

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

МЕТОДІВ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

RESEARCH AND SOFTWARE IMPLEMENTATION OF

IMAGE FILTERING METHODS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-123М

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ОЛІЙНИК С.О.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛІЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ОЛІЙНИКА Сергія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів фільтрації зображень. Research and Software Implementation of Image Filtering Methods

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 02 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “9” вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	9-10 тижні	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Сергій ОЛІЙНИК
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
85 с., 4 табл., 29 рис., 3 дод., 20 джерел.

C#, MICROSOFT VISUAL STUDIO, ЗОБРАЖЕННЯ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СЕРЕДОВИЩЕ РОЗРОБКИ, ФІЛЬТРАЦІЯ ЗОБРАЖЕНЬ.

Об'єкт дослідження – процес розробки програмного забезпечення для обробки зображень.

Предмет дослідження – програмні засоби для підтримки фільтрації зображень.

Мета роботи – дослідження та розробка програмного забезпечення для фільтрації зображень.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Microsoft Visual Studio.

Результати. Розроблено програмне забезпечення для фільтрації зображень за допомогою мови програмування C# та середовища розробки Microsoft Visual Studio.

Практична цінність роботи полягає у розробці програмного забезпечення для фільтрації зображень.

Висновки. Виконано проєктування програмного забезпечення для фільтрації зображень. Розроблено програмне забезпечення для фільтрації зображень. Здійснено тестування розробленого програмного забезпечення для фільтрації зображень.

Галузь використання – обробка зображень.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 85 pages, 4 tables, 29 figures, 3 appendixes, 20 sources.

C#, MICROSOFT VISUAL STUDIO, IMAGES, PROGRAMMING LANGUAGE, SOFTWARE, DEVELOPMENT ENVIRONMENT, IMAGE FILTERING.

The object of research is the process of developing software for image processing.

The subject of the research is software for supporting the image filtering process.

The purpose of this work is to research and develop the software to support the image filtering process.

Materials, methods and technical tools: C# programming language, Microsoft Visual Studio development environment.

Results. The software tools to support the image filtering process using the C# programming language and the Microsoft Visual Studio development environment have been developed.

The practical value of the work lies in the development of image filtering software.

Conclusions. The software to support the image filtering process has been designed. The software to support the image filtering process has been developed. The developed software to support the image filtering process has been tested.

Scope of use – image processing.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Поняття про зображення та якість зображень	11
1.2 Аналіз програмного забезпечення для підтримки процесу обробки зображень.....	19
1.3 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування	31
2.2 Вибір середовища розробки для створення програмного забезпечення для фільтрації зображень	33
2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для фільтрації зображень	35
2.4 Висновки за розділом 2	37
3 Опис програми	38
3.1 Структура програмного забезпечення для фільтрації зображень	38
3.2 Функціонування програмного забезпечення для фільтрації зображень ..	39
3.3 Використання методів фільтрації у програмному забезпеченні для обробки зображень	40
3.4 Проєктування інтерфейсу програмного забезпечення для фільтрації зображень.....	45
3.5 Висновки за розділом 3	47
4 Експлуатація, тестування та експериментальне дослідження програми.....	49
4.1 Призначення й умови застосування програми	49
4.2 Характеристики програми для підтримки процесу фільтрації зображень.....	50
4.3 Інструкція по експлуатації програми.....	50
4.3.1 Звернення до програми.....	50

4.3.2 Вхідні й вихідні дані	51
4.3.3 Повідомлення.....	51
4.4 Виконання програмного забезпечення для підтримки процесу фільтрації зображень.....	51
4.5 Тестування програмного забезпечення для підтримки процесу фільтрації зображень.....	54
4.6 Висновки за розділом 4	55
Висновки	56
Перелік джерел посилання	59
Додаток А Технічне завдання	61
Додаток Б Фрагмент тексту програми	65
Додаток В Слайди презентації	78

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GUI – Graphical User Interface;

VS – Visual Studio;

ПЗ – програмне забезпечення;

ТЗ – технічне завдання;

ФЗ – фільтрація зображень.

ВСТУП

Збільшення використання соціальних мереж, вебсайтів та мобільних застосунків разом з відповідним програмним та апаратним забезпеченням, що дозволяють легко створювати нові зображення, призвело до зростання кількості зображень, які публікуються та обмінюються щодня. Висока якість цих зображень важлива для користувачів, а фільтрація за допомогою відповідного програмного забезпечення дозволяє покращити вигляд фотографій, видалити шум, коригувати кольори та контраст [1]-[7].

З розвитком інформаційних технологій та збільшенням кількості візуального контенту, програмні засоби фільтрації зображень стають все більш важливими для обробки великих обсягів графічних даних. Це особливо важливо для підприємств, які працюють з великими обсягами візуального контенту, такими як платформи соціальних медіа, агентства новин та Інтернет магазини [1]-[7].

Програмне забезпечення фільтрації зображень допомагає виявляти і блокувати небажаний або неприйнятний контент, що є важливим для забезпечення безпеки користувачів, особливо на платформах з великою кількістю користувачів, як-от соціальні мережі та форуми [1]-[7].

Проте використання деяких програмних засобів для фільтрації зображень може призвести до втрати дрібних деталей в зображеннях, особливо якщо вони використовують неефективні методи фільтрації зображень для видалення шуму або інших небажаних елементів. Застосування інших програмних засобів до обробки зображень може призвести до зміни оригінальних кольорів зображення, що може бути небажаним для професійної фотографії або інших застосувань, де точність кольору є критичною. Використання складних методів фільтрації зображень, особливо тих, що використовують алгоритми штучного інтелекту та машинного навчання, можуть вимагати значних обчислювальних ресурсів, що може бути

проблемою для менш потужних пристроїв або для обробки великих обсягів візуальних даних [1]-[7].

Тому актуальною є розробка програмного забезпечення, що реалізує різні методи фільтрації зображень. У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання дослідження та розробки програмного забезпечення для фільтрації зображень.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обробки зображень;
- розробити модель взаємодії користувача з програмним забезпеченням фільтрації зображень;
- здійснити проєктування програмного забезпечення для фільтрації зображень;
- створити програмне забезпечення для фільтрації зображень;
- виконати тестування розробленого програмного забезпечення для фільтрації зображень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про зображення та якість зображень

Зображення – це візуальне представлення об'єктів, сцен або інформації, що зберігається або передається через оптичні або електронні засоби. В контексті візуальних технологій зображення може бути фотографією, малюнком, графікою або будь-яким іншим візуальним відображенням, яке має важливість у різних сферах [1]-[7].

Поняття зображення в контексті візуальної обробки включає в себе представлення візуальної інформації, яку можна сприймати за допомогою людського зору або технічних засобів, таких як камери чи сканери. Якість зображення визначається його розміром, роздільною здатністю, кольоровою гамою, контрастністю та іншими параметрами, які впливають на чіткість і достовірність інформації, яку воно передає [1]-[7].

Обробка зображень відноситься до аналізу зображень за допомогою різних методів. Вона включає використання обчислювальних технік і алгоритмів для виконання різних операцій, таких як покращення якості зображень, виявлення закономірностей і видобування інформації [1]-[7].

Поліпшення якості зображень – це процес обробки зображень, пов'язаний з підвищенням чіткості, роздільної здатності, контрастності та інших важливих параметрів зображення з метою покращення сприйняття інформації, яка міститься на зображенні. Поліпшення якості зображень може включати в себе такі методи як зменшення шуму, виправлення розмиття, підвищення різкості контурів, збільшення яскравості та насиченості кольорів [1]-[7].

Методи покращення зображення можна розділити на дві основні групи: методи, які застосовуються в просторовій області (просторові методи), і методи, що використовуються в частотній області (частотні методи). В термінах просторових методів, це означає покращення якості самого зображення і включає аналіз правильних елементів на зображенні [1]-[7].

Методи, що застосовуються в частотній області, базуються на використанні перетворення Фур'є для обробки зображення [1]-[7]. Стратегії обох типів підходів вимагають не тривіальних комбінацій [1]-[7].

Локальні застосування методів покращення зображень включають: класифікації зображень, контроль над шумом, шумозаглушення за допомогою лінійних і нелінійних методів, виділення окремих меж зображень. Усі ці методи спрямовані на покращення видимості зображень. Наприклад, підкреслення меж може зробити зображення більш чітким і благородним, навіть якщо це необхідно [1]-[7].

Зазвичай методи покращення якості зображення, такі як просторові методи, також спрямовані на покращення видимості, але вони вимагають більше обчислень через використання двовимірних перетворень [1]-[7].

Існує декілька теорій щодо поліпшення зображення. Оцінка якості зображення є дуже суб'єктивним процесом, що робить поняття якості зображення неоднозначним показником ефективності алгоритму. Наприклад, в задачі моделювання оптимальної обробки зображень процесором (з урахуванням вимог до обчислень) вибір буде спрямований на досягнення найточнішого машинного розпізнавання. Проте, не зважаючи на те, що складність дозволяє встановити чіткі стандарти якості, зазвичай потрібно провести декілька тестів, перш ніж буде вибрано надійний метод покращення зображення [1]-[7].

У багатьох програмах якість зображення оцінюється за відстанню між ідеальним та реальним зображеннями, або між модифікованим та оригінальним зображеннями. Це вимірювання середньоквадратичної помилки для фізичних і статистичних змінних. Проте, якщо мова йде про візуальне спостереження зображень, цей критерій не завжди є найкращим, оскільки зорова кора не обробляє просторові характеристики зображення окремо, а витягує з нього інформацію про просторові та часові характеристики, такі як синтетично закодовані кольорові властивості [1]-[7].

Також важкість у поліпшенні якості зображення може вказувати на відсутність вихідного високоякісного зображення. Таким чином, для вирішення проблеми покращення якості зображення суб'єктивне оцінювання залишається найпоширенішим і надійним методом визначення якості зображення. Важливо використовувати не експертні оцінки спостерігачів, які відображають сприйняття якості зображення середнім спостерігачем [1]-[7].

Перетворення зображень, зафіксованих різними оптико-електронними системами і записаних різними рецепторами, часто погіршуються внаслідок впливу різних перешкод. Усі компоненти системи візуалізації, починаючи з освітлення (наприклад, освітлення неоднорідних об'єктів), впливають на зображення. Зміни, внесені оптичною системою під час виробництва, відомі як артефакти. Електронні спотворення, такі як електронний шум, викликані електронним променем, наприклад, у матрицях. Обмеження включають візуальний аналіз та автоматичну обробку зображень [1]-[7].

Мінімізація цих перешкод є важливою задачею покращення зображень. Під час фільтрації яскравість (сигнал) у кожній частині оригінального зображення, що змішується з роздільною здатністю, замінюється іншим значенням яскравості, що менше за відому роздільну здатність. Для пояснення принципів фільтрації зображень необхідно враховувати, що зміни інтенсивності зображення змінюються просторово повільніше, ніж функція інтерференції. У деяких випадках спостерігаються зміни фонового блиску, що характеризують дефекти [1]-[7].

У методах фільтрації зображень, коли реальний сигнал аналізується в певній фіксованій точці, також оцінюються сусідні точки, де похідний сигнал пригнічується. Точність співвідношення досить умовна. Межі можуть утворюватися лише сусідніми областями, що можуть включати як менші, так і далекі частини. У таких випадках вплив розміру (ваги) на віддалені та близькі точки у рішеннях, прийнятих фільтром на заданому точковому кадрі, буде значно різним. Отже, процес фільтрації полягає в максимальному використанні інформації з оточення та самого об'єкта роботи [1]-[7].

У процесі вдосконалення функцій фільтрації зображень використовуються моделі та структури зображень, які базуються на принципах геометричної ефективності. Це пов'язано з активним використанням рідкісних розподілів у середовищі. Різні методи та алгоритми фільтрації зображень ґрунтуються на математичних виразах, моделях невизначеності та різних правилах оптимальності [1]-[7].

Згладжування зображень являє собою метод візуальної обробки, що використовується для зниження різкості переходів між кривими лініями, що складають об'єкти, шляхом видалення ступенів, що можуть виникати на їхніх межах. Основний принцип згладжування полягає у використанні властивостей вивідного пристрою для відображення кольорів, які представлені курсором. У процесі згладжування елементи, що межують з піксельною частинкою зображення, отримують проміжне значення між кольором цієї частини та кольором фону, створюючи плавні переходи та закруглені краї [1]-[7].

Існують різні підходи до згладжування зображень. Загальне згладжування зображень полягає в перемальовуванні фіксованого зображення з поступовим зменшенням різкості переходів. Спеціалізовані методи згладжування зображень використовуються для обробки конкретних типів зображень [1]-[7]. Варто зазначити, що ефективність згладжування також залежить від параметрів гамма-монітора [1]-[7].

При використанні фільтрів або тегів для різних застосувань особливо важливі лінійні напрямки. При проектуванні лінійного компонента системи часто використовуються математичні моделі для перевірки прийнятих рішень і формальних розрахунків. З іншого боку, нелінійні компоненти часто визначаються інтуїцією та емпіричними методами. Концепція загальної суперпозиції іноді дозволяє застосовувати формальні методи до набору завдань нелінійної фільтрації, розширюючи тим самим область застосування лінійних методів фільтрації. При цьому лінійний фільтр часто використовується для виділення сигналу від шуму [1]-[7].

Узагальнений метод нелінійної суперпозиції фільтрації зображень є лише одним із багатьох підходів. Його головна перевага полягає в тому, що він спрощує складність лінійних фільтрів, забезпечуючи аналітичну простоту і інтеграцію сигналу. Хоча більшість задач лінійного фільтрування можуть бути формалізовані алгоритмами для вибору оптимального фільтра, для оцінювання помилок найбільш поширеною метрикою є середньоквадратична помилка (або інтегральна квадратична помилка) [1]-[7].

Масштабування ліній для поліпшення якості зображення базується на попередніх дослідженнях. Його суть полягає у використанні основних зображень. Цей метод дозволяє визначати значення вихідного сигналу на основі значення вхідного сигналу. Іншими словами, він математично перетворює кожен точку вхідного зображення в нове значення вихідного зображення, не залежно від значень інших точок вхідного зображення [1]-[7].

Низький контраст часто стає результатом обробки зображень, які виглядають тільки видимими через обмежений діапазон освітлення тощо. Проблема контрасту також включає управління інтенсивністю зображення та його пристосування до екрану, на який відтворюється сцена. Якщо мінімальна та максимальна яскравість вихідного зображення або будь-які параметри значно відрізняються від нормальних значень, зорове сприйняття зображення може стати темним або ненасиченим, що призводить до неприємного та нудного відчуття [1]-[7].

Рівень яскравості обробленого зображення після цифрової обробки може відрізнятися від рівня яскравості вихідного зображення [1]-[7]. Існують різні методи для вирівнювання яскравості вихідного зображення з яскравістю вхідного: лінійне масштабування (інверсне) та лінійне вирівнювання з коефіцієнтом приглушення [1]-[7]. Загальний підхід до гістограми полягає у звичайному нормуванні кожного значення на кількість кадрів у гістограмі. Гістограми є основою багатьох локальних застосувань [1]-[7]. Обробка гістограми (перетворення гістограми) є ефективним методом для покращення зображень. Окрім корисної статистики про зображення, інформація в

гістограмі є важливою для інших застосувань, таких як аналіз і сегментація зображень. Гістограми відносно просто реалізовувати як у програмному, так і в апаратному забезпеченні, що робить їх відмінними інструментами для обробки зображень у реальному часі [1]-[7].

На гістограмі темного зображення ненульові значення згруповані в області низької яскравості (темряви). Крім того, значні суми в гістограмі переходять у більш високий діапазон. На високо контрастному зображенні гістограма має великий діапазон яскравості і розподіл пікселів, що істотно не відрізняються один від одного, крім окремих піків об'єктів, які виділяються над іншими об'єктами зображення. Загалом, зображення з рівномірним розподілом властивостей із високим контрастом та великим діапазоном освітленості може бути успішно покращене за допомогою інформації з гістограми оригінального зображення [1]-[7].

