Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматики
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Пінтя Д.В.
(прізвище та ініціали)

Керівник: к.ф.-м.н., доцент кафедри КН
Шевчук О.Ф.
(прізвище та ініціали)
«07» червня 2024 р.

Рецензент: к.т.н., доцент кафедри АІІТ
Барабан М.В.
(прізвище та ініціали)
«07» червня 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«07» червня 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра Комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
"07" 03 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Пінті Дмитру Валентиновичу

1. Тема роботи: Програмний модуль цифрової обробки зображень
керівник роботи: Шевчук Олександр Федорович, к.ф.-м.н., доцент
затверджені наказом вищого навчального закладу "11" 03 2024 року №80

2. Строк подання студентом роботи 27.05.2024

3. Вихідні дані до роботи :
формат зображення – jpg;
тестова вибірка – не менше 4 зображень;
Мова програмування – об'єктно-орієнтована.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних технологій цифрової обробки зображень, розробка алгоритму цифрової обробки зображень, програмна реалізація модуля цифрової обробки зображень, висновки, перелік використаних джерел, додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
структура програмного модуля цифрової обробки зображень; схема загального алгоритму цифрової обробки зображень; загальна UML-діаграма класів; приклади роботи програми

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Шевчук О.Ф., к.ф.-м.н., доцент		

6. Дата видачі завдання 07 березня 2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		При
		початок	закінчення	
1	Обґрунтування доцільності розробки програмного модуля цифрової обробки зображень	06.05.2024	09.05.2024	
2	Розробка програмного модуля цифрової обробки зображень	10.05.2024	14.05.2024	
3	Програмна реалізація модуля цифрової обробки зображень	15.05.2024	21.05.2024	
4	Аналіз функціонування програмного модуля цифрової обробки зображень	22.05.2024	24.05.2024	
5	Оформлення додатків	27.05.2024	29.05.2024	
6	Розробка інструкції користувача	30.05.2024	03.06.2024	
7	Оформлення матеріалів до захисту БДР	04.06.2024	06.06.2024	

Студент _____ (підпис) Пінтя Д (прізвище та

Керівник проекту (роботи) _____ (підпис) Шевчук (прізвище та

АНОТАЦІЯ

УДК 621.374.415

Пінтя Д.В. Програмний модуль цифрової обробки зображень. Бакалаврська дипломна робота складається з 80 сторінок формату А4, на яких є 17 рисунок, список використаних джерел містить 21 найменування.

В даній бакалаврській дипломній роботі досліджено процес розробки програмного модуля цифрової обробки зображень.

В роботі проаналізовані сучасні технології цифрової обробки зображень. Обґрунтовано застосування основних етапів цифрової обробки зображень. Здійснено порівняльну характеристику методів, що застосовуються для обробки зображень. Проаналізовано переваги та недоліки програм аналогів для цифрової обробки зображень. Здійснено постановку задачі. Розроблено загальну структурну схему, схему алгоритму функціонування програмного модуля цифрової обробки зображень та схему алгоритму розбиття зображення на пікселі. Розроблено UML-діаграму класів програми. Здійснено програмну реалізацію модуля цифрової обробки зображень мовою програмування Java в середовищі IntelliJ IDEA.

Ключові слова: обробка зображень, цифрова обробка, гістограма, пікселі, корекція яскравості, Java, IntelliJ IDEA.

ABSTRACT

Pintya D.V. Software module for digital image processing. The bachelor thesis consists of 80 pages of A4 format, on which there are 17 figures, the list of used sources contains 21 names.

In this bachelor's thesis, the process of developing a software module for digital image processing is investigated.

Modern technologies of digital image processing are analyzed in the work. The application of the main stages of digital image processing is substantiated. A comparative characterization of the methods used for image processing was carried out. The advantages and disadvantages of analog programs for digital image processing are analyzed. The task has been set. A general structural diagram, a diagram of the algorithm of the functioning of the software module of digital image processing and a diagram of the algorithm for dividing the image into pixels have been developed. A UML class diagram of the program was developed. The software implementation of the digital image processing module was carried out in the Java programming language in the IntelliJ IDEA environment.

Keywords: image processing, digital processing, histogram, pixels, brightness correction, Java, IntelliJ IDEA.

ЗМІСТ

ВСТУП 7
1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ 9
1.1 Аналіз сучасних технологій цифрової обробки зображень 9
1.2 Аналіз основних етапів цифрової обробки зображень 12
1.3 Порівняльна характеристика методів, що застосовуються для цифрової обробки зображень 15
1.4 Аналіз програм аналогів цифрової обробки зображень 22
1.5 Постановка задачі 28
1.6 Висновок 29
2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ 30
2.1 Основні етапи процесу обробки цифрових зображень 30
2.2 Обґрунтування вибору алгоритмів побудови гістограм зображення 31
2.3 Розробка загальної структурної схеми та алгоритмів функціонування програмного модуля цифрової обробки зображень 35
2.4 Висновки 38
3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ 39
3.1 Обґрунтування вибору мови програмування 39
3.2 Обґрунтування вибору середовища програмування 43
3.3 Обґрунтування вибору бібліотек для Java 45
3.4 Програмна реалізація модуля цифрової обробки зображень 50
3.5 Тестування та аналіз результатів роботи програмного модуля цифрової обробки зображень 51
3.6 Висновки 56
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 58
ДОДАТОК А. Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень 60

ДОДАТОК Б. Інструкція користувача 61
ДОДАТОК В. Лістинг програми..... 64
ДОДАТОК Г. Графічна частина 75

ВСТУП

Актуальність досліджень. Розвиток сучасної науки і техніки пов’язаний з необхідністю збору великої кількості інформації, а її ефективна та своєчасна обробка є однією з науково-технічних проблем. Цифрова обробка зображень складається з обробки зображень за допомогою цифрових комп’ютерів. За останні десятиліття його використання зростає в геометричній прогресії. Його застосування варіюється від медицини до розваг, проходячи геологічну обробку та дистанційне зондування. Мультимедійні системи, одні з опор сучасного інформаційного суспільства, значною мірою покладаються на цифрову обробку зображень.

Обробка зображень - це метод виконання деяких операцій із зображенням, щоб отримати покращене зображення або витягти з нього якусь корисну інформацію. Це тип обробки сигналу, в якому вхідним є зображення, а вихідним може бути зображення або характеристики / особливості, пов’язані з цим зображенням. У наш час обробка зображень належить до швидкозростаючих технологій. Це також формує основну наукову область в інженерних та інформаційних дисциплінах.

Обробка зображень в основному включає такі три етапи: імпорт зображення за допомогою інструментів отримання зображень; аналіз та маніпулювання зображеннями; вихідні дані, в яких результат може бути змінено зображення або звіт, який базується на аналізі зображення.

Для обробки зображень використовуються два типи методів, а саме: аналогова та цифрова обробка зображень. Аналогову обробку зображень можна використовувати для друкованих копій, таких як роздруківки та фотографії. Аналітики зображень використовують різні основи інтерпретації, використовуючи ці візуальні прийоми. Методи цифрової обробки зображень допомагають маніпулювати цифровими зображеннями за допомогою комп’ютерів. Три загальні фази, які повинні пройти всі типи даних, використовуючи цифрову техніку, - це попередня обробка, вдосконалення та відображення, вилучення інформації.

Оскільки зображення визначаються у двох вимірах (можливо, більше), цифрова обробка зображень може моделюватися у вигляді багатовимірних систем. На генерацію та розвиток цифрової обробки зображень в основному впливають три фактори: по-перше, розвиток комп’ютерів; по-друге, розвиток математики (особливо

створення та вдосконалення дискретної теорії математики); по-третє, збільшився попит на широкий спектр застосувань у навколишньому середовищі, сільському господарстві, військовій, промисловій та медичній науці.

Наразі існують лише громіздкі системи для обробки зображеннь та створення гістограм, які мають певне призначення та спеціальні функції для подальшої роботи із зображенням. Багато з цих програм мають проблеми зі швидкістю або не мають безкоштовної версії.

Об'єкт дослідження – це процес цифрової обробки зображень.

Предмет дослідження – програмні засоби цифрової обробки зображень.

Мета дослідження –підвищення швидкодії програмного забезпечення цифрової обробки зображень.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати сучасні технологій цифрової обробки зображень;
- проаналізувати переваги та недоліки програм аналогів цифрової обробки зображень;
- розробити алгоритм цифрової обробки зображень;
- здійснити програмну реалізацію модуля цифрової обробки зображень;
- провести тестування нової розробки та проаналізувати отримані результати.

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ

1.1 Аналіз сучасних технологій цифрової обробки зображень

Цифрове зображення - це дискретний простір, що складається з дрібних елементів поверхні, які називаються пікселями. Кожен з цих елементів містить значення або набір значень, що кодують рівень інтенсивності в кожній позиції. Цифрове зображення можна отримати за допомогою великої кількості різних пристроїв, таких як камера, МРТ-апарат або будь-який пристрій з датчиком, здатним фіксувати інтенсивність світла.

