МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи

_____ бакалавр _____

(освітньо-кваліфікаційний рівень)

на тему «Розробка програмного засобу для кольорокорекції зображень»

Виконав: групи ІПЗ-15д
напряму підготовки 121 „ Інженерія програмного забезпечення ”
_____ Хлєбко А.С.
(підпис)

Керівник,
доцент, д.т.н._____ Лифар В.О.
(підпис)

Рецензент,
к.т.н., доцент _____ Митрохін С.О.
(підпис)

СЄВЄРОДОНЕЦЬК
2019 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 „ Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____Лифар В.О.
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
ХЛЄБКО АНДРІЙ СЕРГІЙОВИЧ

1. Тема роботи Розробка програмного засобу для кольорокорекції зображень
керівник роботи доцент Лифар Володимир Олексійович
2. Строк подання студентом роботи 06 червня 2019 р.
3. Вихідні дані до роботи

Об'єктом даної роботи є Методи класифікації і прогнозування інтелектуального аналізу даних на прикладі виявлення мережевих атак.

3.1 Літературні джерела:

Стругайло В. В. Огляд методів фільтрації і сегментації цифрових зображень .// [Електронний ресурс].
URL: http: //cyberleninka.ru/article/n/obzor-metodov-filtratsii-i-segmentatsiitsifrovyh-izobrazheniy
Алгоритмічні основи растрової графіки // [Електронний ресурс]. URL: http://www.intuit.ru/studies/courses/993/163/lecture/4505
Методи і алгоритми фільтрації зображень // [Електронний ресурс]. URL: http://www.mm-dsp.com/2011-09-09-09-27-20/2011-09-0912-16-15.html#2

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   - 4.1 Вступ
   - 4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:
     - Методи отримання даних про атаки.
     - Методи аналізу даних
   - 4.3 Основна частина, в якої висвітлити:
     - Інформаційна модель об’єкту.
     - Реалізація проекта
   - 4.4 Висновки
   - 4.5 Перелік використаних джерел
5. Перелік графічного матеріалу немає
6. Дата видачі завдання 02 лютого 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	01.02.19	
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.02.19	
3	Узагальнення даних літературних джерел, укладання першого розділу	1.03.19	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.03.19	
5	Проектування інфологічної моделі задачі що реалізується.	01.04.19	
6	Укладання та тестування програмного продукту	20.04.19	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	15.05.19	
8	Здача готової пояснювальної записки на кафедру	06.06.19	
9	Укладання доповіді і презентації	10.06.19	

Студент _____ Хлєбко А.С..
(підпис)

Керівник роботи _____ Лифар В.О.
(підпис)

РЕФЕРАТ

Текст – 65., рис. – 6, табл. – 6, додатків – 1, літературних джерел – 12

Розробка програмного забезпечення, що реалізує алгоритми фільтрації зображень шляхом просторового кодування і декодування зображень на основі рекурсивного розбиття на полігони.

Об'єктом розробки є кольорові і чорно-білі зображення різного дозволу і формату.

Мета роботи - розробити програмне забезпечення, що дозволяє отримати зображення максимально наближене до вихідного шляхом застосування алгоритму розподілу зображення на полігони, а також тріангуляції. У вступі обгрунтовується актуальність обраної теми і вказується основна мета роботи. Перший розділ присвячений огляду основних фільтрів зображень і методам обробки зображень. Другий розділ присвячений опису етапів розробки алгоритму розбиття і структуризації зображень. Третій розділ присвячений розробці та тестуванню Win-додатки для моделювання алгоритму обробки зображень. Четвертий розділ присвячений передпроектної оцінки вартості і стандартизації розробленого програмного засобу. У висновку підводяться підсумки виконаної роботи.

ОБРОБКА ЗОБРАЖЕНЬ, КОЛЬОРОКОРЕКЦІЯ, ФІЛЬТРИ ЗОБРАЖЕНЬ, АЛГОРИТМИ РОЗБИТТЯ, СТРУКТУРИЗАЦІЯ ЗОБРАЖЕНЬ

ЗМІСТ

ВСТУП

Ще в середині XX століття обробка зображень була в основному аналогової і проводилася на оптичних пристроях. Схожі способи обробки зображень важливі і донині, наприклад, в такій області як голографія. Однак, з інтенсивним зростанням продуктивності обчислювальних пристроїв, дані методи втратили свою актуальність у зв'язку з появою методів цифрової обробки зображень. Методи комп'ютерної або цифрової обробки зображень мають ряд переваг в порівнянні з застарілими аналоговими методами, а саме: велика точність, надійність, гнучкість і простота в реалізації. При обробці зображень часто застосовується спеціалізоване і високопродуктивне обладнання, наприклад, процесори з конвеєрної обробкою інструкцій і багатопроцесорні системи. [1]

Обробка зображень являє собою сімейство всіх методів, завдань і алгоритмів, які виконуються по відношенню до зображень.

Виконувати обробку зображень слід по ряду причин, таких як:

- поліпшення зображення для сприйняття людиною (збільшення візуалізації і швидкості засвоєння інформації)
- поліпшення зображення для сприйняття комп'ютером (видалення шумів)
- рішення практичних завдань (виявлення об'єктів, розпізнавання і аналіз тексту, ідентифікація особистості)
- спецефекти (отримати естетичне задоволення від красивого ефекту, а також прагматичне використання, пов'язане з відео)

Цифрова обробка зображень є активно розвиваються напрямком в науці. Актуальність завдання обробки зображень полягає в проведених дослідженнях, що розробляються алгоритми і методи обробки та аналізу інформації різного роду, представленої у вигляді зображень.

В результаті виконання даної роботи має бути обгрунтоване рішення про вибір конкретного методу обробки зображень, його переваги перед іншими просторовими методами. У зв'язку з цим так само буде реалізована програма, що виконує моделювання пірамідально-рекурсивного методу кодування і декодування зображення в просторової області.

РОЗДІЛ 1. ОГЛЯД ФІЛЬТРІВ І МЕТОДІВ ОБРОБКИ ЗОБРАЖЕНЬ

1.1. Порівняння лінійних і нелінійних фільтрів зображень.

Незалежно від того, чи використовується видалення шуму, правильний вибір між лінійним або нелінійним фільтром для моделювання методів обробки зображень може бути різницею між отриманням зображення.

Історично, обробка в реальному часі або вбудованого зображення була обмежена з точки зору складності через обмеження витрат / потужності лежить в основі процесора. Однак, з сьогоднішніми потужними обчислювальними системами, можна розглянути складні методи фільтрації, які до сих пір могли бути виконані тільки при аналогової обробки даних зображень. Вивчення відмінностей між лінійними і нелінійними фільтрами може допомогти розробникам впровадити найбільш ефективну технологію фільтрації для виявлення і управління інформацією про зображення.

Фільтрація в обробці зображень - це основна функція, яка використовується для виконання багатьох завдань, включаючи інтерполяцію, придушення шуму і повторну вибірку. Вибір фільтра часто визначається характером завдання, типом і поведінкою даних. Шум, динамічний діапазон, точність кольору, оптичні артефакти і багато іншого впливають на результат функцій фільтра при обробці зображень.

У наступному порівняльному аналізі будуть розглянуті відмінності між двома основними категоріями фільтрації - лінійними і нелінійними, а також основні підходи до обробки зображень, які отримують вигоду від цих типів фільтрів, і виявляють ситуації, коли один фільтр може бути кращим або необхідним в порівнянні з іншим.

1.1.1. Теорія фільтрів

При обробці зображень методи 2D-фільтрації зазвичай розглядаються як розширення теорії обробки 1D-сигналів. Практично вся сучасна обробка зображень

включає в себе дискретну або вибіркову обробку сигналів. Це порівняно з обробкою сигналів, яка була застосована до аналогової або безперервній обробці в тимчасовій області, яка характеризувала телебачення і відео кілька поколінь назад. Обидва вони взаємопов'язані, і основа для обробки дискретного сигналу виходить з безперервної теорії обробки сигналів часу. [2]

1.1.2. лінійні фільтри

Щоб переглянути та порівняти два типи фільтрації, першим кроком є короткий опис атрибутів, які містять лінійну фільтрацію.

Кілька принципів визначають лінійну систему. Перші два є основними визначеннями лінійності:

- Якщо система задає вхідний сигнал як x [n] = ax [n1] + bx [n2], тоді лінійний відгук системи y [n] = ay [n1] + by [n2]. Це відомо як властивість суперпозиції і має фундаментальне значення для проектування лінійної системи.
- Второе свойство - сдвиговая инвариантность. Если y [n] является ответом на линейную, сдвигово-инвариантную систему с входом x [n], то y [n-n0] является ответом на систему с входом x [n-n0].

Крім того, накладаються дві додаткові умови, причинні і стабільні. Каузальне умова необхідно при розгляді систем, в яких майбутні значення невідомі (наприклад, при потокової передачі відео). Можна розглянути систему, яка не є причинного при перегляді знімків із зразками до і після цільового місця розташування (наприклад, в буферизує версії кадру зображення). Стабільність накладена для того, щоб вихід фільтра перевищував кінцевий межа, з огляду на вхід, який також не перевищує кінцевий межа. Це називається умовою обмеженого вхідного обмеження (Уово). [3]

У більшості випадків система оцінюється в просторової частотної області. Для цього використовується теорема згортки, що надає необхідні інструменти для оцінки інформації про частотної області.

Якщо x [n] і h [n] - дві послідовності, їх згортка визначається, виразом 1.1:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{1.1}$$

Відповідна частотна характеристика показана в рівнянні 1.2:

$$Y(e^{-i\omega}) = X(e^{-i\omega})H(e^{-i\omega}) \tag{1.2}$$

У рівнянні 2, e-iw - уявлення частотної області, а w – частотна змінна від -p до p. Це фундаментальне співвідношення описує реакцію фільтра по частоті - низький прохід, високий прохід, смуговий прохід і т. Д. В залежності від характеру ядра фільтра h [n] для будь-якого набору даних зображення може бути реалізована велика кількість відповідей.

Типовий фільтр нижніх частот з 25 відводами (h [0..24]) показаний на рисунку 1.1.2.1. Ідея фільтра нижніх частот полягає в збереженні низькочастотної інформації та зменшенні або виключення високочастотної інформації на зображенні. Фільтр розмиває краї, але зберігає гладкі ділянки зображення неушкодженими. Аналогічним чином фільтри верхніх частот зберігають ребра і іншу високочастотну інформацію, але фільтрують низькочастотні області зображення. [5]

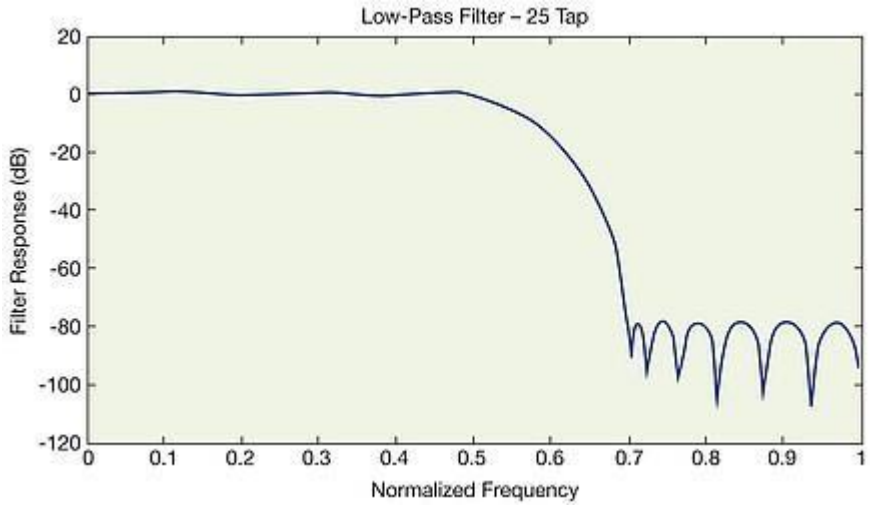

Рис. 1.1.2.1. Фільтр нижних частот.

