Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
другий (магістерський)
(рівень вищої освіти)

на тему: **Попередня обробка зображень для задач технічного зору**

Виконав: студент V курсу, групи КН-61м
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Лях Р.І.
(прізвище та ініціали)
Керівник Яцишин С.І.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Львів – 2022 р.

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра інформаційних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Крошний І. М.

" "

_____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Лях Р.І.

(прізвище, ім'я, по батькові)

1. Тема роботи **Попередня обробка зображень для задач технічного зору**

керівник роботи, канд. техн. наук, доцент Яцишин Світлана Іванівна.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 13.09.2022 року № С-618

2. Термін подання студентом роботи 19. 12. 2022 р.

3. Вихідні дані до роботи:

- провести огляд фільтрів для попередньої обробки зображень;
- дослідити математичну модель побудови фільтрів для попередньої обробки зображень;
- розробити на мові програмування C# застосунок з інтуїтивним інтерфейсом;
- провести тестування роботи розробленого застосунка.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне забезпечення

Розділ 3. Математичне забезпечення

Розділ 4. Програмне забезпечення

5. Перелік графічного матеріалу (системний аналіз, розробка та відображення алгоритмів роботи стеганографічних методів захисту аудіо файлів, структура програмного рішення, експериментальна частина)

Додаток А. Вихідний код розробленого програмного забезпечення

6. Дата видачі завдання 01 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел	01.09-30.09.2022	виконано
2	Розділ 1. Стан проблемної області	01.10-15.10.2022	виконано
3	Розділ 2. Інформаційне забезпечення	16.10-31.10.2022	виконано
4	Розділ 3. Математичне забезпечення	01.11-15.11.2022	виконано
5	Розділ 4. Програмне забезпечення	16.11-30.11.2022	виконано
7	Експериментальна частина	01.12-09.12.2022	виконано
8	Оформлення дипломної роботи	10.12-14.12.2022	виконано
9	Підготовка до захисту дипломної роботи, оформлення презентації	15.05-17.12. 2022	виконано

Студент _____ Лях Р.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яцишин С.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська робота містить 63 сторінок пояснювальної записки, 31 рисунок, 1 таблицю, 1 додаток, 17 літературних джерела.

Мета роботи – розглянути проблеми попередньої обробки зображень для задач технічного зору на основі фільтрації.

Основні завдання: розглянути особливості побудови фільтрів для попередньої обробки зображень для задач технічного зору, запропонувати методи побудови фільтрів для попередньої обробки зображень, створити програму для побудови фільтрів.

Розроблений додаток дозволяє швидко синтезувати фільтри різкості, розмиття, границі, тиснення для попередньої обробки зображень в задачах технічного зору.

Ключові слова: попередня обробка зображень, числова лінійка-в’язанка, фільтр границі, фільтр різкості, фільтр розмиття, фільтр тиснення.

ABSTRACT

The master's thesis contains 63 pages of an explanatory note, 31 figures, 1 table, 1 appendix, and 17 literary sources.

The purpose of the work is to consider the problems of image preprocessing for technical vision problems based on filtering.'

The main tasks: consider the features of building filters for pre-processing images for technical vision problems, propose methods for building filters for pre-processing images, create a program for building filters.

The developed application allows you to quickly synthesize sharpness, blur, border, embossing filters for pre-processing images in technical vision tasks.

Keywords: blur filter, border filter, embossing filter, image pre-processing, number ruler-knitting, sharpness filter**.**

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	програмне забезпечення
ЧЛВ	числова лінійка-в’язанка
ЦФ	цифрова фільтрація

ВСТУП

Методи цифрової обробки дозволяють трансформувати зображення для поліпшення їх візуального сприйняття. Також у цій сфері вирішуються завдання зміни подання зображень з метою забезпечення їх зберігання, передачі, візуалізації в електронному вигляді та подальшого аналізу інформації, що міститься в них. Обробка цифрових зображень є галуззю науки, яка швидко розвивається. Дослідження та розробка методів і алгоритмів обробки та аналізу інформації, представленої у вигляді цифрових зображень, є дуже актуальним завданням.

У роботі наведено огляд методів фільтрації та сегментації зображень. Наведено приклади розглянутих методів вирішення задач удосконалення зорового зображення та виділення контурів об’єктів на зображеннях.

Актуальність теми. Вона полягає в можливості створення власних фільтрів, які можуть обробляти графічні зображення за різними критеріями, такими як збільшення різкості, контрастність, розмиття тощо.

Візуальне представлення даних набагато інформативніше інших методів отримання та сприйняття інформації. В умовах поширення засобів генерації візуальної інформації дуже актуальними є завдання обробки та підвищення якості зображень. Серед цих завдань однією з базових, класичних є проблема збільшення або, навпаки, зменшення контрастності, збентеження зображення та додаткові ефекти, такі як тиснення.

Для запобігання цим проблемам найкраще підходить одна з частин комп'ютерної графіки. Комп’ютерна графіка поділяється на три основні області: візуалізація, обробка зображень і розпізнавання образів. Візуалізація - це створення образу на основі будь-якого опису (моделі). Основним завданням розпізнавання образів є отримання семантичного опису представлених об'єктів. Обробка зображень відповідає за перетворення (фільтрацію) зображень. Розвиток сучасних засобів обчислювальної техніки та

інформаційних технологій сприяє широкому впровадженню в практику систем автоматичної обробки зображень.

Основне призначення такої системи – покращити якість зображення. Але завданням обробки зображень може бути як поліпшення якості (реставрація, реставрація) зображення відповідно до певної умови, так і спеціальне перетворення, яке кардинально змінює зображення. В останньому випадку обробка зображення може бути проміжним етапом для подальшого розпізнавання зображення (наприклад, виділення контуру об’єкта). Методи обробки зображень можуть значно відрізнятися залежно від того, як було отримано зображення - синтезовано системою комп'ютерної графіки чи оцифрованою чорно-білою чи кольоровою фотографією.

Багато підходів до фільтрації зображень діляться на дві великі категорії: обробка в просторовій області та обробка в частотній області. У магістерській роботі будуть розглянуті просторові (матричні) методи обробки зображень. Просторові методи поєднують підходи до обробки, засновані на маніпулюванні пікселями обробленого зображення.

Загальної теорії покращення зображення немає. Коли зображення обробляється для візуальної інтерпретації, спостерігач є остаточним суддею про те, наскільки добре працює певний метод.

Візуальна оцінка якості зображення є досить суб’єктивним процесом, що робить поняття «ідеальне зображення» невловимим еталоном для порівняння продуктивності алгоритму. Але коли проблема дозволяє встановити чіткі критерії якості, зазвичай потрібна певна кількість випробувань, перш ніж буде обрано конкретний підхід до покращення зображення.

Таким чином, актуальність розглянутої теми дослідження зумовлена необхідністю пошуку нових напрямків удосконалення проблеми обробки зображень із застосуванням фільтрів на основі матриці перетворення. Усе це покращить якість та ефективність уже розроблених методів розпізнавання та обробки зображень.

Мета дослідження. Мета магістерської роботи полягає у опрацюванні теоретичних положень за завданням обробки зображень; дослідженні вже існуючих методів фільтрації та застосування числової лінійки-в'язанки для розробки фільтрів.

Об'єкт дослідження – методи та алгоритми цифрової обробки растрових зображень.

Предмет дослідження - теоретичні та методичні основи обробки зображень за допомогою фільтрації у просторовій області.

Задачі дослідження. Вище описана мета визначила необхідність вирішення наступних завдань

- узагальнити теоретичні основи та виявити особливості завдання обробки зображень у сучасних інформаційних умовах;
- дослідити використання методів просторової обробки зображень;
- узагальнити існуючий досвід використання числових лінійок-в'язанок у галузі цифрової обробки зображень, розробити та обґрунтувати досліджені методи фільтрації на основі числових лінійок-в'язанок.

Методи дослідження. Теоретичну і методологічну основу дослідження склали роботи провідних вчених та фахівців в сфері цифрової обробки зображень, а саме такі як Гонсалес Р., Хуанг Т.С., Яне Б., Яншин В.В., Калинин Г.А., Сойфер В.А. ті інші. У роботі були використані загальнонаукові методи: спостереження, порівняння, узагальнення, абстрагування, формалізація, аналіз та синтез, застосовувалися положення теорії множин, економіко-математичне моделювання (для оцінки ефективності системи та вибору технічних та програмних засобів).

Практична цінність одержаних результатів. Виходячи з виконаних теоретичних та експериментальних досліджень, практичний результат магістерської роботи полягає у використанні стандартних алгоритмів

фільтрації при застосуванні нееквідистантних структур типу числових лінійних в'язанок.

В рамках цього напрямку отримано такі основні результати:

- розроблено концептуальну модель фільтрів на основі числових лінійних в'язанок;
- досліджено застосування фільтрації зображень на основі числових лінійних в'язанок, а саме фільтрів розмиття, різкості, меж, тиснення;
- розроблений інтерфейс для обробки зображень за допомогою фільтрів, який наочний і простий у використанні.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

Цифровим зображенням прийнято називати двовимірний, дискретний сигнал, який отримується в результаті перетворення безперервного (аналогово) зображення в еквівалентний цифровий масив [5]. Під цифровою обробкою зображення зазвичай розуміється математична обробка сигналу, заданого матрицею дискретних відліків. Обробка здійснюється шляхом застосування до вхідного зображення різних перетворень, цифрова реалізація яких часто називається цифровою фільтрацією. З математичної точки зору процес фільтрації може бути представлений як перетворення безлічі вхідних зображень на безліч вихідних за допомогою деякого оператора.

Цифрова фільтрація у зображенні зазвичай спрямована на досягнення однієї з таких цілей [5].

1. Перетворення за деяким законом та відповідно до заданих вимог дискретних відліків зображення внаслідок чого формується нове зображення.
2. Отримування з зображення будь-якої корисної інформації про об'єкт дослідження, наприклад, як деякого вектору атрибутів зображення.

Цілі цифрової фільтрації досягаються використанням різних алгоритмів та методів обробки, кожен з яких спрямований на вирішення конкретної задачі. Завдання цифрової обробки зображень за своїм прикладним призначенням може бути класифіковано, наприклад, наступним чином [6, 9]: підвищення якості зображень, препарування зображень, сегментація зображень, морфологічна обробка зображень, виявлення низькорівневих ознак у зображенні, стиск зображень.

Кожна із задач цифрової фільтрації містить багато підзадач, для вирішення яких розроблено велику кількість алгоритмів і методів, заснованих на різних теоретичних підходах, які використовуються залежно від конкретних вимог до результатів їх застосування.

Алгоритми та методи цифрової фільтрації поділяються на дві групи. До першої групи належать методи, засновані на представленні зображень та імпульсних характеристик фільтрів у вигляді матриці дискретних значень, звані методами цифрової фільтрації в просторовій області або методами просторової фільтрації. До другої групи належать методи фільтрації в частотній області або методи частотної фільтрації, засновані на перетвореннях спектру зображення [5].

Процеси просторової обробки, як правило, описуються рівнянням виду (1.1) [4].

$$g[x, y] = T(f[x, y]) \quad (1.1)$$

де f (x, y) і g (x, y) - вхідне і вихідне зображення відповідно,

T - оператор над f, визначений в деякому околі точки [x, y].

Оператор T реалізує будь-яке перетворення, яке обчислює значення g в точці [x, y] на основі значень f у заданому околі.

Одне з основних досліджень такої постановки задачі просторової фільтрації сприяло участі масок, коефіцієнти яких визначають ступінь процесу. У цьому випадку оператор T в (1.1) вигідно скручує зображення f [x, y] з маскою (або фільтром) h [x, y] [5, 9].

Методи частотної фільтрації створені на основі перетворення частоти в зображення. У цьому випадку модель оцінки є результатом рівнянням виду [5, 9]

$$g[x, y] = F^{-1}(F(f[x, y])\, F(h[x, y])) \quad (1.2)$$

де F - пряме частотне перетворення,

F^{-1} - зворотне частотне перетворення.

У цьому випадку частотна фільтрація зводиться до обчислення прямого частотного перетворення вхідного зображення, зміни результуючого зображення шляхом його множення на зображення функції фільтра та обчислення зворотного частотного перетворення.

Якщо F і F^{-1} є простими та оберненими перетвореннями Фур'є, то взаємозв'язок методів фільтрації просторової та частотної області

встановлюється теоремою про згортку, яка стверджує, що просторова згортка двох функцій f[x, y] * h[x, y] може бути отримана з використанням зворотного перетворення Фур'є до добутку їх Фур'є-образів в частотній області, тобто

$$f[x, y] * h[x, y] \leftrightarrow \dot{f}[w1, w2]\, \hat{h}[w1, w2] \qquad (1.3)$$

де $\dot{f}$, $\hat{h}$ - Фур'є-образ деякого сигналу, і навпаки, згортка в частотній області може бути отримана застосуванням прямого перетворення Фур'є до твору в просторовій області [12]:

$$f[x, y]\, h[x, y] \leftrightarrow \dot{f}[w1, w2] * \hat{h}[w1, w2] \qquad (1.4)$$

Вважаючи в (1.3), (1.4) f[x, y] рівним одиничному імпульсу, можна показати взаємозв'язок фільтрів в просторовій і частотній областях, який дозволяє легко здійснити перехід від просторових фільтрів до частотних і навпаки [5]. Це виявляється корисним для вирішення проблем, які важко реалізувати в просторовій області, але тривіально розв’язуваних у частотній області, або навпаки.

При розв’язуванні задач цифрової фільтрації вибір методу фільтрації (просторової чи частотної) базується на вимогах до результатів розрахунку, швидкості обробки даних, алгоритму розв’язання, реалізованого у просторовому чи частотному представленні [9] тощо. описані та вирішені в просторовій області, наприклад, виявлення країв, інші в частотній області, наприклад, виділення меж [5].

