonSaveFiles.Text = "Зберегти";
            this.buttonSaveFiles.UseVisualStyleBackColor = true;
            this.buttonSaveFiles.Click += new
System.EventHandler(this.buttonSaveFiles_Click);

            //
            // radioButton4
            //
            this.radioButton4.AutoSize = true;
            this.radioButton4.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton4.Location = new System.Drawing.Point(8, 158);
            this.radioButton4.Name = "radioButton4";
            this.radioButton4.Size = new System.Drawing.Size(194, 19);
            this.radioButton4.TabIndex = 8;
            this.radioButton4.TabStop = true;
            this.radioButton4.Text = "Зберегти в обрану папку";
            this.radioButton4.UseVisualStyleBackColor = true;

            //
            // radioButton5
            //
            this.radioButton5.AutoSize = true;
            this.radioButton5.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton5.Location = new System.Drawing.Point(8, 132);
            this.radioButton5.Name = "radioButton5";
            this.radioButton5.Size = new System.Drawing.Size(184, 19);
            this.radioButton5.TabIndex = 7;
            this.radioButton5.TabStop = true;
            this.radioButton5.Text = "Зберегти в поточну папку";
            this.radioButton5.UseVisualStyleBackColor = true;

            //
            // radioButton6
            //
            this.radioButton6.AutoSize = true;
            this.radioButton6.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton6.Location = new System.Drawing.Point(8, 106);
            this.radioButton6.Name = "radioButton6";
            this.radioButton6.Size = new System.Drawing.Size(217, 19);
            this.radioButton6.TabIndex = 6;
            this.radioButton6.TabStop = true;
            this.radioButton6.Text = "Зберегти в папку із програмою";
            this.radioButton6.UseVisualStyleBackColor = true;
```

```
            //
            // buttonChooseArchive
            //
            this.buttonChooseArchive.Location = new System.Drawing.Point(128,
59);
            this.buttonChooseArchive.Name = "buttonChooseArchive";
            this.buttonChooseArchive.Size = new System.Drawing.Size(75, 23);
            this.buttonChooseArchive.TabIndex = 2;
            this.buttonChooseArchive.Text = "Огляд";
            this.buttonChooseArchive.UseVisualStyleBackColor = true;
            this.buttonChooseArchive.Click += new
System.EventHandler(this.buttonChooseArchive_Click);

            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label2.Location = new System.Drawing.Point(5, 32);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(215, 15);
            this.label2.TabIndex = 1;

            this.label2.Text = "Виберіть архів для розархівації:";

            //
            // Gzip
            //
            this.AutoScaleDimensions = new System.Drawing.Size(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.White;
            this.ClientSize = new System.Drawing.Size(494, 272);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Margin = new System.Windows.Forms.Padding(4);
            this.MaximizeBox = false;
            this.Name = "Gzip";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Archiver ZIP";
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.ResumeLayout(false);

        }
```

```
        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Button buttonSaveArchive;

        private System.Windows.Forms.RadioButton radioButton3;
        private System.Windows.Forms.RadioButton radioButton2;
        private System.Windows.Forms.RadioButton radioButton1;

        private System.Windows.Forms.Button buttonChooseFiles;

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Button buttonSaveFiles;

        private System.Windows.Forms.RadioButton radioButton4;
        private System.Windows.Forms.RadioButton radioButton5;
        private System.Windows.Forms.RadioButton radioButton6;

        private System.Windows.Forms.Button buttonChooseArchive;
        private System.Windows.Forms.Label label2;
    }
}
```