Фільтрація для поліпшення якості зображень включає в себе різноманітні методи обробки зображень, які спрямовані на покращення різних аспектів зображення, таких як чіткість, контрастність, відновлення деталей і зменшення шуму [1]-[7].

Фільтрація зображень являє собою підхід, який застосовується в обробці зображень для покращення або зміни їх візуального вигляду. Процес фільтрування включає використання фільтра або ядра для кожного пікселя зображення, щоб визначити нове значення пікселя на основі існуючих значень сусідніх пікселів. Фільтр задає ваги, які застосовуються до сусідніх пікселів під час фільтрації [1]-[7].

Застосуваннями фільтрації зображень є такі [1]-[7]:

– зменшення шуму – небажаний шум, як-от випадкові зміни яскравості пікселів, який супроводжує зображення, зроблені камерами або отримані за допомогою цифрової обробки. Фільтри зображень ефективно зменшують або усувають цей шум, створюючи чіткі та приємні для ока зображення [1]-[7];

– згладжування та розмиття – використання фільтрів для пом'якшення або розмиття зображення, що зменшує деталізацію. Це може бути корисним у таких сферах, як живопис, видалення плям та естетична медицина [1]-[7];

– виявлення країв – фільтри для визначення країв виявляють межі, які розділяють різні частини або об'єкти на зображенні. Це важливо для таких процесів, як розпізнавання об'єктів, сегментація зображення та пошук об'єктів, де враховуються порогові значення [1]-[7];

– поліпшення зображення. Фільтри можуть підкреслювати певні деталі або характеристики зображення. Наприклад, вони можуть зменшити різкість і підвищити деталізацію країв, забезпечуючи вищу роздільну здатність зображення. Збільшуючи контрастність зображення між різними частинами зображення, контрастні фільтри можуть зробити певні ділянки сцени більш помітними [1]-[7];

– косметичні зміни. Фільтрування зображень також можна використовувати для косметичних змін, дозволяючи фотографам, дизайнерам та художникам змінювати зовнішній вигляд своїх зображень. Фільтри можна застосовувати для зміни колірної гами, імітації ефектів плівки та створення спеціалізованих зображень [1]-[7].

Фільтрування зображень відіграє ключову роль в обробці зображень, покращуючи важливі характеристики, зменшуючи шум і готуючи зображення для подальшого аналізу чи візуалізації. Фільтрація зображень покращує доступність і зрозумілість візуальних медіа та допомагає досягти бажаних результатів у різних додатках [1]-[7].

Методи обробки зображень шляхом фільтрації відіграють важливу роль у створенні, відтворенні та аналізі цифрових зображень. Лінійні фільтри зображень є корисними інструментами для обробки зображень, особливо коли необхідно зменшити шум. Ці фільтри працюють шляхом застосування операції згортання до зображення з використанням попередньо визначеної маски або ядра фільтра. Перевага лінійних фільтрів полягає в їх простоті та ефективності у зменшенні різних типів шуму [1]-[7].

Процес використання лінійного фільтра для зменшення шуму в зображеннях ґрунтується на методі, що називається згорткою. Згортка – це математична операція, під час якої маска фільтра послідовно застосовується до кожного пікселя зображення. Обчислюється добуток коефіцієнтів фільтра та відповідних значень пікселів. Цей процес повторюється для кожного пікселя, що в підсумку дає нове фільтроване зображення [1]-[7].

Однією з основних переваг лінійної фільтрації для зменшення шуму зображення є можливість виконання обробки зображень у режимі реального часу, що є важливим для програмного забезпечення, де потрібні швидкі обчислення. Фільтри використовують прості арифметичні операції, що забезпечує швидку роботу як у програмному, так і в апаратному забезпеченні. Проте, хоча лінійні фільтри ефективні у багатьох випадках, вони можуть мати проблеми зі складними шумами та нелінійними ситуаціями, де потрібні нелінійні фільтри [1]-[7].

Існують різні методи та підходи до фільтрації зображень. Медіанна фільтрація використовується для зменшення шуму на зображенні шляхом заміни кожного пікселя медіанним значенням в його околі. Фільтрація за розміром застосовується для згладжування зображення шляхом заміни кожного пікселя середнім значенням пікселів у його околі [1]-[7].

Фільтрація Гаусса використовується для створення зваженого середнього значення пікселів у околі, що дозволяє зменшити шум і згладити зображення. Адаптивна фільтрація налаштовує параметри фільтра в залежності від структури зображення для кращого збереження деталей і зменшення розмиття [1]-[7]. Нелінійна фільтрація застосовується для зменшення шуму, зокрема шуму сітки [1]-[7]. Ці методи можуть використовуватися окремо або у поєднанні для досягнення бажаного результату поліпшення якості зображень [1]-[7].

Відзначено, що поліпшення якості зображень є процесом обробки зображень, пов'язаний з підвищенням чіткості, роздільної здатності, контрастності та інших важливих параметрів зображення з метою покращення

сприйняття інформації, яка міститься на зображенні. Фільтрація зображень являє собою підхід, який застосовується в обробці зображень для покращення або зміни їх візуального вигляду. Проаналізовано практичні застосування фільтрації зображень. Відзначено, що фільтрація зображень відіграє ключову роль в обробці зображень, покращуючи важливі характеристики, зменшуючи шум і готуючи зображення для подальшого аналізу чи візуалізації.

1.2 Аналіз програмного забезпечення для підтримки процесу обробки зображень

Проаналізуємо програмне забезпечення для підтримки процесу обробки зображень [8]-[13].

Програмне забезпечення Fotor (рис. 1.1) є універсальним вебсайтом для редагування зображень, який дозволяє створювати колажі, ретушувати фотографії та створювати чудову графіку [8]-[11].

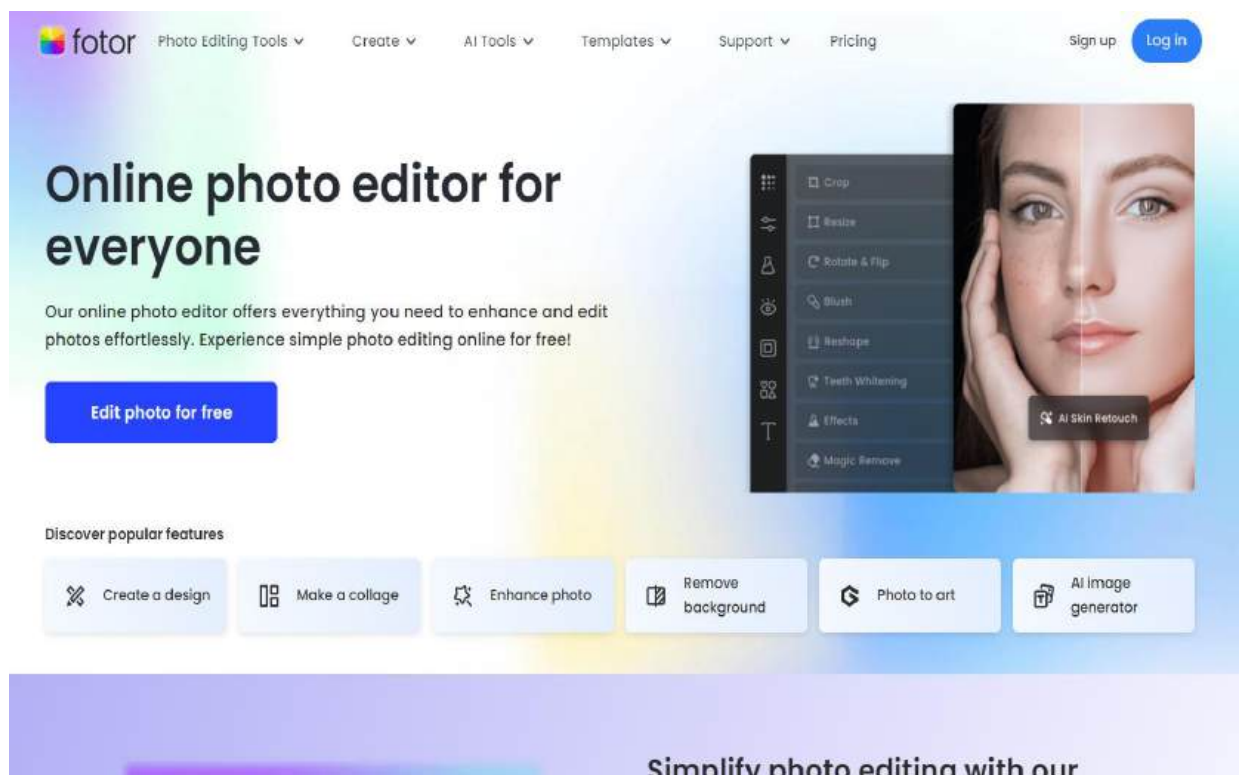


Рисунок 1.1 – Програмне забезпечення Fotor [11]

Програмне забезпечення Fotor надає велику кількість готових шаблонів, які містять сотні попередньо налаштованих шрифтів, зображень і дизайнів. Використання програмного забезпечення Fotor можливе за підпискою на простий місячний або річний план для професіоналів [8]-[11].

Програмне забезпечення Fotor являє собою багатофункціональну платформу для редагування фотографій, яка поєднує можливості редагування, дизайну та штучного інтелекту для простого створення привабливого візуального контенту. Крім вебверсії, Fotor доступний на Windows, Mac, iOS та Android. На сьогодні Fotor налічує велику кількість користувачів. Fotor підходить усім, хто потребує створення або покращення візуального контенту, дозволяючи швидко редагувати зображення, створювати графічні дизайни або генерувати роботи за допомогою штучного інтелекту з мінімальними зусиллями [8]-[11].

Основними характеристиками програмного забезпечення Fotor для редагування зображень є такі [8]-[11]:

- засоби редагування зображень. Основними інструментами редагування програмного забезпечення Fotor є обрізка, поворот, зміна розміру, регулювання яскравості, контрасту, насиченості та різкості. Інструментами ретушування програмного забезпечення Fotor є видалення плям, відбілювання зубів, згладжування шкіри, усунення червоних очей [8]-[11];

- засоби фільтрації: програмне забезпечення Fotor містить широкий вибір фільтрів для покращення фотографій [8]-[11];

- засоби графічного дизайну. програмне забезпечення Fotor надає шаблони для соціальних мереж, візитних карток, флаєрів, запрошень та багато іншого. Текстові інструменти: різноманітні шрифти, стилі та кольори для створення привабливих текстових елементів [8]-[11];

- персоналізація: програмне забезпечення Fotor надає можливість налаштовувати розмір, колір та інтервал між фотографіями [8]-[11];

- штучний інтелект та автоматизація: програмне забезпечення Fotor надає інтелектуальні інструменти для автоматичного покращення фотографій,

генерації зображень на основі тексту, видалення фону, а також функції автоматичної корекції для швидкого покращення фотографій [8]-[11];

- кросплатформенність: програмне забезпечення Fotor є доступним на вебплатформі, а також для операційних систем Windows, Mac, iOS, Android та ін. [8]-[11];

- простота використання: програмне забезпечення Fotor має інтуїтивно зрозумілий інтерфейс, що робить процес редагування легким навіть для новачків [8]-[11];

- підтримка кількох мов: програмне забезпечення Fotor має інтерфейс доступний багатьма мовами [8]-[11];

- інструменти для соціальних мереж: програмне забезпечення Fotor надає шаблони для соціальних мереж, готові шаблони для обкладинок та постів на різних платформах. Програмне забезпечення Fotor надає можливість налаштування розмірів зображень відповідно до вимог різних соціальних мереж [8]-[11];

- обробка зображень у реальному часі: програмне забезпечення Fotor надає можливість швидкого редагування та обробки зображень завдяки ефективним алгоритмам [8]-[11];

- збереження та експорт: програмне забезпечення Fotor надає можливість роботи з форматами зображень у різних форматах [8]-[11];

- хмарне зберігання: програмне забезпечення Fotor надає опції для збереження роботи у хмарі для доступу з будь-якого пристрою [8]-[11].

Серед недоліків програмного забезпечення Fotor можна відзначити обмежені можливості безкоштовної версії [8]-[11], оскільки деякі функції та інструменти доступні тільки у платній версії, що може обмежувати можливості користувачів без підписки. Інтернет версія Fotor потребує стабільного з'єднання для роботи, що може бути незручно в умовах обмеженого доступу до мережі [8]-[11]. Безкоштовна версія програмного забезпечення Fotor може містити рекламу, що може відволікати користувачів або заважати їм під час редагування зображень [8]-[11].

Програмне забезпечення Capture One (рис. 1.2) надає комбінацію інструментів для каталогізації, редагування зображень, локальних коригувань і попередньо встановлених ефектів зображень [8]-[10], [12].



Рисунок 1.2 – Програмне забезпечення Capture One [12]

Робочий процес програмного забезпечення Capture One підходить для професійних комерційних фотографів, які знімають в студії та співпрацюють у режимі реального часу з клієнтами завдяки програмній службі Capture One Live [8]-[10], [12].

Система коригувальних шарів і масок програмного забезпечення Capture One дозволяє застосовувати будь-який інструмент редагування до кожного коригування [8]-[10], [12].

Програмне забезпечення Capture One також має вікно Cull, яке спрощує сортування великих обсягів зображень. Capture One надає перевагу підпискам, а не безстроковим ліцензіям, що не підходить деяким користувачам. Capture One Enterprise розроблено для підвищення ефективності студій, які обробляють великі обсяги зображень. Програмне забезпечення Capture One

Enterprise поєднує потужні можливості обробки зображень Capture One Pro з розширеним набором інструментів і автоматизованими рішеннями, які можна адаптувати до потреб бізнесу. Capture One Enterprise автоматизує дорогі та трудомісткі ручні процеси, дозволяючи швидше виводити продукти на ринок і значно знижуючи вартість обробки зображень [8]-[10], [12].

Програмне забезпечення Capture One підходить для технічних спеціалістів з фотографії, малих та середніх професійних студій, а також корпоративних студій [8]-[10], [12].

Програмне забезпечення Capture One має такі особливості [8]-[10], [12]:

- висока якість обробки графічних файлів: програмне забезпечення Capture One надає можливість покращеної обробки графічних файлів з кращим відтворенням кольорів та деталізацією [8]-[10], [12];

- неруйнівне редагування: програмне забезпечення Capture One надає можливість редагування без зміни оригінальних файлів [8]-[10], [12];

- засоби каталогізації та управління файлами: програмне забезпечення Capture One забезпечує ефективне управління великими бібліотеками зображень із зручними інструментами для сортування, фільтрації та пошуку зображень [8]-[10], [12];

- засоби локального коригування: програмне забезпечення Capture One забезпечує точне редагування вибраних ділянок зображення за допомогою масок і коригувальних шарів [8]-[10], [12];

- можливість знімання з безпосереднім передаванням зображень на комп'ютер, що ідеально підходить для студійної роботи [8]-[10], [12];

- інструмент для спільної роботи з обробки зображень в режимі реального часу з клієнтами [8]-[10], [12];

- розширені інструменти корекції кольорів: програмне забезпечення Capture One надає інструменти для точного налаштування кольорів за допомогою інтуїтивно зрозумілих інструментів [8]-[10], [12];

- шари та маски: програмне забезпечення Capture One надає великий вибір шарів і масок для складних коригувань і ретуші [8]-[10], [12];

- фільтри та попередньо встановлені ефекти: програмне забезпечення Capture One надає великий вибір фільтрів і попередньо встановлених ефектів для швидкого застосування [8]-[10], [12];

- розширені інструменти автоматизації: програмне забезпечення Capture One Enterprise пропонує додаткові інструменти та рішення для автоматизації, що підвищують ефективність роботи студій [8]-[10], [12];

- інструменти для ефективного сортування та відбору великої кількості зображень [8]-[10], [12];

- інтеграція з іншими платформами: можливість інтеграції програмного забезпечення Capture One з іншими платформами та інструментами для більш ефективного робочого процесу [8]-[10], [12].