Як просторові, так і частотні фільтри, які використовуються в практиці цифрової обробки зображень, мають свої переваги і недоліки. Таким чином, методи просторової фільтрації дозволяють аналізувати локальні властивості нестаціонарних і просторово неоднорідних сигналів, але часто вимагають попередньої обробки зображень через вплив характеристик яскравості. З іншого боку, методи частотної фільтрації дозволяють аналізувати глобальні особливості зображень (наприклад, частоту), а також дозволяють аналітично описувати фільтри.

Таким чином, вибір методу фільтрації залежить від розв'язуваної задачі, типу сигналу, вимог до точності та швидкості.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Фільтрація зображень

Цифрові зображення схильні до різного роду шумів, які можуть бути результатом методу отримання зображень, технології передачі інформації, методів оцифровки даних. Процес видалення різного роду шумів із зображень називається фільтрацією.

Під час фільтрації характеристики яскравості кожної точки цифрового зображення замінюються на інше значення яскравості, яке розпізнається як найменш спотворена перешкода [1, 2].

Методи частотного перетворення зображення [1-6] засновані на ідеї перетворення Фур'є, сенс якого полягає в представленні початкової функції у вигляді суми тригонометричних функцій різних частот, помножених на задані коефіцієнти. Якщо функція є періодичною, то таке зображення називається рядом Фур’є. Інакше неперіодичну функцію зі скінченною площею під графіком можна виразити як інтеграл від тригонометричних функцій, помножених на деяку вагову функцію [1].

Ця опція називається перетворенням Фур’є і є більш корисною, ніж ряд Фур’є, у більшості практичних завдань. Важливою властивістю є те, що функція, представлена перетворенням Фур'є, після виконання над нею перетворень може повернутися до свого початкового вигляду. В результаті такий підхід дозволяє обробляти функцію в частотній області, а потім повертати її до початкової форми без втрати інформації. Перетворення Фур'є також можна використовувати для вирішення проблем фільтрації зображень. На практиці реалізація частотних підходів може бути подібною до методів просторової фільтрації [1].

Методи покращення просторового зображення застосовуються до растрових зображень, представлених у вигляді двовимірних матриць. Принцип роботи просторових алгоритмів полягає в застосуванні спеціальних

операторів до кожної точки вихідного зображення. Оператори - це прямокутні або квадратні матриці, які називаються масками, ядрами або вікнами [1, 7, 8]. Зазвичай маска є невеликим двовимірним масивом, і методи покращення, засновані на цьому підході, часто називають обробкою маски або фільтрацією маски.

У лінійній фільтрації відгук маски є сумою добутків пікселів у зоні покриття фільтра. Як лінійний згладжуючий фільтр використовується усереднювальний фільтр з початковим значенням, яке є середнім значенням по краю маски фільтра [1]. Подібний фільтр використовується для видалення зернистості зображення, викликаної імпульсним шумом [9, 7].

Принцип роботи нелінійних просторових методів подібний до лінійних фільтрів. Операції, які виконує нелінійний фільтр, залежать від значень елементів матриці зображення, розташованих в досліджуваному околі. Наприклад, робота нелінійного фільтра може полягати в обчисленні медіани елементів зображення аналізованої околиці [1].

Для медіанної фільтрації значення пікселя є середнім значенням точок у відповідному околі. Часто при вирішенні задач шумозаглушення медіанний фільтр ефективніше простого усереднення, оскільки призводить до меншого спотворення меж виділених об'єктів. В якості маски для медіанної фільтрації використовується двовимірне вікно з центральною симетрією, центр якого знаходиться в поточній точці фільтрації [2].

Адаптивна фільтрація базується на фільтрі Вінера, який є одним із типів лінійних фільтрів для адаптивної локальної обробки зображень [2]. Якщо значення стандартного відхилення інтенсивності пікселя в цій локальній області велике, фільтр Вінера виконує мале згладжування, і навпаки, при меншому відхиленні площа згладжування більша [2]. Цей підхід часто ефективніший, ніж звичайна лінійна фільтрація. Ще однією перевагою адаптивного фільтра є збереження країв та інших частин високочастотних об’єктів на зображенні. Однак для обчислення фільтра Вінера потрібно більше часу, ніж для лінійного фільтра [2].

На рис. 2.1 показані результати фільтрації при накладенні імпульсного шуму на цифрове зображення. На рис. 2.2 представлені результати фільтрації накладеного на цифрове зображення білого шуму Гауса.

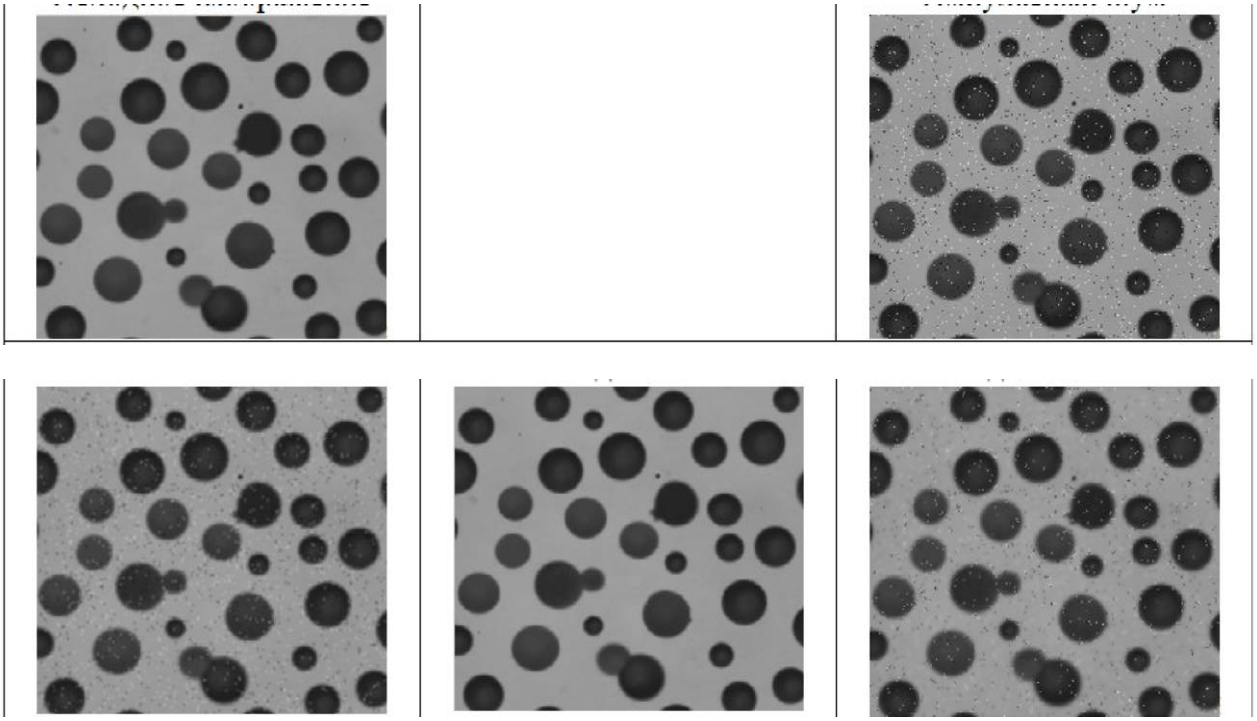

Рис. 2.1. Результати фільтрації імпульсного шуму на зображенні

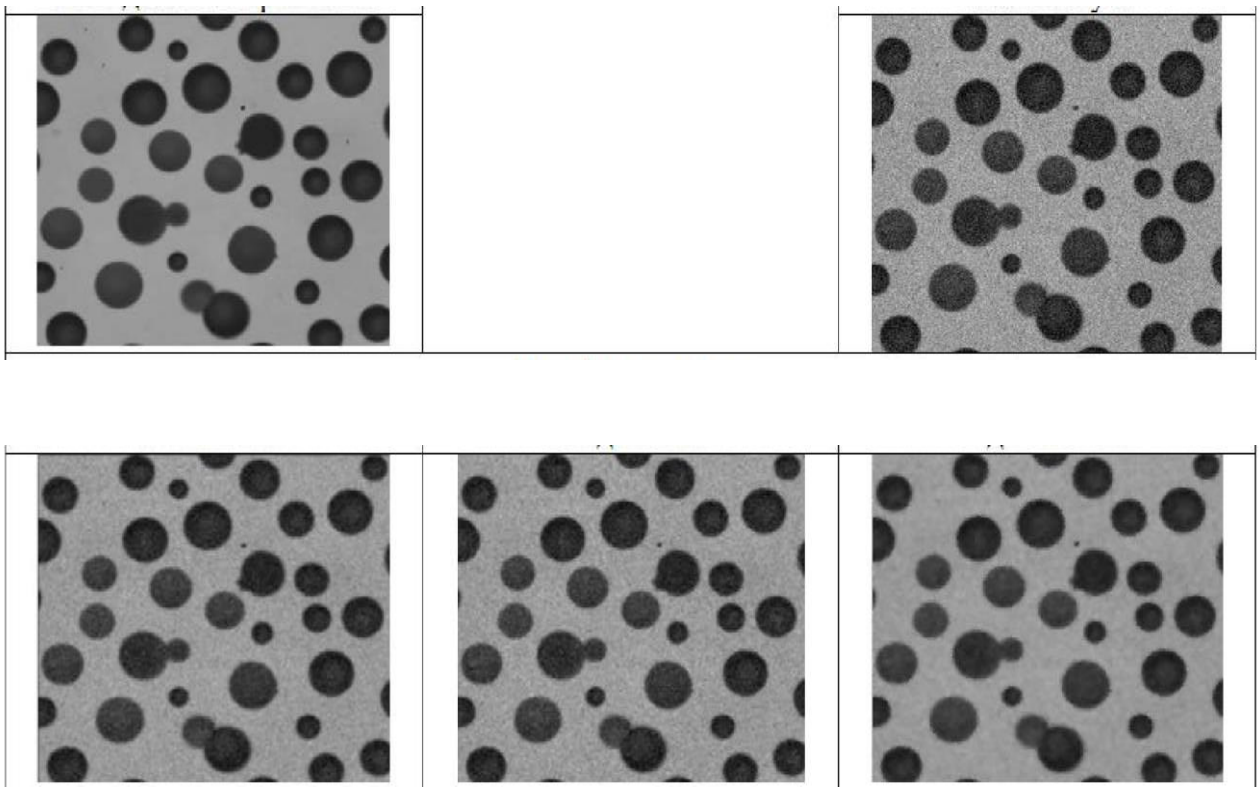

Рис. 2.2. Результати фільтрації білого шуму на зображенні

Цілі просторової фільтрації зазвичай стосуються покращення якості зображення; найчастіше це усунення перешкод або загострення, підкреслення контурів. Просторова фільтрація має досить багатий арсенал засобів обробки для цих цілей, від алгоритмів усереднення країв до операцій локального диференціювання та покращення країв. Розглянемо загальний алгоритм фільтрації та конкретний алгоритм лінійної фільтрації [4].

Фільтрація зображення має багато спільного з добре відомою в радіотехніці фільтрацією сигналу. Відмінність полягає в тому, що замість «одновимірного» фільтра, призначеного для фільтрації зашумленого сигналу з однією змінною (зазвичай часом), для фільтрації зображень використовуються двовимірні фільтри, відповідні двовимірності самого зображення.

Такий двовимірний фільтр встановлюється наступним чином. Береться невелика, зазвичай прямокутна ділянка площини, і на ній визначається якась функція. Зазначена область називається апертурою або вікном, а функція, визначена в ній, називається ваговою функцією або віконною функцією.

Таким чином, кожен елемент відкриття відповідає числу (за бажанням користувача), яке буде називатися ваговим коефіцієнтом. Сума всіх вагових коефіцієнтів є ваговою функцією. Діафрагму разом із певною функцією зважування часто називають маскою. Важливою перевагою останнього терміну є його стислість.

Діафрагма зазвичай невелика, наприклад 3x3 або 5x5 елементів. Лінійні розміри отвору зазвичай приймаються непарними, щоб можна було однозначно визначити його центральний елемент, але ніщо не заважає матриці бути парною.

Фільтрування здійснюється переміщенням діафрагми фільтра по зображенню. У кожному положенні діафрагми виконуються ті самі дії, які визначають реакцію фільтра. При переміщенні вікна вагова функція залишається незмінною (стаціонарний фільтр). Тому фільтрацію ковзного вікна називають просторово інваріантними операціями [2].

Типовим прикладом є алгоритм лінійної фільтрації, який у загальному вигляді виглядає наступним чином. У кожній позиції діафрагми вагова функція поелементно множиться на значення відповідних пікселів вихідного зображення; підводиться підсумок роботи. Сума ділиться на коефіцієнт нормалізації, і отримане значення, яке є відгуком фільтра, присвоюється пікселю нового (відфільтрованого) зображення, що відповідає положенню центру діафрагми [9].

Тут вибрано середину, тому що в іншому випадку (якщо, скажімо, ми присвоїмо значення відповіді пікселю, що відповідає верхньому лівому елементу вікна), фільтрація супроводжується небажаним зсувом відфільтрованого зображення щодо оригіналу.

Нормувальний коефіцієнт, згаданий вище, зазвичай приймається рівним сумі всіх елементів вагової функції (вагових коефіцієнтів). Зазвичай, але не завжди.

У лінійних фільтрах відповідь є лінійною функцією багатьох змінних, що відтворюються пікселями, що потрапляють у вікно. Ваговими коефіцієнтами є коефіцієнти зазначеної лінійної функції.