1.1.3. Нелінійні фільтри

Нелінійні фільтри мають абсолютно іншу поведінку в порівнянні з лінійними фільтрами. Для нелінійних фільтрів вихід фільтра або відповідь фільтра не підкоряються принципам, викладеним раніше, зокрема, до масштабування і сдвиговой інваріантності. Крім того, нелінійний фільтр може давати результати, які змінюються неінтуітівнимі.

Найпростішим нелінійним фільтром, який слід розглянути, є медіанний або ранжуються фільтр. У медіанному фільтрі вихід фільтра залежить від порядку введення вхідних значень, зазвичай від молодшого до найбільшого або навпаки. Використовується діапазон підтримки фільтра з непарною кількістю значень, що спрощує вибір виведення.

Наприклад, припустимо, що фільтр був заснований на п'яти значеннях. У цікавій для області x0..x4 значення впорядковані від найменшого до найбільшого. Як висновок вибирається значення в позиції 2. Розглянемо випадок на низькій частоті: все значення однакові або близькі до нього. В цьому випадку вибране значення буде вихідним значенням ± деяка невелика помилка. У разі високої частоти, такий як

край, значення на одній стороні краю будуть низькими, а значення з іншого боку будуть високими. Коли запит буде виконаний, низькі значення будуть як і раніше перебувати в найнижчому положенні, а високі значення будуть як і раніше перебувати в верхньому положенні. Вибір середнього значення буде або на нижньому боці, або на високій стороні, але не на середині, як у випадку лінійного фільтра нижніх частот. Через це властивості серединний фільтр іноді називають фільтром збереження краю. Це корисно при видаленні викидів, таких як імпульсний шум. [4]

1.1.4. Вибір правильного фільтра

Обидва типи фільтрів мають місце у функціях обробки зображень. У типовому конвеєрі для обробки зображень в реальному часі нерідко доводиться мати десятки обох типів, включених для формування, виявлення та управління інформацією про зображення. Більш того, кожен з цих типів фільтрів може бути параметризрвані, щоб працювати в одному випадку при певних обставинах і іншим способом при інших умовах, використовуючи генерацію правила адаптивного фільтра.

Фільтрація даних зображення - це стандартний процес, який використовується практично у всіх системах обробки зображень. Цілі варіюються від видалення шуму до абстракції об'єктів. Лінійні і нелінійні фільтри є двома найбільш використовуваними формами конструкції фільтра. Щоб визначитися який тип фільтра вибрати, потрібно розглянути цілі, для яких планується використовувати фільтр, а також характер даних зображення. У тих випадках, коли вхідні дані містять великий обсяг шуму, але величина низька, може бути досить лінійного фільтра нижніх частот. І навпаки, якщо зображення містить малу кількість шуму, але з

відносно великою, то середній фільтр може бути більш доречним. У будь-якому випадку процес фільтрації змінює загальне частотне зміст зображення.

1.3. Просторові методи

Просторова область - це безліч пікселів, що становлять цифрове зображення. Процедури просторової обробки описуються загальним рівнянням $g(x,y) = T[f(x,y)]$ де $f(x,y)$ - вхідне цифрове зображення, $g(x,y)$ - обернене, а T – оператор над f , визначений у деякому околі точки (x,y) , для якої ця точка є центром. Центр міста пересувається від пікселя до пікселя, починаючи з верхнього лівого кута.

Оператор T виконується для кожної точки (x,y), що дає в результаті вихідне значення g для даної точки. Процес використовує тільки пікселі всередині області цифрового зображення, обмеженою околицею.

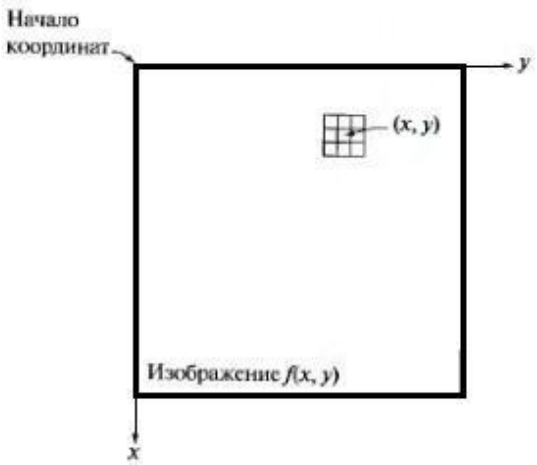

Рис.1.3.1. Окрестність 3*3 навколо точки (x,y) цифрового зображення

Збільшення розмірів околиці призводить до значно більшої гнучкості процесу обробки. Один з основних підходів тут базується на використанні так званих масок (фільтрів, ядер, шаблонів або вікон). Найчастіше маска являє собою

невеликий (наприклад, 3 * 3 елементи) двовимірний масив, значення коефіцієнтів маски всередині якого визначають істота процесу. Методи поліпшення, що базуються на такому підході, часто називають обробкою по масці або фільтрацією по масці.

Один з найбільш часто використовуваних способів обробки зображення, яка виконується з різними цілями, є просторова фільтрація. Просторова фільтрація - фільтрація, яка виконується безпосередньо над елементами зображення. Схема просторової фільтрації представлена на рис.1.3.2. Контроль здійснюється шляхом простому переміщенні маски фільтра від точки до точки зображення; в кожній точці відгук фільтра обчислюється з використанням попередньо заданих зв'язків. [5]

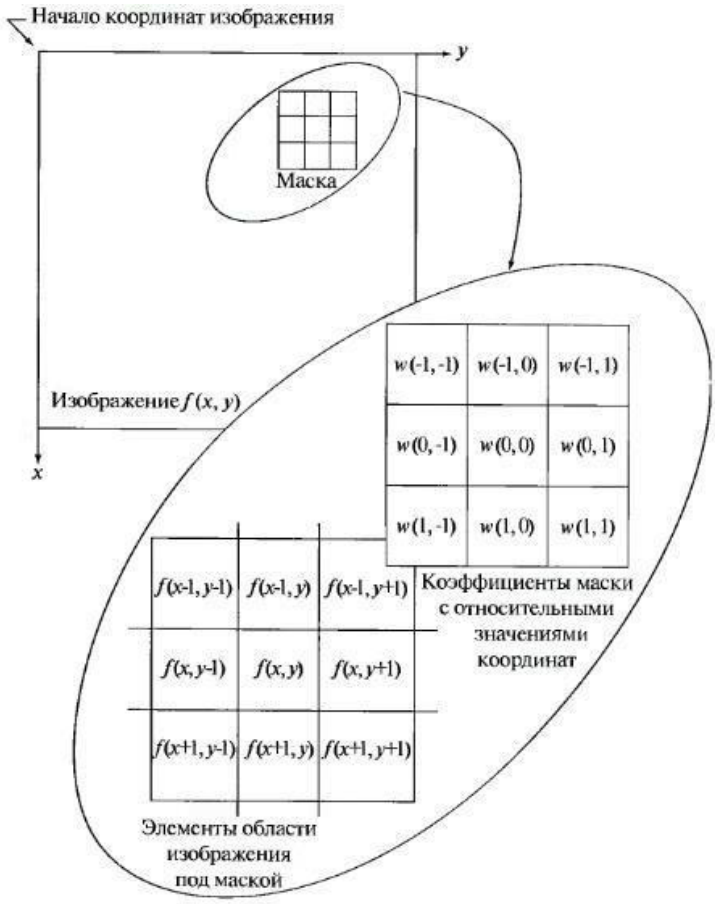

Рис.1.3.2. Схема просторової фільтрації

У разі лінійної просторової фільтрації відгук задається сумою добутків коефіцієнтів фільтра на відповідні значення пікселів в області, покритої маскою. Для маски 3 * 3 (рис.1.6), результат (відгук) R лінійної фільтрації в точке (x, y) обчислюється як

$$R = w(-1,-1) f(x-1, y-1) + w(-1,0) f(x-1, y) + ... + w(0,0) f(x, y) + ... + w(1,0) f(x+1, y) +$$

$$+ w(1,1) f(x+1, y+1)$$

1.3.1. Дискретне перетворення Фур'є

ДПФ є найбільш важливим дискретним перетворенням, використовуваним для виконання аналізу Фур'є в багатьох практичних додатках. У цифровій обробці сигналів функція являє собою будь-яку кількість або сигнал, який змінюється з часом, наприклад тиск звукової хвилі, радіосигнал або щоденні показники температури, відібрані за кінцевий часовий інтервал. При обробці зображень вибірки можуть являти собою значення пікселів уздовж рядка або стовпця растрового зображення. ДПФ також використовується для ефективного вирішення рівнянь в приватних похідних і для виконання інших операцій, таких як згортки або множення великих цілих чисел.

Оскільки метод має справу з кінцевим об'ємом даних, він може бути реалізований на комп'ютерах за допомогою числових алгоритмів або навіть виділеного обладнання. Ці реалізації зазвичай використовують

ефективні швидкі алгоритми перетворення Фур'є. [5]

1.3.2. Дискретне вейвлет-перетворення

Вейвлети часто використовуються для візуалізації двовимірних сигналів, таких як зображення. У наступному прикладі наведені три кроки для видалення небажаних білих гауссовских шумів з показаного шуму.

Перший крок - вибрати тип вейвлета і рівень N розкладання. В цьому випадку біортогональні 3,5 сплеску були обрані з рівнем N в 10. біортогональних сплески зазвичай використовуються при обробці зображень для виявлення і фільтрації білого гауссовского шуму через їх високої контрастності значень інтенсивності сусідніх пікселів. Використовуючи ці сплески, на двовимірному зображенні виконується вейвлетпреобразованіе.

Після декомпозиції файлу зображення наступним кроком є визначення порогових значень для кожного рівня від 1 до N. Стратегія Біргіт-Массарта є досить поширеним методом вибору цих порогових значень. Використовуючи цей процес, індивідуальні порогові значення для N = 10 рівнів. Застосування цих порогових значень є здебільшого фактичної фільтрації сигналу.

Останній крок - відновити зображення з змінених рівнів. Це виконується з використанням інверсного вейвлет-перетворення. Отримане зображення з віддаленим гауссовским шумом показано нижче вихідного зображення. При фільтрації будь-якої форми даних важливо кількісно оцінити ставлення сигнал / шум до результату. В цьому випадку СКО шумового зображення в порівнянні з оригіналом становило 30,4958%, а СКО зменшеного зображення становило 32,5525%.

Результатом поліпшення фільтрації вейвлета є посилення СКО 2.0567%.

Важливо відзначити, що вибір інших стратегій сплесків, рівнів і порогових значень може призводити до різних типів фільтрації. У цьому прикладі був обраний білий гауссовский шум. Хоча, з різним граничним значенням, він просто міг бути легко посилений. [4]

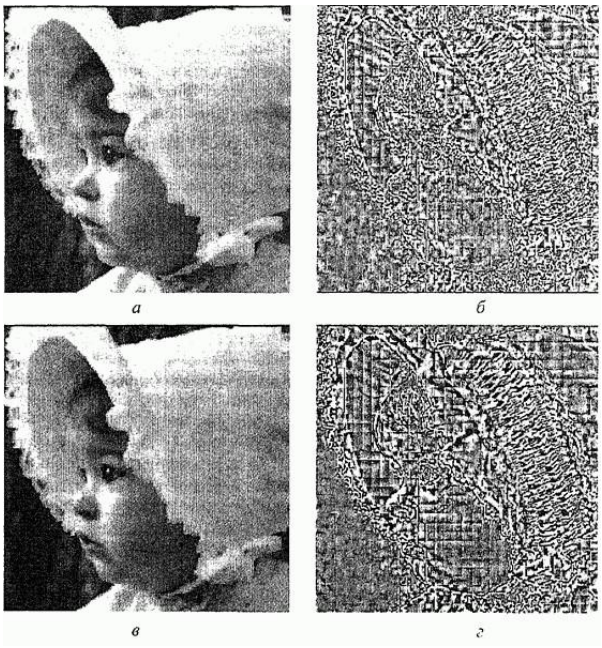

Рис. 1.3.2.1. Стиснення зображення із застосуванням ДВП: а - відтворене зображення, коефіцієнт стиснення 4.2, б - різницеве зображення - різниці між яркостями початкового і відновленого зображень; в - відтворене зображення, коефіцієнт стиснення 5.6, г – разностное зображення між вихідним і відновленим зображеннями.