Через свою дискретну природу теорія, що використовується для обробки цифрового зображення, буде покладатися на дискретну область, навіть якщо аналогія з неперервною областю можлива.

Масштаб сірого - це цифрове зображення, в якому кожен піксель містить лише одне скалярне значення, яке є його інтенсивністю. Кількість можливих рівнів (значень інтенсивності) залежить від числового типу, що кодує зображення.

Наприклад, зображення, закодоване з n = 8 біт, матиме лише $L = 2^8 = 256$ можливих значень інтенсивності, що переходять від 0, що представляє чорний до L-1 = 255, що представляє білий.

Кольорове зображення - це цифровий масив пікселів, що містить інформацію про колір. Кожне зображення можна розкласти на три різні шари відповідно до трьох кодованих каналів: червоного, зеленого та синього.

Наприклад, 8-бітові кольорові зображення кодують червоний і зелений канал трьома бітами, а синій - двома. Який може кодувати 256 різних кольорів.

Область використання технологій цифрової обробки значно розширюється і замінює аналогові методи обробки сигналів зображення. Цифрові методи обробки широко використовуються у промисловій, мистецькій, медицинській сферах та космічних подорожах. Їх використовують в управлінні процесами, в автоматизації розпізнавання об'єктів, в розпізнаванні образів та в багатьох інших сферах. Цифрова

передача зображень з космічних кораблів, цифрові канали передачі зображень вимагають передачі все більших потоків інформації. Візуалізація, покращення якості та автоматизація обробки медичних зображень, включно з зображеннями, створеними електронними мікроскопами, рентгенівськими апаратами, томографами тощо, є предметом сучасних досліджень та розробок. Автоматичний аналіз у системах дистанційного зондування широко використовується в польових аналізах та лісовому господарстві, наприклад, для автоматичного обчислення площі, що обробляється, у сільському господарстві для моніторингу стиглості врожаю, в освіті та в системах протипожежного захисту. Контроль якості виробленої продукції базується на автоматичних методах поетапного аналізу.

У наш час важко уявити сферу діяльності, в якій можна давати раду без обробки комп'ютерних зображень. Комп’ютерна обробка зображень вирішує різні завдання, наприкдлад, покращення якості зображення; обчислення параметрів зображення; спектральний аналіз багатовимірних сигналів; розпізнавання зображень; стиснення зображення.

Першим кроком цифрової обробки зображень є трансформування оптичного зображення у форму, яка може зберігатися в пам'яті машини. Це трансформування здійснюється світлочутливою системою, яка називається електронно-оптичний цифровий перетворювач, що створює закодовані числа, які є показниками інтенсивності світла. Програма реєструє оптичне зображення через рівні проміжки часу, а отриманий електричний струм, який перетворюється в послідовність неперервних числових значень, зображає розподіл інтенсивності в оптичному зображенні. Даний процес називається оцифруванням або квантуванням, а збережене числове подання вихідного оптичного зображення - цифровим. Цифрове зображення - це числова абстракція, якою, як і будь-які інші дані, що зберігаються в пам'яті апарата, можна керувати. Існує велика кількість операцій, які можна виконати над цифровим зображенням. Однак усі ці перетворення належать до однієї з двох загальних категорій: аналіз зображень або обробка зображень.

Методи обробки зображень та аналізу зображень можуть сильно відрізнятися за своєю складністю і, отже, за вимогами до пристрою. Найпростіші методи аналізу

зображень використовуються для підвищення продуктивності та точності вирішення проблем, які інакше можна було б зробити вручну, наприклад: Б. підрахунок кількості колоній бактерій у чашках Петрі для значного збільшення. Типова система цього вигляду може мати у складі тільки оптичний проектор, цифровий графічний планшет та міні-комп'ютера, або ж відеокамери, відеомонітора та міні-комп'ютера. Така система не вимагає особливих витрат і може бути негайно використана кваліфікованим персоналом [2].

З метою обробки графічної інформації застосовують різні програмні засоби, пакети комп'ютерного дизайну та програмні функції. При комп’ютерній обробці зображення розглядається як двовимірний сигнал, в якому кожна інформаційна одиниця обробляється незалежно від інших і визначається лише алгоритмом [3].

У цій роботі використана модель RGB, так як дана модель найкраще відповідає структурі людського ока. Основу цієї моделі складають три кольори: червоний - червоний, зелений - зелений і синій - синій. У моделі RGB кольори утворюються додаванням трьох компонентів, що визначають відносну яскравість червоного, зеленого та синього кольорів. Ці компоненти називаються первинними. Комп'ютерна модель RGB представляє кожен основний колір у 256 градаціях яскравості, що відповідає 8-бітовому режиму [4].

Наведемо основні переваги моделі RGB є:

- його відношення до пристрою (сканера та монітора);

- широке кольорове покриття (можливість відображати найрізноманітніші кольори, близькі до людського зору);

- можливість реалізації багатьох методів обробки зображень (фільтрів) у програмах растрової графіки;

- невеликий обсяг (у порівнянні з моделлю CMYK), який займає зображення в оперативній пам’яті та на жорсткому диску комп’ютера.

Недоліками моделі RGB є:

- співвідношення кольорових каналів (із збільшенням яскравості одного каналу, зменшенням інших);

- можливість помилки у відображенні кольорів на екрані монітора щодо

кольорів, отриманих при переході на модель CMYK;

- існують кольори, створені в цьому режимі, що не можуть відображатися під час друку [5].

Основною проблемою обробки зображень є проблема збільшення роздільної здатності окремих фрагментів. Причинами погіршення картини є:

- порушення технічного шуму;
- мало або забагато освітлення предметів;
- відсутність чіткості при отриманні зображення;
- розмір частин, які слід розрізнити, недостатній.

Програмні рішення для обробки зображень зазвичай складається із спеціальних модулів, які здійснюють певні процеси. Розроблені програмні пакети також мають інструменти, що дозволяють користувачеві самостійно розробляти програми, що запускають принаймні спеціальні системні модулі.

1.2 Аналіз основних етапів цифрової обробки зображень

Методи цифрової обробки зображень можна поділити на дві великі категорії: методи, при яких зображення є на вході та виході, та методи, в яких вхідні зображення та вихід мають функції та атрибути, вибрані на основі цих зображень (рис. 1.1).

Зображена схема не має на увазі застосування кожного процесу до зображення, а лише наводить принципи всіх методів, що можуть бути застосовані з метою обробки зображень для отримання різних результатів.

У більшості автоматизованих та автоматизованих систем обробки зображень достатньо мати інформацію про його контур, щоб розпізнати та класифікувати об’єкт. Ця інформація дозволяє легко розрахувати площу, центр ваги та коефіцієнт форми об’єкта або енергетичний спектр текстури. Вибір та аналіз обрисів об’єктів дозволяє уникнути надмірної відеоінформації.

Рисунок 1.1 – Основні стадії цифрової обробки зображень

Однак при отриманні та передачі сигналів зображення виглядають як мультиплікативні перешкоди, пов'язані з власним шумом датчика, проходженням світлових променів та шумом каналу зв'язку [6]. Межі об'єкта можуть бути розмитими. Дія цих факторів створює прогалини і дефекти контурів. Існуючі алгоритми присвоєння контурів не працюють в умовах шуму, і має сенс використовувати двійкове зображення для розпізнавання. Використання бінаризаційних та контурних процесів підвищує імунітет і швидкість, а також спрощує структуру пристроїв виявлення.

З метою обробки цифрового зображення слід використовувати гістограми зображення. Що таке гістограма? Це схема розподілу пікселів різної яскравості на зображенні.

Горизонтальна вісь графіка показує яскравість відтінків на зображенні. Від найтемнішого (чорного) до найсвітлішого (білого). На цій осі є стовпці, в яких відображається кількість пікселів певної яскравості. Чим вище смуга, тим більше

пікселів цієї яскравості на фотографії. Наприклад, у наведеному прикладі графіком на малюнку є середнє значення відтінків легкості - півтонів. Тоді як темних і світлих тонів небагато.

Оскільки відтінки розташовані по-різному в різних кадрах, кожне зображення має свою унікальну схему гістограми. Проте все ж існують деякі закономірності.

Щоб краще зрозуміти, як ця гістограма працює на практиці, давайте спочатку навчимося оцінювати яскравість і контрастність фотографії на ній.

Гістограма - це схема розподілу яскравості пікселів зображення. Камера може відображати як загальну гістограму складеного каналу RGB, так і гістограми окремих каналів (червоний, зелений та синій). Горизонтальна вісь - це значення яскравості пікселів від чорного до білого через проміжні градації (для кольорових каналів від чорного до найбільш насиченого кольору). Вертикальна вісь - це кількість пікселів, які відповідають заданій яскравості, виражена у відносних одиницях [7].