Фільтри, у яких відповідь не можна виразити як лінійну функцію значень пікселів, є нелінійними за визначенням.

Залежно від того, де (в якому полі) зберігається відповідь фільтра, розрізняють звичайні та рекурсивні фільтри.

Якщо в простих фільтрах відгук записується у вихідне зображення, то в рекурсивних фільтрах він записується назад у вихідне зображення, змінюючи значення пікселя безпосередньо в процесі фільтрації. Тому в рекурсивних фільтрах вже оброблені пікселі впливають на результат фільтрації наступних.

Іноді корисно повторити процедуру фільтрації багато разів; у таких випадках говорять про ітераційне застосування фільтрів.

Фільтрування вимагає переміщення діафрагми по зображенню, що обробляється. У програмах ми використовуємо один тип руху, схожий на телевізійну прогресивну розгортку. Діафрагма рухається вздовж лінії зліва направо з кроком 1 піксель; коли він досягає кінця одного рядка, він переходить до початку наступного. Це найпоширеніший, але далеко не єдиний вид руху. Траєкторія переміщення вікна може бути будь-якою, необхідно лише, щоб центр вікна відвідав усі пікселі один раз.

З точки зору фільтрації результатів, при програмуванні простих (тобто нерекурсивних) алгоритмів неважливо, який вид руху, яку траєкторію вікна вибрати - результати від цього не залежать. Тип трафіку може впливати на інші характеристики, наприклад на швидкість алгоритму.

Тепер розглянемо ще один аспект руху вікон, який рідко згадується в літературі. Мова піде про так званий ефект краю та як обробляти пікселі поблизу меж поля.

Ефект крайових ефектів особливо помітний на маленьких зображеннях. Якщо розмір зображення набагато більший за розмір вікна, то частка області відфільтрованого поля, де видно краєві ефекти, мала. У цьому випадку

ефектами країв можна знехтувати, а алгоритм обробки можна вибрати на основі інших міркувань, таких як максимальна швидкість [14].

Проблему крайового ефекту важко назвати фундаментальною, але її потрібно якось вирішувати в програмуванні.

Елементи діафрагми, які виходять за межі поля, не беруть участь в обробці, тому відгук фільтра формується на основі меншої кількості точок, ніж у центральній зоні. Іншими словами, фактичний розмір вікна мініатюризовано по краях зображення. Крайовий ефект зберігається, хоча й у значно послабленій формі.

Відгук лінійного фільтра залежить від зображення, яке обробляється. Розглянемо фільтри, в яких для кожного положення діафрагми здійснюється елементарне множення вагової функції на значення відповідних елементів зображення, підсумовування добутків і нормалізація отриманої суми.

Введемо необхідні позначення. Нехай апертура має розмір M_p x M_q елементів; поточний елемент апертури позначимо через:

p=1, 2,... M_p - поточна стрічка;

q=1,2,…M_q – поточний стовпчик.

Давайте визначимо метод вказівки положення діафрагми на зображенні. Вибирається певний орієнтир діафрагми (зазвичай центр, рідше один з кутових елементів). Тепер достатньо розташувати цю опорну точку в системі координат зображення, щоб визначити положення всієї апертури.

Цю точку відліку будемо називати умовним центром отвору; координати умовного центру позначимо через (p_m, q_m). Умовний центр може (але не обов'язково) збігатися з геометричним центром отвору. Взагалі умовним центром можна вважати будь-яку точку апертури; більше того, умовний центр навіть не обов'язково знаходиться всередині неї - можна, скажімо, задати p_m=0, q_m=0. Ми, однак, визначаємо умовний центр так, щоб при непарних розмірах апертури він збігався з її центральним пікселем:

$$p_m = \left[\frac{M_p+1}{2}\right] \qquad q_m = \left[\frac{M_q+1}{2}\right],$$

де квадратні дужки позначають цілу частину числа.

Різні типи мережевих фільтрів відрізняються ваговими функціями та коефіцієнтом нормування [15].

Як правило, використовуються елементи отвори 3x3; збільшення розміру апертури значно збільшує обсяг обчислень, при цьому якість обробки трохи покращується. Втім, із зростанням продуктивності процесорів ЕОМ обчислювальні витрати все менше лімітують розмір застосовуваних апертур, і останнім часом у вживання входять апертури розміром 5x5 і навіть 7x7 елементів.

Тепер розглянемо деякі конкретні приклади лінійних фільтрів.

Одним із найпоширеніших застосувань лінійних фільтрів є згладжування шуму. У частотній інтерпретації процесів фільтрації перешкодозахисним фільтром є фільтр нижніх частот [4, 9].

Ідея використання згладжуючих фільтрів досить зрозуміла. Заміною вихідних значень елементів зображення на середні значення маски фільтра досягається зменшення «різких» переходів рівнів яскравості. Оскільки випадковий шум характеризується просто різкими стрибками яскравості, найбільш очевидним застосуванням згладжування є шумозаглушення. Однак контури, які зазвичай представляють інтерес для зображення, також мають різкі зміни яскравості, тому недоліком використання фільтрів згладжування є те, що контури розфокусовані. Іншим застосуванням такої обробки може бути згладжування хибних контурів, які виникають під час трансформацій з недостатнім рівнем яскравості. Основним використанням фільтрів згладжування є придушення «нерелевантних» деталей зображення. Під «незначними» тут ми маємо на увазі набори точок, малі порівняно з розміром маски фільтра.

Для цього застосовуються вагові функції наступного вигляду:

$$H = \frac{1}{9} \| 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \|.$$

Цей фільтр дає середнє нормальне значення відносно маски. Зауважте, що коефіцієнти фільтра вказані в одиницях замість 1/9. Причина в тому, що цей варіант більш ефективний для комп'ютерних розрахунків. Після завершення процесу підсумовування отримане значення ділиться на 9. Маска розмірами mxn матиме нормувальний коефіцієнт, що дорівнює 1/mn. Такий просторовий фільтр, всі коефіцієнти якого однакові, іноді називають однорідним усереднюються фільтром:

$$\square = \frac{1}{16}\|1\ 2\ 1\ 2\ 4\ 2\ 1\ 2\ 1\| \qquad \square = \frac{1}{16}\|2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2\|.$$

Друга маска цікавіша. Ця маска дає середнє зважене; цей термін використовується, щоб показати, що значення елементів множаться на різні коефіцієнти, що дозволяє їм надавати різні ваги порівняно з іншими. У масці фактор у середині маски має найбільше значення (вагу), що надає відповідному елементу більшої ваги при обчисленні середнього. Значення решти коефіцієнтів в масці зменшуються в міру віддалення від центру маски. Діагональні смуги, порівняно з ортогональними, розташовані далі від центру і тому «важать» менше, ніж найближчі сусіди центрального елемента. Основною стратегією надання найбільшої ваги центральному пікселю та іншим пікселям, обернено пропорційним їх відстані, є зменшення розфокусування під час згладжування. Ви можете вибрати інші значення для коефіцієнтів маски для досягнення ваших цілей, але сума коефіцієнтів дорівнює 16, що зручно для комп’ютерної реалізації, оскільки це ступінь двійки. Зауважте, що на практиці досить важко побачити різницю між зображеннями, згладженими фільтрами за допомогою однієї з масок або іншого подібного дизайну, тому що розмір області, охопленої маскою при фільтрації одного елемента, дуже малий [13].

Фільтри підвищення різкості (частота інтерпретується як фільтри високих частот) використовують такі три типові вагові функції:

$$\square = \|0\ -1\ 0\ -1\ 5\ -1\ 0\ -1\ 0\| \square = \|-1\ -1\ -1\ -1\ 9\ -1\ -1\ -1\ -1\| \square = \|1\ -2\ 1\ -2\ 5\ -2\ 1\ -2\ 1\|.$$

Основна мета підвищення різкості полягає в тому, щоб виділити дрібні деталі на зображенні або підкреслити ті деталі, які не сфокусовані через помилки або недосконаліть самого методу зйомки. Підвищення чіткості зображення має широкий спектр застосування – від електронного друку та медичної інтроскопії до технічного контролю промисловості та систем автоматичного наведення у військовій сфері [10].

Згладжуючий фільтр Гаусса також відноситься до категорії лінійних фільтрів. Замість рівномірного розподілу цей фільтр надає найбільшу вагу центральному пікселю. Розмиття за Гауссом швидко розмиває виділення на змінну величину. Використання цього фільтра зменшує кількість деталей і дозволяє створити ефект занурення в туман:

$$H = \|1\ \ 2\ \ 3\ \ 2\ \ 1\,2\ \ 4\ \ 5\ \ 4\ \ 2\,3\ \ 5\ \ 6\ \ 5\ \ 3\,2\ \ 4\ \ 5\ \ 4\ \ 2\,1\ \ 2\ \ 3\ \ 2\ \ 1\|.$$

Фільтр визначення краю належить до категорії фільтрів виявлення краю з коефіцієнтами матриці з нульовою сумою. Фільтр замінює ділянки суцільного кольору чорним, а області з варіаціями – нечорним кольором:

$$\mathrm{H} = \|-1\ \ -1\ \ -1\ 0\ 0\ 0\ 1\ 1\ 1\|.$$

Фільтр тиснення реалізовується за допомогою матриці:

$$\mathrm{H} = \|-1\ 0\ 1\ \ -2\ 0\ 2\ \ -1\ 0\ 1\| \text{ або } \mathrm{H} = \|-1\ \ -1\ \ -1\ \ -1\ 8\ \ -1\ \ -1\ \ -1\ \ -1\|.$$

У той час як ядра розмиття та підвищення різкості мали суму коефіцієнтів, рівну одиниці, у цьому випадку сума вагових коефіцієнтів у ядрі тиснення дорівнює нулю. Якщо сума коефіцієнтів не дорівнює 0, то ми отримаємо відхилення до будь-якого або конкретного кольору. Отримане значення кольору буде додатково оброблено (усереднено) і доведено до діапазону 0-255.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Алгоритми фільтрації графічних зображень

Процес просторової фільтрації заснований на простому переміщенні маски фільтра від точки до точки зображення; у кожній точці (x, y) відповідь фільтра обчислюється за допомогою попередньо визначених посилань. Для лінійної просторової фільтрації зворотний зв'язок є сумою добутку коефіцієнтів фільтра і відповідних значень пікселів в області, охопленої маскою фільтра.

Для маски 3x3 елемента, показаної на рис. 3.1, результат (відгук) R лінійної фільтрації в точці (x, y) зображення складе:

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

який, очевидно, є сумою добутків коефіцієнтів маски та значень пікселів безпосередньо під маскою. Зокрема, зауважте, що коефіцієнт w(0,0) слідує за f(x, y), вказуючи, що маска центрується в точці (x, y).

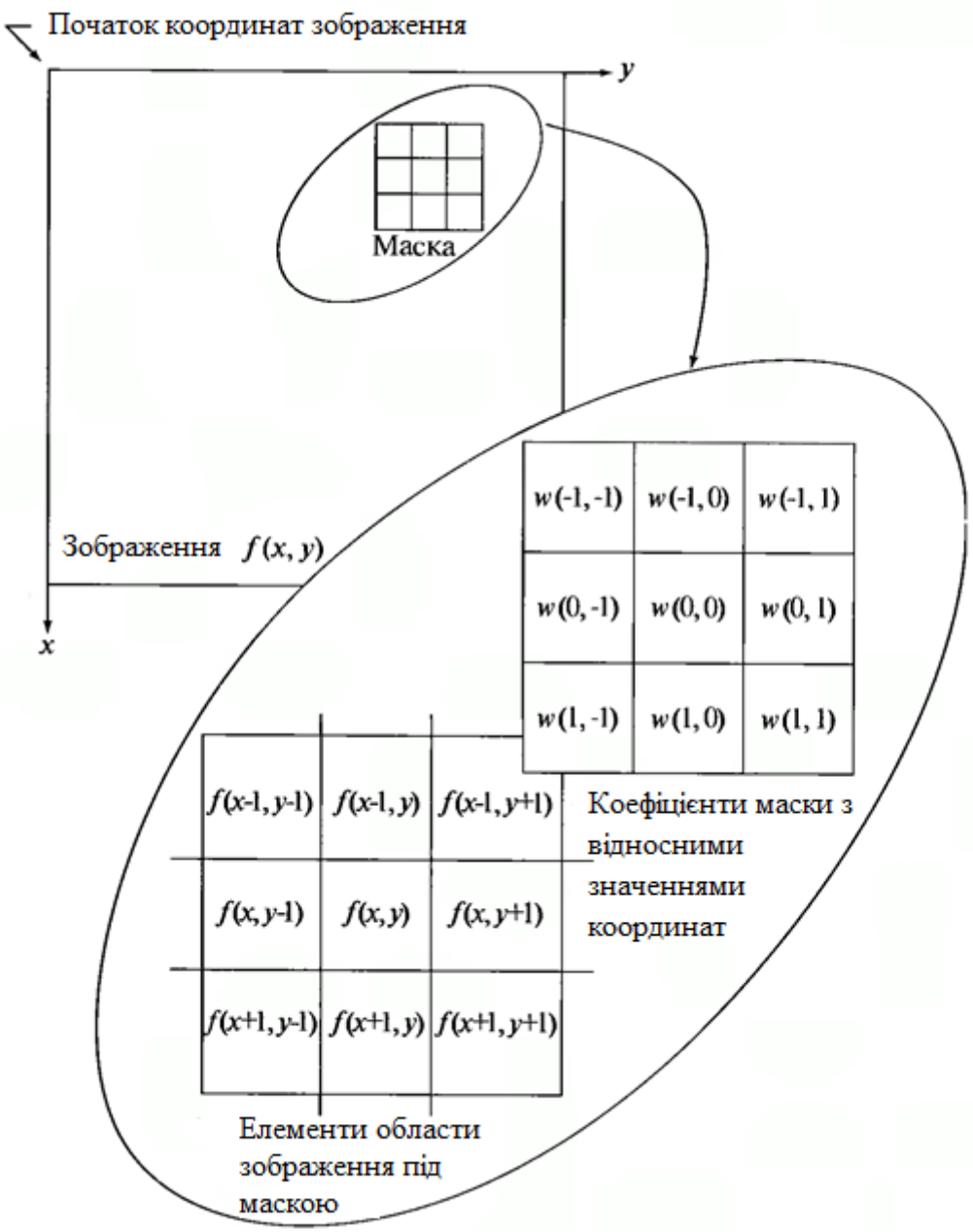

Рис. 3.1. Схема просторової фільтрації

Фільтрація виконується переміщенням маски зліва направо (або зверху вниз) на один піксель. У кожній позиції діафрагми виконується вищезгадана операція, а саме: вагові коефіцієнти множаться на відповідні значення яскравості вихідного зображення, а результати підсумовуються. Отримане значення присвоюється середньому (i, j) пікселю. Зазвичай це значення ділять на певне число (коефіцієнт нормування). Маска містить непарну кількість рядків і стовпців, щоб центральний елемент був однозначно визначений, але це не обов’язково.