Серед недоліків програмного забезпечення Capture One можна відзначити обмежені можливості безкоштовної версії [8]-[10], [12], оскільки деякі функції та інструменти доступні тільки у платній версії, що може обмежувати можливості користувачів без підписки. Інтернет версія Capture One потребує стабільного з'єднання для роботи, що може бути незручно в умовах обмеженого доступу до мережі [8]-[10], [12]. Безкоштовна версія Capture One може містити рекламу, що може відволікати користувачів або заважати їм під час редагування зображень [8]-[10], [12].

Програмне забезпечення PhotoLab (рис. 1.3) являє собою досит потужний і ефективний засіб для обробки зображень. Лабораторно розроблені корекції об'єктивів вражають своєю точністю, також програмне забезпечення PhotoLab виправляє м'якість країв. Засоби покращення зображень забезпечують чудові результати для зображень з високим контрастом, а функція вкладених проектів значно покращує організацію фотографій. Усі коригування зображень можна змінювати, видаляти або додавати в будь-який час, а також створювати віртуальні копії для випробування різних налаштувань без дублювання оригінальних файлів [8]-[10], [13].

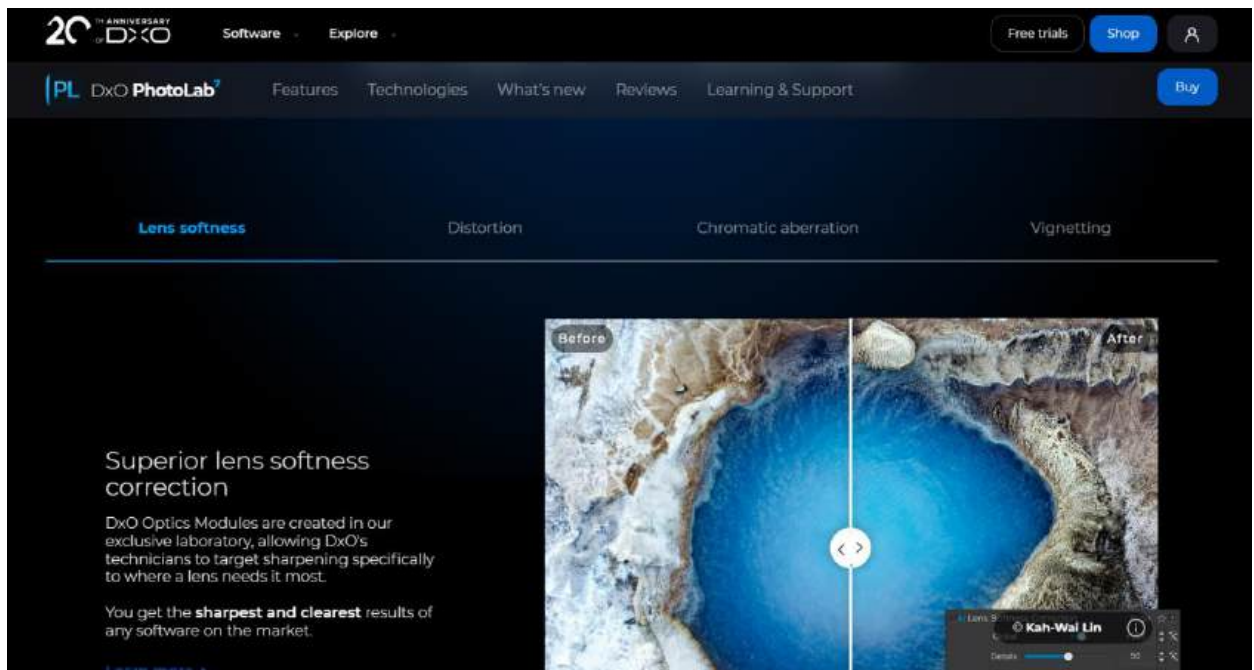


Рисунок 1.3 – Програмне забезпечення PhotoLab [13]

Програмне забезпечення PhotoLab є менеджером цифрових ресурсів і редактором зображень, програмою для управління цифровими активами та дозволяє впорядковувати цифрові зображення, призначаючи їм кольорові мітки, рейтинги та ключові слова, що допомагає швидше знаходити потрібні файли за допомогою фільтрації [8]-[10], [13].

Програмне забезпечення PhotoLab надає можливість фільтрувати файли за метаданими, наприклад, відображати зображення відповідно до їхньої швидкості, що допомагає знайти фотографії з шумами для корекції. PhotoLab має потужний набір інструментів цифрової темної кімнати. Програмне забезпечення PhotoLab використовує повну колекцію оптичних модулів для швидкого та ефективного виправлення артефактів об'єтивів, таких як спотворення та хроматична аберация. Ці оптичні модулі особливо корисні для тих, хто знімає у високій якості [8]-[10], [13].

Основною особливістю програмного забезпечення PhotoLab є оптичні модулі, які автоматично коригують поширені артефакти, такі як бочкоподібне або подушкоподібне спотворення об'єктива. При цьому програма автоматично пропонує завантажити відповідний оптичний модуль для конкретної камери та

налаштування об'єктива, використаного для цієї фотографії. Ще однією важливою функцією програмного забезпечення PhotoLab є усунення шумів. Завдяки великій бібліотеці зображень, інструменти шумозаглушення на основі штучного інтелекту в PhotoLab здатні ефективно згладжувати яскраві та кольорові шуми [8]-[10], [13].

Інша корисна функція програмного забезпечення PhotoLab полягає в автоматично встановленому плагіні, який дозволяє експортувати відредаговані файли з PhotoLab 7. Таким чином, користувач може скористатися потужними інструментами обробки високоякісних зображень у PhotoLab та завершити редагування в інших програмних засобах, якщо це необхідно [8]-[10], [13].

Програмне забезпечення PhotoLab дозволяє переглядати фотографії, збережені в папках на комп'ютері користувача або підключеному зовнішньому жорсткому диску, що означає відсутність додаткових файлів при імпорті в програму. Інтерфейс програмного забезпечення PhotoLab являє собою чистий сіро-чорний дизайн, знайомий користувачам сучасних програм для керування цифровими зображеннями і цифрових темних кімнат. Інтерфейс організований у дві основні робочі області з вкладками. Програмне забезпечення PhotoLab дозволяє переглядати та впорядковувати зображення за допомогою класичних інструментів, таких як рейтинги, кольорові мітки та ключові слова [8]-[10], [13].

Програмне забезпечення PhotoLab має такі особливості [8]-[10], [13]:

- засоби неруйнівного редагування: усі коригування можуть бути змінені, видалені або додані в майбутньому без втрати якості оригінального зображення. Можливість створювати віртуальні копії, щоб випробувати різні коригування без необхідності дублювання файлів [8]-[10], [13];

- передова обробка зображень у високій якості. Програмне забезпечення PhotoLab підтримує високоякісну обробку зображень, покращене зменшення шумів, особливо для зображень з високим рівнем шуму [8]-[10];

- вбудована бібліотека оптичних модулів для автоматичного виправлення спотворень об'єктива та хроматичних аберацій [8]-[10], [13];

- оптичні модулі постійно оновлюються, що дозволяє обробляти навіть старі файли поліпшеною якістю [8]-[10], [13];
- засоби управління цифровими активами: програмне забезпечення PhotoLab надає можливість впорядковувати цифрові зображення за допомогою кольорових міток, рейтингів та ключових слів [8]-[10], [13];
- функції фільтрації файлів за метаданими для швидкого пошуку конкретних зображень [8]-[10], [13];
- підтримка класичних інструментів для редагування зображень, таких як корекція кольору, експозиції, контрасту тощо [8]-[10], [13];
- зручний інтерфейс, організований у дві основні робочі області: програмне забезпечення PhotoLab надає засоби PhotoLibrary для організації зображень та PhotoLab для їх редагування [8]-[10], [13];
- можливість експорту редагованих файлів в інші програмні засоби для подальшого редагування. Інтеграція з іншими популярними інструментами для редагування фотографій [8]-[10], [13];
- інтуїтивний інтерфейс: програмне забезпечення PhotoLab має сіро-чорний дизайн, який виглядає знайомим для користувачів інших програм для редагування зображень [8]-[10], [13];
- висока продуктивність: програмне забезпечення PhotoLab підходить для обробки великої кількості фотографій, включаючи серійні зйомки. програмне забезпечення PhotoLab оптимізовано для швидкої та ефективної роботи як на апаратному, так і на програмному рівні [8]-[10], [13];
- розширені інструменти для професійних користувачів: програмне забезпечення PhotoLab підтримує функції для професійних комерційних фотографів, таких як зйомка в прив'язці до комп'ютера [8]-[10], [13];
- можливість співпраці з клієнтами в режимі реального часу через відповідний сервіс [8]-[10], [13].

Серед недоліків програмного забезпечення PhotoLab можна відзначити обмежені можливості безкоштовної версії [8]-[10], [13], оскільки деякі функції та інструменти доступні тільки у платній версії, що може обмежувати

можливості користувачів без підписки. Інтернет версія PhotoLab потребує стабільного з'єднання для роботи, що може бути незручно в умовах обмеженого доступу до мережі [8]-[10], [13]. Безкоштовна версія PhotoLab може містити рекламу, що може відволікати користувачів або заважати їм під час редагування зображень [8]-[10], [13].

Порівняльну характеристику програмного забезпечення для обробки зображень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для підтримки процесу обробки зображень

Критерій порівняння	Fotor [11]	Capture One [12]	PhotoLab [13]
Зручний інтерфейс	+—	+	+—
Кросплатформність	+	—	+—
Інтеграція з іншими платформами	—	+	+—
Інструменти для соціальних мереж	+	—	—
Хмарне зберігання даних	+—	+	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для підтримки процесу обробки, зокрема фільтрації, зображень. Проте використання деяких програмних засобів для фільтрації зображень може призвести до втрати дрібних деталей в зображеннях, особливо якщо вони використовують неефективні методи фільтрації зображень для видалення шуму або інших небажаних елементів. Застосування інших програмних засобів до обробки зображень може призвести до зміни оригінальних кольорів зображення, що може бути небажаним для професійної фотографії або інших застосувань, де точність кольору є критичною. Використання складних методів фільтрації

зображень, особливо тих, що використовують алгоритми штучного інтелекту та машинного навчання, можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для менш потужних пристроїв або для обробки великих обсягів візуальних даних.

Тому актуальною є розробка програмного забезпечення, що реалізує різні методи фільтрації зображень.

При розробці програмного забезпечення для фільтрації зображень необхідно забезпечити такі функціональні вимоги:

- підтримка можливості завантаження зображення для обробки;
- можливість вибору методу (фільтру) для обробки зображення шляхом фільтрації;
- можливість відображення результатів фільтрації за допомогою обраного фільтра;
- підтримка можливості скидання застосованого до зображення фільтру;
- можливість збереження відфільтрованого зображення у файл.

1.3 Висновки за розділом 1

Відзначено, що поліпшення якості зображень є процесом обробки зображень, пов'язаний з підвищенням чіткості, роздільної здатності, контрастності та інших важливих параметрів зображення з метою покращення сприйняття інформації, яка міститься на зображенні. Фільтрація зображень являє собою підхід, який застосовується в обробці зображень для покращення або зміни їх візуального вигляду. Проаналізовано практичні застосування фільтрації зображень. Відзначено, що фільтрація зображень відіграє ключову роль в обробці зображень, покращуючи важливі характеристики, зменшуючи шум і готуючи зображення для подальшого аналізу чи візуалізації.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки процесу обробки,

зокрема фільтрації, зображень. Проте використання деяких програмних засобів для фільтрації зображень може призвести до втрати дрібних деталей в зображеннях, особливо якщо вони використовують неефективні методи фільтрації зображень для видалення шуму або інших небажаних елементів. Застосування інших програмних засобів до обробки зображень може призвести до зміни оригінальних кольорів зображення, що може бути небажаним для професійної фотографії або інших застосувань, де точність кольору є критичною. Використання складних методів фільтрації зображень, особливо тих, що використовують алгоритми штучного інтелекту та машинного навчання, можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для менш потужних пристроїв або для обробки великих обсягів візуальних даних. Тому актуальною є розробка програмного забезпечення, що реалізує різні методи фільтрації зображень.

Сформульовано функціональні вимоги до програмного забезпечення для фільтрації зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для фільтрації зображень було використано мову програмування C# [14], [15].

C# – є об'єктно-орієнтованою мовою програмування зі стабільною типізацією, яка дозволяє розробникам створювати безпечні та надійні програми на платформі .NET. Гнучкість C# дозволяє створювати надійні та адаптивні програми, особливо корисно при управлінні пам'яттю для невідомих, недоступних об'єктів. Типи допомагають уникнути змінних, не пов'язаних із звичайними об'єктами [14], [15].

Програми на C# працюють на платформі .NET, використовуючи Common Language Runtime та набір бібліотек класів [14], [15].

Обробка винятків забезпечує систематичний і масштабований підхід до виявлення та відновлення помилок, а лямбда-вирази дозволяють програмістам писати код на основі функцій [14], [15].

Як і Objective-C, C# належить до сімейства мов C, що робить її знайомою для розробників C, C++, Java та JavaScript [15], [16].

Мова програмування C# пропонує високу продуктивність і швидкість на iOS і Android. C# має полегшену підтримку коду, що допомагає скоротити час розробки. Мова програмування C# дозволяє працювати з API [14], [15].

Це невдалий вибір для програм зі складними комбінаціями [14], [15].

Мова програмування C# заснована на мовах C++ та C. Спочатку ця мова була створений для спеціалізованого програмування, але з часом він став загальною та універсальною мовою. Її використовують для написання коду в різних сферах: від управління ресурсами до обробки даних та загального програмування. Технічно C# є мовою програмування, яка дозволяє користувачеві компілювати дані та код у зрозумілу програму. Потім код транслюється компілятором у комп'ютерно-виконувані інструкції, що робить мову швидшою та ефективнішою [14], [15].

Мова програмування C# є портативною мовою програмування, що дозволяє запускати одну й ту саму програму на різних операційних системах. Мова програмування C# підтримує загальну парадигму, яка дозволяє кодувати різними способами. Мова програмування C# є об'єктно-орієнтованою мовою, що дозволяє повторно використовувати дані та методи організації коду. Мова програмування C# заснована на мові C++, що є близькою до машинної мови, тому вона полегшує маніпуляції з об'єктами. Мова програмування C# дозволяє програмісту мати контроль над управлінням пам'яттю [14], [15].

Мова програмування C# Мова має велику спільноту розробників, багато ресурсів, безкоштовних посібників та книг [14], [15].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для фільтрації зображень наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для фільтрації зображень

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Зручність розробки GUI	+	+—	+
Висока продуктивність	+	—	+—
Зручність для створення ПЗ для фільтрації зображень	+	—	—
Засоби обробки зображень	+	+	+—
Підтримка засобів штучного інтелекту та машинного навчання	+	+	+—

Отже, для реалізації програмного забезпечення для фільтрації зображень обрано мову програмування C#, оскільки C# має доступ до потужних бібліотек для роботи з графічними зображеннями, може інтегруватися з бібліотеками для роботи з графікою на низькому рівні. Крім

того, C# підтримує багатопоточність, що дозволяє ефективно розподіляти навантаження при обробці зображень, та має велику кількість готових бібліотек та фреймворків для обробки зображень.

2.2 Вибір середовища розробки для створення програмного забезпечення для фільтрації зображень

В якості середовища розробки для створення програмного забезпечення для фільтрації зображень було обрано середовище Microsoft Visual Studio [16], [17].

Середовище розробки Microsoft Visual Studio є інтегрованою платформою розробки від Microsoft, створеною для підтримки розробників програмного забезпечення. Середовище розробки Microsoft Visual Studio використовує штучний інтелект для аналізу коду, що полегшує автоматичне виконання фрагментів коду [16], [17].