Бувають ситуації, коли гістограма значно полегшує створення високоякісних зображень:

- Зйомка вночі або при яскравому освітленні: Важко визначити яскравість і контрастність зображень як без додаткових джерел світла, так і з освітленим екраном.
- Студійні кадри: Коли ви працюєте без лічильника світла, вам потрібно зосередитись на результатах на дисплеї. Графік точніше відображає ситуацію з експозицією.
- Запис об’єкта: Якщо ви записуєте об’єкти на білому фоні, ви можете чітко судити за гістограмою, наскільки білий колір на дисплеї насправді білий.

Гістограма є дуже важливим інструментом обробки зображень. Це графічне зображення розподілу даних. Гістограма зображення дає графічне представлення розподілу інтенсивності пікселів у цифровому зображенні.

Вісь x вказує діапазон значень, які може приймати змінна. Цей діапазон можна розділити на ряд інтервалів, які називаються контейнерами. Вісь y показує кількість, скільки значень потрапляє в цей інтервал або бін [8].

При побудові гістограми ми маємо інтенсивність пікселів по осі X і частоту по

осі Y. Як і будь-яка інша гістограма, ми можемо вирішити, скільки бункерів використовувати.

Гістограму можна розрахувати як для шкали сірого, так і для кольорового зображення. У першому випадку ми маємо єдиний канал, отже, єдину гістограму. У другому випадку ми маємо три канали (червоний, зелений та синій), отже, три гістограми.

Розрахунок гістограми зображення дуже корисний, оскільки дає інтуїцію щодо деяких властивостей зображення, таких як тональний діапазон, контрастність та яскравість.

1.3 Порівняльна характеристика методів, що застосовуються для цифрової обробки зображень

Здійснення комп'ютерної обробки зображень можливе після трансформування сигналу зображення з аналогової у цифрову форму. Забезпечення ефективності обробки залежить від адекватності моделі, що описує зображення, необхідної для розробки алгоритмів обробки. Моделлю зображення називають функціональну систему, що описує основні характеристики зображення: функцію яскравості, що зображає зміну яскравості в площині зображення, просторові спектри та спектральну інтенсивність зображень, функцію автокореляції. Канал зображення містить оптичну систему, оптико-електричний перетворювач, аналого-цифровий перетворювач (АЦП) та цифрову обробку сигналу зображення [9].

Розглянемо детальніше основні методи цифрової обробки зображень:

1. Попередня обробка - майже завжди використовується після видалення інформації з відеодатчика і має на меті зменшити спотворення зображення, спричинене дискретизацією та квантуванням, а також придушити зовнішні шуми. Зазвичай це операції усереднення та вирівнювання над гістограмами.

2. Сегментація - процес пошуку однорідних ділянок на зображенні. Ця фаза дуже складна і зазвичай не алгоритмізована для будь-яких зображень.

Найпоширеніші методи сегментації засновані на визначенні однорідних кольорів або текстур.

3. Покращення / фільтрація - може використовуватися після сегментації та має ту ж мету, що і попередня обробка: зменшення шуму зображення, спричиненого дискретизацією та квантуванням, а також придушення зовнішніх шумів.

4. Розпізнавання часто є останнім етапом обробки, який лежить в основі процесів інтерпретації та розуміння. Вхід для розпізнавання вибрано та частково відновлено в результаті сегментації зображення. Вони відрізняються від еталонних зображень тим, що мають геометричне та яскраве спотворення, а також шум, який вони отримують.

Кінцевий результат аналізу зображень багато в чому визначається якістю сегментації, а рівень деталізації основних особливостей залежить від поставленого завдання. Як результат, не існує єдиного методу чи алгоритму, які б підходили для вирішення всіх типів задач сегментації. Кожна процедура має свої переваги та недоліки. У більшості випадків один або кілька алгоритмів вибираються та модифікуються відповідно до конкретних умов задачі [10].

Сегментація вирішує в загальному розумінні дві основні проблеми:

• ділення об’єкта на частини для подальшого аналізу;

• зміна форми опису елементів зображення, що дозволяє представити точки як структури високого рівня, що забезпечують ефективність подальшої обробки зображення.

Існує кілька класифікацій методів, але більшість із них базуються на наступних двох властивостях сигналу яскравості - розривності та однорідності.

Методи, засновані на неперервності яскравості, включають виявлення лінійних точок та різниць. При пошуку точок і ліній за допомогою спеціальних масок організовується відповідний пошук. Похідні та градієнти функцій яскравості використовуються як методи розпізнавання відмінностей. Ці методи базуються на більш загальних уявленнях [11]. В даний час до основних методів сегментації зображень належать такі класи (методи групуються від найпростіших до найбільш трудомістких) [12]:

1. Морфологічні методи - в основному застосовуються для роботи з бінарними (чорно-білими) зображеннями. Ці методи дають вам компоненти зображення, які ви можете використовувати для представлення форми об’єкта.

2. Граничні методи - мають інтуїтивні властивості та прості у реалізації. Існує декілька основних типів сегментування порогових значень, але лише два основних типи: оптимальний метод порогового значення та метод адаптивного порогового значення. Усі інші методи цього класу походять із двох згаданих алгоритмів.

3. Методи створення області - це алгоритми, які рекурсивно виконують процес групування пікселів в області за заздалегідь визначеними критеріями. Одним з основних методів тут є метод ін’єкції.

4. Текстурні методи - засновані на аналізі дифузних (кольорових, відбивних) властивостей поверхні об'єкта. Методи, представлені в цій категорії, являють собою набори складних операторів, які можуть звести процес виявлення поверхні до простого завдання розрізнення рівнів яскравості.

Цифрові зображення можуть бути чутливими до різних типів шумів, які можуть виникати внаслідок методів захоплення зображень, технологій передачі інформації та методів оцифрування даних. Усунення різних типів шуму на зображеннях називається фільтрацією.

Фільтрація замінює властивості яскравості кожної точки зображення в цифровому форматі різним значенням яскравості, яке визнається найменш спотвореною перешкодою. Існує частотна та просторова фільтрація.

Частотні методи перетворень зображень засновані на ідеї перетворення Фур'є, зміст якої полягає у поданні вихідної функції як суми тригонометричних функцій різних частот, помножених на задані коефіцієнти. Якщо функція періодична, таке подання називається рядом Фур'є. В іншому випадку неперіодична функція з кінцевою площею під діаграмою може бути виражена як інтеграл тригонометричних функцій, помножений на вагову функцію. Цей варіант називається перетворенням Фур'є і в більшості практичних задач корисніший за ряд Фур'є. Важливою властивістю є те, що функція, представлена перетворенням Фур'є, може бути повернута до початкового стану після перетворень. Це дозволяє вам обробити

функцію в частотній області, а потім повернутися у початковий стан, не втрачаючи жодної інформації. Перетворення Фур'є також можуть бути використані для вирішення проблем фільтрації зображень.

У практичному застосуванні реалізація частотних підходів може бути подібною до методів просторової фільтрації [13].

Цифрові зображення можуть бути чутливими до різних типів шумів, які можуть виникати внаслідок методу отримання зображень, технологій передачі інформації та методів оцифрування даних.

Методи просторового покращення зображення застосовуються до цифрових зображень, які представлені у вигляді двовимірних матриць. Принцип просторових алгоритмів полягає у застосуванні спеціальних операторів до кожної точки вихідного зображення. Оператори - це прямокутні або квадратні матриці, які називаються масками, ядрами або вікнами. Найчастіше маска являє собою невеликий двовимірний масив, і методи вдосконалення, засновані на цьому підході, часто називають обробкою маски або фільтруванням маски. Просторова фільтрація є найкращим рішенням у випадках, коли присутня лише адитивна складова шуму.

Використовуються різні фільтри:

1. Фільтр, заснований на обчисленні середнього арифметичного
2. Фільтри на основі розрахунку середнього геометричного
3. Фільтри на основі розрахунку середньої гармоніки
4. Фільтри на основі статистики замовлень
5. Середній фільтр.
6. Фільтри на основі розрахунку максимуму та мінімуму
7. Фільтри на основі вибору центральної точки

Слід зазначити, що медіанний фільтр часто є більш ефективним, ніж звичайне усереднення, при вирішенні проблем придушення шуму, оскільки це призводить до меншого спотворення меж вибраних об'єктів, саме тому цей фільтр був використаний у цій роботі [14].

Моделі імовірнісних образів часто застосовують для опису зображень. У цьому випадку зображення аналізується у вигляді випадкової функції просторових

координат (*x*, *y*) і часу *t*. Випадковий процес називають стаціонарним у ширшому розумінні, якщо він має постійні значення математичного сподівання та дисперсії, в той час коли функція автокореляції залежить не від координат, а від їх різниці. Випадковий процес називають нерухомим, якщо його n-мірна щільність ймовірності переміщення є постійною. Випадковий процес описується розподілом щільності ймовірності яскравості на зображенні за просторовими координатами протягом фіксованого часу t.