Давайте розглянемо деякі фільтри. Першим фільтром буде ефект розмиття. Для того, щоб розмити зображення нам потрібно занести в пам'ять значення RGB складових кольору кожного пікселя.

Тоді ядро розмиття буде застосовано до всіх колірних компонентів усіх пікселів відредагованого зображення:

$$H = \frac{1}{9}\|1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\| \qquad H = \frac{1}{25}\|1\ 3\ 1\ 3\ 9\ 3\ 1\ 3\ 1\|.$$

Матриця для фільтру розмиття може бути не тільки 3x3, але й 5x5, 7x7 і більше. Для прикладу матриця 5x5:

$$H = \frac{1}{158}\|0\ \ 1\ \ 2\ \ 1\ \ 0\ 1\ \ 3\ \ 10\ 3\ \ 1\ 2\ \ 10\ 90\ 10\ 2\ 1\ \ 3\ \ 10\ 3\ \ 1\ 0\ \ 1\ \ 2\ \ 1\ \ 0\|.$$

Щоб визначити колір пікселя під центром ядра, помножте вагові коефіцієнти ядра на відповідні значення кольору відредагованого зображення. Потім підводяться підсумки.

Отримане зображення «змазане» порівняно з оригіналом, оскільки колір кожного обробленого пікселя поширюється на сусідні пікселі.

Щоб збільшити ядро розмиття, ви можете використовувати ядро більшого розміру (таким чином поширюючи колір на більшу кількість суміжних пікселів) і змінюючи фактори таким чином, щоб зменшити вплив центрального фактору.

Підкреслення і різкість контурів або підвищення різкості зображення відбувається за рахунок збільшення високочастотних складових сигналу, до складу яких входять не тільки компоненти контурів і кордонів, а й шуми. Можливості реалізації цієї процедури за допомогою локальних фільтрів дуже різноманітні. Диференціальний оператор, написаний для обробки дискретного зображення, обчислює різницю яскравості у вікні. У загальному вигляді диференціальні оператори є лінійними:

$$d = \frac{\partial^n f(x,y)}{\partial x^k \partial y^{n-k}}.$$

Ізотропні алгоритми потрібні для чіткості меж довільно орієнтованих структур. Вони можуть бути непарними (оператори градієнта) або парними (оператори Лапласа). Недоліком диференціальних операторів є посилення шуму (високочастотної інтерференції) в результаті підсилення високих просторових частот. Чим вищий порядок диференціації, тим сильніше оператори реагують на високочастотний шум. Щоб зменшити ці ефекти, має сенс виконати шумозаглушення перед використанням диференціальних операторів. Іншим способом зниження чутливості до шуму є розробка диференціальних операторів, які створюють різницю середніх значень.

Під час створення ефекту різкості ми використовуємо той самий алгоритм, але з іншим ядром, оскільки тепер наша мета — підвищити різкість зображення.

Ядро для збільшення різкості:

$$\square = \| -1\ -1\ -1\ -1\ 9\ -1\ -1\ -1\ -1 \| \qquad \square = \| -1\ -3\ -1\ -3\ 41\ -3\ -1\ -3\ -1 \|.$$

Як і в попередньому випадку, ми обробляємо компоненти RGB окремо, а потім створюємо значення кольору обробленого пікселя. Щоб збільшити контраст між центральним пікселем і сусідніми пікселями, ми використовуємо негативні вагові коефіцієнти.

Це робить отримане зображення чіткішим за вихідне. По суті, додаткова деталізація з’явилася нізвідки – це просто підвищений контраст між кольорами пікселів.

Матриця для виділення контурів зображення.

$$\square = \| -1\ -1\ -1\ 0\ 0\ 0\ 1\ 1\ 1 \|.$$

Тиснення виконується аналогічно. Але його матриця:

$$\square = \|0\ 1\ 2\ 0\ 0\ 0\ 0\ -1\ -2\ \|.$$

У той час як ядра розмиття та підвищення різкості мали суму коефіцієнтів, рівну одиниці, у цьому випадку сума вагових коефіцієнтів у ядрі тиснення дорівнює нулю. Якщо сума коефіцієнтів не дорівнює 0, ми отримаємо відхилення до якогось конкретного кольору.

Недоліком таких прямокутних фільтрів є те, що при високих просторових частотах в зображеннях можуть виникнути помилки. Істотним недоліком лінійної фільтрації зображення є те, що разом із шумозаглушенням одночасно розмиваються контури зображення. Це пояснюється тим, що всі елементи вихідного зображення обробляються з однаковим фактором.

3.2. Концептуальна модель цифрової обробки зображень

Концептуальна модель цифрової обробки зображень представлена на рис. 3.2.

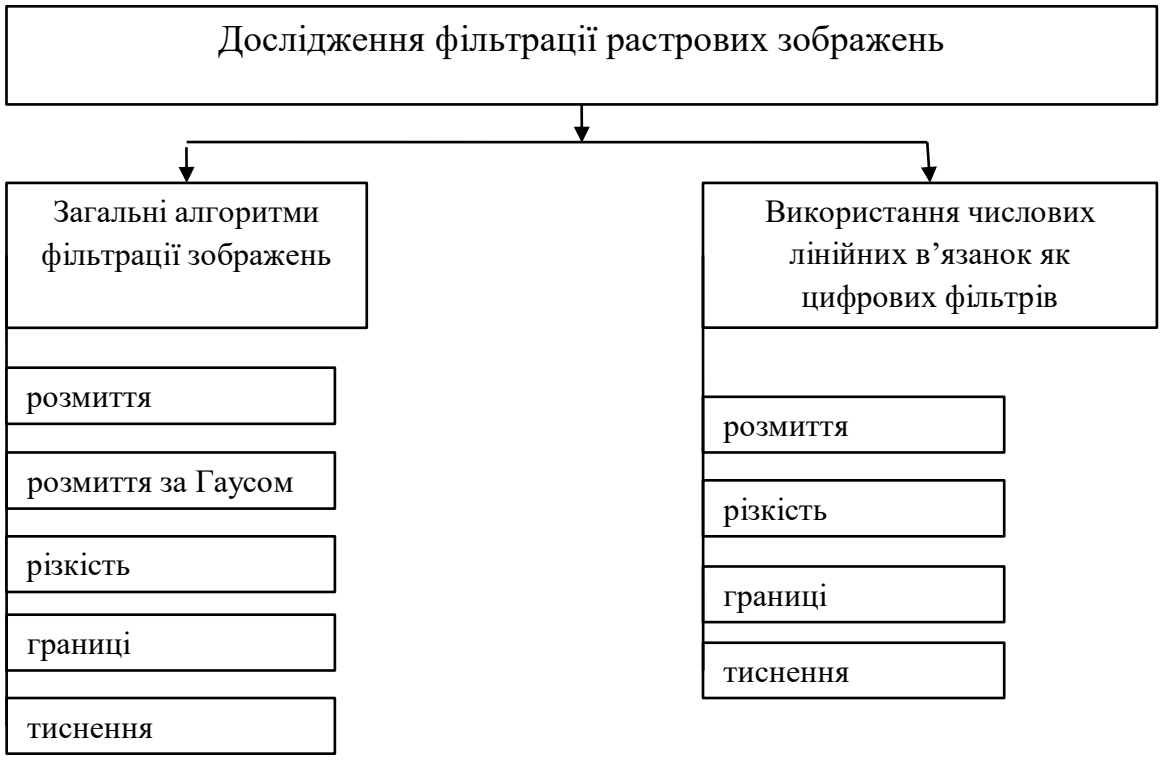

Рис. 3.2. Концептуальна модель

У результаті досліджень із застосуванням розробленої програми обробки зображень з використанням числових лінійок-в'язанок як недистанційних структур на стандартному алгоритмі просторової фільтрації отримано нові типи матриць перетворення, які виявилися ефективнішими за відомі стандартні. А саме, розмиття матриці фільтра, збільшення різкості, контурування зображення та тиснення.

В якості коефіцієнтів матриць перетворення були взяті числові лінійки в'язанки, а саме (табл. 4.1.):

Таблиця 4.1.

N (порядок)	Прості компактні ЧЛВ
3	1, 3, 2
4	3, 1, 5, 2
5	1, 3, 6, 2, 5 1, 7, 3, 2, 4
6	1, 10, 5, 3, 4, 2
6	1, 4, 2, 8, 3, 9
7	1, 3, 5, 6, 7, 10, 2
	4, 1, 12, 2, 6, 3, 7
8	1, 4, 7, 13, 12, 8, 6, 3
	2, 8, 14, 1, 4, 7, 6, 3

Використання цих лінійок дало досить хороші результати. Для досягнення ефекту розмиття можна використовувати наступні матриці:

$$H = \|1\ 3\ 2\ 2\ 1\ 3\ 3\ 2\ 1\ \| \quad H = \|3\ 1\ 5\ 2\ 2\ 3\ 1\ 5\ 5\ 2\ 3\ 1\ 1\ 5\ 2\ 3\ \|$$

$$H = \|1\ 3\ 6\ 2\ 5\ 5\ 1\ 3\ 6\ 2\ \ 2\ 5\ 1\ 3\ 6\ 6\ 2\ 5\ 1\ 3\ 3\ 6\ 2\ 5\ 1\ \| \ H =$$

$$\|1\ 7\ 3\ 2\ 4\ 4\ 1\ 7\ 3\ 2\ 2\ 4\ 1\ 7\ 3\ 3\ 2\ 4\ 1\ 7\ 7\ 3\ 2\ 4\ 1\ \|$$

$$H =$$

$$\|1\ 10\ 5\ 3\ 4\ 2\ 2\ 1\ 10\ 5\ 3\ 4\ 4\ 2\ 1\ 10\ 5\ 3\ 3\ 4\ 2\ 1\ 10\ 5\ 5\ 3\ 4\ 2\ 1\ 10\ \ 10\ 5\ 3\ 4\ 2\ 1\ \|$$

$$H = \|1\ 4\ 2\ 8\ 3\ 9\ \ 9\ 1\ 4\ 2\ 8\ 3\ 3\ 9\ 1\ 4\ 2\ 8\ 8\ 3\ 9\ 1\ 4\ 2\ 2\ 8\ 3\ 9\ 1\ 4\ 4\ 2\ 8\ 3\ 9\ 1\ \|$$

$$H =$$

$$\|1\ 3\ 5\ 6\ 7\ 10\ 2\ 2\ 1\ 3\ 5\ 6\ 7\ 10\ 10\ 2\ 1\ 3\ 5\ 6\ 7\ 7\ 10\ 2\ 1\ 3\ 5\ 6\ \ 6\ 7\ 10\ 2\ 1\ 3\ 5\ 5\ 6\ 7\ 10\ 2\ 1\ 3$$

$$H =$$

$$\|4\ 1\ 12\ 2\ 6\ 3\ 7\ 7\ 4\ 1\ 12\ 2\ 6\ 3\ 3\ 7\ 4\ 1\ 12\ 2\ 6\ 6\ 3\ 7\ 4\ 1\ 12\ 2\ 2\ 6\ 3\ 7\ 4\ 1\ 12\ 12\ 2\ 6\ 3\ 7\ 4\ 1$$

$$H =$$

$$\|1\ 4\ 7\ 13\ 12\ 8\ 6\ 3\ 3\ 1\ 4\ 7\ 13\ 12\ 8\ 6\ 6\ 3\ 1\ 4\ 7\ 13\ 12\ 8\ 8\ 6\ 3\ 1\ 4\ 7\ 13\ 12\ 12\ 8\ 6\ 3\ 1\ 4\ 7\ 13$$

$$H =$$

$$\|2\ 8\ 14\ 1\ 4\ 7\ 6\ 3\ 3\ 2\ 8\ 14\ 1\ 4\ 7\ 6\ 6\ 3\ 2\ 8\ 14\ 1\ 4\ 7\ 7\ 6\ 3\ 2\ 8\ 14\ 1\ 4\ 4\ 7\ 6\ 3\ 2\ 8\ 14\ 1\ 1\ 4\ 7$$

Після порівняння зображень опрацьованих фільтром розмиття, використовуючи кожну з цих матриць, можна сказати, що зі збільшенням порядку матриці зростає ефект фільтру. Результати роботи програми фільтрації зображень проілюстровані у 5 розділі.