Однією з основних переваг Visual Studio, яку відзначають багато користувачів, є можливість спільної розробки в реальному часі між командами. Ця функція особливо корисна для налагодження. Середовище розробки Microsoft Visual Studio також дозволяє користувачам взаємодіяти з серверами, комп'ютерами та терміналами [16], [17].

Середовище розробки Microsoft Visual Studio підтримує розробку програм для персональних комп'ютерів, мобільних застосунків, вебзастосунків та ігор. Середовище розробки Microsoft Visual Studio підтримує мови C#, Python, Node.js, ASP.NET і Azure. Використовуючи середовище розробки Visual Studio, користувачі можуть легко створювати хмарне середовище для розробки [16], [17].

Завдяки підтримці кількох мов середовище розробки Visual Studio повністю інтегрується з різними фреймворками, що дозволяє використовувати мову C# для програмування під різні операційні системи, зокрема, операційні системи Mac, Windows і Linux [16], [17].

Середовище розробки Microsoft Visual Studio забезпечує високий рівень інтелектуального аналізу коду, швидке встановлення, надійні функції тестування [16], [17].

Середовище розробки Microsoft Visual Studio є одним з найпопулярніших середовищ розробки. Це кросплатформенний інструмент із сильною підтримкою функції перетягування. Середовище розробки Microsoft Visual Studio також має багато корисних функцій, таких як статичне тестування, налагодження та профілювання. Середовище розробки Microsoft Visual Studio орієнтований на розробку програмних проєктів різного призначення і дозволяє розробникам легко працювати над великими та складними проєктами [16], [17].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для фільтрації зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для фільтрації зображень

Критерій порівняння середовищ розробки	Середовища розробки		
	Microsoft Visual Studio	Xcode	Android Studio
Зручність використання	+	—	—
Розширені можливості для роботи з графікою	+	+—	—
Підсвічування синтаксису коду на C#	+	+—	+—
Автозавершення коду	+	+	+—
Засоби рефакторингу	+	+	+—

Таким чином, для створення програмного забезпечення для фільтрації зображень обрано середовище розробки Microsoft Visual Studio, що надає потужний редактор коду з підтримкою підсвічування синтаксису, автозавершення коду, рефакторингу та інші функції, що значно підвищують продуктивність розробки програм на мові C#. Крім того, середовище розробки Microsoft Visual Studio підтримує велику кількість розширень, які можна легко завантажити, що дозволяє додавати нові функції та інструменти, які можуть бути корисними для розроблюваного програмного продукту.

2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для фільтрації зображень

З метою можливості розробки програмного забезпечення для фільтрації зображень створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для фільтрації зображень враховуємо те, що користувачі програми мають можливості завантаження зображення для обробки, вибору методу (фільтру) для обробки зображення шляхом фільтрації, відображення результатів фільтрації за допомогою обраного фільтра, скидання застосованого до зображення фільтру. Крім того, повинна бути забезпечена можливість збереження відфільтрованого зображення у файл.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для фільтрації зображень можна описати таким чином:

- користувач програмного забезпечення для фільтрації зображень запускає програму;
- після запуску програми користувачу відображається головний екран;
- користувач бачить відповідну форму, де йому пропонується завантажити зображення;
- користувач завантажує зображення для обробки;

- користувач здійснює вибір методу (фільтру) для обробки зображення шляхом фільтрації;
- користувачу відображаються результати фільтрації за допомогою обраного фільтра;
- користувач зберігає відфільтроване зображення у файл
- користувач завершує роботу з програмним забезпеченням для фільтрації зображень.

Враховуючи сценарії роботи користувачів програми для фільтрації зображень, модель взаємодії користувача з програмним забезпеченням для фільтрації зображень подамо у вигляді діаграми прецедентів, зображеної на рис. 2.1.

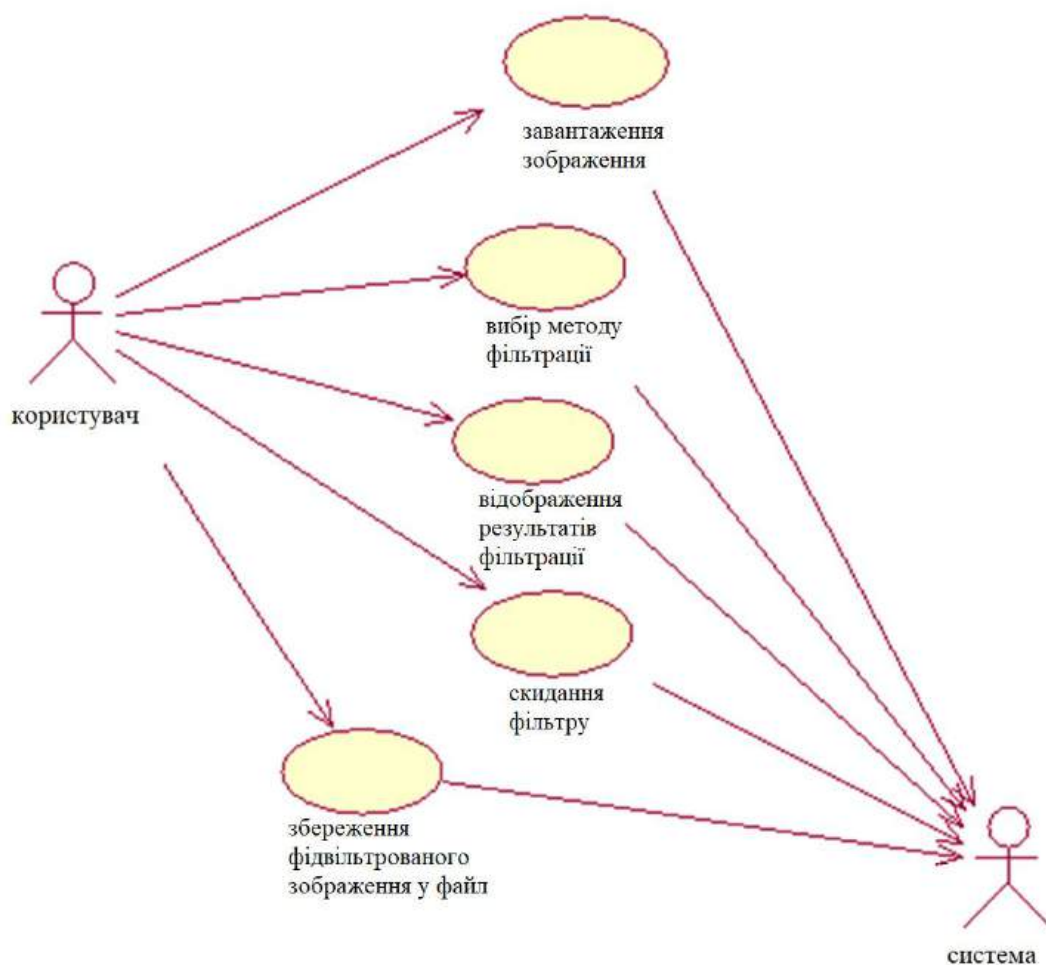


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням фільтрації зображень

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для фільтрації зображень, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для фільтрації зображень.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для фільтрації зображень обрано мову програмування C#, оскільки C# має доступ до потужних бібліотек для роботи з графічними зображеннями, може інтегруватися з бібліотеками для роботи з графікою на низькому рівні. Крім того, C# підтримує багатопоточність, що дозволяє ефективно розподіляти навантаження при обробці зображень, та має велику кількість готових бібліотек та фреймворків для обробки зображень.

Для створення програмного забезпечення для фільтрації зображень обрано середовище розробки Microsoft Visual Studio, що надає потужний редактор коду з підтримкою підсвічування синтаксису, автозавершення коду, рефакторингу та інші функції, що значно підвищують продуктивність розробки програм на мові C#. Крім того, середовище розробки Microsoft Visual Studio підтримує велику кількість розширень, які можна легко завантажити, що дозволяє додавати нові функції та інструменти, які можуть бути корисними для розроблюваного програмного продукту.

Створено модель взаємодії користувача з програмним забезпеченням для фільтрації зображень, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для фільтрації зображень.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для фільтрації зображень

Структуру програмного забезпечення для фільтрації зображень наведено на рис. 3.1.

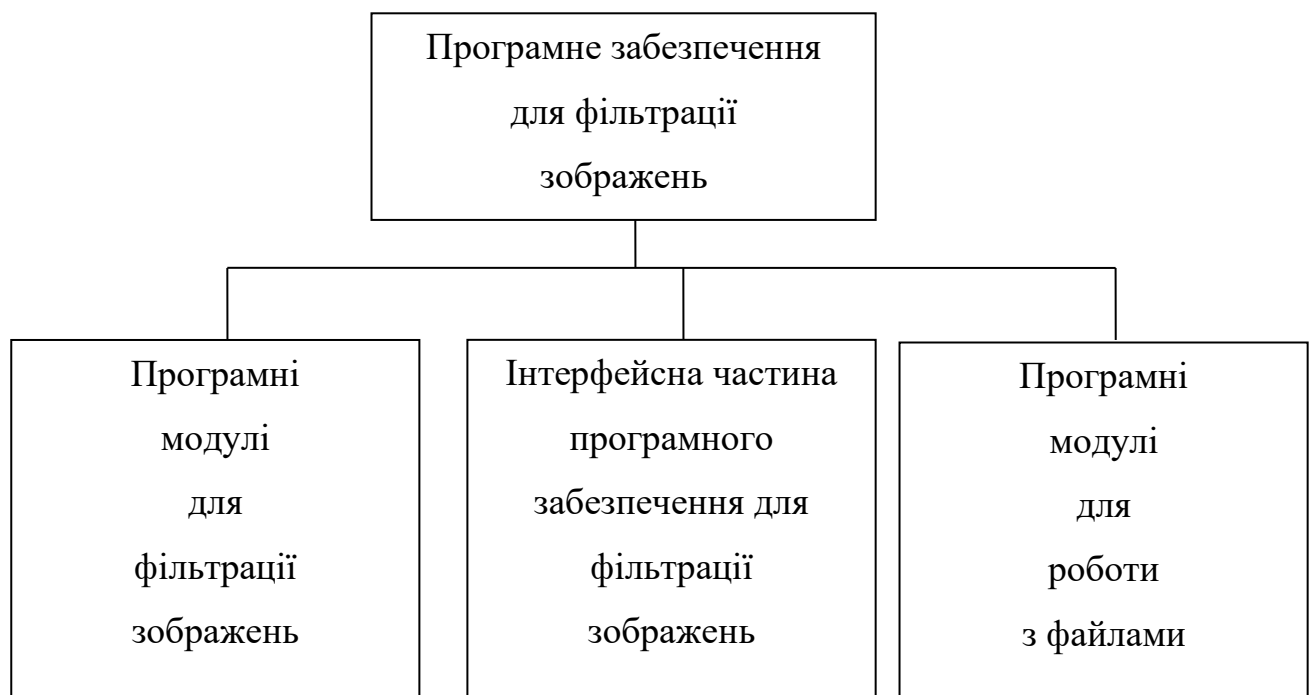


Рисунок 3.1 – Структура програмного забезпечення для фільтрації зображень

Як видно з рис. 3.1, програмне забезпечення для фільтрації зображень складається з трьох основних частин:

- програмні модулі для фільтрації зображень – містять програмну реалізацію методів фільтрації зображень, зокрема, медіанного методу, методу розмиття по рамці, методу Гаусса, оператора Прюїтта та рельєфного методу;
- інтерфейсна частина – містить структуру інтерфейсних форм користувача для роботи з системою;

– програмні модулі для роботи з файлами – містять засоби для роботи з файлами, що містять графічні зображення, зокрема, засоби для відкриття та збереження графічних файлів.

3.2 Функціонування програмного забезпечення для фільтрації зображень

Функціонування програмного забезпечення для фільтрації зображень подамо за допомогою схеми, зображеної на рис. 3.2.



Рисунок 3.2 – Схема функціонування програмного забезпечення для фільтрації зображень

Робота програмного забезпечення для фільтрації зображень розпочинається з введення користувачем файлу з зображенням для фільтрації. На наступному етапі обирається метод фільтрації. Після цього виконується фільтрація зображення на основі обраного методу. Далі за бажанням користувача відфільтроване зображення зберігається у файлі на диску, після чого відбувається завершення роботи програми.

3.3 Використання методів фільтрації у програмному забезпеченні для обробки зображень

При розробці програмного забезпечення для фільтрації зображень було використано різні методи фільтрації [18]–[20].

Фільтрація зображень заснована на виконанні декількох локальних перетворень над місцевими значеннями кольору зображення та відповідними значеннями декількох масивів однакового розміру. Така матриця називається фільтром, маскою, ядром, шаблоном або вікном. Значення елементів матриці фільтрації називаються коефіцієнтами. Операції фільтрації тут виконуються безпосередньо над компонентами зображення [18]–[20].

Для фільтрації зображень може використовуватися локальна фільтрація або доменна фільтрація. Метод заснований на масці, яка переміщується по зображенню, полягає у тому, що у кожній точці відгук фільтра обчислюється за допомогою відповідних коефіцієнтів. У лінійній просторовій фільтрації результат досягається шляхом поєднання отриманих коефіцієнтів фільтра з відповідними пікселями у зоні, охопленій маскою фільтра [18]–[20].

Це означає, що всі елементи зображення обробляються відповідно до заданої маски. Процес операції лінійного фільтрування є еквівалентом згортки у частотній області. З цієї причини лінійну просторову фільтрацію часто називають згорткою. Подібним чином фільтр зображення іноді називають маскою або згорткою ядра [18]–[20].

Для покращення якості зображення зазвичай використовують фільтри для видалення шуму та пом'якшення зображення. Оскільки спектр шуму містить багато локальних частот, простий фільтр низьких частот може зменшити шум [18]–[20].

Коефіцієнти фільтра можуть мати різну вагу, що в деяких випадках робить фільтр більш чутливим до змін в центрі зображення. Деякі фільтри адаптовані для комп'ютерної реалізації і ефективно зменшують шум, залишаючи тонкі деталі зображення недоторканими. Такі фільтри часто використовуються в нелінійних застосуваннях [18]–[20].

Нелінійні фільтри працюють подібно до лінійних, але їхні функції залежать від властивостей аналізованих факторів зовнішнього середовища і використовують лінійні комбінації. Наприклад, для видалення шуму може бути ефективніше використовувати нелінійний фільтр, основне завдання якого полягає в обчисленні середнього значення локальних змінних. Середня оцінка є нелінійною функцією [18]–[20].

Медіанна фільтрація – це нелінійний метод обробки зображень, корисний для придушення шуму на зображеннях. Центральна частина фільтра – це горизонтальне вікно, яке аналізує певну кількість пікселів. Центральний елемент розташований посередині цього вікна [18]–[20].

Частини зображення потрапляють у вікно під певними кутами. Медіанна фільтрація розраховує медіанне значення, яке замінить центральний елемент. Медіанний фільтр найкраще працює для видалення шуму з малюнків. Проте, якщо обране невірне медіанне значення, яке може бути частиною природного сигналу (наприклад, коли використовуються великі датчики), фільтрація може призвести до спотворення зображення. Тому медіанний фільтр у деяких випадках ефективно пригнічує шум, а в інших може пригнічувати важливі сигнали [18]–[20].

Розмір фільтра не впливає на чутливість до фільтрації зображення, що зазвичай є бажаним. Проте цей фільтр знижує вплив сигналів з періодом, що

не перевищує половину ширини вікна. Медіанний фільтр також добре підходить для згладжування поверхонь трикутної функції [18]–[20].

Для зменшення шуму можна використовувати різні методи центральної фільтрації. Один з них починається з центрального фільтра з вікном, яке охоплює три частини зображення. Якщо ослаблення сигналу незначне, розмір вікна збільшується. Цей процес триває, поки центр фільтра не почне погіршуватися [18]–[20].

Іншим методом є каскадна медіанна фільтрація із застосуванням вікна фіксованого або змінного розміру. Загалом, ділянки, які залишаються незмінними після одного проходу фільтра, не змінюються після повторних проходів. Якщо тривалість сигналів не перевищує половину ширини вікна, коригування проводяться після кожної операції [18]–[20].