1.3.3. Дискретне косинусное перетворення

Як і будь-який пов'язаний з Фур'є перетворення, дискретні косинусні перетворення (ДКП) висловлюють функцію або сигнал у вигляді суми синусоїд з різними частотами і амплітудами. Подібно дискретного перетворення Фур'є (ДПФ), ДКП працює на функції з кінцевим числом дискретних точок даних. Очевидна відмінність між ДКП і ДПФ полягає в тому, що перший використовує тільки функції косинуса, а останній використовує як косинуси, так і синуси (у вигляді комплексних експонент). Однак це видиме відмінність є просто наслідком більш глибокого відмінності: ДКП має на увазі різні граничні умови від ДПФ або інших пов'язаних перетворень.

Фур'є-пов'язані перетворення, які працюють над функцією над кінцевою областю, такі як ДПФ або ДКП або ряд Фур'є, можна розглядати як неявне визначення розширення цієї функції поза домену. Тобто, як тільки ви напишете функцію f (x) як суму синусоидов, ви можете оцінити цю суму в будь-якому x, навіть для x, де орігіналльная f (x) не визначена. ДПФ, як і ряд Фур'є, має на увазі періодичне розширення вихідної функції. ДКП, як і косинусное перетворення, має на увазі парне розширення вихідної функції.

Ілюстрація неявних парних / непарних розширень вхідних даних ДКП (DCT) для N = 11 точок даних (червоні точки) для чотирьох найбільш поширених типів DCT (типи I-IV) представлена на рисунку 1.3.3.1:

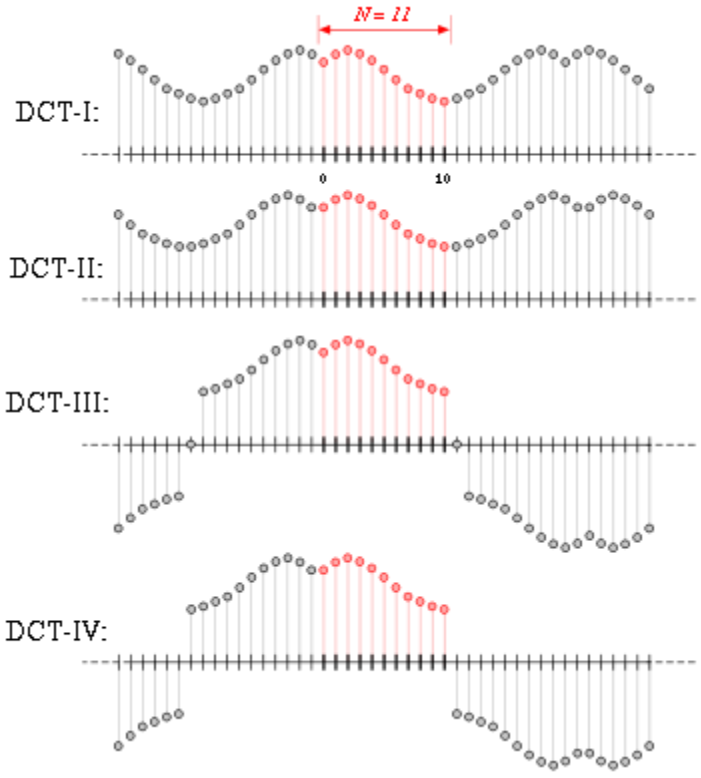

Рис. 1.3.3.1. Порівняння різних типів ДКП.

Однак, оскільки ДКП працюють на кінцевих дискретних послідовностей, виникають два питання, які не застосовуються для безперервного косинусного перетворення.

По-перше, потрібно вказати, чи є функція парній або непарній як на лівій, так і на правій межах області.

По-друге, потрібно вказати, в якій точці функція парна або непарна. Зокрема, розглянемо послідовність abcd з чотирьох однаково розподілених точок даних і скажемо, що ми вказуємо парну ліву кордон. Є дві розумні можливості: або дані відносяться до зразка a, і в цьому випадку парне розширення є dcbabcd, або дані про точку на півдорозі між a і попередньої точкою, і в цьому випадку парне розширення dcbaabcd (a повторюється).

Ці вибори призводять до всіх стандартних варіацій ДКП, а також до дискретних синусоїдальним перетворенням (ДСП). Кожна межа може бути або парної, або непарної (2 варіанти на кордон) і може бути симетричною відносно точки даних або точки на півдорозі між двома точками даних (2 варіанти на кордон), в цілому $2 \times 2 \times 2 \times 2 = 16$ можливості . Половина цих можливостей, ті, де ліва межа парна, відповідають 8 типам ДКП; Інша половина - це 8 типів ДСП.

Ці різні граничні умови сильно впливають на додатки перетворення і призводять до унікальних корисних властивостях для різних типів ДСП. Найбільш безпосередньо при використанні перетворень Фур'є для вирішення рівнянь в приватних похідних спектральними методами граничні умови безпосередньо визначаються як частина розв'язуваної задачі. Граничні умови відповідають за властивості «енергетичної компактификации», які роблять ДКП корисними для стиснення зображень і звуку, оскільки кордону впливають на швидкість збіжності будь-яких Фур'є-подібних рядів.

Зокрема, добре відомо, що будь-які розриви в функції зменшують швидкість збіжності рядів Фур'є, так що для представлення функції з заданою точністю потрібно більше синусоид. Той же принцип визначає корисність ДПФ і інших перетворень для стиснення сигналу; Чим більше гладка функція, тим менше членів в

її ДПФ або ДКП повинні представляти її точно, і тим більше вона може бути стиснута.

Ось чому ДКП і, зокрема, ДКП типів I, II, V і VI (типи, які мають дві парні кордону), як правило, краще працюють для стиснення сигналу, ніж ДПФ і ДСП. На практиці, ДКП типу II зазвичай краще для таких додатків, частково з міркувань обчислювального зручності. [5]

1.4. частотні методи

Поліпшення зображення в частотній області є лінійним. Ми просто обчислюємо перетворення Фур'є зображення, яке потрібно посилити, множимо результат на фільтр (а не згортаємо в просторової області) і виконуємо зворотне перетворення для створення розширеного зображення.

Ідея розмиття зображення шляхом зменшення його високочастотних компонентів або заточування зображення за рахунок збільшення величини його високочастотних компонентів інтуїтивно зрозуміла. Однак, в обчислювальних цілях, часто більш ефективно виконувати ці операції як згортки за допомогою невеликих просторових фільтрів в просторової області. Розуміння концепції частотної області важливо і призводить до методів вдосконалення, про які, можливо, не думали, обмежуючи увагу просторової областю. [2]

1.4.1. Фільтрація в частотної області

Фільтр нижніх частот включає в себе усунення високочастотних компонентів зображення. Це призводить до розмиття зображення (і, отже, до зменшення різких переходів, пов'язаних з шумом). Ідеальний фільтр нижніх частот зберігає всі низькочастотні компоненти і усуває всі високочастотні компоненти. Однак ідеальні

фільтри страждають від двох проблем: розмиття і шумів. Ці проблеми викликані формою пов'язаного фільтра просторової області, який має велику кількість волнистостей. Більш плавні переходи в фільтрах частотної області, такі як фільтр Баттерворта, досягають набагато кращих результатів. [3]

1.4.2. гомоморфності фільтрація

Зазвичай зображення складаються зі світла, відбитого від об'єктів. Основна природа зображення F (x, y) може бути охарактеризована двома компонентами: (1) кількістю світла джерела, падаючим на переглядається сцену, і (2) кількістю світла, відбитого об'єктами в сцені. Ці частини світу називаються компонентами освітлення і відображення і позначаються відповідно i (x, y) і r (x, y). Функції i і r мультиплікативно об'єднуються, щоб дати функцію зображення F:

$$F(x, y) = i(x, y) * r(x, y),$$

де 0 <i (x, y) <беск. і 0 <r (x, y) <1. Ми не можемо легко використовувати вказане вище твір для роздільної роботи на частотних складових освітлення і відображення, оскільки перетворення Фур'є твори двох функцій невіддільне. [4]

Процес гомоморфной фільтрації можна показати на рис. 1.4.2.1:

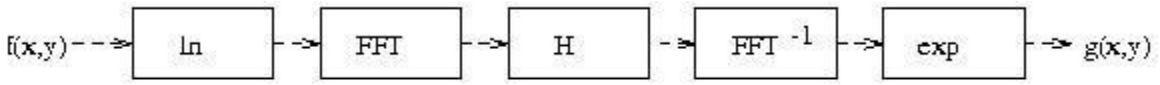

Рис. 1.4.2.1. Процес гомоморфної фільтрації.

РОЗДІЛ 2. ЕТАПИ РОЗРОБКИ АЛГОРИТМА РОЗБИТТЯ ТА СТРУКТУРИЗАЦІЇ ЗОБРАЖЕНЬ

В даному розділі буде розглянуто метод фільтрації зображення, який заснований на полигонально-рекурсивном розбитті вихідного зображення. Кодування проводиться за допомогою знаходження опорних точок, а декодування - виконанням тріангуляції. Також в результаті виконання даної глави буде представлений порівняльний аналіз виконання моделювання алгоритму з основними просторовими методами.

Важливим і одним з найперспективніших напрямків в області кодування і декодування зображень є методи, які використовують алгоритм відновлення зображень, заснований на тріангуляції, який застосовується в задачах розпізнавання образів, а також відновленні двовимірних зображень по нерегулярно розташованим опорних точок (ОТ). Головні переваги триангуляційних методів відновлення:

• Триангуляція ВІД автоматично пристосовується під вихідні дані - в тій області, де ВІД розріджені, трикутники мають велику площу. А там, де є згущення - меншу. Кількість утворених трикутників не перевищує подвоєного числа опорних точок.

• У прямокутної сітки для достовірного відображення досить багато мінливих поверхонь, тому її потрібно подрібнювати, що веде до високих вимог продуктивності обчислювального пристрою, а отже до утворення нестійкості, яка відсутня в разі трикутної сітки.

Розширення можливостей алгоритмів, заснованих на тріангуляції стиснення і відновлення зображень, пов'язані з блоковим методом проектування систем, коли проектується комбінація нових сложнофункціональних блоків і блоків або багаторазові бібліотек. Всілякі стандарти і відсутність зручного інструменту

застосування цих блоків ускладнює завдання. Для проектування систем, які засновані на пірамідально-рекурсивном підході (ПРП) підвищення ступеня стиснення і можливість не втратити точність кодування безпосередньо корелює з обчислювальними потужностями системи. Стрімкий розвиток технології систем на основі процесорів і збільшення щільності розміщення вентилів на процесорах ускладнює виконання завдання, а розвиток методів, які використовують тріангуляцію і розробка алгоритмів кодування і декодування по опорних точках, на основі ПРП більш актуальними.

Розробляються системи ідентифікації і аналізу реальних об'єктів на зображеннях повинні надавати передачу всіх властивостей модельованого об'єкта: обсяг, місце положення, тінь, яскравість, матеріал поверхні. Чим реалістичніше зображення, тим більших потужностей системи вимагає його обробка.

Як підходу для розвитку триангуляційних методів в даному розділі пропонується використання пірамідально-рекурсивного методу, помітно знижує обчислювальні ресурси для знаходження опорних точок в процесі кодування і надається ефективний алгоритм реалізації регулярної триангуляционной сітки при відновленні зображення.