Результатом застосування матриці:

$$H = \|1\ \ 4\ \ 7\ \ 13\ \ 12\ \ 8\ \ 6\ \ 3\ 3\ \ 1\ \ 4\ \ 7\ \ 13\ \ 12\ \ 8\ \ 6\ 6\ \ 3\ \ 1\ \ 4$$
$$-8-6-3-1-4-7-13-12\ -12-8-6-3-1-4-7$$
$$-13\ 13\ \ 12\ \ 8\ \ 6\ \ 3\ \ 1\ \ 4\ \ 7\ 7\ \ 13\ \ 12\ \ 8\ \ 6\ \ 3\ \ 1\ \ 4\ 4\ \ 7\ \ 13$$

є фільтр з ефектом горизонтального розмиття.

Для отримання ефекту вертикального розмиття необхідно застосувати матрицю:

$$H = \| -1-4-7-13-12-8-6$$
$$-3\ 3\ \ 1\ \ 4\ \ 7\ \ 13\ \ 12\ \ 8\ \ 6\ 6\ \ 3\ \ 1\ \ 4\ \ 7\ \ 13\ \ 12\ \ 8\ 8\ \ 6\ \ 3\ \ 1$$
$$-4-7-13-12-8-6-3-1\ \|$$

Для збільшення різкості зображення підходить така матриця:

$$H = \| -1\ 3\ \ -2\ 2\ 1\ 3\ \ -3\ 2\ \ -1\ \|$$

Порівнюючи отриману матрицю з еталонною матрицею, можна побачити, що велика різниця між центральним елементом і його оточенням не потрібна.

Наступні матриці

$$H = \|3 \quad 1 \quad 5 \quad 2 \; -2 -3 -1 -5 \; -5 -2 -3 -1 \; 1 \quad 5 \quad 2 \quad 3\|$$

$$H = \| -1 -3 -6 -2 -5 \; 5 \quad 1 \quad 3 \quad 6 \quad 2 \; 2 \; -5 -1 - 3 \quad 6 \; 6 \quad 2 \quad 5 \quad 1 \quad 3 \; -3 -6 -2 -5 -1\|$$

$$H = \| -4 -1 -12 -2 -6 -3 -7 \; -7 \quad 4 \quad 1 \quad 12 \quad 2 \quad 6 -3 \; -3 \quad 7 \quad 4 \quad 1 \quad 12 \quad 2 -6 \; -6 \quad 3 \quad 7 \quad 4 \quad 1 \quad 12 -2 \; -2 \quad 6 \quad 3 \quad 7 \quad 4 \quad 1 -12 \; -12 \quad 2 \quad 6 \quad 3 \quad 7 \quad 4 -1 \; -1 -12 -2 -6 -3 -7 -4\|$$

можна використовувати для підкреслення контурів зображення. Ці фільтри можна використовувати для будь-якого растрового зображення, але найкраще вони працюють із чорно-білими зображеннями.

Фільтр тиснення (його графічний ефект схожий на гравюру зображення) реалізовується за допомогою матриць:

$$H = \|1 \quad 7 \quad 3 \quad 2 \quad 4 \; -4 -1 -7 -3 -2 \; 2 \quad 4 \quad 1 -7 \quad 3 \; -3 -2 -4 -1 -7 \; 7 \quad 3 \quad 2 \quad 4 \quad 1\| \qquad H = \| -1 -3 -6 -2 -5 \; 5 \quad 1 \quad 3 \quad 6 \quad 2 \; -2 \quad 5 \quad 1 \quad 3 -6 \; 6 \quad 2 \quad 5 \quad 1 \quad 3 \; -3 -6 -2 -5 -1\| \; H = \|1 \quad 10 \quad 5 \quad 3 \quad 4 \quad 2 \; -2 -1 -10 -5 -3 -4 \; 4 \quad 2 \quad 1 \quad 10 \quad 5 \quad 3 \; -3 -4 -2 -1 -10 -5 \; -5 -3 -4 -2 -1 -10 \; 10 \; 10 \quad 5 \quad 3 \quad 4 \quad 2 \quad 1\|$$

Результати застосування даних матриць перетворення представлені у експериментальній частині роботи**.**

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Розробка та реалізація програмних засобів

Загальний розроблений алгоритм фільтрації зображень складається з наступних елементів:

• отримання програмою вхідного зображення;

• вибір фільтра для зображення користувачем - передача відповідної матриці перетворення в основний модуль обробки зображення;

• обробка зображення обраним фільтром;

• виведення отриманого зображення на екран.

Сам алгоритм фільтрації зображення:

• отримання граничних розмірів зображення та матриці перетворення;

• множення значення кожного пікселя на відповідний центральний коефіцієнт матриці фільтра, крім того, навколишні елементи множаться на відповідні значення;

• підсумовування всіх результатів множення;

• ділення результату на суму коефіцієнтів матриці перетворення;

• перевірка або значення пікселя не виходить за межі діапазону (0 і 255);

• збереження нового значення пікселя на зображенні.

Цей алгоритм відповідає блок-схемі:

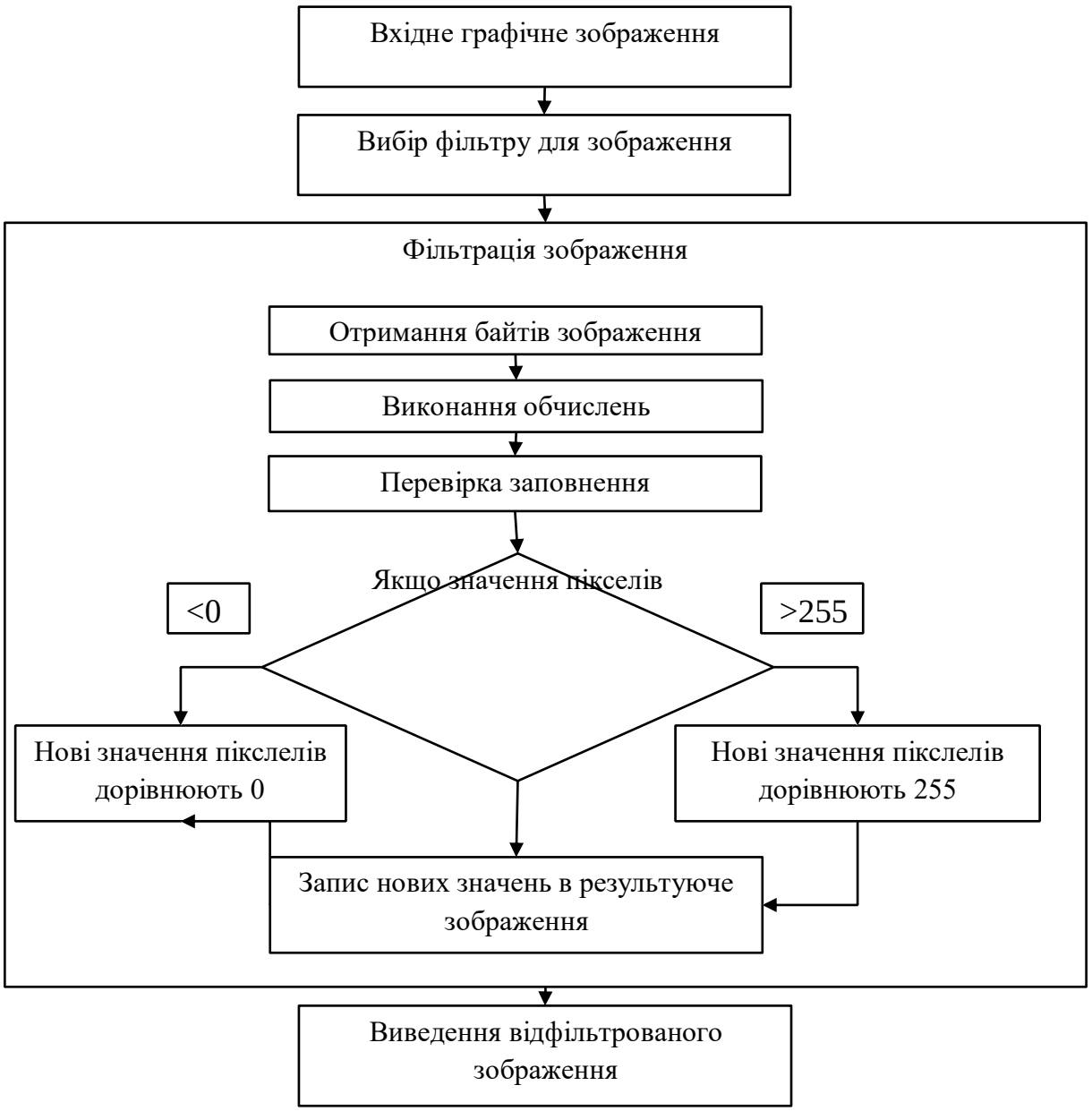

Рис. 4.1. Блок-схема роботи програми фільтрації графічного зображення

Програмна реалізація цього алгоритму наведена в додатку А.

4.2. Вибір платформи програмування

У програмній частині виконується розробка та написання алгоритму фільтрації зображень. Для розв’язання цієї задачі обрана платформа .NET Framework та мова програмування Microsoft Visual C#.

Платформа .NET Framework.

Базовою платформою, що об’єднує використані при розробці технології та інструментальні засоби, які описані нижче, є Microsoft .NET Framework (при

розробці нами використано версію .NET Framework 3.5). Вибір цієї платформи як основи для реалізації представленого програмного проекту обумовлений головним чином тим, що вона поєднує в собі набір зручних і ефективних засобів програмування та обробки даних, надає розробнику потужний сервер баз даних MS SQL Server, а також повноцінно підтримує інтегроване середовище розробки MS Visual.Studio, за допомогою якого можна виконувати практично всі завдання на етапі кодування.

Microsoft .NET Framework – розвинена платформа розробки, розгортання та виконання розподілених програм. Серцем .NET Framework є Common Language Runtime (CLR) або .NET Runtime. Код, який виконується в CLR, часто називають керованим кодом (IL або MSIL). Проміжна мова MSIL подібна до байт-коду Java тим, що вона заснована на ідеї низькорівневого представлення мови у формі числових кодів, які можна легко перевести в машинні коди, що відповідають платформі.

Наявність IL розкрило перед .NET широкі можливості, зокрема:

1. кросплатформеність;
2. можливості взаємодії різних мов програмування, що підтримуються в .NET, в рамках одного проекту;
3. введення спільної системи типів (CST – Common System Types), що описує типи даних, які доступні в IL, тому всі мови програмування, орієнтовані на NET, генерують код, який базується на єдиний системи типів.

Основним аргументом на користь .NET є його технологія управління пам'яттю, яка заснована на принципі збирання сміття. Основна ідея цього підходу полягає в тому, що якщо об’єкт не використовується (тобто кількість посилань на нього дорівнює нулю), то його слід знищити. Такий підхід звільняє програміста від відповідальності за звільнення зарезервованої динамічної пам'яті і, як наслідок, значно спрощує процес розробки.

Мова програмування C#.

Основною мовою програмування для цього програмного проекту було обрано C# 2010.

Офіційно Microsoft декларує C# як «просту, сучасну, об’єктно-орієнтовану та безпечну для типів мову програмування, яка емулює мови C/C++».

C# на даний момент є однією з найпотужніших об'єктно-орієнтованих мов програмування. Крім того, ця мова повністю орієнтована на .NET Framework, забезпечуючи ефективне використання максимальної кількості можливостей останнього. Також досить комфортною в програмуванні є мова C#, що в поєднанні з інструментами середовища MS Visual Studio робить процес розробки максимально «приємним» для програміста [6].

C# – відносно нова мова програмування, що характеризується наступними основними можливостями:

– повна підтримка об’єктно-орієнтованого програмування;
– чітко визначено набір основних типів;
– вбудована підтримка автоматичної генерації XML-документації;
– автоматичне звільнення виділеної динамічної пам'яті;
– можливість позначати властивості та методи атрибутами;
- повна підтримка бібліотеки класів .NET;
– можливість прямого доступу до пам'яті через покажчики;
- управління подіями.

Зауважте, що хоча C# розроблено для створення коду, який виконується в .NET, сам він не є частиною .NET. Деякі функції, які підтримує .NET, не підтримуються C#. І, як не дивно, є функції C#, які не підтримуються .NET.

В якості основного інструменту для реалізації представленого програмного проекту було обрано інтегроване середовище розробки MS Visual Studio. Такий вибір був продиктований гнучкими та потужними можливостями цієї системи, які значно підвищують ефективність роботи програміста та якість виробленого програмного забезпечення та його повну інтеграцію з .NET Framework.

Основними можливостями MS Visual Studio є:

- Інтуїтивно зрозумілий стиль кодування. За промовчанням Visual Studio форматує код розробника в міру його введення, забезпечуючи зручну структуру коду.
- Збільшення темпів розвитку. Більшість функцій Visual Studio призначено для прискорення процесу розробки.
- Налагодження. Система налагодження Visual Studio дозволяє швидко знаходити та виправляти помилки у програмах. Особливий інтерес представляє механізм налагодження процесів, що запускаються з операційної системи, а не серед Visual Studio.

4.3. Діаграма послідовностей для роботи системи

Розглянемо схему послідовності процесу обробки графічних зображень за допомогою фільтрів у вигляді кодування (вибір маски фільтра), декодування файлу (застосування обраного фільтра), зображену на рис. 4.1.