Концепцію центрального фільтра можна легко застосувати у двійковій формі з двовимірним вікном потрібної форми, наприклад, прямокутника або кола. Очевидно, що фільтр із квадратним вікном розміру краще пригнічує шум, ніж ряд горизонтальних і вертикальних центральних ліній. Проте подвійний процес призводить до значно більшого зниження якості вихідного зображення [18]–[20].

Фільтр Гауса являє собою тип фільтра, де передавальна функція представлена функцією Гауса. Його основне призначення полягає в уникненні тимчасових перенапруг і збільшенні постійної часу завдяки малим груповим затримкам. Фільтр Гауса часто застосовується в цифровій обробці зображень для зменшення шуму. Однак, при перевищенні вибірки він може призвести до розмиття зображення [18]–[20].

Розмиття за Гаусом є популярним методом розмиття зображень, використовуючи простий Гаусів розподіл для обчислення варіацій кожного пікселя. Фільтр Гауса створює поверхню з центром у точці розподілу Гауса. Елементи з ненульовим розподілом застосовуються для згортки матриці з вихідним зображенням. Значення центрального пікселя мають найбільшу вагу

(велике значення Гауса), а сусідні пікселі мають меншу вагу, що зменшується із відстанню [18]–[20].

На практиці для обчислення Гаусового розподілу ігноруються елементи, які відрізняються більше ніж на три середньоквадратичні відхилення, оскільки їх вплив незначний. Тому для ефективного розрахунку достатньо використовувати певну матрицю. Також, через кругову симетрію, розмиття за Гаусом можна виконати як два незалежних одновимірних розмиття, спочатку по горизонталі, потім по вертикалі. Це дозволяє зменшити складність обчислень [18]–[20].

Оператор Прюїтта у комп'ютерному зорі являє собою метод виявлення країв в обробці зображень, що обчислює кілька значень за допомогою різних ядер, щоб визначити орієнтацію краю для кожного пікселя [18]–[20].

Оператор Прюїтта використовує різні фільтри для отримання кількох відповідей зображення. За допомогою зміни відношень ваг у кожному куті, можна досягти різних орієнтацій з певним кроком. Максимальна відповідь для кожного пікселя стає значенням пікселя у вихідному зображенні. Ці значення варіюються, залежно від кількості ядер, які дали максимальну відповідь. Цей метод виявлення країв також називають узгодженням країв, оскільки зображення зіставляється з набором графіків, кожен з яких представляє певну орієнтацію краю. Потім алгоритм визначає ширину та орієнтацію краю пікселя, який найкраще відповідає оточенню пікселя [18]–[20].

Правило краю Прюїтта дозволяє просто оцінити ширину та орієнтацію краю. У той час як диференційний детектор нахилу потребує складних обчислень для визначення орієнтації за вертикальними та горизонтальними значеннями, граничний детектор Прюїтта дозволяє здійснювати пряме вилучення з ядра з високою швидкістю. Набір орієнтацій обмежений вісьмома можливими варіантами, але практика показує, що більшість горизонтальних оцінок також є недостатньо точними [18]–[20]. На відміну від цього, ядро Прюїтта вимагає вісім згорток на піксель, тоді як режим градієнтного ядра потребує лише дві, чутливих до вертикальних та горизонтальних змін.

Пороговий ефект для обох методів схожий, оскільки вони використовують однакові ядра згортки [18]–[20]. Користувач використовує два ядра для перетворення оригінального зображення в наближені значення, одне для горизонтальних та одне для вертикальних компонентів [18]–[20].

Порівняння методів фільтрації зображень, що використовуються у програмному забезпеченні фільтрації зображень, наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння методів фільтрації зображень

Критерій порівняння методів фільтрації зображень	Методи фільтрації зображень				
	Медіанний метод	Метод розмиття по рамці	Метод Гаусса	Оператор Прюїтта	Рельєфний метод
Основний принцип	медіана для зменшення шуму	середнє значення в рамці	гаусів розподіл для згладжування	ядра для виявлення країв	амплітудні та просторові зміни
Ефективність зменшення шуму	добре знімає шуми від стаціонарних об'єктів	ефективно зменшує м'які шуми	ефективно згладжує зображення	менше ефективний на відсутність орієнтації	може впливати на зовнішній шум
Час обробки	швидкий, оскільки потребує лише порівняння	швидкий, ідеально підходить для великих об'єктів	може бути повільним для великих вікон	швидкий, бо застосовується до кожного пікселя	може бути повільним через складні алгоритми
Збереження різкості границь	зберігає границі об'єктів, відсутність змазки	згладжує границі, може втратити деталі	згладжує границі, але менше, ніж метод розмиття	відновлює розмежування границь для деталей	зберігає границі
Застосування	видалення шуму з малюнків	згладжування об'єктів	зменшення шуму	виявлення країв	аналіз топографії

3.4 Проектування інтерфейсу програмного забезпечення для фільтрації зображень

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для фільтрації зображень виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для підтримки процесу фільтрації зображень враховані функціональні вимоги до програми:

- підтримка можливості завантаження зображення для обробки;
- можливість вибору методу (фільтру) для обробки зображення шляхом фільтрації;
- можливість відображення результатів фільтрації за допомогою обраного фільтра;
- підтримка можливості скидання застосованого до зображення фільтру;
- можливість збереження відфільтрованого зображення у файл.

Інтерфейс головної форми програмного забезпечення для фільтрації зображень наведено на рис. 3.3.

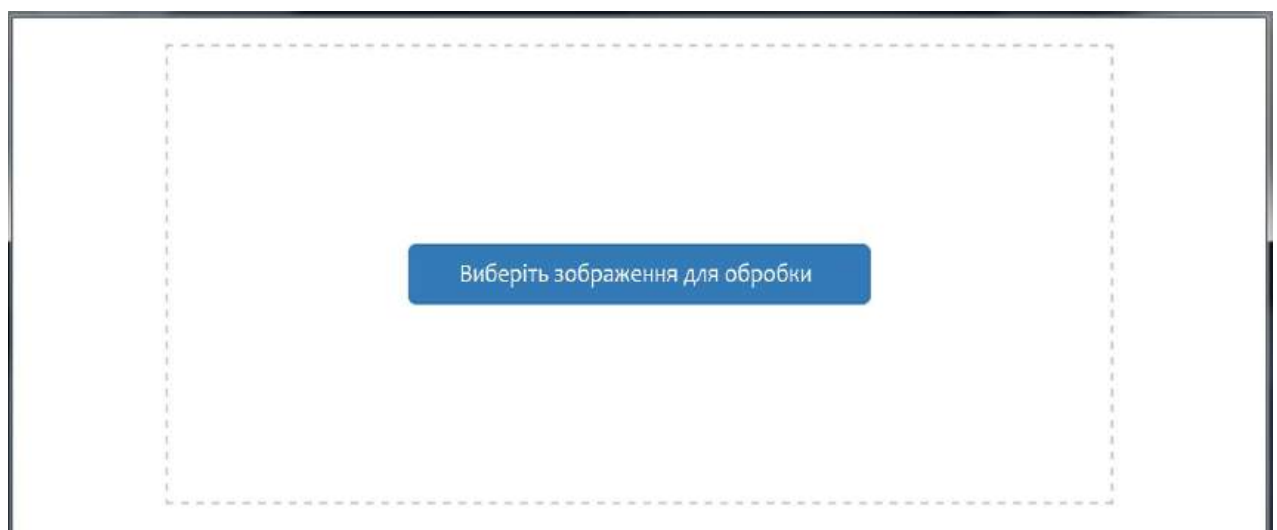


Рисунок 3.3 – Інтерфейс головної форми програмного забезпечення для фільтрації зображень

Інтерфейс області панелі інструментів фільтрації зображень наведено на рис. 3.4, де користувач може обрати різні методи фільтрації, а також дії з завантаженим зображенням.

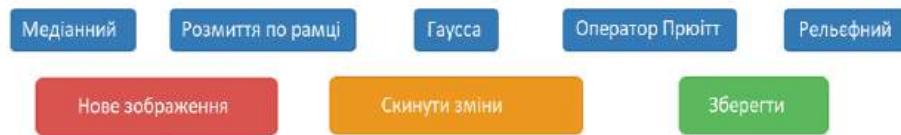


Рисунок 3.4 – Інтерфейс області панелі інструментів фільтрації зображень

Інтерфейс форми обробки завантаженого зображення за допомогою різних методів фільтрації наведено на рис. 3.5.

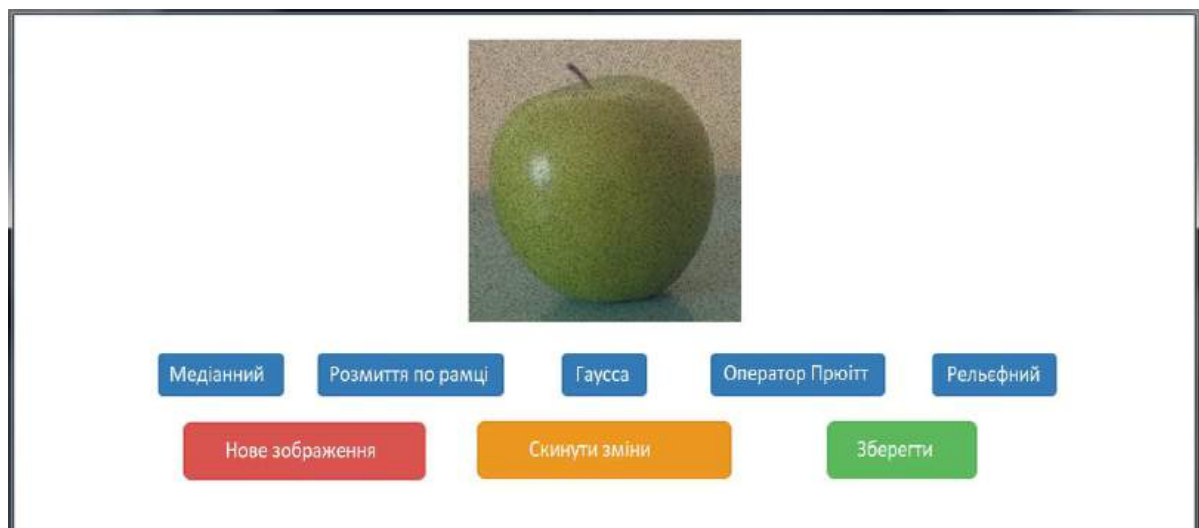


Рисунок 3.5 – Інтерфейс прототипу форми обробки завантаженого зображення за допомогою різних методів фільтрації

При розробці інтерфейсу програмного забезпечення для фільтрації зображень прийнято рішення про використання стандартного діалогового вікна операційної системи для збереження у файл відфільтрованих зображень (рис. 3.6).

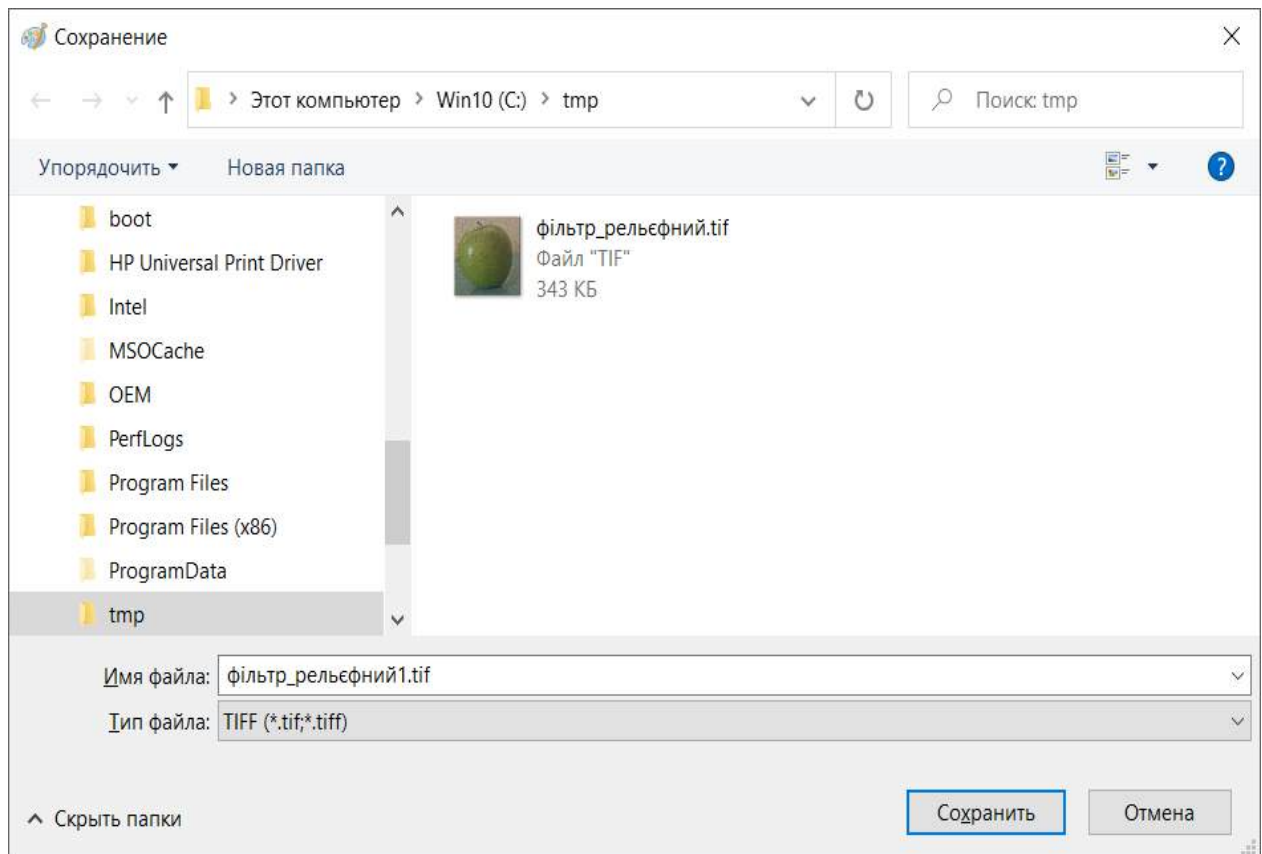


Рисунок 3.6 – Діалогове вікно для збереження у файл відфільтрованих зображень

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для фільтрації зображень.

Запропоновано структуру програмного забезпечення для фільтрації зображень. Визначено, що програмне забезпечення для фільтрації зображень складається з трьох основних частин: програмні модулі для фільтрації зображень, інтерфейсна частина, програмні модулі для роботи з файлами – містять засоби для роботи з файлами, що містять графічні зображення, зокрема, засоби для відкриття та збереження графічних файлів.

Описано функціонування програмного забезпечення для фільтрації зображень.

Описано особливості реалізації програмного забезпечення для фільтрації зображень. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для фільтрації зображень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу фільтрації зображень. Програма написана на мові C#, що має доступ до потужних бібліотек для роботи з графічними зображеннями, може інтегруватися з бібліотеками для роботи з графікою на низькому рівні, підтримує багатопоточність, що дозволяє ефективно розподіляти навантаження при обробці зображень, та має велику кількість готових бібліотек та фреймворків для обробки зображень. В якості середовища розробки для створення програмного забезпечення для підтримки процесу фільтрації зображень обрано Microsoft Visual Studio, що надає потужний редактор коду з підтримкою підсвічування синтаксису, автозавершення коду, рефакторингу та інші функцій, що значно підвищують продуктивність розробки програм на мові C#. Крім того, середовище розробки Microsoft Visual Studio підтримує велику кількість розширень, які можна легко завантажити, що дозволяє додавати нові функції та інструменти, які можуть бути корисними для розроблюваного програмного продукту.