Найкраще наближення до епсилон- ентропії сигналу зображення досягається розбивкою зображень на трикутники.

Це можна пояснити наступними факторами:

• трикутник є найбільш простим полігоном, вершини якого однозначно задають грань

• будь-яку область можна розділити на трикутники

• виходячи з першого пункту, потужності, які потрібні для моделювання алгоритмів розбиття на трикутники помітно нижче, ніж при використанні інших полігонів;

• для трикутника неважко визначити три його найближчих сусіда, що мають з ним спільні грані.

2.1. Пирамидально-рекурсивний підхід

На відміну від триангуляційних методів регулярність пірамідальних структур визначає їх раціональну реалізацію і дієве використання в системах обробки нестаціонарних зображень за рахунок паралельної обробки. Ієрархія опису різних ступенів спільності дозволяє здійснювати незалежну структуризацію зображень. Проектована структуризація є основоположним елементом для моделювання вхідних і вихідних процесів людського сприйняття.

Розглянемо основні переваги пірамідально-рекурсивного методу:

• невибагливість і швидкість алгоритмів декодування (стиснення, оцінка та з'єднання зображення);

• практичне відображення закодованої інформації

(З'єднання, збереження і виявлення зображення);

• найголовнішою причиною використовуваних в системах кодування алгоритмів пірамідально-рекурсивного розбиття є можливість забезпечення покрокової обробки зображення, коли наступна операція починає виконуватися після закінчення попередньої

• можливість обробки закодованих даних без відновлення вихідного зображення для більшої частини операцій обробки.

Суть пірамідально-рекурсивного методу полягає в тому, що ми маємо безліч опорних точок на оригінальному документі, після виконання тріангуляції яких ми маємо регулярну двовимірну полигональную сітку. Усереднення вершин сітки по яскравості сприяє відновленню результуючого зображення. [6]

Кількість і положення опорних точок на зображенні визначають точність і швидкодію алгоритмів кодування і декодування зображення.

Щоб знайти опорні точки опишемо емпіричні варіанти:

• знайдемо рівняння площині за допомогою методу найменших квадратів, яке проходить по точках полігону з найменшим відхиленням від заданого порогу;

• розбиваємо полігон на рівні по площі полігони, якщо звіти перевищили значення яскравості по заданому порогу (перша ступінь деталізації)

• якщо яскравості звітів не перевищують заданого порогу, то в межах полігону опорна точка визначається найменшою відстанню від площині (друга ступінь деталізації). [8]

Спільність підходу до вирішення даного завдання знайшла своє відображення в запропонованій їх систематизації.

2.2. Алгоритми кодування та декодування по опорних точках

Процедура кодування і декодування нестаціонарних сигналів зображень полягає в наступному: вся поверхня зображення сприймається як одна область (полігон) і на ній відбувається перевірка критерію однорідності (чи присутні опорні

точки на зображенні чи ні). На кроці кодування, якщо критерій виконується, тоді зображення можна вважати що містить тільки фон (далі будемо вважати як порожній полігон). В іншому випадку проводиться розбиття цього полігону на однакові рівні полігони і на кожному з них знову обчислюється критерій однорідності. Далі, після виконання перевірки подальшого розбиття піддаються тільки ті полігони, для яких критерій не виконується. Потім в межах кожного з отриманих полігонів виконується пошук оптимальної точки, максимально характеризує даний полігон. В результаті аналізу всієї поверхні зображення ми маємо полигональную двовимірну сітку, яка містить як порожні полігони, так і полігони, які містять ВІД. [7]

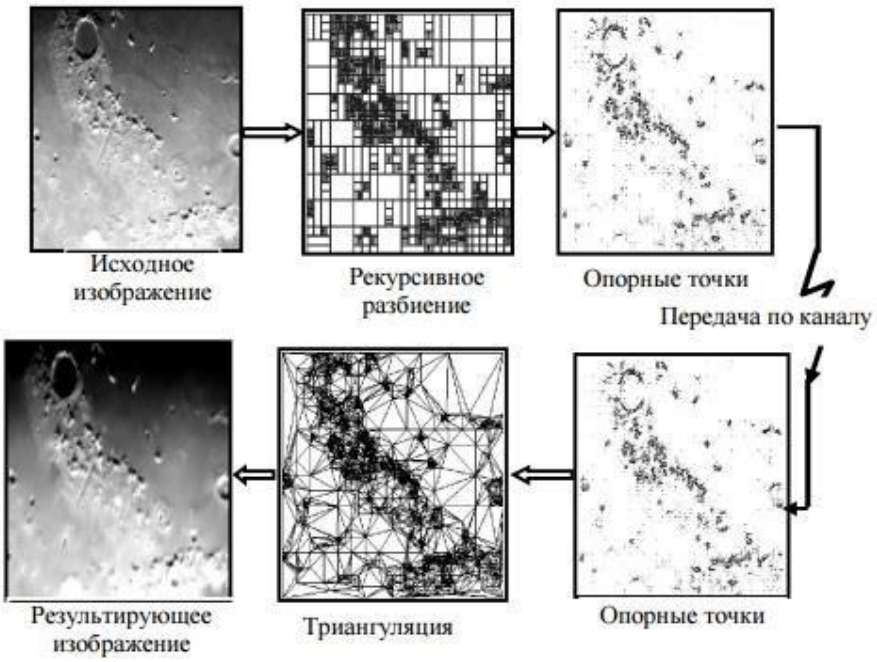

Рис. 2.2.1 Процес кодування і декодування по опорних точках

Кожна отримана ВІД має свої унікальні координатами і яскравістю в межах полігону. Далі на кроці відновлення зображення кожна опорна точка з'єднується з найближчими сусідніми точками, в результаті всі крапки будуть з'єднані сіткою, званої регулярної тріангуляцією. Отримана трикутна сітка шляхом апроксимації двовимірної поверхні за яскравістю відновлює результуюче зображення.

Існує досить відомий, але мало вивчений метод тріангуляції Делоне. Даний метод є ефективним з точки зору досягнення високого коефіцієнта стиснення (КСЖ). Однак, за це доводиться платити складністю реалізації. Якщо на прямокутній сітці можна за кілька операцій визначити полігон, в який потрапила точка, для якої ми хочемо дізнатися значення яскравості, то в разі тріангуляції знаходження трикутника і аналіз форми аппроксимирующих трикутників створюють проблеми в реалізації.

Відзначимо і що сама процедура побудови триангуляционной сітки з непересічними трикутниками і з вершинами, що знаходяться в опорних точках досить складна.

2.3. Структурна схема системи кодування по опорних точках

В даний час, з появою нової елементної бази у вигляді систем на процесорах і сучасних САПР на їх основі, відкриваються нові варіанти створення спеціалізованих систем обробки і практично зняті багато обмежень по швидкості і складності для проектування систем обробки інформації, що, здавалося б, повинно привести до вирішення основних завдань сприйняття даних технічними системами. Хоча проблема не вирішена, а тільки зросла розуміння її значущості та своєрідних складнощів. В даний час стан речей змушує розробників витрачати великі зусилля на вдосконалення новітніх методів і способів не тільки в обробці зображень, але і в ефективному поданні стислих даних, щоб якомога більше скоротити розрив між «комп'ютерним» і «людським» сприйняттям зображень.

Пірамідально-рекурсивні методи розбиття базуються на одному з двох основних властивостей сигналу яскравості:

• розривна (нестаціонарність). Суть підходу полягає в розбитті зображень на основі помітних змін сигналу, наприклад перепади яскравості на зображенні.

• Однорідність. Дана категорія методів використовує розбиття зображення на області, однорідні в плані попередньо підібраних критеріїв, наприклад фільтрація по порогу розбиття, збільшення областей за площею, злиття і розбиття областей.

Регулярність пірамідальних структур надає можливість створювати ефективні системи обробки зображень через наявність розпаралелювання алгоритмів. На кроці кодування распараллеливанию підлягає процес розбиття і пошуку опорних точок на оригінальному документі, а на кроці декодування - процес тріангуляції (апроксимації полігонів) опорних точок. Так як процесорний час в основному займають конкретно ці два процеси, значить, збільшення числа елементарних процесорів для виконання цих функцій і застосування модульного проектування призводить до збільшення продуктивності системи кодування і декодування в системах реального часу.

Щоб ефективно використовувати пам'ять її необхідно розділити на три частини: пам'ять заявок на розбиття полігонів, пам'ять опорних точок і пам'ять зображень. Завдяки розвитку технологій систем на процесорі і наявності сучасних САПР ми можемо перейти від теоретичного моделювання до практичного побудови швидких пірамідальнорекурсівних алгоритмів обробки інформації і в перспективі вирішувати такі проблеми:

• реалізувати алгоритми пірамідальних уявлень у вигляді параметризованих бібліотечних СФ-блоків на базі технології «система на процесорі» як паралельних, так і послідовних типів

• створювати мультимедійні системи, ґрунтуючись на чотиривимірної просторової обробці даних.

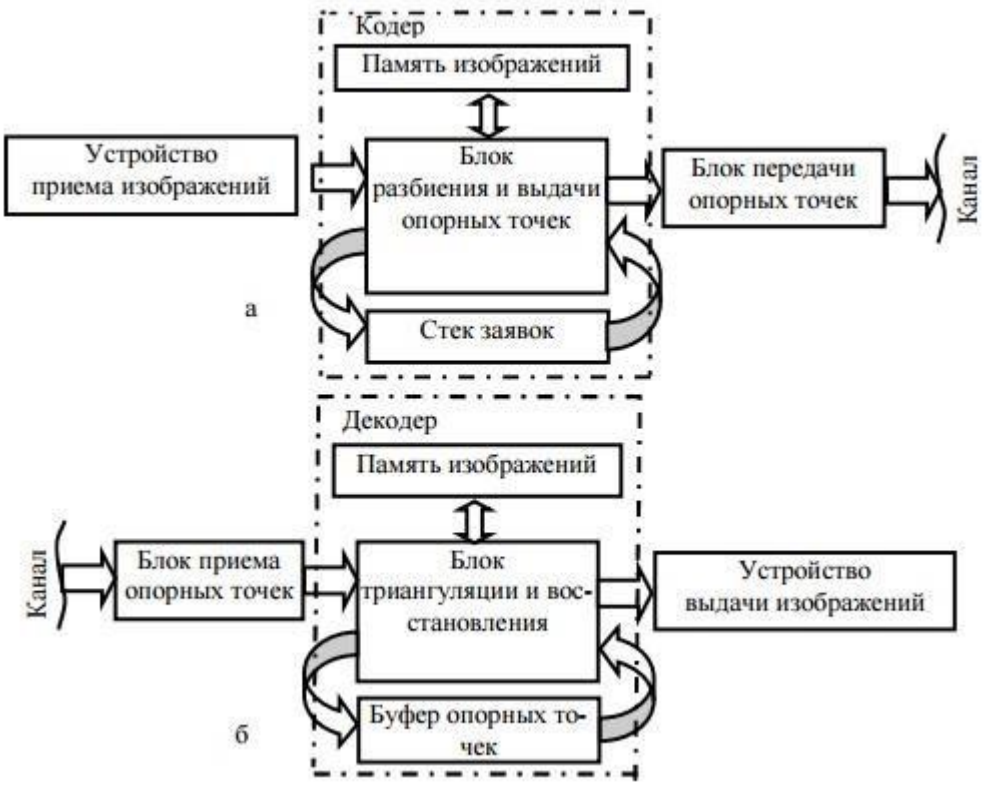

Рис. 2.3.1. Структурна схема системи кодування і декодування зображень

2.4. Триангуляція Делоне

Триангуляція широко використовується в якості основи для представлення геометрії та іншої інформації, що з'являється у величезній різноманітності додатків. У географічних інформаційних системах (ГІС) тріангуляція використовуються для подання місцевості і зв'язків між географічними об'єктами. Системи для моделювання геологічних структур в нафтовій і газовій промисловості використовують тріангуляцію для представлення поверхні, що відокремлює різні геологічні структури, і для подання властивості цих структур. Системи автоматизованого проектування (САПР) використовують тріангуляцію є основними в обробній промисловості і зокрема, в автомобільній промисловості, яка є рушійною

силою для дослідження тріангуляції протягом багатьох десятиліть. Ми також можемо знайти програму для використання тріангуляцію в технічних областях, які моделюють фізичні явища, використовуючи кінцеві методи елементів (МСЕ) і, нарешті, відзначимо важливість тріангуляції в області візуалізації і комп'ютерної графіки. [11]

Визначення тріангуляції Делоне - всередину кола описаної навколо будь-якого її трикутника не повинна потрапляти жодна з вершин графа.