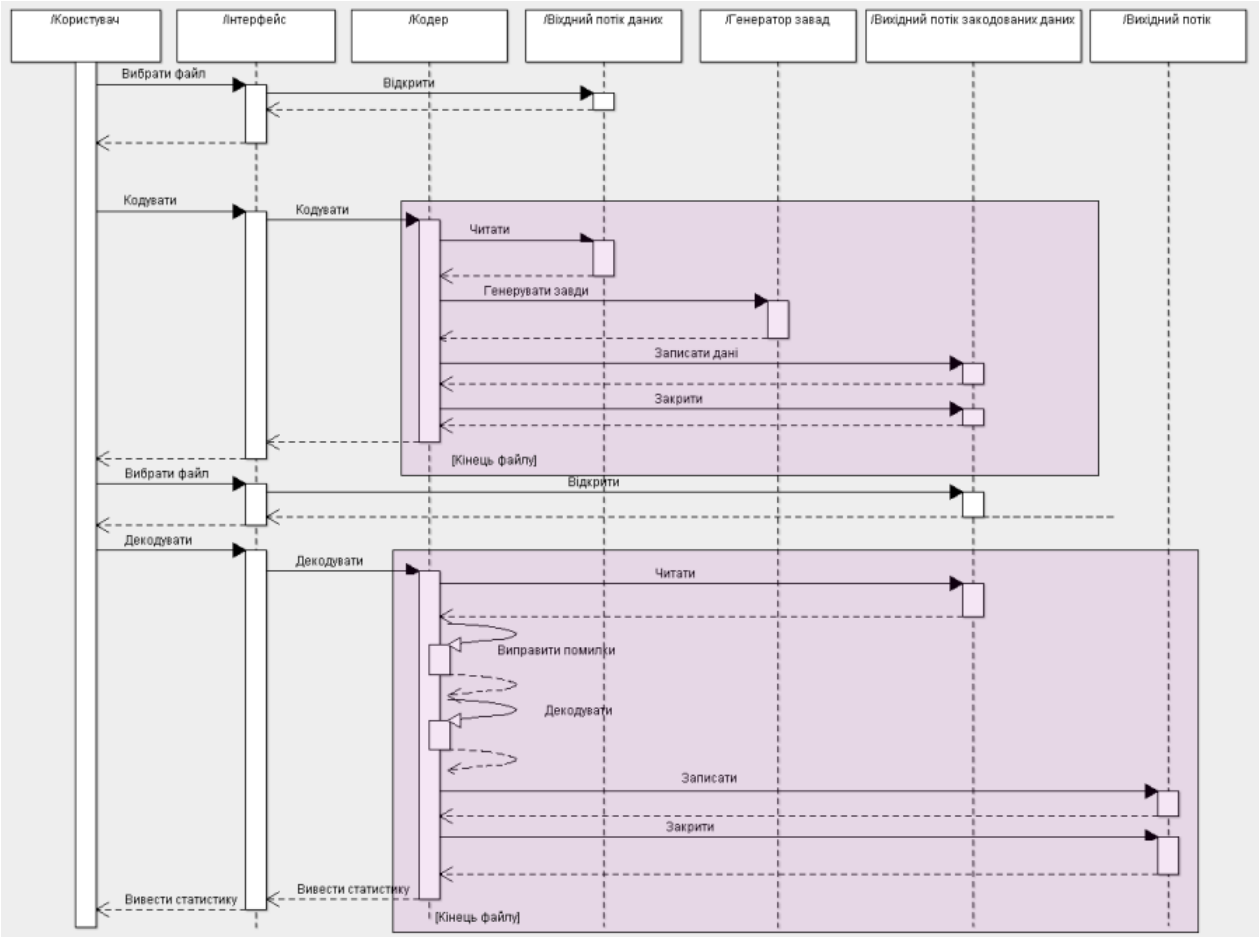

Рис. 4.1. Діаграма послідовностей для роботи системи

Як ви можете бачити на діаграмі послідовності, першою дією з боку користувача є вибір графічного файлу BMP і відкриття потоку для читання файлу. Наступним кроком є вибір маски фільтра (кодування графічного файлу). Після закінчення файлу читаємо блок даних. Ми випадково генеруємо шум (погіршення якості графічного зображення) і зберігаємо його в оригінальний файл потоку. Ми закриваємо потоки.

Процес фільтрації (кодування графічного файлу) починається з команди відкрити файл. Потім користувач дає команду вибрати маску фільтра (декодування графічного файлу). Зашифровані дані з файлу зчитуються блоками до кінця файлу.

Після завершення процесу декодування користувачеві відображається графічне представлення результату операції фільтрації.

4.4. Опис програмного продукту

Графічна оболонка програми обробки зображень з використанням фільтрації дуже проста та інтуїтивно зрозуміла користувачеві. Загальний вигляд оболонки програми:

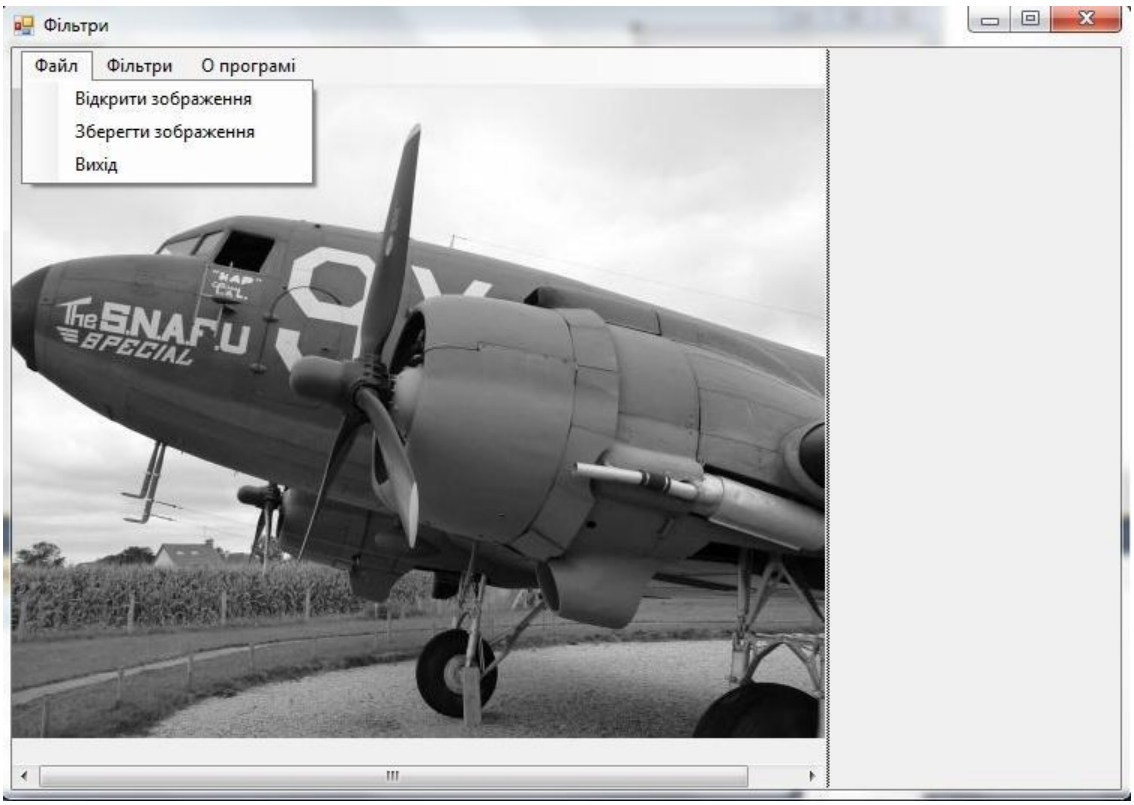

Рис. 4.2. Вигляд вікна меню Файл

Вибравши в меню Файл / Відкрити зображення користувач вибирає файл, який слід відкрити.

За допомогою пункту меню Файл / Зберегти зображення зберігається поточний вигляд зображення під новим ім'ям.

При виборі користувачем пункту Файл / Вихід виконується закриття програми.

В пункті Про програму знаходяться дані про тему магістерської роботи та її автора.

Вигляд пункту Фільтри:

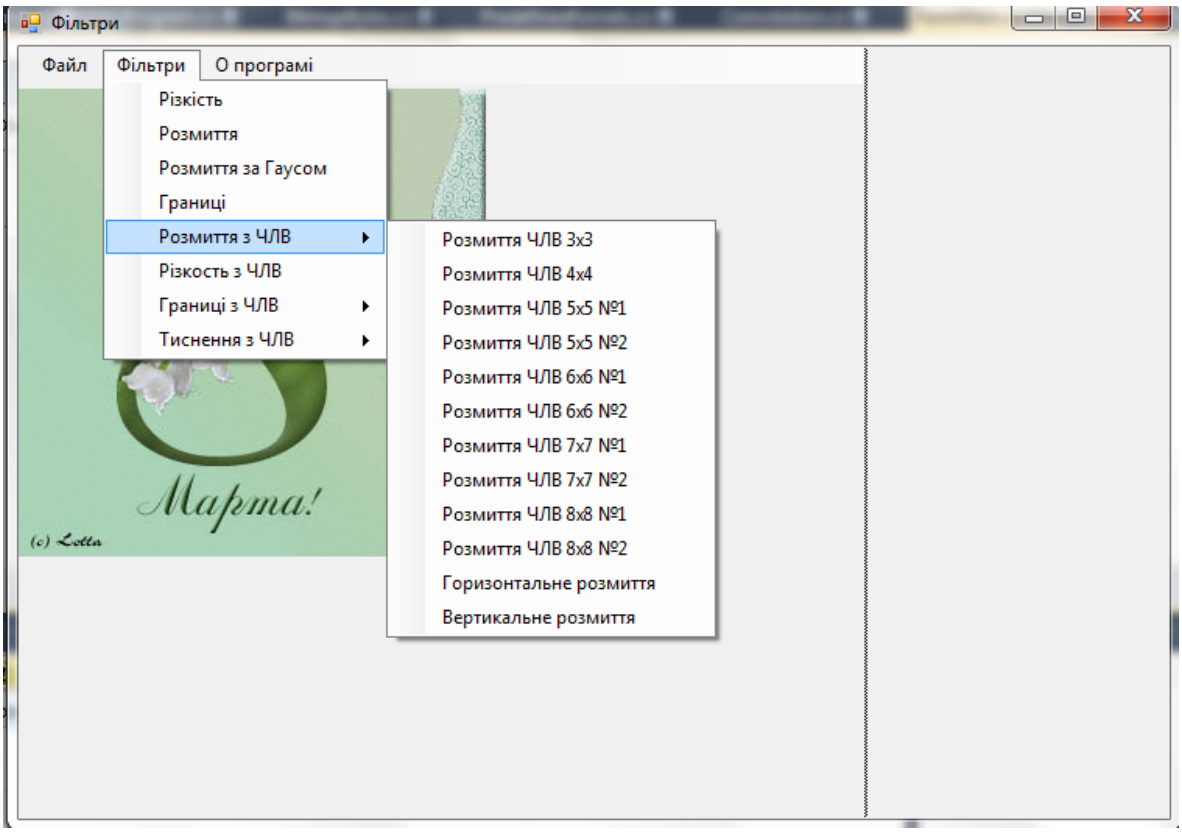

Рис. 4.3. Вигляд вікна меню Фільтри

У цьому пункті знаходяться всі фільтри, які можна застосувати до зображення у даній реалізації програми, а саме:

- різкість;
- розмиття;
- розмиття за Гаусом;
- границі;
- тиснення;
- розмиття з ЧЛВ;
- різкість з ЧЛВ;
- границі з ЧЛВ;
- тиснення з ЧЛВ.

4.5. Опис проведених експериментів

За допомогою розробленої програми обробки зображень з фільтрами досліджено стандартні відомі фільтри та фільтри на основі числових ліній перетину. Для стандартних фільтрів отримані такі результати.

Вхідне зображення :

Рис. 4.4. Зображення, що буде опрацьовуватися

Використовуючи фільтри розмиття зображення можна отримати такі результати:

Рис. 4.5. Фільтр «Розмиття»

Рис. 4.6. Фільтр «Розмиття за Гаусом»

Результат роботи фільтру підвищення різкості (рис. 4.7):

Рис. 4.7. Зображення опрацьоване фільтром «Різкість»

Фільтр для виділення контурів зображення можна застосувати як до кольорового зображення (рис. 4.9), так і до чорно-білого, але в другому випадку він більш ефективний і якісний за обраним ефектом фільтрації (рис. 4.11). Це видно з результату роботи фільтра:

Рис. 4.8. Вхідне зображення

Рис. 4.9. Зображення опрацьоване фільтром «Границі»

Рис. 4.10. Вхідне зображення

Рис. 4.11. Зображення опрацьоване фільтром «Границі»

При використанні фільтра для поліпшення зображення результат не гірший, ніж при використанні стандартних фільтрів:

Рис. 4.12. Вхідне зображення

Рис. 4.13. Зображення опрацьоване фільтром «Тиснення»

Фільтри реалізовані за допомогою числових ліній (CLL). Ми отримали ці результати зображення за допомогою матриці перетворення за допомогою CLI. Отримані фільтри дали ефект розмиття (рис. 4.14), горизонтального розмиття (рис. 4.16) і вертикального розмиття (рис. 4.17), збільшення різкості (рис. 4.18), визначення країв (рис. 4.20 - 4.22) і опуклості зображення (рис. 4.24 - 4.26).

Ефект розмиття виглядає порівняно з найкращими платними стандартними фільтрами:

Рис. 4.14. Фільтр «Розмиття» (матриці порядку від 3x3 до 8x8)

Помітно, що чим більший порядок матриці, тим сильніший ефект розмиття.

Використовуючи модифіковану матрицю 8x8 отримали такі результати ефекту розмиття на рис. 4.17.