Програмне забезпечення для фільтрації зображень забезпечує виконання таких функцій:

- підтримка можливості завантаження зображення для обробки;
- можливість вибору методу (фільтру) для обробки зображення шляхом фільтрації;
- можливість відображення результатів фільтрації за допомогою обраного фільтра;
- підтримка можливості скидання застосованого до зображення фільтру;
- можливість збереження відфільтрованого зображення у файл.

Для експлуатації програмного забезпечення для фільтрації зображень необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 16 Гб і більше;
- не менше 1 Гб вільного простору жорсткому диску;
- процесор з високою тактовою частотою (3.0 ГГц або вище);
- потужний графічний процесор;
- монітор з роздільною здатністю мінімум 1920x1080.

Функціональні характеристики розробленого програмного забезпечення для фільтрації зображень наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для фільтрації зображень наведено у додатку Б.

4.2 Характеристики програми для підтримки процесу фільтрації зображень

Програмне забезпечення для підтримки процесу фільтрації зображень має мати певні характеристики, щоб дозволяють йому бути ефективним і зручним у використанні. Програмне забезпечення для підтримки процесу фільтрації зображень підтримує здатність працювати з широким спектром форматів зображень (JPEG, PNG, TIFF, та інші). Програмне забезпечення для підтримки процесу фільтрації зображень надає можливість використання різноманітних фільтрів для обробки зображень, включаючи фільтри для зменшення шуму, підвищення різкості та ін.

Програмне забезпечення для підтримки процесу фільтрації зображень має зручний та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє легко знаходити та застосовувати потрібні функції.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для фільтрації зображень на комп'ютері користувача необхідно запустити відповідний виконуваний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для фільтрації зображень є зображення для фільтрації, а також обраний метод фільтрації.

Вихідними даними програмного забезпечення для фільтрації зображень є відфільтроване зображення, а також файл із збереженим відфільтрованим зображенням.

4.3.3 Повідомлення

Користувач програмного забезпечення для фільтрації зображень може отримати такі повідомлення: повідомлення про успішне збереження відфільтрованого зображення.

4.4 Виконання програмного забезпечення для підтримки процесу фільтрації зображень

Після запуску програмного забезпечення для підтримки процесу фільтрації зображень користувачу відображається головна форма (рис. 4.1), де пропонується завантажити зображення для фільтрації.

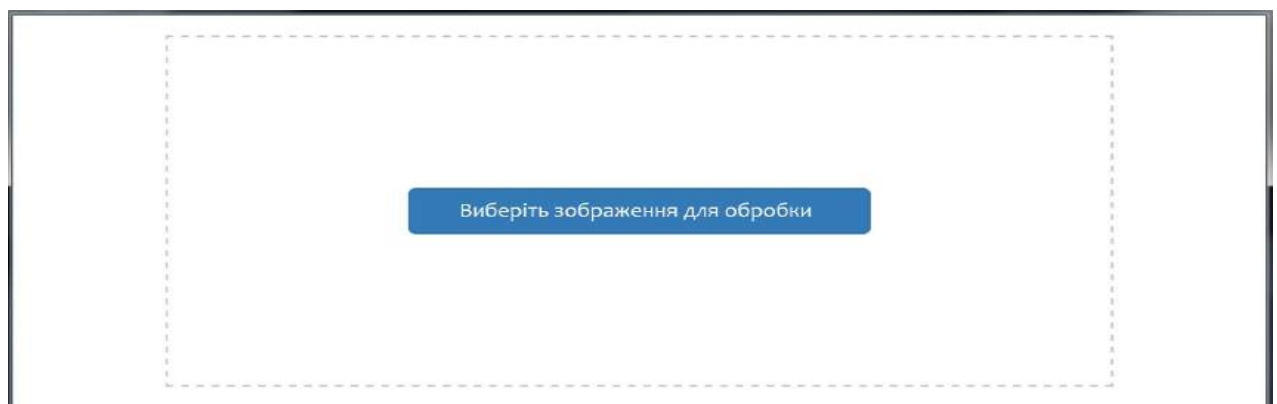


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для фільтрації зображень

Після завантаження користувачем зображення воно відображається. Програма пропонує користувачу скористатися декількома фільтрами для фільтрації обраного зображення (рис. 4.2).

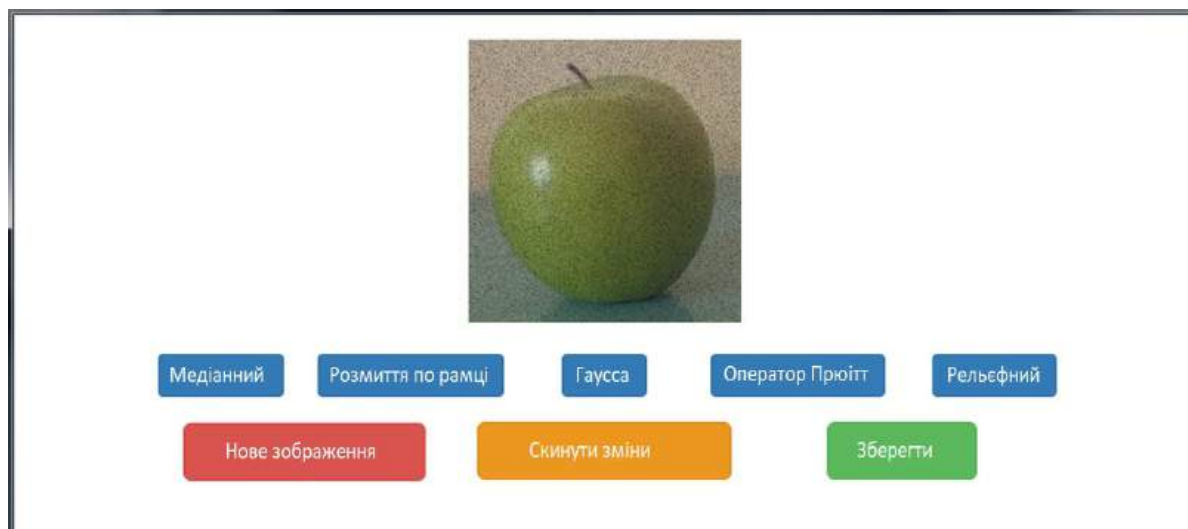


Рисунок 4.2 – Відображення завантаженого зображення

Для вибору методу фільтрації користувачу необхідно натиснути відповідну кнопку. Результат фільтрації зображення за допомогою медіанного фільтру наведено на рис. 4.3.



Рисунок 4.3 – Результат фільтрації зображення за допомогою медіанного фільтру

Результат фільтрації зображення за допомогою фільтру Гаусса наведено на рис. 4.4.

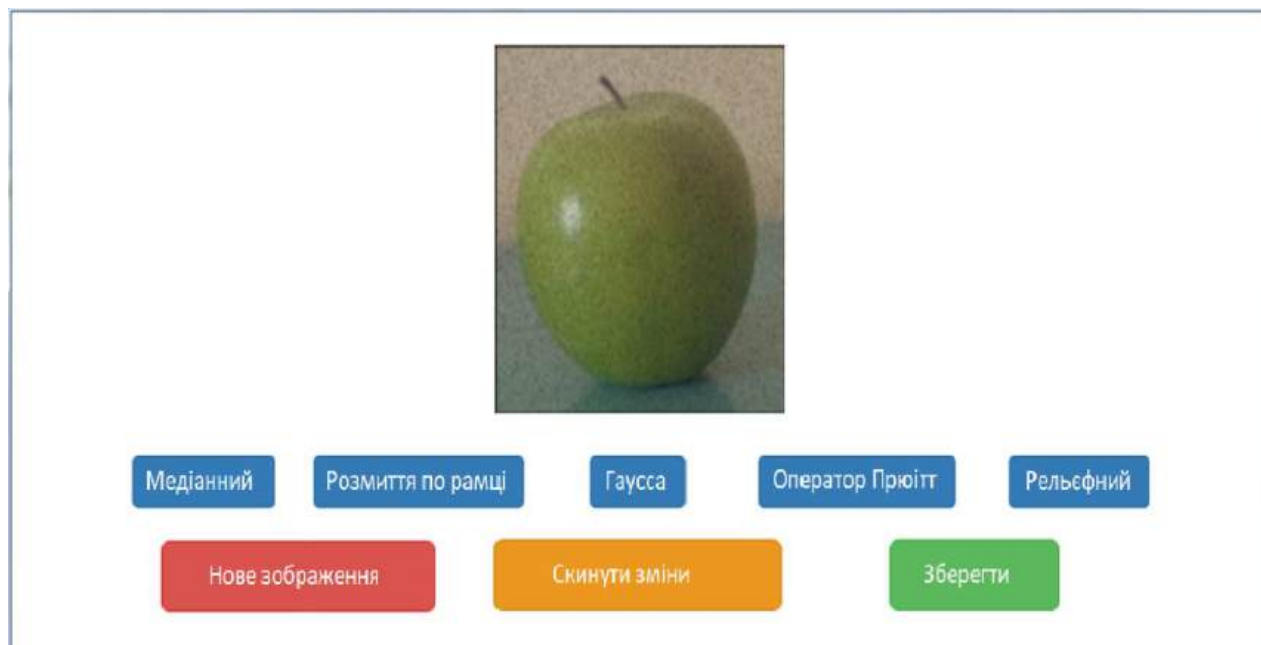


Рисунок 4.4 – Результат фільтрації зображення за допомогою фільтру Гаусса

Результат фільтрації зображення за рельєфного фільтру наведено на рис. 4.5.

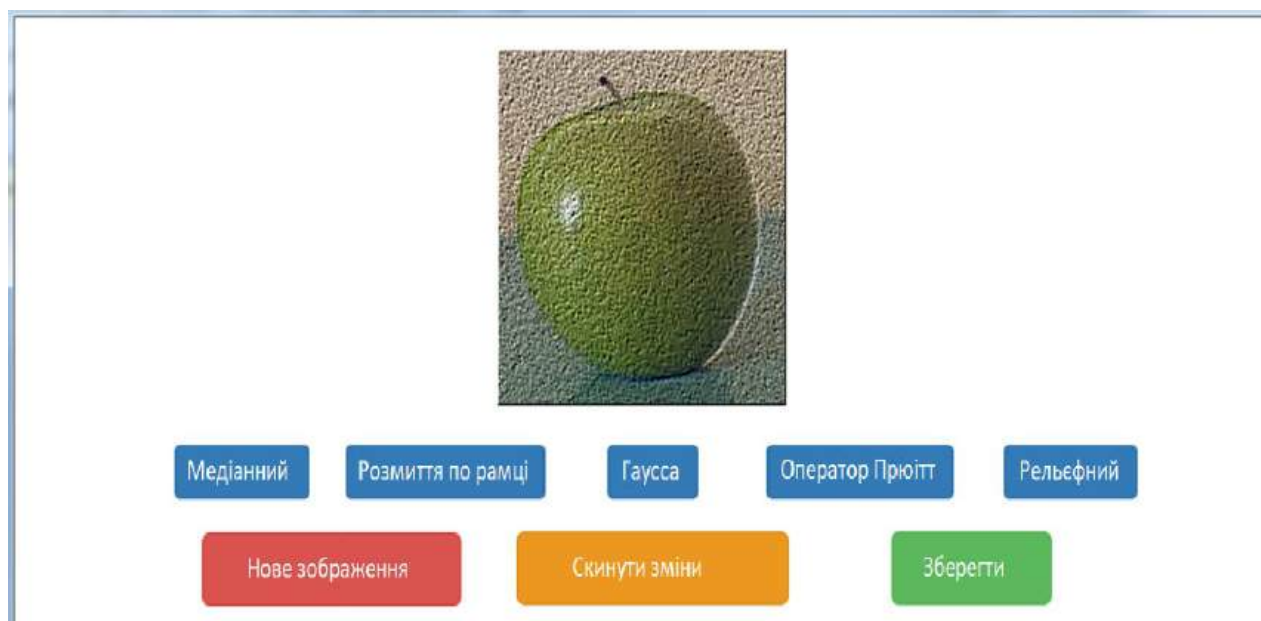


Рисунок 4.5 – Результат фільтрації зображення за рельєфного фільтру

Після обробки зображення шляхом його фільтрації користувач може зберегти отриманий результат, натиснувши кнопку Зберегти (рис. 4.6). Користувач також має можливість скинути зміни, отримані в результаті фільтрації, а також завантажити інше зображення для його обробки.

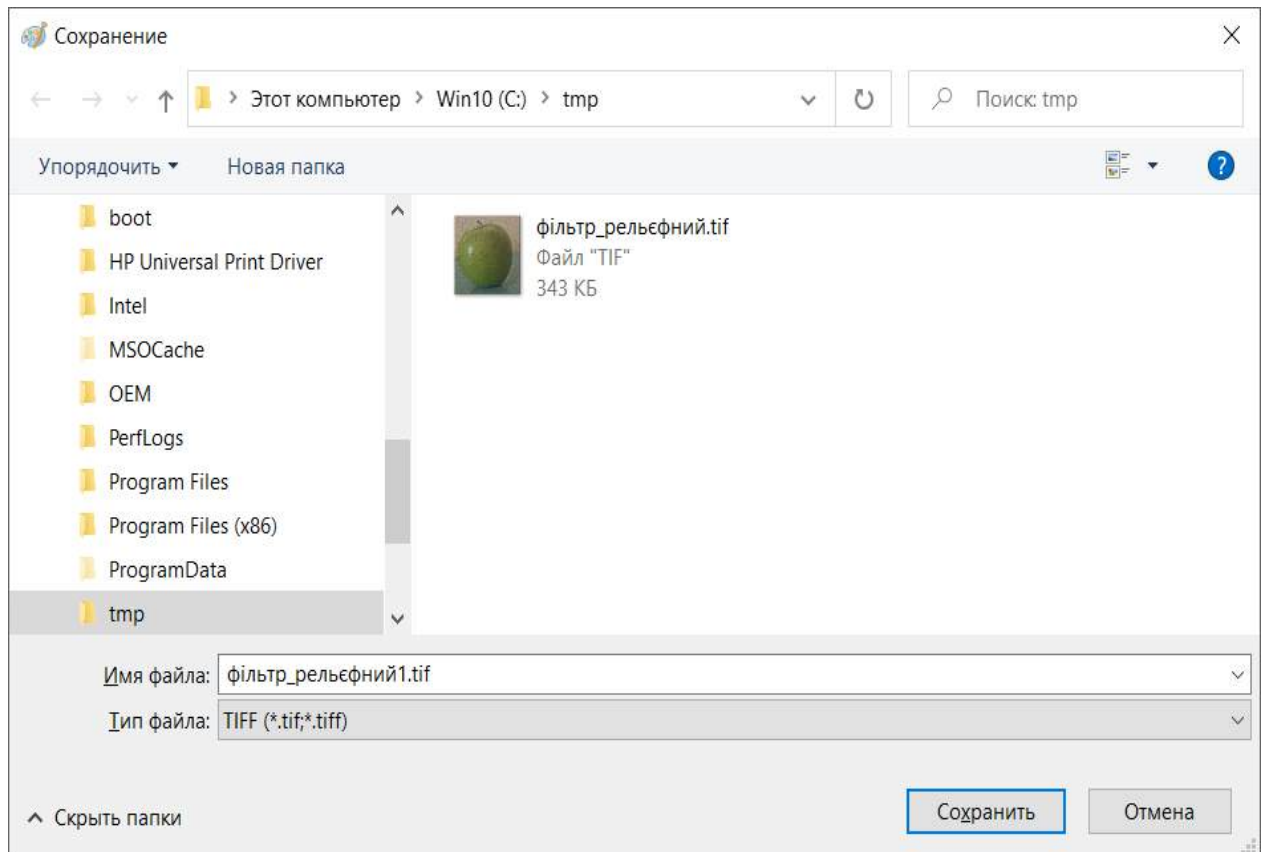


Рисунок 4.6 – Збереження відфільтрованого зображення

4.5 Тестування програмного забезпечення для підтримки процесу фільтрації зображень

Виконано тестування програмного забезпечення для підтримки процесу фільтрації зображень.