Після визначення математичного сподівання (середнє, математичне очікування), стаціонарний процес обчислюється в ширшому розумінні

$$Mf = \xi = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)p(x,y)dxdy = const. \quad (1.1)$$

Аналіз спектра зображення широко використовується при обробці зображень. Спектр зображення можна отримати, використовуючи перетворення Фур'є.

$$F(u,v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I_1(x,y)e^{-j(ux+vy)}dxdy, \quad (1.2)$$

Функція $e^{-j(ux+vy)}dxdy$ при сталому значенні просторових частот зображає плоску хвилю в площині зображення (рис. 1.2). За використання такого аналізу є можливість визначити також амплітуду і фазу спектрів. Обернене перетворення Фур’є допомагає відновити зображення за його спектром, а інтенсивність зображення відповідає розподілу енергії по просторових частотах.

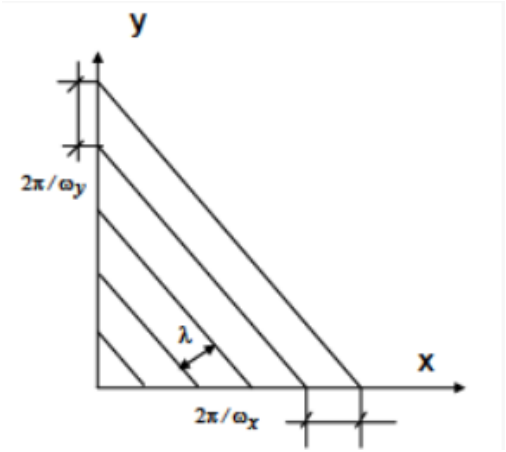

Рисунок 1.2 – Визначення просторових частот зображення

Безперервне зображення (I (x, y)) – коли координати x і y можуть приймати будь-які значення, а дискретним – коли x і y визначені лише для певного набору значень. Як правило, зразок зображення знаходиться не в часовій області, а в просторі, і частота дискретизації вказує, скільки зразків зображення вимірюється на одиницю довжини в кожній координаті. Величина частоти дискретизації в цьому випадку становить 1 / M, а крок дискретизації виражається в одиницях довжини.

$$\delta[k,l] = \begin{cases} 1, k = 0, l = 0, \\ 0, k \neq 0, l \neq 0. \end{cases} \tag{1.2}$$

$$I_1[n,m] = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} I_1(x,y)\delta(k-n, l-m), k, l \in Z, \tag{1.3}$$

Реальні програми дискретного образу містять кінцеву кількість елементів, які звуться пікселями:

$$I_1[n,m], n = \overline{0, N-1}, m = \overline{0, M-1}\,. \tag{1.4}$$

Якщо для отримання яскравості пікселя обробленого зображення використовується яскравість лише одного пікселя вихідного зображення, це перетворення градації.

Зв'язок вхідних та вихідних зображень для корекції яскравості пікселів для різних значень гами (> 1 та <1) наведено на рисунку 1.3.

Гістограмою розподілу яскравості цифрового зображення називають дискретну функцію, що описує частоту рівня сірого на зображенні і представляється у вигляді графіки, абсциса якої містить кількість рівнів сірого у порядку зростання (інтенсивності) та містить вісь ординат – кількість сірих пікселів (частота інтенсивності) [15]. Гістограма має у складі n стовпців для напівтонового зображення n = 256.

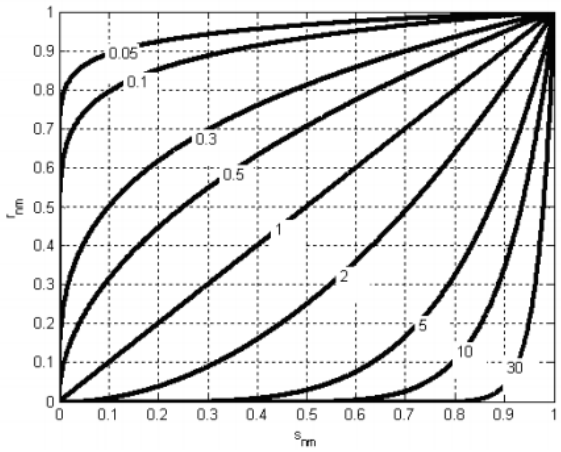

Рисунок 1.3 – Гамма-корекція зображення

З точки зору зорового сприйняття людини, оптимальними є зображення, елементи яких мають рівномірний розподіл світлих кольорів. Покращена ультразвукова візуалізація дозволяє досягти більш високого вирівнювання, яке забезпечує рівномірну здатність створювати яскраве зображення.

Для середньої фільтрації використовують сусідство відповідного пікселя вихідного зображення для отримання яскравості пікселя вихідного зображення. Порядок їх від найменшого до найбільшого визначається значеннями яскравості оточуючих пікселів. Для цієї послідовності існує медіана, тобто визначається, який піксель знаходиться в точці послідовності, що відповідає половині його довжини. Наприклад, якщо є близько 9 пікселів, медіаною є піксель, який є п'ятим у рейтингу. Яскравість цього пікселя - це значення яскравості пікселів відфільтрованого зображення [16].

1.4 Аналіз програм аналогів цифрової обробки зображень

Програми аналізу зображень потрібні для обробки зображень, щоб отримати деякі кількісні властивості. Окрім металургії, обробка зображень використовується в багатьох сферах: у медицині (можливо, навіть більше, ніж у металургії), в біології та в геології.

Існує багато потужних спеціальних платних продуктів, які можна використовувати для вимірювання об’єктів на зображенні, навіть в автоматичному режимі. Існуючі програми дещо відрізняються, розглянемо деякі з них:

IC Measure - програмне забезпечення, що показує тональний розподіл кольорів як зображень, так і відео. Крім того, воно також підтримує зображення різних форматів (рис. 1.4) [17].

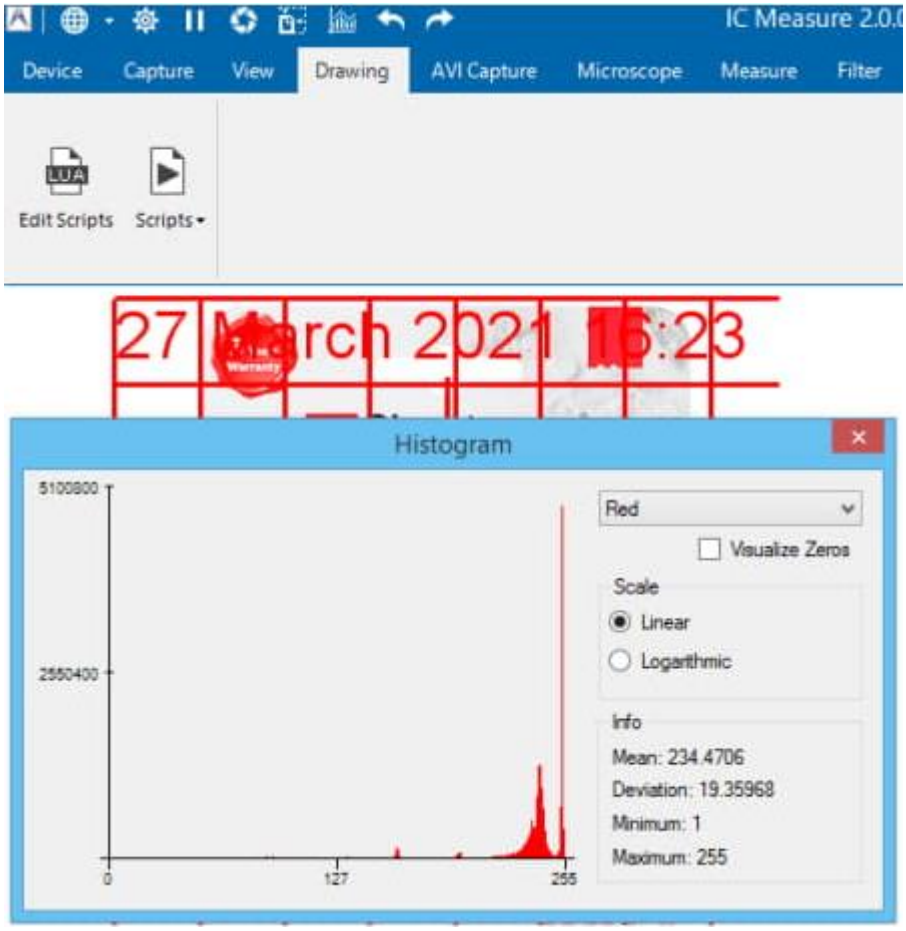

Рисунок 1.4 – Приклад роботи програми IC Measure

IC Measure - це безкоштовне програмне забезпечення для гістограм зображень для Windows. В основному це програмне забезпечення для вимірювання зображень, за допомогою якого користувачі можуть вимірювати відстань та форми, наявні на зображеннях. Крім цього, його також можна використовувати для аналізу різних аспектів відео- та аудіофайлів. Тепер для перегляду та аналізу тонального розподілу кольорів, присутніх на зображенні, користувачі можуть використовувати вбудовану гістограму. На відміну від більшості інших програм для гістограм зображень, це програмне забезпечення також дозволяє користувачам переглядати тональний розподіл кольорів відео в реальному часі на своїй гістограмі. Ще однією хорошою особливістю цього програмного забезпечення є його здатність підтримувати зображення різних форматів, таких як BMP, JPEG, JFIF, PNG, TIFF тощо.