Рис. 4.15. Вхідне зображення

Рис. 4.16. Ефект «Горизонтального розмиття»

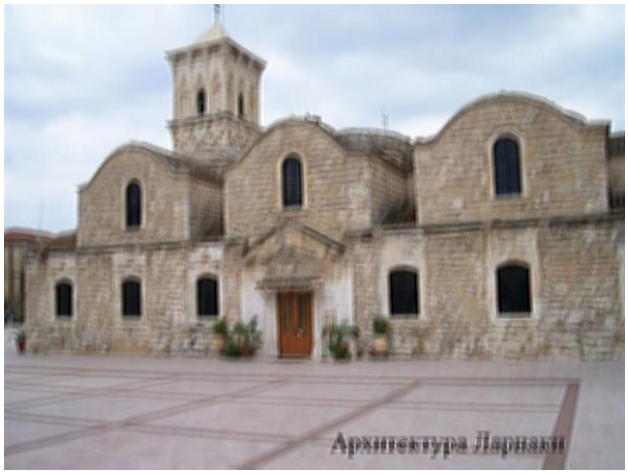

Рис. 4.17. Ефект «Горизонтального» та «Вертикального розмиття»

Фільтр різкості дав такий результат (рис. 4.18):

Рис. 4.18. Зображення опрацьоване фільтром «Різкість»

Досліджуючи фільтри виділення контурів отримано такі результати (рис. 4.20 - 4.22). Для наглядного прикладу краще використовувати чорно-біле зображення.

Рис. 4.19. Вхідне зображення

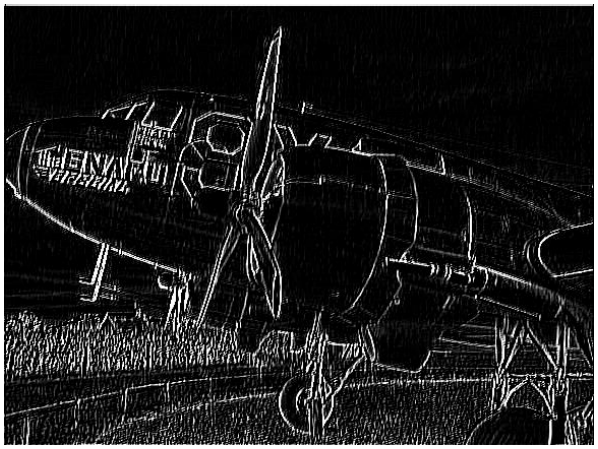

Рис. 4.20. Зображення опрацьоване фільтром «Границі» (матриця 4x4)

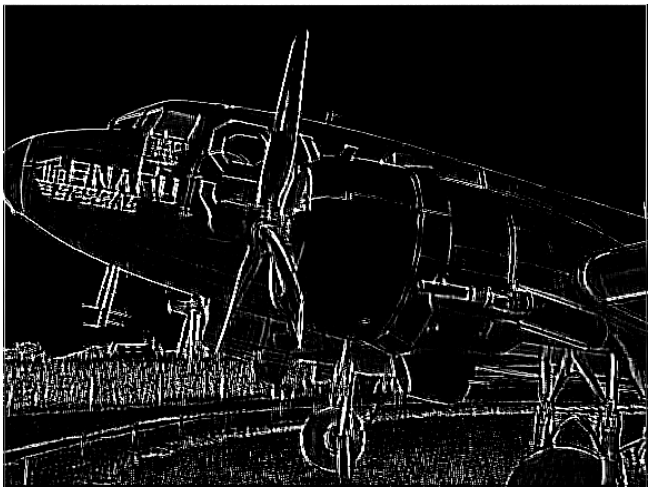

Рис. 4.21. Зображення опрацьовані фільтром «Границі» (матриці 5x5)

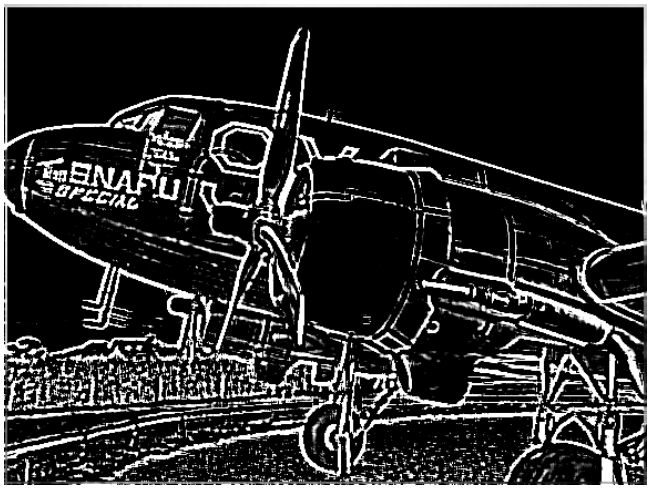

Рис. 4.22. Зображення опрацьовані фільтром «Границі» (матриці 7x7)

ЧЛВ також можна застосовувати для отримання фільтру тиснення, застосувавши різні маски фільтрів на основі ЧЛВ (рис. 4.24, рис. 4.25):

Рис. 4.23. Вхідне зображення

Рис. 4.24. Зображення опрацьоване фільтром «Тиснення» (матриці 5x5)

Рис. 4.25. Зображення опрацьовані фільтром «Тиснення» (матриці 5x5)

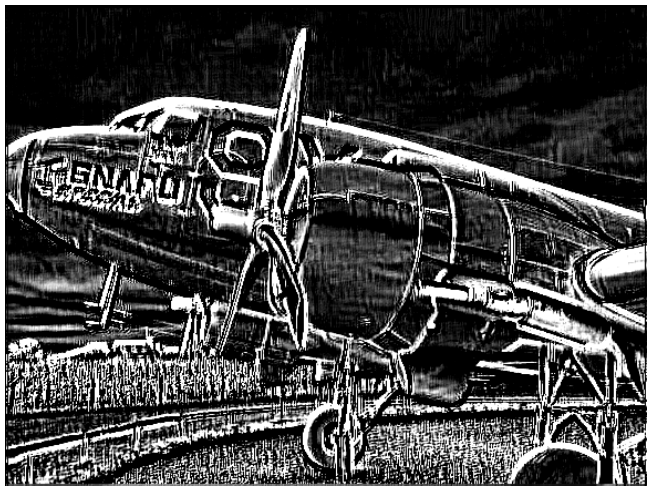

Рис. 4.26. Зображення опрацьовані фільтром «Тиснення» (матриці 6x6)

Після порівняння отриманих результатів можна зробити висновок, що деякі фільтри з використанням числових ліній зв'язку виявилися кращими за відомі раніше стандартні фільтри. Також для розроблених фільтрів можна дотримуватися формули, що чим більша розмірність матриці перетворення, тим сильніший ефект фільтра.

ВИСНОВОК

Предметом магістерської роботи було дослідження та розробка алгоритмів вирішення задачі фільтрації растрових зображень з метою забезпечення відповідної якості зображення.

Відповідно до поставленої мети вирішувалися наступні завдання:

- вивчено теоретичні основи та визначено особливості проблеми обробки зображень у сучасних інформаційних умовах;
- досліджували використання методів обробки просторових зображень;
- досліджено досвід використання числових сполучних ліній в області обробки цифрових зображень, розроблено та обґрунтовано методи фільтрації на основі числових сполучних ліній;
- розроблений метод фільтрації зображення за допомогою числових ліній кадрування дозволяє використовувати його як один із компонентів загальної оцінки візуальної якості зображення, що не залежить від суб’єктивного сприйняття.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Анисимов Б.В., Курганов В.Д., Злобин В.К. Распознавание и цифровая обработки изображений. - М.: Высшая школа, 1983. - 295 с.
2. Архипов, А.Е. Методы цифровой обработки изображений : учеб. пособие / А.Е. Архипов, С.В. Дегтярев, С.С. Садыков. Курск : Изд-во Курск, гос. техн. ун-та, 2002. - 118 с.
3. М.А. Баньщикова. Компьютерная геометрия и графика: Учебно-методический комплекс-Томск, 2009.
4. В.Т. Фисенко, Т.Ю. Фисенко. Компьютерная обработка и распознавание изображений - СПб: СПбГУ ИТМО, 2008.-192 с.
5. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М: Техносфера, 2005. –1072 с.
6. Грузман И.С, В.С. Киричук, В.П. Косых, Г.И. Перетягин, А.А.Спектор.
7. Мак-Дональд, Мэтью. Шпушта, Марио. Microsoft ASP.NET 2.0 с примерами приложений на C# 2005 для профессионалов.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2006. – 1408 с.: ил.
8. Методичні вказівки з виконання магістерської кваліфікаційної роботи по спеціальності ІУС. – Львів,кафедра АСУ,2011.
9. Методичні рекомендації до виконання економічної частини бакалаврської роботи для студентів Інституту комп‘ютерних наук та іноформаційних технологій/ Укл.: З.М. Скибінська, Н.П. Білецька, О.І. Тивончук, З.О. Коваль, О.О. Цогла, Т.І. Свідрик. – Львів: Видавництво Національного університету «Львівська політехніка», 2010. – 16с
10. Пат. 2249850 РФ, МПК7 G 06 F 17/14. Способ последовательно-параллельного вейвлет-преобразования /Квашенников В.В., Яковлев В.Г. ; опубл. 10.04.2005, Бюл. №10.
11. Прэтт У. Цифровая обработка изображений: пер. с англ. / У. Прэтт. – М.: Мир, 1982. – Кн. 2. – 480 с.

12. Різник В.В. Синтез оптимальних комбінаторних систем. - Львів: Вища школа, 1989. - 168 с.
13. Сойфер В.А. Компьютерная обработка изображений. – Соросовский образовательный журнал №3, 1996.
14. Фомин, А.А. Алгоритмы и методы вейвлет-фильтрации изображений / А.А. Фомин // Тез. докл. науч.-практ. конф. Муром, 2 февр. 2007 г. Муром : Изд-во МИ (ф) ВлГУ, 2007. - С. 175-176.
15. Хуанг Т.С. и др. Быстрые алгоритмы в цифровой обработке изображений. – М.: Радио и связь, 1984. – 224 с.
    Цифровая обработка изображений в информационных системах - Новосибирск: Изд-во НГТУ, 2000. — 168.
16. Яне Б. Цифровая обработка изображений - Техносфера,2007. -584 с.
17. Яншин В.В., Калинин Г.А. Обработка изображений на языке СИ для IBM PC: Алгоритмы и программы - М.: Мир,1994. - 240стр.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ ФІЛЬТРАЦІЇ ГРАФІЧНОГО ЗОБРАЖЕННЯ НА ОСНОВІ НЕЕКВІДИСТАНТНИХ СТРУКТУР

Модуль підтримки роботи з bmp зображеннями:

```
using System.Drawing;
using System.Drawing.Imaging;

namespace Convolution.DSP
{
   public static class BitmapBytes
   {
      public static byte[] GetBytes(Bitmap input)
      {
         int bytesCount = input.Width * input.Height * 3;
         BitmapData inputData = input.LockBits(
            new Rectangle(0, 0, input.Width, input.Height),
            ImageLockMode.ReadOnly,
            PixelFormat.Format24bppRgb);

         byte[] output = new byte[bytesCount];
         System.Runtime.InteropServices.Marshal.Copy(inputData.Scan0, output, 0, bytesCount);
         input.UnlockBits(inputData);
         return output;
      }

      public static Bitmap GetBitmap(byte[] input, int width, int height)
      {
         if (input.Length % 3 != 0) return null;
         Bitmap output = new Bitmap(width, height);
         BitmapData outputData = output.LockBits(
            new Rectangle(0, 0, width, height),
            ImageLockMode.ReadWrite,
            PixelFormat.Format24bppRgb);

         System.Runtime.InteropServices.Marshal.Copy(input, 0, outputData.Scan0, input.Length);
         output.UnlockBits(outputData);
         return output;
      }
   }
}
```

Модуль алгоритма обробки зображення:

```
using System.Drawing;

namespace Convolution.DSP
{
   public static class Convolution
   {
      public static Bitmap Apply(Bitmap input, double[,] kernel)
      {
         byte[] inputBytes = BitmapBytes.GetBytes(input);
         byte[] outputBytes = new byte[inputBytes.Length];

         int width = input.Width;
         int height = input.Height;
```

```
            int kernelWidth = kernel.GetLength(0);
            int kernelHeight = kernel.GetLength(1);

            for (int x = 0; x < width; x++)
            {
               for (int y = 0; y < height; y++)
               {
                  double rSum = 0, gSum = 0, bSum = 0, kSum = 0;

                  for (int i = 0; i < kernelWidth; i++)
                  {
                     for (int j = 0; j < kernelHeight; j++)
                     {
                        int pixelPosX = x + (i - (kernelWidth / 2));
                        int pixelPosY = y + (j - (kernelHeight / 2));
                        if ((pixelPosX < 0) ||
                           (pixelPosX >= width) ||
                           (pixelPosY < 0) ||
                           (pixelPosY >= height)) continue;

                        byte r = inputBytes[3 * (width * pixelPosY + pixelPosX) + 0];
                        byte g = inputBytes[3 * (width * pixelPosY + pixelPosX) + 1];
                        byte b = inputBytes[3 * (width * pixelPosY + pixelPosX) + 2];

                        double kernelVal = kernel[i, j];

                        rSum += r * kernelVal;
                        gSum += g * kernelVal;
                        bSum += b * kernelVal;

                        kSum += kernelVal;
                     }
                  }

                  if (kSum <= 0) kSum = 1;

                  rSum /= kSum;
                  if (rSum < 0) rSum = 0;
                  if (rSum > 255) rSum = 255;

                  gSum /= kSum;
                  if (gSum < 0) gSum = 0;
                  if (gSum > 255) gSum = 255;

                  bSum /= kSum;
                  if (bSum < 0) bSum = 0;
                  if (bSum > 255) bSum = 255;

                  outputBytes[3 * (width * y + x) + 0] = (byte)rSum;
                  outputBytes[3 * (width * y + x) + 1] = (byte)gSum;
                  outputBytes[3 * (width * y + x) + 2] = (byte)bSum;
               }
            }
            return BitmapBytes.GetBitmap(outputBytes, width, height);
         }
      }
   }
```