Виявлено, що програма для підтримки процесу фільтрації зображень функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу фільтрації зображень.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою фільтрації зображень.

4.6 Висновки за розділом 4

Описано програмне забезпечення для підтримки процесу фільтрації зображень. Виконано тестування розробленого програмного забезпечення для фільтрації зображень. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою фільтрації зображень.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для обробки зображень.

Відзначено, що поліпшення якості зображень є процесом обробки зображень, пов'язаний з підвищенням чіткості, роздільної здатності, контрастності та інших важливих параметрів зображення з метою покращення сприйняття інформації, яка міститься на зображенні. Фільтрація зображень являє собою підхід, який застосовується в обробці зображень для покращення або зміни їх візуального вигляду. Проаналізовано практичні застосування фільтрації зображень. Відзначено, що фільтрація зображень відіграє ключову роль в обробці зображень, покращуючи важливі характеристики, зменшуючи шум і готуючи зображення для подальшого аналізу чи візуалізації.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для підтримки процесу обробки, зокрема фільтрації, зображень. Проте використання деяких програмних засобів для фільтрації зображень може призвести до втрати дрібних деталей в зображеннях, особливо якщо вони використовують неефективні методи фільтрації зображень для видалення шуму або інших небажаних елементів. Застосування інших програмних засобів до обробки зображень може призвести до зміни оригінальних кольорів зображення, що може бути небажаним для професійної фотографії або інших застосувань, де точність кольору є критичною. Використання складних методів фільтрації зображень, особливо тих, що використовують алгоритми штучного інтелекту та машинного навчання, можуть вимагати значних обчислювальних ресурсів, що може бути проблемою для менш потужних пристроїв або для обробки великих обсягів візуальних даних. Тому актуальною є розробка програмного забезпечення, що реалізує різні методи фільтрації зображень.

Сформульовано функціональні вимоги до програмного забезпечення для фільтрації зображень.

Для реалізації програмного забезпечення для фільтрації зображень обрано мову програмування C#, оскільки C# має доступ до потужних бібліотек для роботи з графічними зображеннями, може інтегруватися з бібліотеками для роботи з графікою на низькому рівні. Крім того, C# підтримує багатопоточність, що дозволяє ефективно розподіляти навантаження при обробці зображень, та має велику кількість готових бібліотек та фреймворків для обробки зображень.

Для створення програмного забезпечення для фільтрації зображень обрано середовище розробки Microsoft Visual Studio, що надає потужний редактор коду з підтримкою підсвічування синтаксису, автозавершення коду, рефакторингу та інші функції, що значно підвищують продуктивність розробки програм на мові C#. Крім того, середовище розробки Microsoft Visual Studio підтримує велику кількість розширень, які можна легко завантажити, що дозволяє додавати нові функції та інструменти, які можуть бути корисними для розроблюваного програмного продукту.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для фільтрації зображень, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для фільтрації зображень.

Практична цінність результатів роботи полягає у тому, що розроблено програмне забезпечення для фільтрації зображень.

Запропоновано структуру програмного забезпечення для фільтрації зображень. Визначено, що програмне забезпечення для фільтрації зображень складається з трьох основних частин: програмні модулі для фільтрації зображень, інтерфейсна частина, програмні модулі для роботи з файлами – містять засоби для роботи з файлами, що містять графічні зображення, зокрема, засоби для відкриття та збереження графічних файлів.

Описано функціонування програмного забезпечення для фільтрації зображень.

Описано особливості реалізації програмного забезпечення для фільтрації зображень. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для фільтрації зображень.

Виконано тестування розробленого програмного забезпечення для фільтрації зображень. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою фільтрації зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Understanding Image Filtering Techniques in Image Processing [Electronic resource]. – Access mode: <https://www.imageprovision.com/articles/understanding-image-filtering-techniques-in-image-processing>.
2. Image Filtering Techniques in Image Processing [Electronic resource]. – Access mode: <https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-1-d03362fc73b7>.
3. Tutorial 1: Image Filtering [Electronic resource]. – Access mode: <https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html>.
4. Image Filtering [Electronic resource]. – Access mode: <https://www.sciencedirect.com/topics/computer-science/image-filtering>.
5. Image Filtering Using Convolution in OpenCV [Electronic resource]. – Access mode: <https://learnopencv.com/image-filtering-using-convolution-in-opencv/>.
6. Image Filtering Algorithms and Techniques: A Review [Electronic resource]. – Access mode: https://www.researchgate.net/publication/325681876_Image_Filtering_Algorithms_and_Techniques_A_Review.
7. Introduction to Image Processing [Electronic resource]. – Access mode: <https://towardsdatascience.com/introduction-of-image-processing-with-python-image-filtering-193e9108ea1d>.
8. Best Online Photo Editors for Filters and Effects [Electronic resource]. – Access mode: <https://design.tutsplus.com/articles/best-online-photo-editors-for-filters-and-effects-free-paid--cms-31126>.
9. The best photo editing software [Electronic resource]. – Access mode: <https://www.digitalcameraworld.com/buying-guides/the-best-photo-editing-software>.
10. Photo Editing Software [Electronic resource]. – Access mode: <https://www.capterra.com/photo-editing-software/>.

11. Fotor [Electronic resource]. – Access mode: <https://www.fotor.com/>.
12. Capture One [Electronic resource]. – Access mode: <https://www.captureone.com/en>.
13. PhotoLab [Electronic resource]. – Access mode: <https://www.dxo.com/dxo-photolab/>.
14. Documentation C# [Electronic resource]. – Access mode: <https://learn.microsoft.com/fr-ca/dotnet/csharp/tour-of-csharp/>.
15. C#. Langage de programmation open source moderne, innovant pour la création de toutes vos applications [Electronic resource]. – Access mode: <https://dotnet.microsoft.com/fr-fr/languages/csharp>.
16. GitHub Copilot intégré à Visual Studio [Electronic resource]. – Access mode: <https://visualstudio.microsoft.com/fr/>.
17. Microsoft Visual Studio Reviews [Electronic resource]. – Access mode: <https://www.softwareadvice.com/game-development/microsoft-visual-studio-profile/reviews/>.
18. A Survey On Image Filtering Algorithms [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/document/9096792>.
19. Image Filtering and Edge Detection [Electronic resource]. – Access mode: https://sbme-tutorials.github.io/2018/cv/notes/4_week4.html.
20. Image Filtering and Noise Reduction [Electronic resource]. – Access mode: <https://www.talentica.com/blogs/image-filtering-and-noise-reduction/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу фільтрації зображень.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів фільтрації зображень», затверджене наказом Національного університету «Запорізька політехніка» № 398 від 07 жовтня 2024 р.

A.2 Призначення розробки

Програмний продукт призначений для підтримки процесу фільтрації зображень.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для підтримки процесу фільтрації зображень забезпечує виконання таких функцій:

- підтримка можливості завантаження зображення для обробки;
- можливість вибору методу (фільтру) для обробки зображення шляхом фільтрації;
- можливість відображення результатів фільтрації за допомогою обраного фільтра;
- підтримка можливості скидання застосованого до зображення фільтру;
- можливість збереження відфільтрованого зображення у файл.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для підтримки процесу фільтрації зображень повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для підтримки процесу фільтрації зображень повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для підтримки процесу фільтрації зображень необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного забезпечення для підтримки процесу фільтрації зображень необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 16 Гб і більше;
- не менше 1 Гб вільного простору жорсткому диску;
- процесор з високою тактовою частотою (3.0 ГГц або вище);
- потужний графічний процесор;
- монітор з роздільною здатністю мінімум 1920x1080.

A.3.6 Вимоги до маркування і пакування

Програма для підтримки процесу фільтрації зображень може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для підтримки процесу фільтрації зображень».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для фільтрації зображень є зображення для фільтрації, а також обраний метод фільтрації.

Вихідними даними програмного забезпечення для фільтрації зображень є відфільтроване зображення, а також файл із збереженим відфільтрованим зображенням.

ДОДАТОК Б
Фрагмент тексту програми

```

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

class ZobrFltring
{
    static void Main(string[] args)
    {
        string vhidZobrPath = "vhid.jpg";
        string vyhidZobrPathMedian = "vyhid_median.jpg";
        string vyhidZobrPathBoxBlur = "vyhid_box_blur.jpg";
        string vyhidZobrPathGaussian = "vyhid_gaussian.jpg";
        string vyhidZobrPathEmboss = "vyhid_emboss.jpg";
        Bitmap vhidZobr = new Bitmap(vhidZobrPath);
        Bitmap vyhidZobrMedian = ZastMedianFltr(vhidZobr);
        vyhidZobrMedian.Save(vyhidZobrPathMedian, ZobrFormat.Jpeg);
        Bitmap vyhidZobrBoxBlur = ZastBoxBlurFltr(vhidZobr);
        vyhidZobrBoxBlur.Save(vyhidZobrPathBoxBlur, ZobrFormat.Jpeg);
        Bitmap vyhidZobrGaussian = ZastGaussianBlurFltr(vhidZobr);
        vyhidZobrGaussian.Save(vyhidZobrPathGaussian, ZobrFormat.Jpeg);
        Bitmap vyhidZobrEmboss = ZastEmbossFltr(vhidZobr);
        vyhidZobrEmboss.Save(vyhidZobrPathEmboss, ZobrFormat.Jpeg);
        Console.WriteLine("Фільтрація завершена. Відфільтровані зображення збережено.");
    }

    static Bitmap ZastMedianFltr(Bitmap dgerBitmap)
    {
        BitmapData dgerdt = dgerBitmap.LockBits(new Rectangle(0, 0,
            dgerBitmap.Width, dgerBitmap.Height),
            ZobrLockMode.ReadOnly, dgerBitmap.PixelFormat);

        Bitmap rsltZobr = new Bitmap(dgerBitmap.Width, dgerBitmap.Height);

        BitmapData rsltDt = rsltZobr.LockBits(new Rectangle(0, 0,
            rsltZobr.Width, rsltZobr.Height),
            ZobrLockMode.WriteOnly,
            PixelFormat.Format32bppArgb);

        int btsPrPxl = Bitmap.GetPixelFormatSize(dgerBitmap.PixelFormat) / 8;
        int btKlk = dgerdt.Stride * dgerBitmap.Height;
        byte[] pxlBfr = new byte[btKlk];
    }
}

```

```

byte[] rsltBfr = new byte[btKlk];

IntPtr dgerScan0 = dgerdt.Scan0;
IntPtr rsltScn0 = rsltDt.Scan0;

System.Runtime.InteropServices.Marshal.Copy(dgerScan0,    pxlBfr,    0,
btKlk);
dgerBitmap.UnlockBits(dgerdt);

int fltrZmish = 1;
int calcZmish = 0;

int byteZmish = 0;

byte[] nghbrPxls = new byte[9];
byte[] mddPxl;

for (int zmishY = fltrZmish; zmishY <
    dgerBitmap.Height - fltrZmish; zmishY++)
{
    for (int zmishX = fltrZmish; zmishX <
        dgerBitmap.Width - fltrZmish; zmishX++)
    {
        byteZmish = zmishY *
            dgerdt.Stride +
            zmishX * btsPrPxl;

        for (int fltrY = -fltrZmish;
            fltrY <= fltrZmish; fltrY++)
        {
            for (int fltrX = -fltrZmish;
                fltrX <= fltrZmish; fltrX++)
            {
                calcZmish = byteZmish +
                    (fltrX * btsPrPxl) +
                    (fltrY * dgerdt.Stride);

                nghbrPxls[fltrY + fltrZmish +
                    (fltrX + fltrZmish) * 3] =
                    pxlBfr[calcZmish];
            }
        }
    }
}

```

```

        Array.Sort(nghbrPxls);

        mddPx1 = nghbrPxls[4];

        rsltBfr[byteZmish] = mddPx1[0];
        rsltBfr[byteZmish + 1] = mddPx1[1];
        rsltBfr[byteZmish + 2] = mddPx1[2];
        rsltBfr[byteZmish + 3] = 255;
    }
}

System.Runtime.InteropServices.Marshal.Copy(rsltBfr, 0, rsltScn0,
btKlk);
rsltZobr.UnlockBits(rsltDt);

return rsltZobr;
}

static Bitmap ZastBoxBlurFltr(Bitmap dgerBitmap)
{
    BitmapData dgerdt = dgerBitmap.LockBits(new Rectangle(0, 0,
        dgerBitmap.Width, dgerBitmap.Height),
        ZobrLockMode.ReadOnly, dgerBitmap.PixelFormat);

    Bitmap rsltZobr = new Bitmap(dgerBitmap.Width, dgerBitmap.Height);

    BitmapData rsltDt = rsltZobr.LockBits(new Rectangle(0, 0,
        rsltZobr.Width, rsltZobr.Height),
        ZobrLockMode.WriteOnly,
PixelFormat.Format32bppArgb);

    int btsPrPx1 = Bitmap.GetPixelFormatSize(dgerBitmap.PixelFormat) / 8;
    int btKlk = dgerdt.Stride * dgerBitmap.Height;
    byte[] pxlBfr = new byte[btKlk];
    byte[] rsltBfr = new byte[btKlk];

    IntPtr dgerScan0 = dgerdt.Scan0;
    IntPtr rsltScn0 = rsltDt.Scan0;

    System.Runtime.InteropServices.Marshal.Copy(dgerScan0, pxlBfr, 0,
btKlk);
    dgerBitmap.UnlockBits(dgerdt);

```

```

int fltrZmish = 1;
int calcZmish = 0;

int byteZmish = 0;
int syn = 0;
int zel = 0;
int cherv = 0;

for (int zmishY = fltrZmish; zmishY <
    dgerBitmap.Height - fltrZmish; zmishY++)
{
    for (int zmishX = fltrZmish; zmishX <
        dgerBitmap.Width - fltrZmish; zmishX++)
    {
        syn = 0;
        zel = 0;
        cherv = 0;

        byteZmish = zmishY *
            dgerdt.Stride +
            zmishX * btsPrPxl;

        for (int fltrY = -fltrZmish;
            fltrY <= fltrZmish; fltrY++)
        {
            for (int fltrX = -fltrZmish;
                fltrX <= fltrZmish; fltrX++)
            {
                calcZmish = byteZmish +
                    (fltrX * btsPrPxl) +
                    (fltrY * dgerdt.Stride);

                syn += pxlBfr[calcZmish];
                zel += pxlBfr[calcZmish + 1];
                cherv += pxlBfr[calcZmish + 2];
            }
        }

        syn = syn / 9;
        zel = zel / 9;
        cherv = cherv / 9;

        rsltBfr[byteZmish] = (byte)syn;
    }
}

```

```

        rsltBfr[byteZmish + 1] = (byte)zel;
        rsltBfr[byteZmish + 2] = (byte)cherv;
        rsltBfr[byteZmish + 3] = 255;
    }
}

System.Runtime.InteropServices.Marshal.Copy(rsltBfr,    0,    rsltScn0,
btKlk);
rsltZobr.UnlockBits(rsltDt);

return rsltZobr;
}

static Bitmap ZastGaussianBlurFltr(Bitmap dgerBitmap)
{
    BitmapData dgerdt = dgerBitmap.LockBits(new Rectangle(0, 0,
        dgerBitmap.Width, dgerBitmap.Height),
        ZobrLockMode.ReadOnly, dgerBitmap.PixelFormat);

    Bitmap rsltZobr = new Bitmap(dgerBitmap.Width, dgerBitmap.Height);

    BitmapData rsltDt = rsltZobr.LockBits(new Rectangle(0, 0,
        rsltZobr.Width, rsltZobr.Height),
        ZobrLockMode.WriteOnly,
PixelFormat.Format32bppArgb);

    int btsPrPxl = Bitmap.GetPixelFormatSize(dgerBitmap.PixelFormat) / 8;
    int btKlk = dgerdt.Stride * dgerBitmap.Height;
    byte[] pxlBfr = new byte[btKlk];
    byte[] rsltBfr = new byte[btKlk];

    IntPtr dgerScan0 = dgerdt.Scan0;
    IntPtr rsltScn0 = rsltDt.Scan0;

    System.Runtime.InteropServices.Marshal.Copy(dgerScan0,    pxlBfr,    0,
btKlk);
    dgerBitmap.UnlockBits(dgerdt);

    int fltrZmish = 1;
    int calcZmish = 0;

    int byteZmish = 0;
    double syn = 0.0;