Основні переваги:

- користувачі можуть відкрити гістограму, перейшовши на вкладку Вид. Використовуючи цю гістограму, користувачі можуть переглядати значення розподілу кольорів RGB, присутні на зображенні, у вигляді гістограми. За замовчуванням ця гістограма використовує лінійну шкалу для відображення розподілу кольорів. Хоча користувачі можуть вибрати логарифмічну шкалу для перегляду розподілу кольорів RGB зображення.
- інформація гістограми - показує середнє значення та відхилення значень тонального розподілу кольору;
- підтримка послідовності відео та зображень: У цьому програмному забезпеченні користувачі можуть також вводити послідовність зображень та відеофайлів для перегляду та аналізу значень розподілу кольорів.

LightBox Image Editor - це мила маленька утиліта з вишуканим інтерфейсом. Програма використовує розділений екран попереднього перегляду для показу зображень до та після налаштування. Особливості: Видалення інструменту Color Cast Tool за допомогою алгоритму сірої точки, який видаляє небажані тони обличчя. Технологія розумного контрасту; Smart Fill Light – для застосовування тіні, зберігаючи реальність зображення обличчя; Видалення явища червоних очей, що використовується разом із технологією розпізнавання обличчя тощо (рис. 1.5).

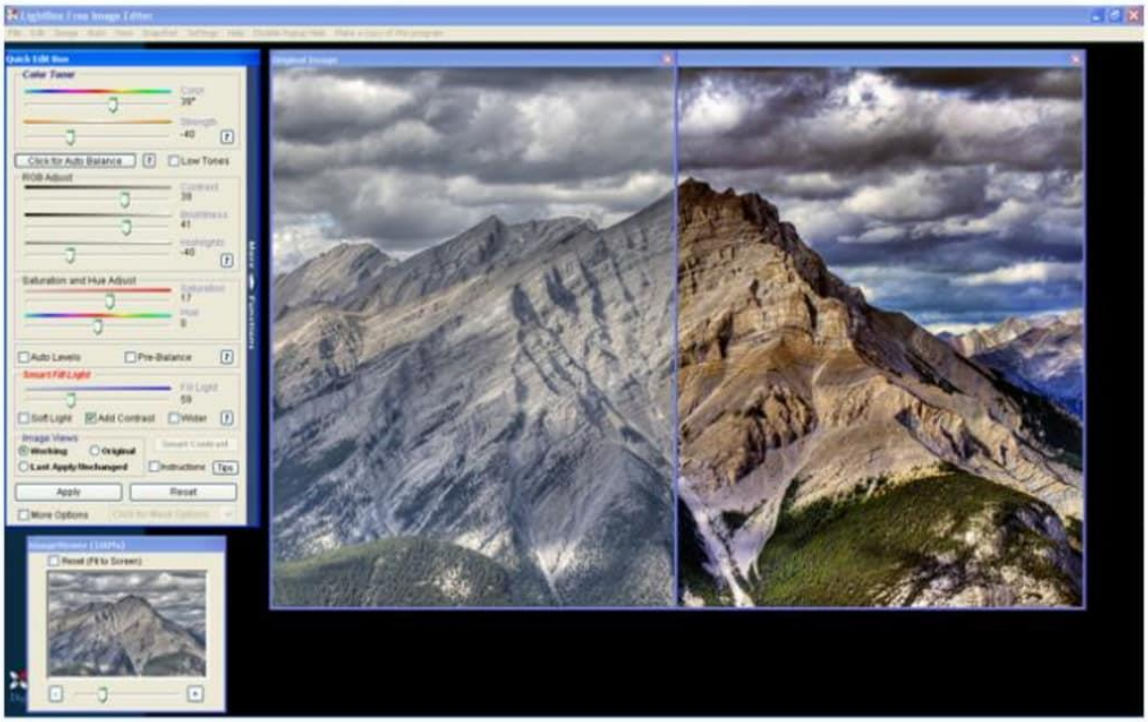

Рисунок 1.5 – Приклад роботи програми LightBox Image Editor

Jmicrovision (рис.1.6) – програма аналізу зображень, що має наступні можливості [19]:

- дозволяє провести лінійні вимірювання і довжин довільних кривих;

- забезпечує вимірювання площ довільних фігур, та інших геометричних характеристик (овальність, периметр і т.д.);

- виділяє елементи структури за кольором.

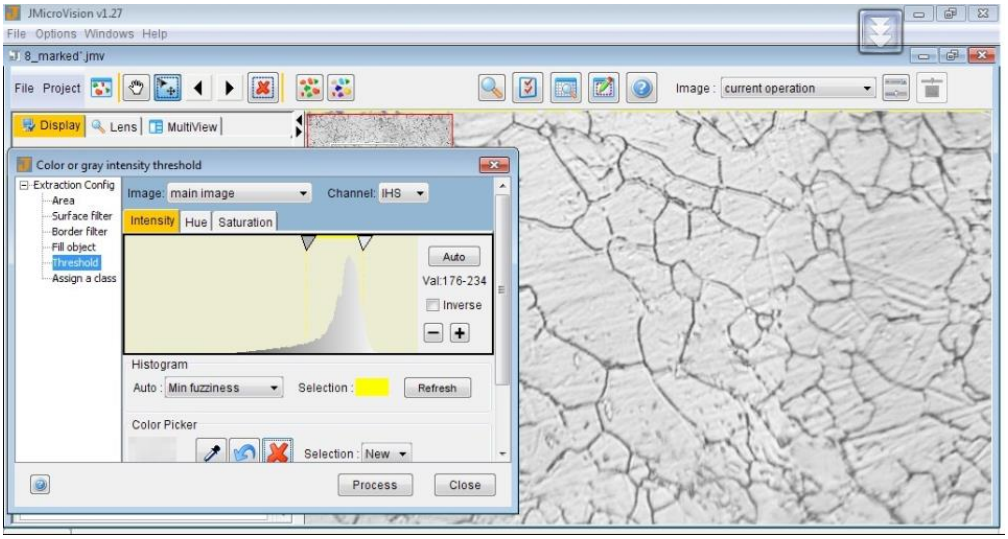

Рисунок 1.6 – Приклад роботи програми Jmicrovision

На сьогоднішній день ChaosPro є одним з найкращих фрактальних генераторів зображень з широким спектром функцій, що постійно оновлюється, а також є безкоштовним [20]. Його використання забезпечує легке створення безлічі дивовижних фрактальних образів (рис. 1.7). Програма має дуже простий та зручний інтерфейс.

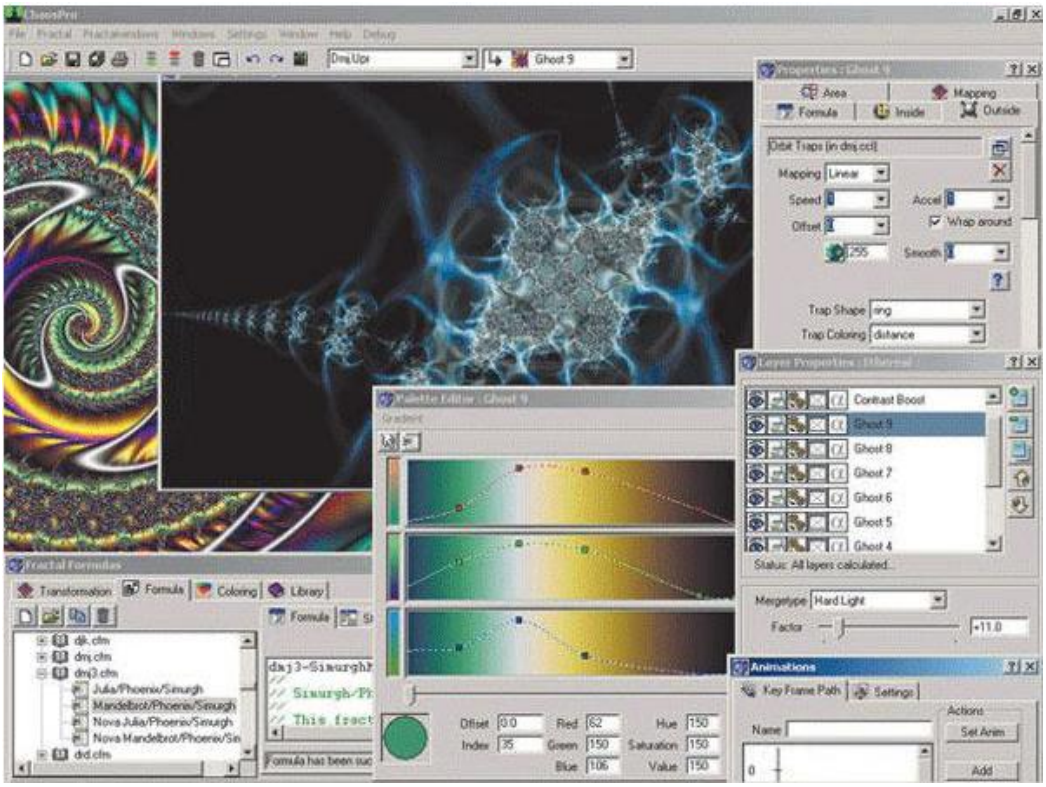

Рисунок 1.7 – Приклад роботи програми ChaosPro

У списку функцій програми:

- точне узгодження кольорів, що забезпечує плавний перехід градієнта один в одного, що дозволяє створювати плавні та красиві фрактальні зображення;
- одночасне побудова декількох фракталів, відкритих у різних вікнах;
- існує можливість створити анімацію на базі фрактальних зображень з визначенням ключових фаз анімації, що можуть відрізнятися за всіма змінними параметрами: кутами повороту та повороту, колірними параметрами тощо;
- створення тривимірних зображень фракталів на основі загальних двовимірних зображень.
- забезпечення повної або майже повної підтримки стандартних міжнародних форматів фрактальних зображень *.frm, *.map, *.ifs та *.l;

- існує можливість розробляти всі типи фракталів завдяки вбудованому компілятору.

KStars (рис. 1.8) – програмний інструмент, головним призначенням якого є перегляд та вивчення даних про небесні об'єкти та явища. Це потужний планетарій, який дозволяє користувачам спостерігати за зоряним небом у будь-який момент часу і з будь-якого місця на Землі. Програма надає детальну інформацію про зірки, планети, комети, астероїди та інші небесні тіла, а також дозволяє відслідковувати їх рух у реальному часі.

KStars є особливо корисним для астрономів-любителів та студентів, які вивчають астрономію, оскільки надає зручний інтерфейс для доступу до астрономічних даних і можливостей для проведення віртуальних спостережень. За допомогою KStars можна планувати реальні астрономічні спостереження, отримуючи точні координати та інформацію про видимість небесних об'єктів.

Серед функцій KStars можна виділити можливість створення астрономічних подій, таких як затемнення або транзити, моделювання руху Сонячної системи, а також інтеграцію з телескопами для автоматизованого наведення на об'єкти. Програма підтримує велике число каталогів зірок і інших небесних тіл, що робить її універсальним інструментом для дослідження космосу.

Таким чином, KStars є незамінним засобом для всіх, хто цікавиться астрономією, забезпечуючи глибоке розуміння зоряного неба та сприяючи розвитку наукових знань у цій галузі.

Особливостями програми слід назвати [2]:

- Підтримку 8, 16, 32, IEEE 32- та IEEE 64-бітних форматів.
- Підтримку різнокольорових зображень FITS та мозаїчних зображень FITS.
- Ведення статистики.
- Наявність прямокутних та екваторіальних сіток.

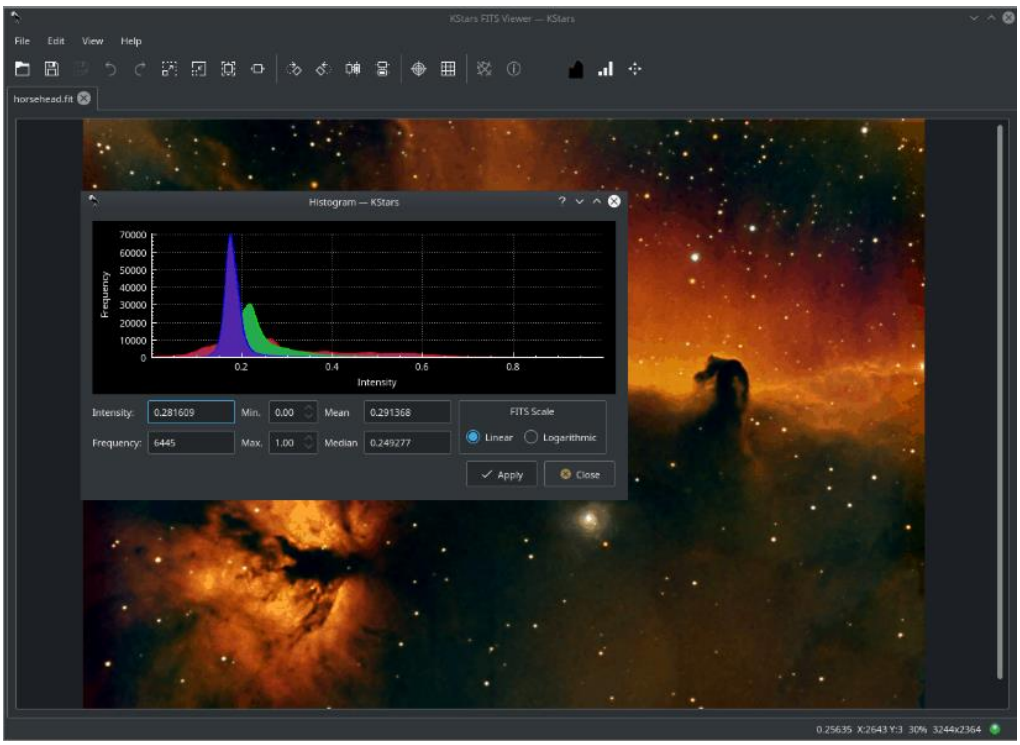

Рисунок 1.8 – Приклад роботи програми Kstars

Artweaver (рис. 1.9) - програма, схожа за функціональністю та користувальницьким інтерфейсом до Photoshop [4]. Він має багатомовний інтерфейс, підтримує плагіни та має всі атрибути програми середнього рівня. Особливості: Підтримує роботу з шарами та прозорість (але тільки для «рідного» формату програми - AWD); містить різноманітні налаштування для малювання (вугілля, олівець, губка, олія, акрилове волокно) та фільтри (розмиття, ефект вітру, хвилі, ефект масла, грануляція тощо).

Основними особливостями Artweaver є:

- «Піпетка», щоб взяти зразок кольору із зображення: бере колір пікселя або в середньому 3 х 3 та 5 х 5 пікселів.

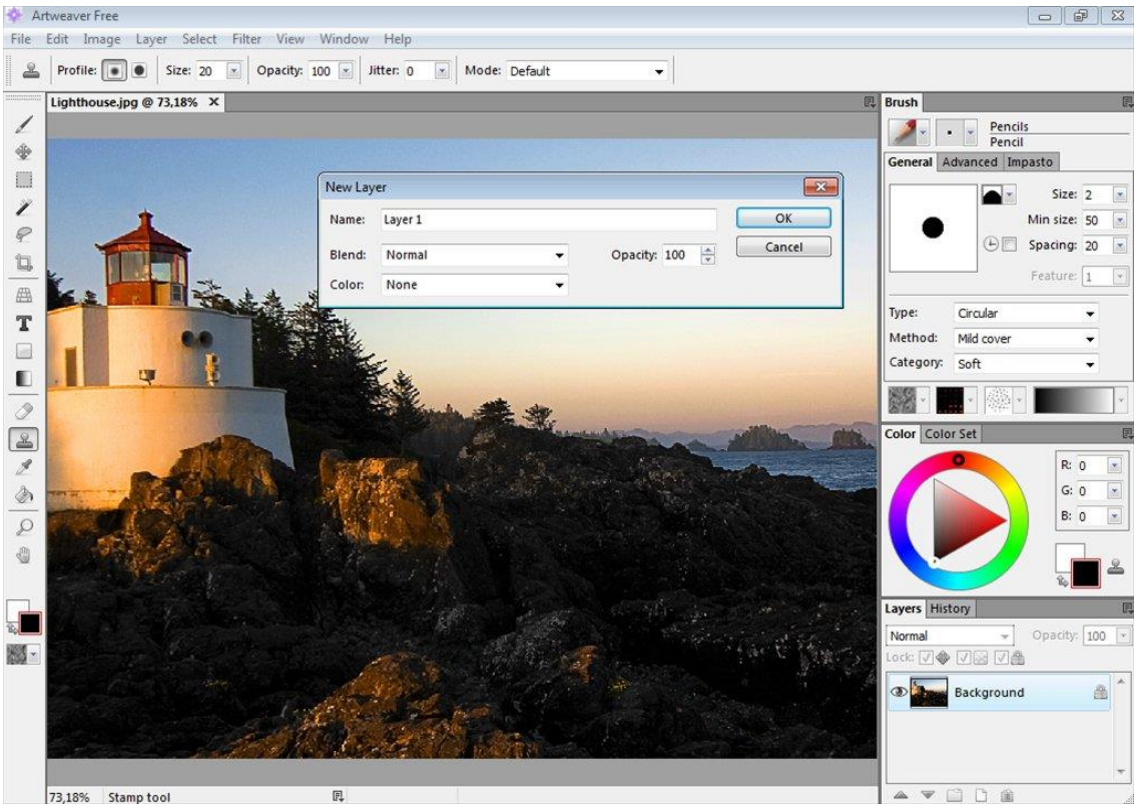

Рисунок 1.9 – Приклад роботи програми Artweaver

Серед проаналізованих програмних систем, IC Measure, Jmicrovision та ChaosPro є максимально наближеними за своїми характеристиками до розроблюваного програмного модуля обробки цифрових зображень. Отже, необхідно обрати ці системи як прототип.