Модуль роботи графічного інтерфесу програми:

```
using System;
using System.Drawing;
```

```
using System.Windows.Forms;

namespace Convolution
{
   public partial class FormMain : Form
   {
      public FormMain()
      {
         InitializeComponent();
      }
      private Bitmap picture;


      private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void відкритиЗображенняToolStripMenuItem_Click(object sender, EventArgs e)
      {
         OpenFileDialog dialog = new OpenFileDialog();
         if (dialog.ShowDialog() == DialogResult.OK)
         {
            picture = (Bitmap)Image.FromFile(dialog.FileName);
            pictureBox.Size = picture.Size;
            pictureBox.Image = picture;
         }
      }

      private void зберігтиЗображенняToolStripMenuItem_Click(object sender, EventArgs e)
      {
         SaveFileDialog savedialog = new SaveFileDialog();
         savedialog.Title = "Зберігти зображення як ...";
         savedialog.OverwritePrompt = true;
         savedialog.CheckPathExists = true;
         savedialog.Filter =
            "Bitmap File(*.bmp)|*.bmp|" +
            "GIF File(*.gif)|*.gif|" +
            "JPEG File(*.jpg)|*.jpg|" +
            "TIF File(*.tif)|*.tif|" +
            "PNG File(*.png)|*.png";
         savedialog.ShowHelp = true;

         if (savedialog.ShowDialog() == DialogResult.OK)
         {

            string fileName = savedialog.FileName;
            string strFilExtn =
               fileName.Remove(0, fileName.Length - 3);
            switch (strFilExtn)
            {
               case "bmp":
                  pictureBox.Image.Save(fileName, System.Drawing.Imaging.ImageFormat.Bmp);
                  break;
               case "jpg":
                  pictureBox.Image.Save(fileName, System.Drawing.Imaging.ImageFormat.Jpeg);
                  break;
               case "gif":
                  pictureBox.Image.Save(fileName, System.Drawing.Imaging.ImageFormat.Gif);
                  break;
               case "tif":
                  pictureBox.Image.Save(fileName, System.Drawing.Imaging.ImageFormat.Tiff);
                  break;
```

```
            case "png":
                pictureBox.Image.Save(fileName, System.Drawing.Imaging.ImageFormat.Png);
                break;
            default:
                break;
        }

    }
}

private void вихідToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    Application.Exit();
}

private void різкістьToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (pictureBox.Image == null) return;

    double[,] kernel = new double[0, 0];
    kernel = DSP.PredefinedKernels.Sharpen;
    pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void розмиттяToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

    double[,] kernel = new double[0, 0];
    kernel = DSP.PredefinedKernels.BoxBlur;
    pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void розмиттяЗаГаусомToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (pictureBox.Image == null) return;

    double[,] kernel = new double[0, 0];
    kernel = DSP.PredefinedKernels.GaussianBlur;
    pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void границіToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (pictureBox.Image == null) return;

    double[,] kernel = new double[0, 0];
    kernel = DSP.PredefinedKernels.EdgeDetect;
    pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void відмінитиДіюToolStripMenuItem_Click(object sender, EventArgs e)
{

}

private void чЛВ3х3ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (pictureBox.Image == null) return;

    double[,] kernel = new double[0, 0];
    kernel = DSP.PredefinedKernels.CLV3x3;
    pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
```

```
}

private void чЛВ4х4ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV4x4;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ5х51ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV5x5_1;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ5х52ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV5x5_2;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ6х61ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV6x6_1;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ6х62ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV6x6_2;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ7х71ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV7x7_1;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ7х72ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV7x7_2;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
```

```
}

private void чЛВ8х81ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV8x8_1;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void чЛВ8х82ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLV8x8_2;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void горизонтальнеРозмиттяToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLVrozmgor;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void вертикальнеРозмиттяToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLVrozmvert;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void фільтрРізкостіЗЧЛВToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLVrizk;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void границіЧЛВToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLVgran4x4;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
}

private void границіЧЛВ5х5ToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pictureBox.Image == null) return;

   double[,] kernel = new double[0, 0];
   kernel = DSP.PredefinedKernels.CLVgran5x5;
   pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
```

```
        }

        private void границіЧЛВ7х7ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (pictureBox.Image == null) return;

            double[,] kernel = new double[0, 0];
            kernel = DSP.PredefinedKernels.CLVgran7x7;
            pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
        }

        private void тисненняЧЛВToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (pictureBox.Image == null) return;

            double[,] kernel = new double[0, 0];
            kernel = DSP.PredefinedKernels.CLVtusn5x5_1;
            pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
        }

        private void тисненняЧЛВ5х52ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (pictureBox.Image == null) return;

            double[,] kernel = new double[0, 0];
            kernel = DSP.PredefinedKernels.CLVtusn5x5_2;
            pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
        }

        private void тисненняЧЛВ6х6ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (pictureBox.Image == null) return;

            double[,] kernel = new double[0, 0];
            kernel = DSP.PredefinedKernels.CLVtusn6x6;
            pictureBox.Image = DSP.Convolution.Apply((Bitmap)pictureBox.Image, kernel);
        }


    }
}
```

Модуль матриць перетворення :

```
namespace Convolution.DSP
{
    static class PredefinedKernels
    {
          public readonly static double[,] Sharpen = {{ -1, -1, -1},
                                                      { -1, 16, -1},
                                                      { -1, -1, -1} };
        public readonly static double[,] EdgeDetect = { {-1,-1,-1 },
                                                        { 0, 0, 0 },
                                                        { 1, 1, 1}};
        public readonly static double[,] Tusnenya = { {-1, 0, 1},
                                                      {-2, 0, 2},
                                                      {-1, 0, 1}};
        public readonly static double[,] BoxBlur = { {1, 1, 1, 1, 1 },
                                                     {  1, 1, 1, 1, 1 },
                                                     {  1, 1, 1, 1, 1 } ,
                                                     {  1, 1, 1, 1, 1 },
                                                     {  1, 1, 1, 1, 1 }};
        public readonly static double[,] GaussianBlur = { { 1, 2, 3, 2, 1 },
                                                          { 2, 4, 5, 4, 2 },
```

```
                                        { 3, 5, 6, 5, 3 },
                                        { 2, 4, 5, 4, 2 },
                                        { 1, 2, 3, 2, 1 }};
public readonly static double[,] CLV3x3 = {{1, 3, 2},
                                    {2, 1, 3},
                                    {3, 2, 1}};
      public readonly static double[,] CLV4x4 = { {3, 1, 5, 2 },
                                    {2, 3, 1, 5 },
                                    {5, 2, 3, 1 },
                                    {1, 5, 2, 3 }};
public readonly static double[,] CLV5x5_1 = { {1, 3, 6, 2, 5},
                                    {5, 1, 3, 6, 2},
                                    {2, 5, 1, 3, 6},
                                    {6, 2, 5, 1, 3},
                                    {3, 6, 2, 5, 1}};
public readonly static double[,] CLV5x5_2 =   { {1, 7, 3, 2, 4},
                                     {4, 1, 7, 3, 2},
                                     {2, 4, 1, 7, 3},
                                     {3, 2, 4, 1, 7},
                                     {7, 3, 2, 4, 1}};
public readonly static double[,] CLV6x6_1 = { {1, 10, 5, 3, 4, 2},
                                    {2, 1, 10, 5, 3, 4},
                                    {4, 2, 1, 10, 5, 3},
                                    {3, 4, 2, 1, 10, 5},
                                    {5, 3, 4, 2, 1, 10},
                                    {10, 5, 3, 4, 2, 1}};
public readonly static double[,] CLV6x6_2 = { {1, 4, 2, 8, 3, 9},
                                    {9, 1, 4, 2, 8, 3},
                                    {3, 9, 1, 4, 2, 8},
                                    {8, 3, 9, 1, 4, 2},
                                    {2, 8, 3, 9, 1, 4},
                                    {4, 2, 8, 3, 9, 1}};
public readonly static double[,] CLV7x7_1 = { {1, 3, 5, 6, 7, 10, 2},
                                    {2, 1, 3, 5, 6, 7, 10},
                                    {10, 2, 1, 3, 5, 6, 7},
                                    {7, 10, 2, 1, 3, 5, 6},
                                    {6, 7, 10, 2, 1, 3, 5},
                                    {5, 6, 7, 10, 2, 1, 3},
                                    {3, 5, 6, 7, 10, 2, 1}};
public readonly static double[,] CLV7x7_2 = { {4, 1, 12, 2, 6, 3, 7},
                                    {7, 4, 1, 12, 2, 6, 3},
                                    {3, 7, 4, 1, 12, 2, 6},
                                    {6, 3, 7, 4, 1, 12, 2},
                                    {2, 6, 3, 7, 4, 1, 12},
                                    {12, 2, 6, 3, 7, 4, 1},
                                    {1, 12, 2, 6, 3, 7, 4}};
public readonly static double[,] CLV8x8_1 =  { {1, 4, 7, 13, 12, 8, 6, 3},
                                     {3, 1, 4, 7, 13, 12, 8, 6},
                                     {6, 3, 1, 4, 7, 13, 12, 8},
                                     {8, 6, 3, 1, 4, 7, 13, 12},
                                     {12, 8, 6, 3, 1, 4, 7, 13},
                                     {13, 12, 8, 6, 3, 1, 4, 7},
                                     {7, 13, 12, 8, 6, 3, 1, 4},
                                     {4, 7, 13, 12, 8, 6, 3, 1}};
public readonly static double[,] CLV8x8_2 = { {2, 8, 14, 1, 4, 7, 6, 3},
                                    {3, 2, 8, 14, 1, 4, 7, 6},
                                    {6, 3, 2, 8, 14, 1, 4, 7},
                                    {7, 6, 3, 2, 8, 14, 1, 4},
                                    {4, 7, 6, 3, 2, 8, 14, 1},
                                    {1, 4, 7, 6, 3, 2, 8, 14},
                                    {14, 1, 4, 7, 6, 3, 2, 8},
                                    {8, 14, 1, 4, 7, 6, 3, 2}};
```

```
        public readonly static double[,] CLVrizk = {{-1, 3, -2},
                                                    {2, 1, 3},
                                                    {-3, 2, -1}};
        public readonly static double[,] CLVrozmgor =  { {1, 4, 7, 13, 12, 8, 6, 3},
                                                         {3, 1, 4, 7, 13, 12, 8, 6},
                                                         {6, 3, 1, 4, 7, 13, 12, 8},
                                                        {-8, -6, -3, -1, -4, -7, -13, -12},
                                                        {-12, -8, -6, -3, -1, -4, -7, -13},
                                                         {13, 12, 8, 6, 3, 1, 4, 7},
                                                         {7, 13, 12, 8, 6, 3, 1, 4},
                                                         {4, 7, 13, 12, 8, 6, 3, 1}};

      public readonly static double[,] CLVrozmvert = { {-1, -4, -7, -13, -12, -8, -6, -3},
                                                       {3, 1, 4, 7, 13, 12, 8, 6},
                                                       {6, 3, 1, 4, 7, 13, 12, 8},
                                                       {8, 6, 3, 1, 4, 7, 13, 12},
                                                       {12, 8, 6, 3, 1, 4, 7, 13},
                                                       {13, 12, 8, 6, 3, 1, 4, 7},
                                                       {7, 13, 12, 8, 6, 3, 1, 4},
                                                      {-4, -7, -13, -12, -8, -6, -3, -1}};
    public readonly static double[,] CLVgran4x4 = { {3, 1, 5, 2 },
                                                    {-2, -3, -1, -5 },
                                                    {-5, -2, -3, -1 },
                                                    {1, 5, 2, 3 }};
        public readonly static double[,] CLVgran5x5 = { {-1, -3, -6, -2, -5},
                                                        {5, 1, 3, 6, 2},
                                                        {2, -5, -1, -3, 6},
                                                        {6, 2, 5, 1, 3},
                                                        {-3, -6, -2, -5, -1}};
        public readonly static double[,] CLVgran7x7 = { {-4, -1, -12, -2, -6, -3, -7},
                                                       {-7, 4, 1, 12, 2, 6, -3},
                                                       {-3, 7, 4, 1, 12, 2, -6},
                                                       {-6, 3, 7, 4, 1, 12, -2},
                                                       {-2, 6, 3, 7, 4, 1, -12},
                                                       {-12, 2, 6, 3, 7, 4, -1},
                                                       {-1, -12, -2, -6, -3, -7, -4}};
        public readonly static double[,] CLVtusn5x5_1 =  { {1, 7, 3, 2, 4},
                                                           {-4, -1, -7, -3, -2},
                                                           {2, 4, 1, -7, 3},
                                                           {-3, -2, -4, -1, -7},
                                                           {7, 3, 2, 4, 1}};
        public readonly static double[,] CLVtusn5x5_2 = { {-1, -3, -6, -2, -5},
                                                          {5, 1, 3, 6, 2},
                                                          {-2, 5, -1, 3, -6},
                                                          {6, 2, 5, 1, 3},
                                                          {-3, -6, -2, -5, -1}};

        public readonly static double[,] CLVtusn6x6 = { {1, 10, 5, 3, 4, 2},
                                                        {-2, -1, -10, -5, -3, -4},
                                                        {4, 2, 1, 10, 5, 3},
                                                        {-3, -4, -2, -1, -10, -5},
                                                        {-5, -3, -4, -2, -1, -10},
                                                        {10, 5, 3, 4, 2, 1}};
    }
}
```