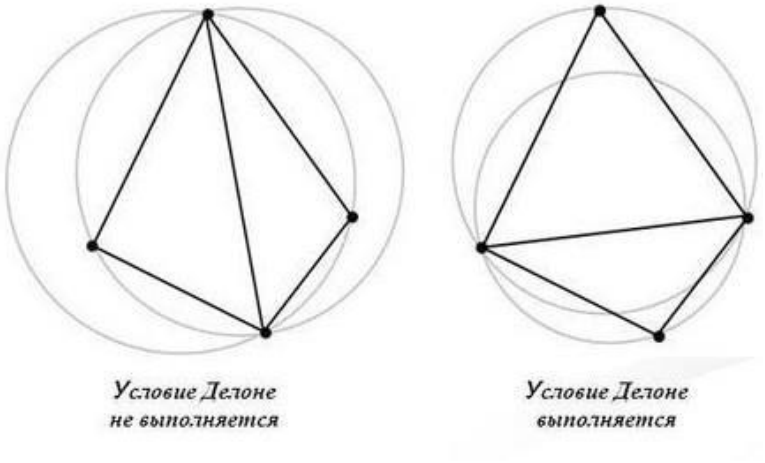

Рис. 2.4.1. Триангуляція за критерієм Делоне.

2.5. Вибір ефективного методу

Не вдаючись в описані раніше переваги обраного методу для стиснення і відновлення зображень, корисно буде проаналізувати значення коефіцієнтів стиснення на прикладах зображень, пропонованих в розробленій класифікації за двома ознаками: міра нестаціонарності і широкополосность сигналів зображень. Також проведемо порівняння значень коефіцієнтів стиснення для методів в просторової області, які ми позначили раніше, а саме: (дискретне вейвлетпреобразованіе (ДВП), дискретне косинусное перетворення (ДКП) і ПЗМ).

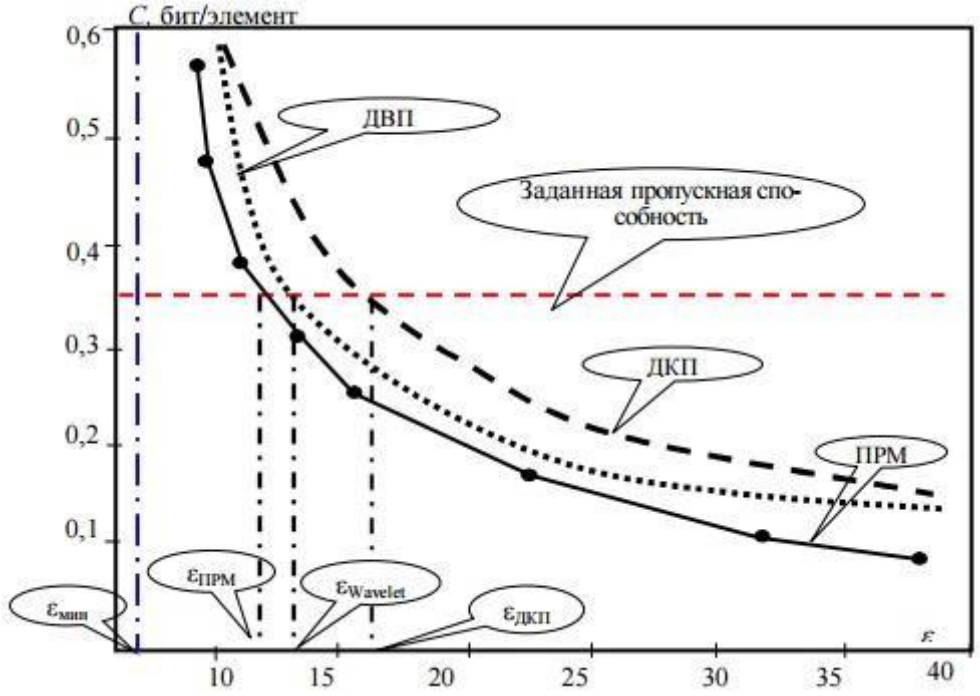

Рис. 2.5.1. Залежність необхідної пропускної спроможності від допустимої СКО

Слід зазначити важливість впливу геометричних і амплітудних відхилень, які враховуються в процесі стиснення, на якість відновленого зображення. При цьому найбільш істотним є знаходження оптимального поєднання геометричних і амплітудних спотворень, яке характеризується їх рівним впливом на якість відновленого зображення.

Під амплітудним відхиленням мається на увазі той діапазон допустимих спотворень яскравості точок вихідного зображення при стисканні для отримання полігональної сітки. Отже, геометричні відхилення - це допустимі відхилення в системі координат (т. Е. Задані в межах полігону). Слід зазначити, що сукупність цих відхилень створює своєрідну апертуру в тривимірному просторі, за межі якого в процесі стиснення не допускаються відхилення яскравості і координат точок зображення. Сама апертура може бути витягнута уздовж координатних осей або осі яскравості і має різні форми і простору.

У підсумку, моделювання етапів ПЗМ визначило основні характеристики системи для апаратної реалізації алгоритмів, що задаються в основу оцінки потрібної нам продуктивності НВІС кодера і декодера на базі технології «система на процесорі» (перше значення для гладких зображень, друге - для контрастних):

• обсяг робочої пам'яті = (61, тисячі шістсот сорок два)

• кількість заявок, звернених в пам'ять = (90399, 143037)

• вага стисненого опису вихідного зображення в бітах = (2412, 61306)

• щільність опису опорних точок в бітах = (1,39, 1,25)

• кількість опорних точок = (326, 8451)

У тому числі знайдено оптимальне поєднання числа полігонів, які були отримані при розбитті вихідного полігону, кількості рівнів розбиття, СКО кодування і потрібної пропускної здатності С при багаторазовому виконанні експериментів в алгоритмах з фіксованим розташуванням ВІД з метою досягти високого ступеня стиснення і прийнятній якості.

ПЗМ алгоритми кодування і декодування реалізовані на мові високого рівня. Перевірка на практиці ПЗМ проведена при використанні розбивки на кожному етапі кодування конкретного інтервалу на 2, 3 і 4 частини на зображеннях двох типів контрастності (гладкі і контрастні). В результаті помітно, що при хорошому суб'єктивному якості відновлених зображень коефіцієнт стиснення з використанням ПЗМ на 2% більше, ніж ДВП, і на 1,5% більше, ніж ДКП для гладких знімків, а для контрастних зображень на 1,4% більше, ніж ДВП і ДКП.

Дослідження проводилися над зображеннями розміром 256 × 256 точок (полігон нульового рівня) і яскравість дозволом 8 градацій. Ємність пам'яті для зберігання цифрового зображення N = 2562 • C. [8]

Висновки

В результаті виконання даного розділу сформулюємо висновки, які є основними властивостями пропонованого методу при проектуванні системи кодування і декодування зображень.

1. Дані представляються у вигляді якоїсь регулярної ієрархічної структури, яка не залежить від змісту даних в ній. Дана структура включає в себе безліч опорних точок і способи їх знаходження, а також компактно розташовану в пам'яті ЕОМ, причому структура і спеціальна нумерація дозволяють абстрагуватися від вихідного багатовимірного скалярного або векторного інформаційного поля і зберігають в той час все властивості вихідного зображення.

2. Для того, щоб надати можливість покрокового наближення до епсилон-ентропії вихідного сигналу, на кожному етапі розбиття вихідного зображення аналізується результат відновлення за величиною помилки кодування і при необхідності проводиться подальше розбиття і уточнення результуючого зображення.

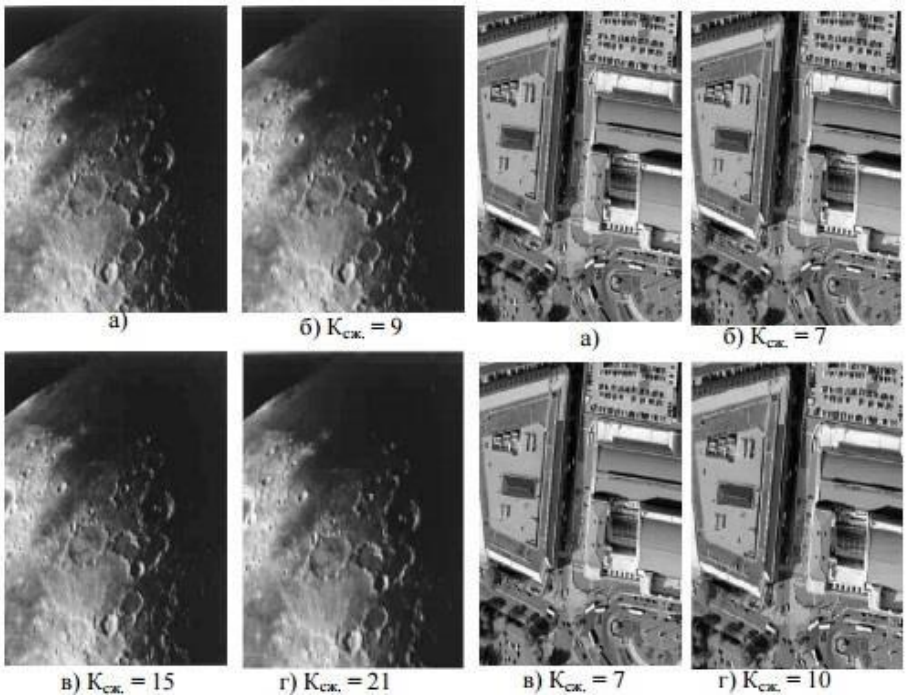

Рис. 2.5.2. Результати стиснення двох класів зображень: а – вихідне зображення; б - ДВП; в - ДКП; г - ПЗМ.

3. Елементами стисненого опису не є фрагменти зображення, а опорні точки, які мають схожі або однакові властивості. Виходячи з цього, складність ПЗМ визначена не роздільною здатністю апаратури (числом звітів зображення), а числом опорних точок, т. Е. Кількістю елементів самої структури, яке для конкретного класу може бути значно меншою за кількість звітів вихідного зображення.

РОЗДІЛ 3 ОПИС ПРОГРАМИ

Лістинг програми показаний в додатку А.

3.1 Вибір інструментальних засобів розробки

Нині існує чимало якісних засобів розробки програмного забезпечення. Розглянемо особливості деяких з них.

В даному проекті використовується Borland Delphi.

Borland Delphi — це інтегроване середовище швидкої розробки програмного забезпечення в Microsoft Windows. Delphi являє собою актуальну і легку у використанні програму, яка необхідна для генерації автономних програм графічного інтерфейсу або 32-бітових консольних додатків - програм, які існують поза рамками GUI, замість цього, відповідно до так званим «DOS вікна».

Delphi є першою мовою програмування, що забезпечує знищення бар'єру між додатками комплексного і спрощеного характеру у використанні і низькорівневими бітовими програмними засобами.