1.5 Постановка задачі

У даній дипломній роботі повинен бути реалізований програмний модуль для цифрової обробки зображень. Програмний модуль розроблений для додатків, які спеціалізуються на обробці зображень і потребують швидкого інструменту для перетворення та аналізу даних. Таким чином, програмний модуль обробляє зображення, реалізує гістограми RGB-каналів та яскравість для подальшої обробки фотографій або використання цієї технології у фото-програмах для кращого автофокусування. Програмний модуль також містить обробку зображень, яка полягає у зміні яскравості зображення, щоб мати можливість продовжувати працювати з його

обробкою.

Програмний модуль обробки цифрових зображень повинен бути зручним та зрозумілим для кінцевого користувача і не вимагати особливих навичок чи знань про алгоритми побудови гістограм. Вхідні дані повинні бути імпортовані в програму у вигляді вхідного зображення, а вихідні дані повинні характеризуватися чіткістю подання. Максимально можлива швидкість цифрової обробки зображень важлива для користувача створеного програмного модуля. Програмний модуль повинен автоматично обробляти вхідну інформацію та виводити результат у доступному форматі.

Для вирішення задачі, наведеної в даному розділі, необхідно вирішити наступні завдання:

1. Розробити математичну модель цифрової обробки зображень;
2. Спроектувати структуру програмного модуля;
3. Програмно реалізувати модуль цифрової обробки зображень;
4. Провести тестування програмного модуля.

1.6 Висновок

У даному розділі проаналізовані сучасні технології цифрової обробки зображень. Обґрунтовано застосування основних етапів цифрової обробки зображень. Здійснено порівняльну характеристику методів, що застосовуються для обробки зображень. Проаналізовано переваги та недоліки програм аналогів для цифрової обробки зображень. Здійснено постановку задачі.

2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ

2.1 Основні етапи процесу обробки цифрових зображень

Обробка зображень – поетапна процедура, що залежить від результатів попереднього етапу, а також знань та досвіду оператора. За допомогою фази попередньої обробки покращується якість зображення, а етап сегментації виділяє елементи та компоненти, які в кінцевому підсумку покращують якість та точність обробки зображення. Безпосередня участь у створенні деяких типів зображень забезпечується комп’ютерами, так як їх неможливо отримати іншим способом. У галузі медицини виділяють наступні типи: комп'ютерна томографія, позитронно-емісійна томографія (ПЕТ), ядерно-магнітний резонанс [23].

Цифрова обробка зображення може використовуватися для:

1. покращення якості зображення, компенсація дефектів у системі запису та зменшення шуму;

2. обрахування критично важливих кількісних параметрів (відстань, площа, об'єм тощо);

3. спрощення визначення (розпізнавання структури, розрахунок дози для променевої терапії);

4. зворотний зв’язок (автоматизована хірургія).

Розглянемо наступні етапи обробки зображень:

Попередня обробка. Даний етап виключає відхилення, що зв’язані з системою формування зображення, і зменшує шум.

Зміна контрастності зображення. З допомогою аналізу гістограм можна робити висновок про якість оцифрування. Коли гістограма має нелінійний розподіл, можлива втрата багатьох деталей. Операції вирівнювання гістограми поліпшують контрастність і, як наслідок, сприяють кращому виділенню деталей.

Сегментація. Даний етап обробки зображень виділяє окремі деталі зображення (в медицині – органи, клітини тощо). Сегментація відбувається за допомогою

ідентифікації однакових пікселів з прийнятним рівнем помилки. Важливий етап особливо для зображень в медичній галузі, адже порівнюючи два сегментовані за часом зображення, виявляється динаміка.

Автоматична комп’ютерна інтерпретація – це все ще проблема. Для її якісного виконання потрібна база знань, що має містити порівняльну та патологічну анатомії. Визначені структури та параметри необхідно порівнювати з відомими структурами та класифікувати.

Для оцінки якості зображення рідко використовують точність вихідного зображення або лінійності зміни параметрів.

Для прикладу, зображення, що отримують в сутінках або в тумані, матиме низьку контрастність, розмиті контури та бліді кольори. Відтворюючи сигнали безпосередньо, зображення, отримане за допомогою інфрачервоного перетворювача, матиме низьку градацію яскравості, що унеможливить підкреслення деталей з низькою контрастністю. Іноді корисно виділити або підсилити певні характеристики новоствореної сцени, щоб покращити їх суб'єктивне сприйняття [24].

Суб'єктивне сприйняття зображення при його візуальному аналізі та обробці ускладнює застосування формалізованих підходів до формування властивостей відтвореного зображення. Тож набувають широкого поширення методи, що не потребують строгих математичних критеріїв при здійсненні візуалізації зображень. Застосування певних методів, установок та режимів зазвичай базується на досвіді, суб’єктивному сприйнятті та художніх навичках.

2.2 Обґрунтування вибору алгоритмів побудови гістограм зображення

У контексті обробки зображення гістограма зображення, як правило, відноситься до гістограми значень інтенсивності пікселів. Ця гістограма являє собою графік, що показує кількість пікселів на зображенні при кожному різному значенні інтенсивності, знайденому на цьому зображенні. Для 8-бітового зображення у градаціях сірого існує 256 різних можливих інтенсивностей, тому гістограма графічно відображатиме 256 чисел, що показують розподіл пікселів серед цих значень сірого.

Гістограми також можуть бути зроблені з кольорових зображень - можуть бути зроблені або окремі гістограми червоних, зелених та синіх каналів, або може бути створена тривимірна гістограма з трьома осями, що представляють червоний, синій та зелений канали, а також яскравість у кожній точці, що представляє кількість пікселів. Точний результат операції залежить від реалізації - це може бути просто зображення необхідної гістограми у відповідному форматі зображення, або це може бути файл даних, що представляє статистику гістограми.

Математично гістограму сприймають як одновимірний масив *H(I)*, усі значення якого є ймовірностями, з якими матриця цифрового зображення містить значення, рівні рівню яскравості *I* [25].

Гістограма дискретного зображення - це дискретна функція $H(b_k) = \frac{N_k}{N}$, де b - k-й рівень яскравості пікселів, N_k - кількість пікселів із яскравістю b_k, а N - загальна кількість піксельних зображень. Значення $H\,(b_k)$ - це оцінка ймовірності пікселя яскравості b_k на зображенні.

Гістограма еквалізується (лінеаризується), якщо зображення має багато пікселів з подібною яскравістю та кілька пікселів з різною яскравістю. На гістограмі зображається, що багато пікселів групуються в певні інтервали яскравості, тоді як деякі інтервали яскравості майже незайняті.

У такому випадку деталі зображення, представленого в цих кольорах, важко розрізнити. Натомість у зображенні є пробіли яскравості без пікселів. Ці вільні інтервали яскравості можна «зайняти» для поліпшення якості зображення. Для цього виконують вирівнювання гістограм.

Якщо існує піксель вихідного зображення з яскравістю b_k, тобто k-м рівнем яскравості в гістограмі ($k = 0 \ldots N - 1$), тоді обчислюється яскравість відповідного пікселя отриманого зображення за формулою:

$$r_k = \sum_{p=0}^{k} H(b_p) = \sum_{p=0}^{k} \frac{N_p}{N}. \qquad (2.1)$$

Алгоритмом побудови гістограми називають послідовну верифікацію цифрового масиву зображення *A (M, N)* для обрахунку кількості його елементів, рівних *I {0,1, ... R}*, де *R = max A (M, N)*. В результаті дістаємо функцію розподілу яскравості *I {0,1, ... R}* цифрового зображення *A (M, N)* - гістограму *H(I)*. Наведемо приклад читання таких значень. Значення гістограми *H(k) = L* показує, що цифрова матриця містить елементи, значення яких дорівнює *k*, що зустрічаються *L* разів.