```

```

double zel = 0.0;
double cherv = 0.0;

double[,] gaussianYadr = new double[,]
{
    { 0.0625, 0.125, 0.0625 },
    { 0.125, 0.25, 0.125 },
    { 0.0625, 0.125, 0.0625 }
};

double YadrSum = 1.0;
int YadrSize = 3;

for (int zmishY = fltrZmish; zmishY <
    dgerBitmap.Height - fltrZmish; zmishY++)
{
    for (int zmishX = fltrZmish; zmishX <
        dgerBitmap.Width - fltrZmish; zmishX++)
    {
        syn = 0;
        zel = 0;
        cherv = 0;

        byteZmish = zmishY *
            dgerdt.Stride +
            zmishX * btsPrPxl;

        for (int fltrY = -fltrZmish;
            fltrY <= fltrZmish; fltrY++)
        {
            for (int fltrX = -fltrZmish;
                fltrX <= fltrZmish; fltrX++)
            {
                calcZmish = byteZmish +
                    (fltrX * btsPrPxl) +
                    (fltrY * dgerdt.Stride);

                syn += (pxlBfr[calcZmish]) *
                    gaussianYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

                zel += (pxlBfr[calcZmish + 1]) *
                    gaussianYadr[fltrY + fltrZmish,

```

```

        fltrX + fltrZmish];

        cherv += (pxlBfr[calcZmish + 2]) *
            gaussianYadr[fltrY + fltrZmish,
                fltrX + fltrZmish];
    }
}

rsltBfr[byteZmish] = (byte) (syn / YadrSum);
rsltBfr[byteZmish + 1] = (byte) (zel / YadrSum);
rsltBfr[byteZmish + 2] = (byte) (cherv / YadrSum);
rsltBfr[byteZmish + 3] = 255;
    }
}

System.Runtime.InteropServices.Marshal.Copy(rsltBfr,    0,    rsltScn0,
btKlk);
rsltZobr.UnlockBits(rsltDt);

return rsltZobr;
}

static Bitmap ZastEmbossFltr(Bitmap dgerBitmap)
{
    BitmapData dgerdt = dgerBitmap.LockBits(new Rectangle(0, 0,
        dgerBitmap.Width, dgerBitmap.Height),
        ZobrLockMode.ReadOnly, dgerBitmap.PixelFormat);

    Bitmap rsltZobr = new Bitmap(dgerBitmap.Width, dgerBitmap.Height);

    BitmapData rsltDt = rsltZobr.LockBits(new Rectangle(0, 0,
        rsltZobr.Width, rsltZobr.Height),
        ZobrLockMode.WriteOnly,
PixelFormat.Format32bppArgb);

    int btsPrPxl = Bitmap.GetPixelFormatSize(dgerBitmap.PixelFormat) / 8;
    int btKlk = dgerdt.Stride * dgerBitmap.Height;
    byte[] pxlBfr = new byte[btKlk];
    byte[] rsltBfr = new byte[btKlk];

    IntPtr dgerScan0 = dgerdt.Scan0;
    IntPtr rsltScn0 = rsltDt.Scan0;

```

```

        System.Runtime.InteropServices.Marshal.Copy(dgerScan0,    pxlBfr,    0,
btKlk);
        dgerBitmap.UnlockBits(dgerdt);

        int fltrZmish = 1;
        int calcZmish = 0;

        int byteZmish = 0;
        double syn = 0.0;
        double zel = 0.0;
        double cherv = 0.0;

        double[,] embossYadr = new double[,]
        {
            { -2, -1, 0 },
            { -1,  1, 1 },
            {  0,  1, 2 }
        };

        double YadrSum = 1.0;
        int YadrSize = 3;

        for (int zmishY = fltrZmish; zmishY <
            dgerBitmap.Height - fltrZmish; zmishY++)
        {
            for (int zmishX = fltrZmish; zmishX <
                dgerBitmap.Width - fltrZmish; zmishX++)
            {
                syn = 0;
                zel = 0;
                cherv = 0;

                byteZmish = zmishY *
                    dgerdt.Stride +
                    zmishX * btsPrPxl;

                for (int fltrY = -fltrZmish;
                    fltrY <= fltrZmish; fltrY++)
                {
                    for (int fltrX = -fltrZmish;
                        fltrX <= fltrZmish; fltrX++)
                    {
                        calcZmish = byteZmish +

```

```

        (fltrX * btsPrPxl) +
        (fltrY * dgerdt.Stride);

    syn += (pxlBfr[calcZmish]) *
        embossYadr[fltrY + fltrZmish,
            fltrX + fltrZmish];

    zel += (pxlBfr[calcZmish + 1]) *
        embossYadr[fltrY + fltrZmish,
            fltrX + fltrZmish];

    cherv += (pxlBfr[calcZmish + 2]) *
        embossYadr[fltrY + fltrZmish,
            fltrX + fltrZmish];
}

}

    rsltBfr[byteZmish] = (byte)(syn + 128);
    rsltBfr[byteZmish + 1] = (byte)(zel + 128);
    rsltBfr[byteZmish + 2] = (byte)(cherv + 128);
    rsltBfr[byteZmish + 3] = 255;
}

}

static Bitmap ZastPrewittFltr(Bitmap dgerBitmap)
{
    BitmapData dgerdt = dgerBitmap.LockBits(new Rectangle(0, 0,
        dgerBitmap.Width, dgerBitmap.Height),
        ZobrLockMode.ReadOnly, dgerBitmap.PixelFormat);

    Bitmap rsltZobr = new Bitmap(dgerBitmap.Width, dgerBitmap.Height);

    BitmapData rsltDt = rsltZobr.LockBits(new Rectangle(0, 0,
        rsltZobr.Width, rsltZobr.Height),
        ZobrLockMode.WriteOnly,
PixelFormat.Format32bppArgb);

    int btsPrPxl = Bitmap.GetPixelFormatSize(dgerBitmap.PixelFormat) / 8;
    int btKlk = dgerdt.Stride * dgerBitmap.Height;
    byte[] pxlBfr = new byte[btKlk];
    byte[] rsltBfr = new byte[btKlk];

    IntPtr dgerScan0 = dgerdt.Scan0;

```

```

IntPtr rsltScn0 = rsltDt.Scan0;

System.Runtime.InteropServices.Marshal.Copy(dgerScan0,    pxlBfr,    0,
btKlk);
dgerBitmap.UnlockBits(dgerdt);

int fltrZmish = 1;
int calcZmish = 0;

int byteZmish = 0;
double synX = 0.0;
double zelX = 0.0;
double chervX = 0.0;

double synY = 0.0;
double zelY = 0.0;
double chervY = 0.0;

double[,] xYadr = new double[,]
{
    { -1, 0, 1 },
    { -1, 0, 1 },
    { -1, 0, 1 }
};

double[,] yYadr = new double[,]
{
    { 1, 1, 1 },
    { 0, 0, 0 },
    { -1, -1, -1 }
};

for (int zmishY = fltrZmish; zmishY <
    dgerBitmap.Height - fltrZmish; zmishY++)
{
    for (int zmishX = fltrZmish; zmishX <
        dgerBitmap.Width - fltrZmish; zmishX++)
    {
        synX = zelX = chervX = 0.0;
        synY = zelY = chervY = 0.0;

        byteZmish = zmishY *
            dgerdt.Stride +

```

```

        zmishX * btsPrPxl;

    for (int fltrY = -fltrZmish;
        fltrY <= fltrZmish; fltrY++)
    {
        for (int fltrX = -fltrZmish;
            fltrX <= fltrZmish; fltrX++)
        {
            calcZmish = byteZmish +
                        (fltrX * btsPrPxl) +
                        (fltrY * dgerdt.Stride);

            synX += (pxlBfr[calcZmish]) *
                    xYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

            zelX += (pxlBfr[calcZmish + 1]) *
                    xYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

            chervX += (pxlBfr[calcZmish + 2]) *
                    xYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

            synY += (pxlBfr[calcZmish]) *
                    yYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

            zelY += (pxlBfr[calcZmish + 1]) *
                    yYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];

            chervY += (pxlBfr[calcZmish + 2]) *
                    yYadr[fltrY + fltrZmish,
                        fltrX + fltrZmish];
        }
    }

    int syn = (int)Math.Sqrt((synX * synX) + (synY * synY));
    int zel = (int)Math.Sqrt((zelX * zelX) + (zelY * zelY));
    int cherv = (int)Math.Sqrt((chervX * chervX) + (chervY *
chervY));

```

```

        syn = syn > 255 ? 255 : (syn < 0 ? 0 : syn);
        zel = zel > 255 ? 255 : (zel < 0 ? 0 : zel);
        cherv = cherv > 255 ? 255 : (cherv < 0 ? 0 : cherv);

        rsltBfr[byteZmish] = (byte)syn;
        rsltBfr[byteZmish + 1] = (byte)zel;
        rsltBfr[byteZmish + 2] = (byte)cherv;
        rsltBfr[byteZmish + 3] = 255;
    }
}

System.Runtime.InteropServices.Marshal.Copy(rsltBfr,    0,    rsltScn0,
btKlk);
rsltZobr.UnlockBits(rsltDt);

return rsltZobr;
}
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**Дослідження та програмна реалізація
методів фільтрації зображень**

Виконав:
студент групи КНТ-123м

Сергій ОЛІЙНИК

Керівник:
к. т. н., доцент, доцент кафедри ПЗ

Тетяна КОЛПАКОВА

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для обробки зображень.

Предмет дослідження – програмні засоби для підтримки фільтрації зображень.

Мета роботи – дослідження та розробка програмного забезпечення для фільтрації зображень.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обробки зображень;
- розробити модель взаємодії користувача з програмним забезпеченням фільтрації зображень;
- здійснити проектування програмного забезпечення для фільтрації зображень;
- створити програмне забезпечення для фільтрації зображень;
- виконати тестування розробленого програмного забезпечення для фільтрації зображень.

3

Рисунок В.3 – Слайд 3

Порівняння ПЗ для підтримки процесу обробки зображень

Критерій порівняння	Fotor	Capture One	PhotoLab
Зручний інтерфейс	+–	+	+–
Кросплатформність	+	–	+–
Інтеграція з іншими платформами	–	+	+–
Інструменти для соціальних мереж	+	–	–
Хмарне зберігання даних	+–	+	+

4

Рисунок В.4 – Слайд 4

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Зручність розробки GUI	+	+–	+
Висока продуктивність	+	–	+–
Зручність для створення ПЗ для фільтрації зображень	+	–	–
Засоби обробки зображень	+	+	+–
Підтримка засобів штучного інтелекту та машинного навчання	+	+	+–

5

Рисунок В.5 – Слайд 5

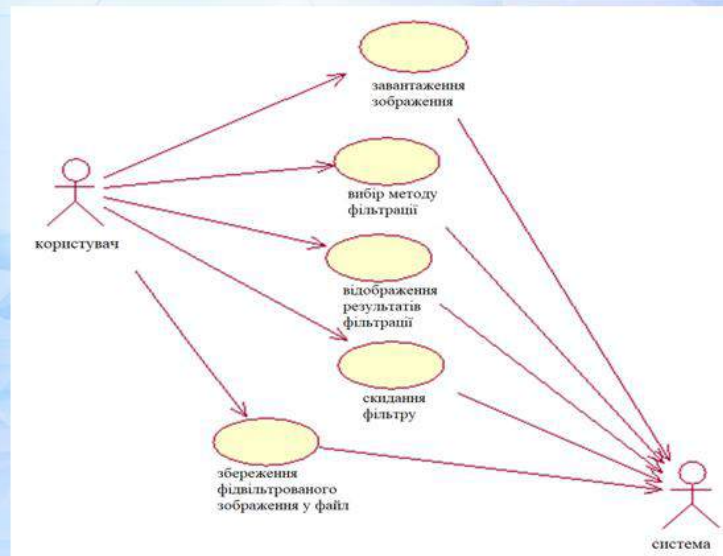
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Microsoft Visual Studio	Xcode	Android Studio
Зручність використання	+	–	–
Розширені можливості для роботи з графікою	+	+–	–
Підсвічування синтаксису коду на C#	+	+–	+–
Автозавершення коду	+	+	+–
Засоби рефакторингу	+	+	+–

6

Рисунок В.6 – Слайд 6

Модель взаємодії користувача з ПЗ фільтрації зображень



7

Рисунок В.7 – Слайд 7

Структура програмного забезпечення для фільтрації зображень



8

Рисунок В.8 – Слайд 8

Схема функціонування ПЗ для фільтрації зображень



9

Рисунок В.9 – Слайд 9

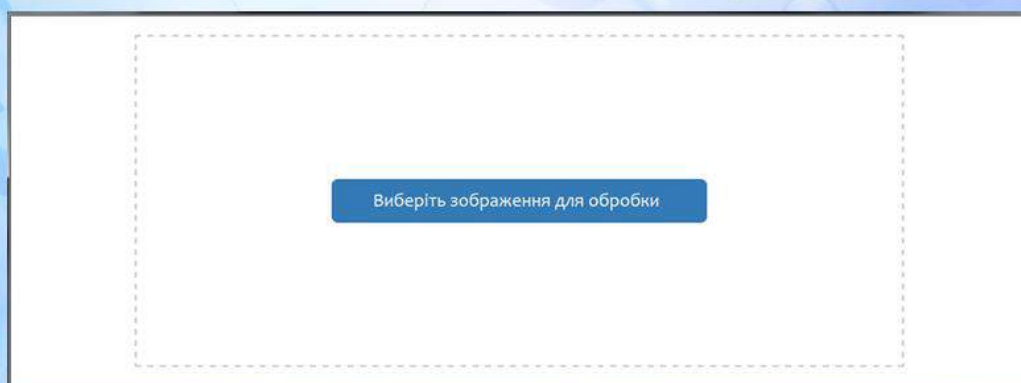
Порівняння методів фільтрації зображень, що використовуються у ПЗ

Критерій порівняння методів фільтрації зображень	Методи фільтрації зображень				
	Медіанний метод	Метод розмиття по рамці	Метод Гаусса	Оператор Прюїтта	Рельєфний метод
Основний принцип	медіана для зменшення шуму	середнє значення в рамці	гаусів розподіл для згладжування	ядра для виявлення країв	амплітудні та просторові зміни
Ефективність зменшення шуму	добре знімає шуми від стаціонарних об'єктів	ефективно зменшує м'які шуми	ефективно згладжує зображення	менше ефективний на відсутність орієнтації	може впливати на зовнішній шум
Час обробки	швидкий, оскільки потребує лише порівняння	швидкий, ідеально підходить для великих об'єктів	може бути повільним для великих вікон	швидкий, бо застосовується до кожного пікселя	може бути повільним через складні алгоритми
Збереження різкості границь	зберігає границі об'єктів, відсутність змазки	згладжує границі, може втратити деталі	згладжує границі, але менше, ніж метод розмиття	відновлює розмежування границь для деталей	зберігає границі
Застосування	видалення шуму з малюнків	згладжування об'єктів	зменшення шуму	виявлення країв	аналіз топографії

10

Рисунок В.10 – Слайд 10

Робота з програмою. Головна форма



11

Рисунок В.11 – Слайд 11

Робота з програмою. Відображення зображення. Результат фільтрації



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для обробки зображень.

Новизна роботи полягає в тому, що створено модель взаємодії користувача з програмним забезпеченням для фільтрації зображень, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для фільтрації зображень.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення для фільтрації зображень.

Виконано тестування розробленого програмного забезпечення для фільтрації зображень. Після проведення тестування встановлено, що програма для фільтрації зображень відповідає технічним вимогам і може успішно використовуватися за призначенням.

13

Рисунок В.13 – Слайд 13