Створюючи GUI-додатки за допомогою Delphi, трансльований мова програмування існує в рамках RAD-середовища (мова Паскаль). Delphi включає в себе такі компоненти, як основні елементи графічного інтерфейсу користувача системи Windows, які представлені у вигляді екранного бланка, кнопок і ін. Це означає, що користувачеві не потрібно організовувати написання кодування в разі приєднання цих елементів до визначеному додатком. Користувач просто розробляє їх в програмі малювання. Можливо також використання керуючих елементів

ActiveX з метою створення таких спеціальних програм, як веб-браузери. Delphi дозволяє користувачеві розробляти весь інтерфейс візуально, а також швидко складати код.

Delphi існує в безлічі конфігурацій, які використовуються як у відомчих, так і у виробничих установах. За допомогою Delphi користувач може написати програму в рамках ОС Windows на багато швидше і простіше, ніж це коли-небудь було можливо.

В основі Delphi-середовища лежить мова програмування Паскаль. Delphi має можливість використання безлічі баз даних. Прикладами можуть бути локальні бази даних - Paradox, Dbase, мережеві серверні бази даних SQL - InterBase, SysBase.

Незважаючи на те, що написання комп'ютерної програми за допомогою Delpi представляється полегшеним заходом, не варто забувати, що подібне програмний засіб вимагає упевнених знань системи Windows.

3.2 Опис програми

При розробці інтерфейсу додатку можна обмежитися єдиним головним вікном, яке буде містити в собі активні елементи для настройки параметрів роботи програми, і області виведення результатів виконання конкретних функцій.

Інтерфейс програми містить кілька областей: область вибору вхідного зображення, область налаштування параметрів розбиття зображення, область роботи з опорними точками, область з висновком результуючого зображення. Загальний

вигляд головної форми додатка рисунок 3.1

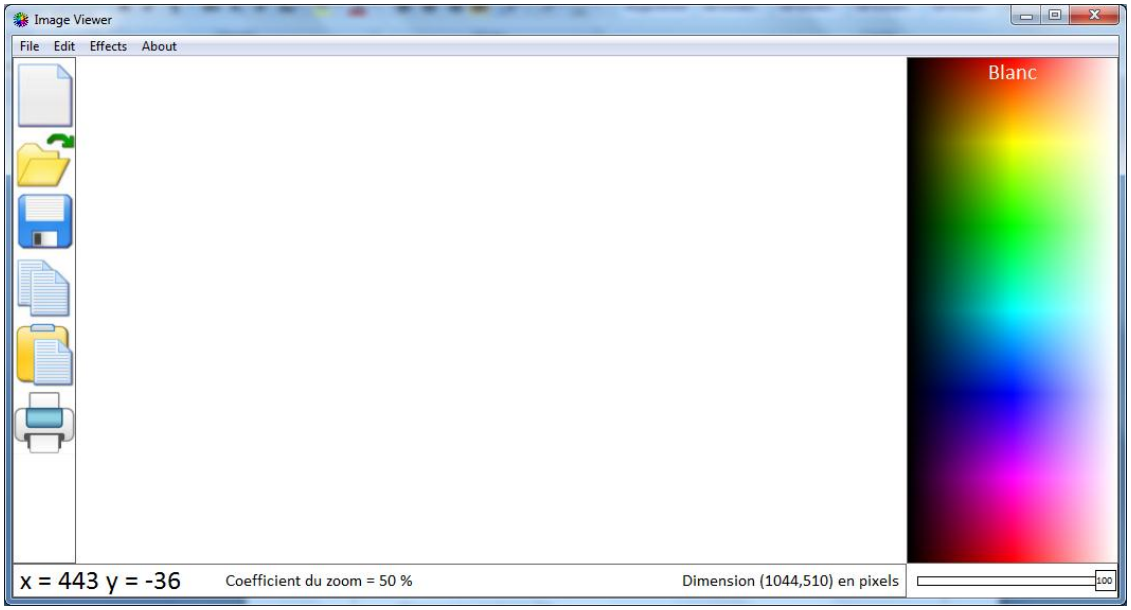

Рисунок 3. 1 –Вікно програми

Розглянемо докладніше різні області інтерфейсу розробленої системи.

Область вибору вхідного зображення призначена для наочної демонстрації роботи програми - щоб звірити вихідне зображення з кінцевим. Також область містить кнопки, за допомогою яких можна відкрити потрібне зображення, відновити зображення з файлу опорних точок, попередньо зберігши його і кнопку скасування останнього виробленого дії.

Для вибору нас зображення потрібно натиснути кнопку  «Відкрити» або в меню File кнопку Open, потім за допомогою провідника знайти зображення і натиснути кнопку ОК.

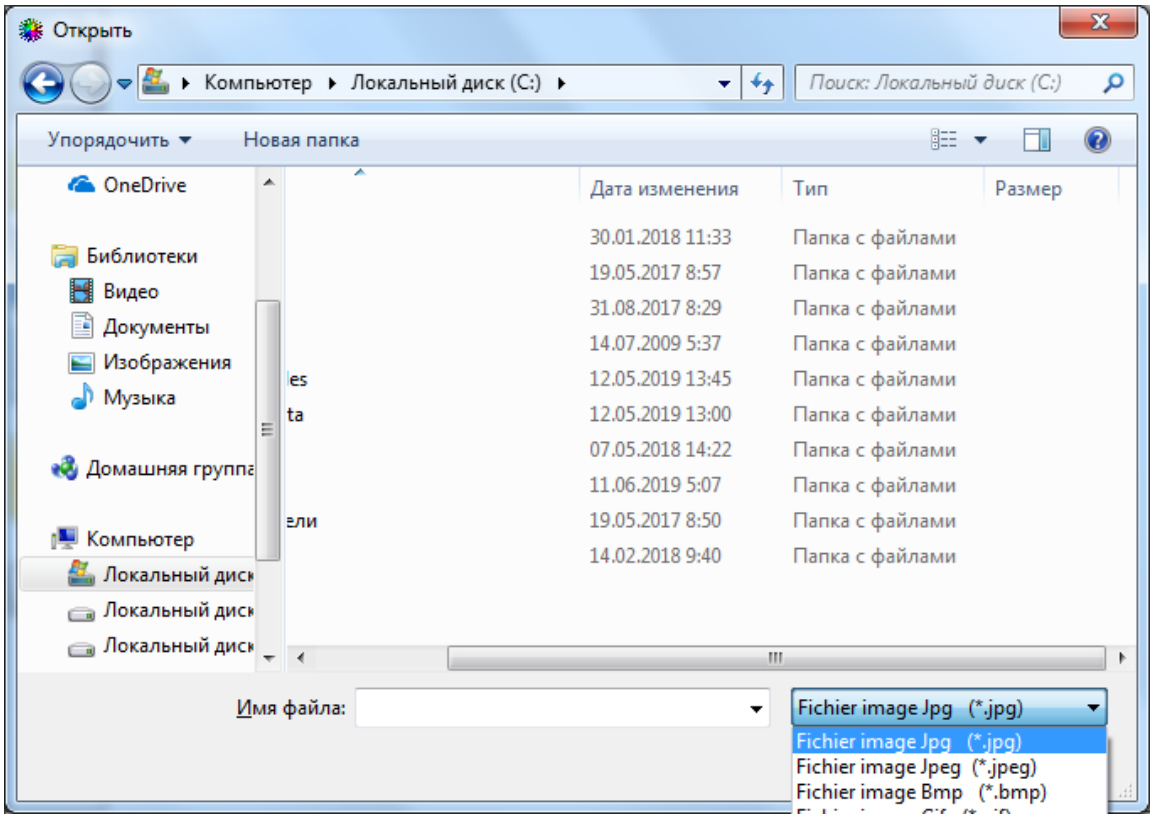

Рисунок 3. 2 –Вікритя зображення

Відзначимо що програма підтримує такі формати зображення jpg, jpeg, bmp, gip, png

Після виконання даної операції вибране зображення з'явиться в області інтерфейсу додатку (приклад у формте jpeg)

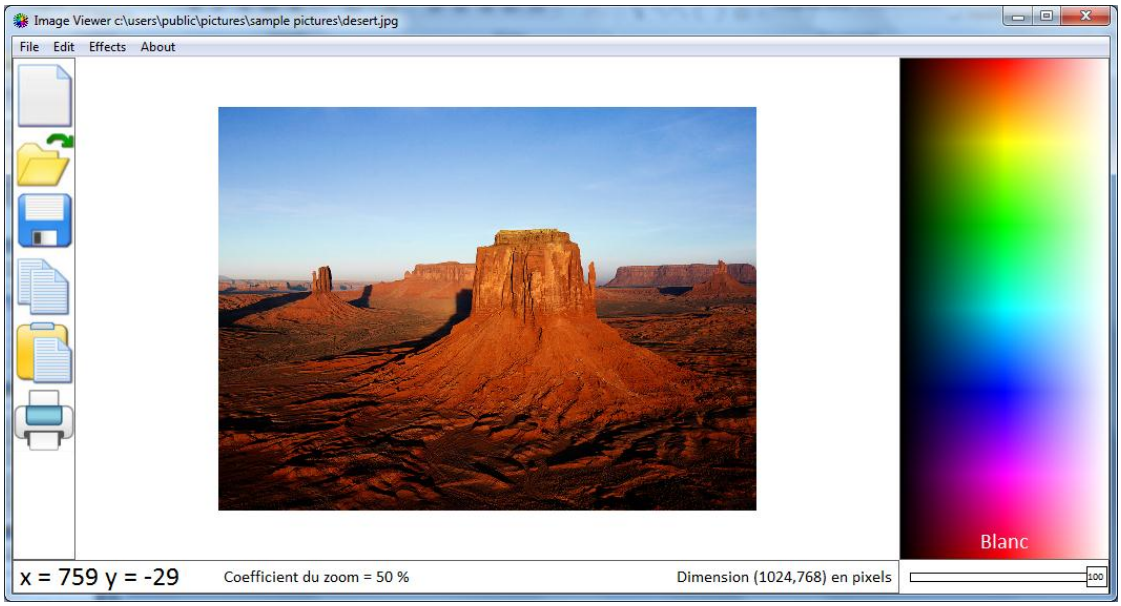

Рисунок 3.3 – Приклад

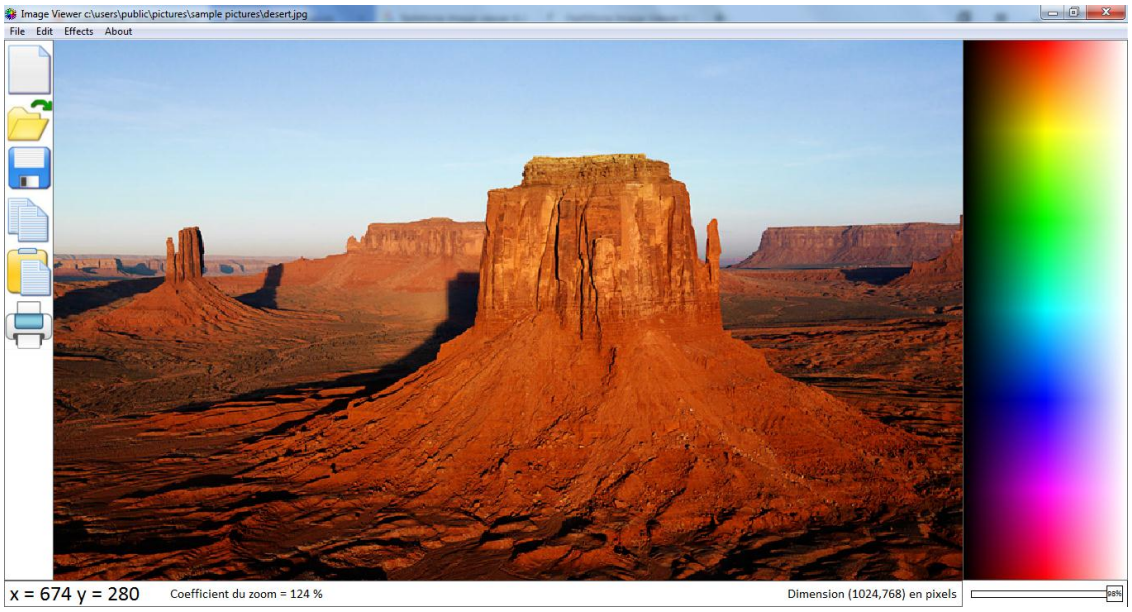

Рисунок 3.4 – Масштабування картинки в головному вікні.

В вкладці Effects знаходяться дії доступні в даній програмі рисунок 3.5

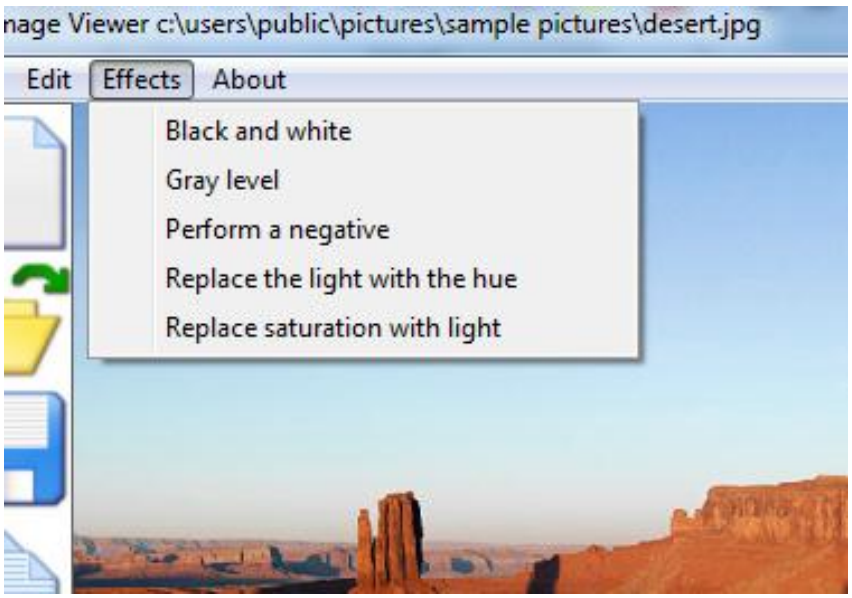

Рисунок 3.5 – Вкладця Effects

Black and while - Чорний і білий - Рисунок 3.6

Gray level - Рівень сірого- Рисунок 3.7

Perform a negative - Виконайте негатив- Рисунок 3.8

Replace the light with the hue - Замініть світло на відтінок- Рисунок 3.9

Replace saturation with light - Замініть насичення світлом

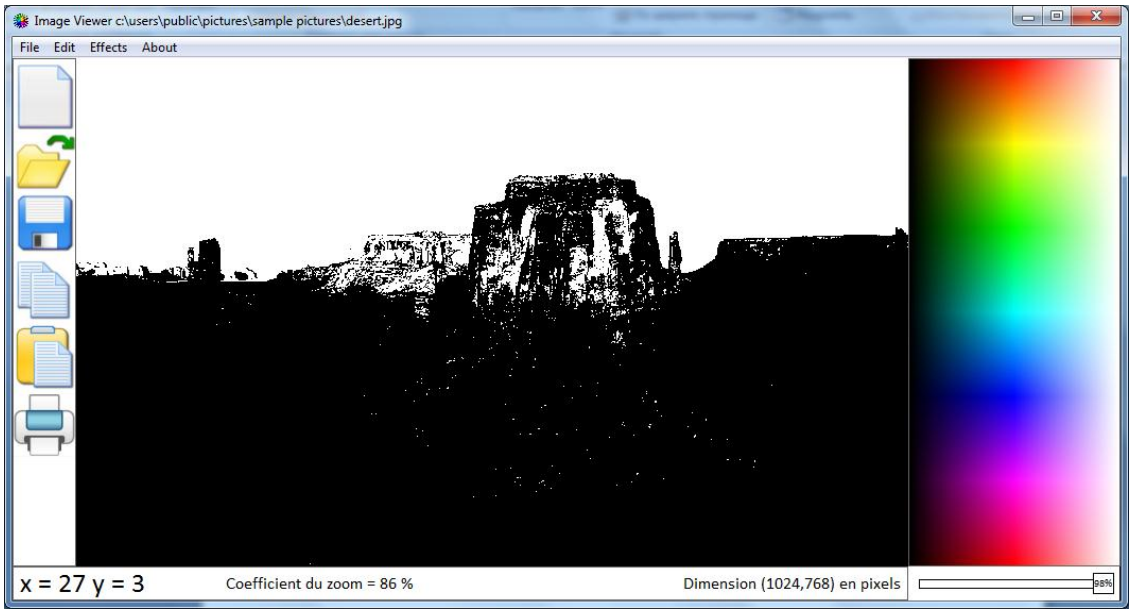

Рисунок 3.6 – ефект чорний і білий

Рисунок 3.7 – рівень сірого

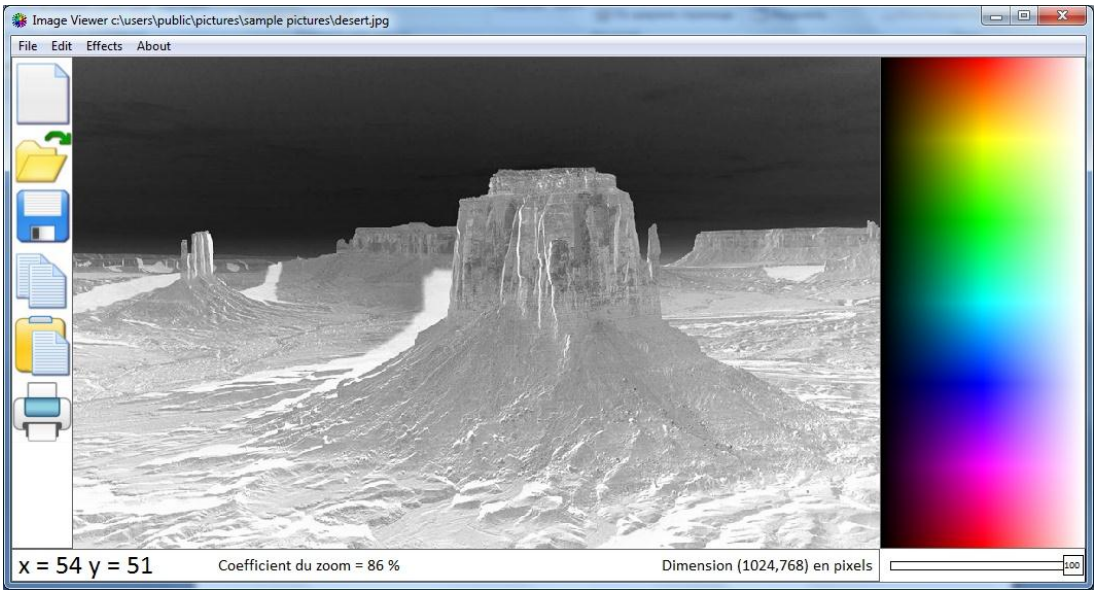

Рисунок 3.8 – ефект негатив

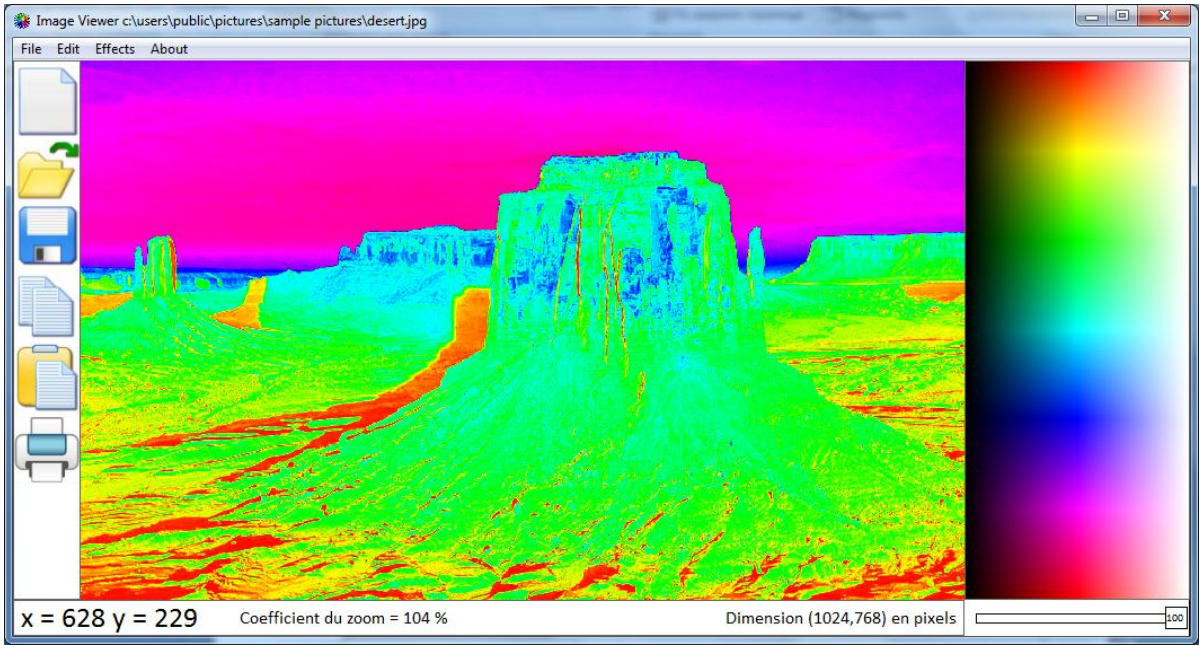

Рисунок 3.9 – ефект заміна світло на відтінок

Перед застосуванням ефекта заміна світло на відтінок попередньо вибирається в правому вікні відтінок.

За допомогою бігунка можна змінювати затененность рисунок 3.10

Рисунок 3.10 – затененность

У нижньому рядку відображається координати курсору щодо зображення і розширення обраного зображення в пикселях рисунок 3.11

Рисунок 3.11 – нижний рядок

Варто відзначити що інтерфейс програми є інтуїтивно зрозумілимі включає в себе основні дії відкрити файл, зберегти і т.д

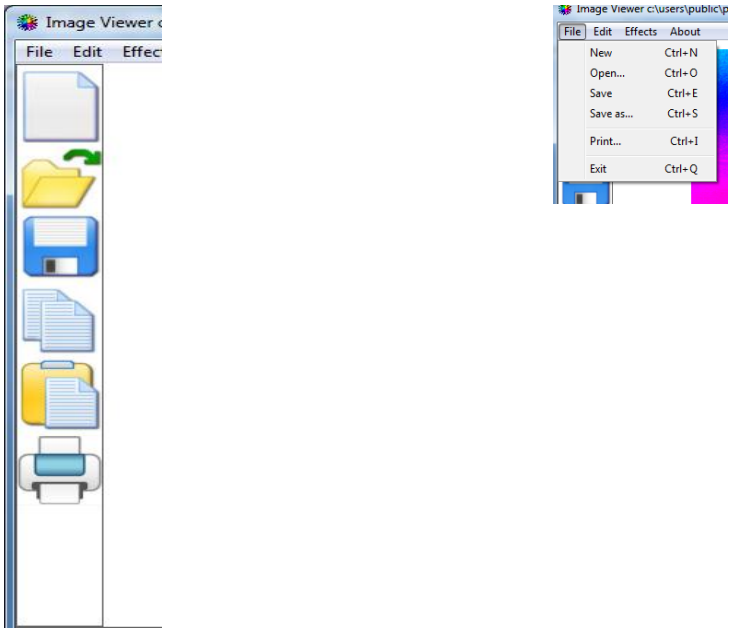

Рисунок 3.12 – основні функції дій над файлом при збереженні файлу так само можна вибрати різні формати збереження

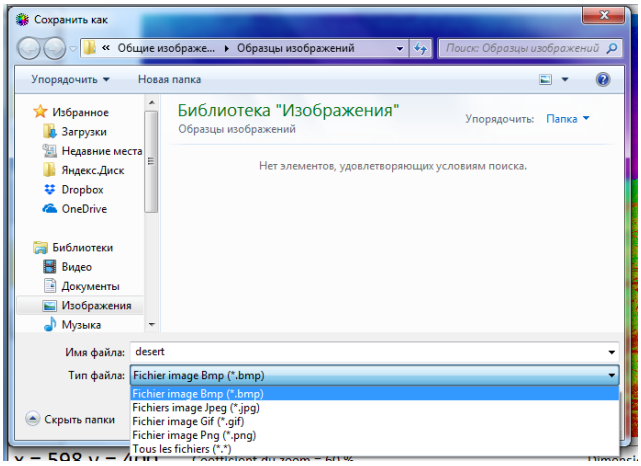

Рисунок 3.13 – формати збереження

ВИСНОВКИ

1. Зроблено аналіз предметної області, вивчені основні джерела й особливості розробки програмного засобу для кольорокорекції зображень.

2. У роботі була розроблена програма для кольорокорекції зображень

3. Одержана програма може бути як як безкоштовна програма для редагування цифрових зображень.

СПИСОК ЛІТЕРАТУРИ

1. Б/а Википедия // [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Обработка_изображений
2. Стругайло В. В. Обзор методов фильтрации и сегментации цифровых изображений.// [Электронный ресурс]. URL:http://cyberleninka.ru/article/n/obzor-metodov-filtratsii-i-segmentatsiitsifrovyh-izobrazheniy
3. Алгоритмические основы растровой графики // [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/993/163/lecture/4505
4. Методы и алгоритмы фильтрации изображений // [Электронный ресурс]. URL: http://www.mm-dsp.com/2011-09-09-09-27-20/2011-09-0912-16-15.html#2
5. Пространственные методы улучшения изображения. Некоторые градационные преобразования. Гистограмма изображения. Основы пространственной фильтрации // [Электронный ресурс]. URL: http://helpiks.org/5-26602.html
6. Александров В.В., Горский И.Д. Представление и обработка изображений. Рекурсивный подход. Л.: Наука, 1985. – 192 с. Фролов
7. Фахми, Ш. С. Классификация нестационарных изображений и разработка методики оценки алгоритмов кодирования источника [Текст] / Ш. С. Фахми, И. А. Зубакин // Науч. тех. вестник СПбГУ ИТМО.- 2010.- № 2(66).- С. 54- 59.
8. Фахми, Ш. С. Оценка степени приближения к энтропии на основе пирамидально-рекурсивного метода кодирования изображений [Текст] / Ш. С. Фахми // Изв. СПбГЭТУ «ЛЭТИ».- 2010.- Вып. 4.- С. 8-17.
9. Фахми, Ш. С. Развитие триангуляционного подхода для кодирования и декодирования нестационарных изображений [Текст] / Ш. С. Фахми // Вестник ТОГУ.- 2010.- № 3 (18).- С. 81–90.

10. Hjelle O., Dæhlen M. Triangulations and Applications. Springer-Verlag, 2006.
11. Demaret L., Iske A. Anisotropic triangulation methods in image approximation. In: Approximation Algorithms for Complex Systems. E.H. Georgoulis, A. Iske and J. Levesley (etc). – Berlin: Springer, 2010. – P. 47–68.
12. Скворцов А. В. Триангуляция Делоне и ее применение. Томск: Изд-во Том. ун-та, 2002.
13. Разработка и стандартизация программных средств / сост.: В.И. Фомин. СПб: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина), 2015. 35 с.

ДОДАТОК А

Листинг програми

```
unit Voir_une_image;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, jpeg;
type
  TForm_ouvrir_l_imprimante = class(TForm)
    PrintDialog1: TPrintDialog;
    Timer1: TTimer;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Image1Click(Sender: TObject);
  private
  public
  end;
var
  Form_ouvrir_l_imprimante : TForm_ouvrir_l_imprimante;
implementation
procedure TForm_ouvrir_l_imprimante.Button1Click(Sender: TObject);
begin
if PrintDialog1.Execute then begin end;
end;
procedure TForm_ouvrir_l_imprimante.Timer1Timer(Sender: TObject);
begin
```

```
self.Hide; {L'йvйnement timer a une pйriode de 50 millisecondes}
Timer1.Enabled:=false;
end;
procedure TForm_ouvrir_l_imprimante.Image1Click(Sender: TObject);
begin
showwindow(self.Handle,sw_hide);
end;
end.
interface

uses wintypes,sysutils,contnrs,classes,math,utile,forms;

type

 node=class(Classes.Tcollectionitem)
   end;

T_color_name = class(node)
   public
         name:pc50;
         color:TColorRef;
         constructor Create(un_nom_couleur:pchar; une_numeric_couleur:tcolorref);
         end; {T_color_name}

 TO_COL = class(contnrs.TObjectList)
   function at(index:integer):pointer;
   procedure insert(AObject: TObject); overload;
   procedure Freeall;
   end;
```

```
  T_Col_Named = class(TO_COL)
    function Lecture_fichier_couleur:boolean;
    procedure Recherche_couleur_plus_proche(find_this_color:TColorRef; var result_named_color:T_color_name);
    end;

procedure pchar_to_real(apc:pchar; var areal:real);
procedure remplace_pt_window_pt_decimal(apc:pchar); overload; {1,00 -> 1.00}
procedure remplace_pt_window_pt_decimal(var str:string); overload; {1,00 -> 1.00}
function Get_exe_path(path_exe:pchar):pchar;
function Hexvalue(hex:string):longint;
procedure bip;

implementation

function get_exe_path(path_exe:pchar):pchar;
var p:pchar;
begin
strpcopy(path_exe,paramstr(0));
P := StrRScan(path_exe, '\');
inc(p); p^:=#0;
get_exe_path:=path_exe;
end;

procedure pchar_to_real(apc:pchar; var areal:real);
var astring:string;
        code:integer;
begin
```

```
remplace_pt_window_pt_decimal(apc);
astring:=strpas(apc);
system.val(astring,areal,code);
if code<>0 then
        areal:=0.0;
end;

procedure remplace_pt_window_pt_decimal(apc:pchar); {1,00 -> 1.00}
var pc:pchar;
begin
if sDecimal[0]='.' then exit;
 repeat
  pc:=strpos(apc,sDecimal);
  if pc<>nil then pc^:='.';
   pc:=strpos(apc,sDecimal);
 until pc=nil;
end;

procedure remplace_pt_window_pt_decimal(var str:string);
 var apc:pc1024; {Must be enought}
 begin
 strpcopy(apc,str);
 remplace_pt_window_pt_decimal(apc);
 str:=strpas(apc);
 end;

function  Hexvalue(hex:string):longint;
var i:integer;
        val:longint;
```

```
            c:char;
        begin
        val:=0;
        for i:=length(hex) downto 1 do
            begin
            c:=upcase(hex[i]);
            case c of
                '0'..'9': val := round( val + ( ord(c) - ord('0') ) * math.power(16,length(hex)-i));
                'A'..'F': val := round( val + ( ord(c) - ord('A') + 10 ) * math.power(16,length(hex)-
i));
                end;
            end;
        HexValue := val;
        end;

        procedure bip;
        begin
         messagebeep(0);
         end;

        function TO_COL.at(index:integer):pointer;
         begin
         try
           if (index<count) and (index>=0) then
             at:=Self.items[index]
           else
             at:=nil;
         except
           on EAccessViolation do
```

```
      begin
      at:=nil;
      end;
    end;
  end;

procedure TO_COL.insert(AObject: TObject);
  begin
  self.add(AObject);
  end;

procedure TO_COL.Freeall;
  var i:integer;
      un:node;
  begin
  for i:=pred(Count) downto 0 do
    begin
    un:=node(Items[i]);
    if un<>nil then
      remove(un);
    end;
 self.Clear;
  end;

constructor T_color_name.Create(un_nom_couleur:pchar; une_numeric_couleur:tcolorref);
 begin
 inherited Create(nil);
 strcopy(Self.name,un_nom_couleur);
 self.color:=une_numeric_couleur;
```

```
end; {T_color_name.Create}


function T_Col_Named.Lecture_fichier_couleur:boolean;
var
    pc_file_name: string;
 pc_input:pc100;
 num_buf:pc20;
F:Text;
 P,Q,R:pchar;
 un_color_name:T_color_name;
 index:integer;
 un_composant_de_couleur:real;
 Couleur_courant:array[0..3] of real;
 an_str,an_strR,an_strV,an_strB:string[12];
begin
Lecture_fichier_couleur:=FALSE;
pc_file_name := ExtractFilePath(Application.ExeName) + 'ansicoul.txt'; {colors.inc}
if FileExists((pc_file_name)) then
begin
System.Assign(F,pc_file_name);
System.Reset(F);
While Not(EOF(F)) do
begin
        readln(F,pc_input);
        P:=Strpos(pc_input,kpc_diese);
if p<>nil then
begin
q:=p;
dec(q);
```

```
q^:=#0;
inc(p);
an_str:=strpas(p);
      an_strR:=an_str;
an_strR:=an_str[1]+an_str[2];
an_strV:=an_str[3]+an_str[4];
          an_strB:=an_str[5]+an_str[6];
      un_color_name:=T_color_name.Create(
        pc_input,RGB(Hexvalue(an_strR),Hexvalue(an_strV),Hexvalue(an_strB)));
insert(un_color_name);
end
else
begin
P:=Strpos(pc_input,kpc_equal);                  if P<>nil then
begin
Q:=P;
dec(P);
inc(Q);
P^:=#0;
R:=@num_buf;
index:=0;
while (Q^<>chr(0)) do
                              begin
inc(Q);
if Q^ in ['0'..'9','.'] then
begin
R^:=Q^;
inc(R);
end
```

```
else if R<>@num_buf then
begin
R^:=#0;
pchar_to_real(num_buf,un_composant_de_couleur);
if index<3 then
Couleur_courant[index]:=un_composant_de_couleur;
inc(index);
R:=@num_buf;
end;
                    end; {while}
      un_color_name:=T_color_name.Create(pc_input,
        RGB(
round(Couleur_courant[0]*255),
round(Couleur_courant[1]*255),
round(Couleur_courant[2]*255)));
insert(un_color_name);
end;
end;
 end;
Close(F);
Lecture_fichier_couleur:=TRUE;
end;
end; {T_Col_Named.Extrated_Lecture_fichier_couleur}


procedure T_Col_Named.recherche_couleur_plus_proche(find_this_color:TColorRef; var
result_named_color:T_color_name);
var i:integer;
    distance_maxi:longint;
    une_couleur:lec_color.T_color_name;
```

```
{Parcours l'intйgralitй du tableau, trouve la couleur la plus proche (c) dB}
procedure first_name_color(named_color:T_color_name);
var distance_local:longint;
begin
distance_local:=(abs(GetRvalue(find_this_color)-GetRvalue(named_color.color)))
+(abs(GetGvalue(find_this_color)-GetGvalue(named_color.color)))
+(abs(GetBvalue(find_this_color)-GetBvalue(named_color.color)));
if distance_local<distance_maxi then
begin
distance_maxi:=distance_local;
result_named_color:=named_color;
end;
end; {first_name_color}
begin {recheche_couleur_plus_proche}
result_named_color:=nil;
  for i:=0 to pred(self.Count) do
    begin
    une_couleur:=self.at(i);
    {Est-ce que cette couleur correspond?}
    if une_couleur<>nil then      first_name_color(une_couleur);
    end; {for i}
end;  {T_Col_Named.recheche_couleur_plus_proche}
end.
```