За допомогою аналізу гістограми яскравості можна отримати необхідну інформацію про вихідне зображення, з якого його отримали, а також довести, яким ефективним є оптичний цифровий перетворювач.

Для зображень з низькою контрастністю є типовим незначний рівень значень рівня сірого кольору. З цього випливає, що існує ймовірність рівних або близьких до них значеннь сірого у сусідніх пікселів.

Гістограма сірого каналу теж може бути застосована для показу, що зображає наявність перевищення динамічного діапазону пристрою оцифровки під час створення цифрового зображення. Дискримінацію гістограми можна використовувати в тих випадках, коли деталі втрачаються тільки на декількох невеликих ділянках зображення. Необхідно уникати можливості відсікання гістограми рівня сірого, урегулювавши систему формування та реєстрації зображень так, щоб у результаті оцифровки не було пікселів як із дуже низькими, так і з високими значеннями сірого кольору.

Кожне цифрове зображення має унікальну гістограму яскравості, тому для багатьох використань вона має всю необхідну інформацію, потрібну для опису зображення.

Контраст називають одним із параметрів, що суттєво визначає якість зображення. Так як зображення має непростий сюжетний характер, виникає потреба визначати контраст на основі контрасту окремих поєднань елементів зображення. Зображення, що утворюються при різних дослідженнях, часто не застосовують повний спектр можливих рівнів яскравості, що зумовлює їх низьку інформативність. Контрастність зображення, яскравість компонентів якого розміщена у вузькій області можливих значень, низька. Одним із способів покращення якості таких зображень є

нелінійне перетворення значень відеосигналу, зокрема, розширення діапазону застосованих значень елементів яскравості до максимально можливого діапазону. Зазвичай такі перетворення базуються на лінійному розтягуванні або гамма-корекції [26].

Після трансформування зображення в цифровий вигляд його можна обробити методами, які впливають на яскравість та контрастність зображення. Таку процедуру використовують лише для значення рівня сірого кольору окремих пікселів та не змінюють просторових особливостей цифрового зображення.

Функцією *f(I)* називають функцію обертання інтенсивності або функцією поступового перетворення. Її суть зазначається в тому, як перетворюється значення рівня сірого кожного пікселя в вихідному документі I_1 для того щоб дістати отримане зображення I_2. Ця функція має вигляд

$$I_2(x, y) = f[I_1(x, y)], \tag{2.2}$$

де $I_1(x,y)$ – піксель в рядку x і колонці y в вихідному документі, а $I_2(x,y)$ – результат використання функції перетворення інтенсивності *f(I)*. Необхідно підкреслити, що значення вихідного пікселя $I_2(x,y)$ залежить тільки від вхідного значення рівня сірого відповідного пікселя I_1 та не має зв’язку з іншими оточуючими пікселями. Дане перетворення називають точковим, що пояснюється несуттєвою просторовою інформацією зображення.

Отже, функція перетворення інтенсивності для окремих пікселів виглядає наступним чином

$$Gv_2 = f(Gv_1), \tag{2.3}$$

де Gv_1 – початкове значення і Gv_2 – результуюче значення в зображеннях.

Загалом функцію перетворення інтенсивності описують у вигляді лінійної функції типу

$$Gv_2 = mGv_1 + b. \quad (2.4)$$

При нахилу m цієї функції рівному одиниці, позитивне значення точки обмеження b може призвести до підвищення яскравості цілого зображення, а від'ємне значення b навпаки може зменшити значення рівня сірого кожного пікселя і призвести до більш темного зображення на виході. На рисунку 2.1 показано приклад, коли цей ефект можна спостерігати при аналізі гістограми зображення до і після застосування функції перетворення інтенсивності.

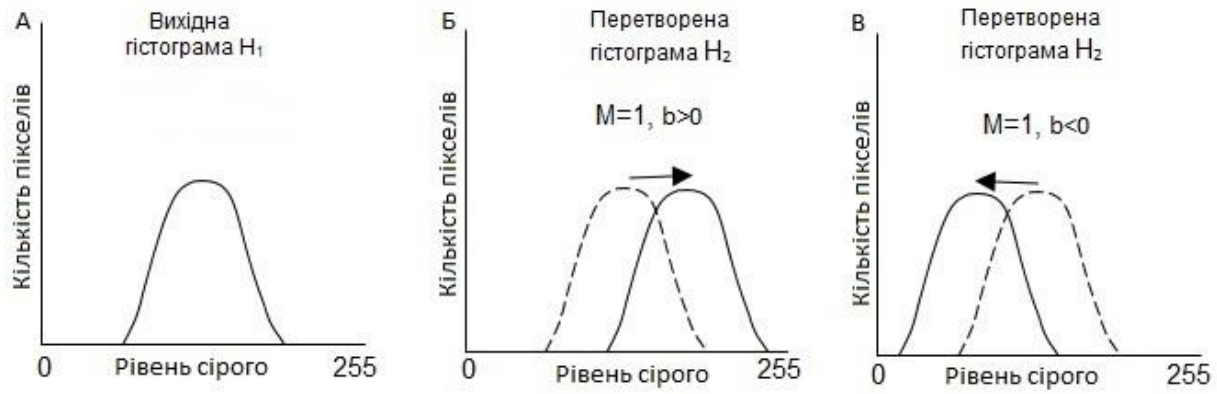

Рисунок 2.1 – Зміни в тоновій гістограмі

Функції перетворення інтенсивності широко розповсюджені в програмах для обробки цифрових зображень з декількох причин. Їх можна застосувати для покращення візуального сприйняття зображення та розрізнення півтонів, що можуть візуально бути невидимими для ока в первісному вигляді.

2.3 Розробка загальної структурної схеми та алгоритмів функціонування програмного модуля цифрової обробки зображень

Програма міститиме модуль введення зображення, модуль обробки зображення, модуль інтерфейсу та модуль побудови гістограм. Загальну структурну схему програмного модуля обробки цифрових зображень наведено на рисунку 2.2

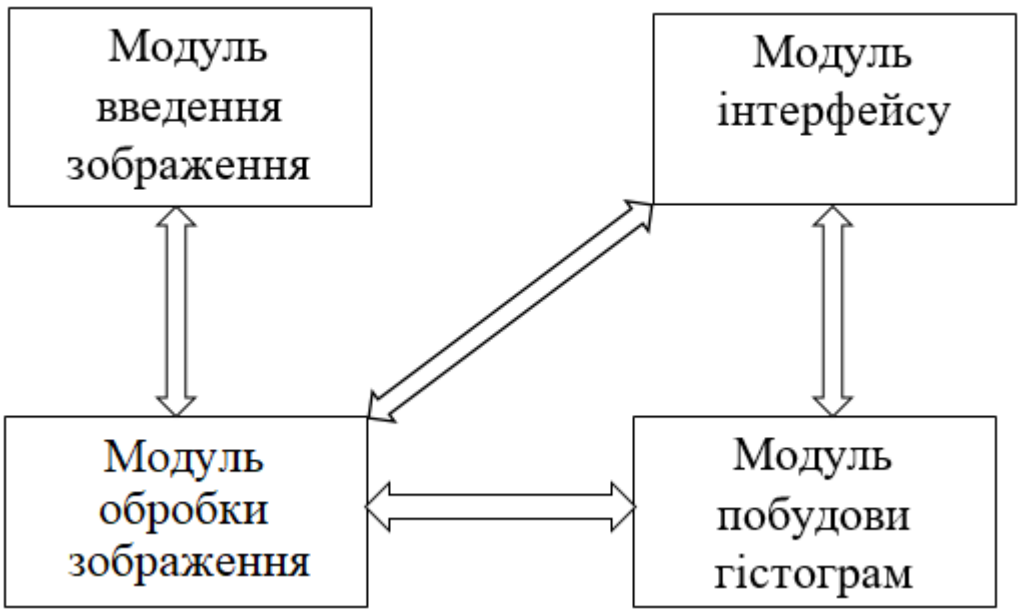

Рисунок 2.2 – Загальна структурна схема програмного модуля цифрової обробки зображень

Модуль графічному інтерфейсу, що має зв’язок з модулем обробки та модулем побудови гістограм, дозволяє отримувати всю необхідну інформацію для користувача. Даний відповідає за коректне відображення результатів обробки та введення даних на екрані.

Модуль обробки зображення отримує зображення від модуля введення зображення та має можливість передання результатів обробки до модуля побудови гістограм. Даний модуль є основним модулем програми та містить інтелектуальний алгоритм обробки зображення. Обробка відбується при застосуванні розробленої математичної моделі.

Завдяки модулеві побудови гістограм відбувається перетворення зображення в масив даних, у якому елементи показані у вигляді стовпців гістограми. Модуль побудови гістограм пов’язаний з модулем інтерфейсу та модулем обробки зображення.

Завдяки модулю введення зображення відбувається імпорт зображень, які необхідно обробити.

Розроблену схему загального алгоритму функціонування системи зображено на рисунку 2.3

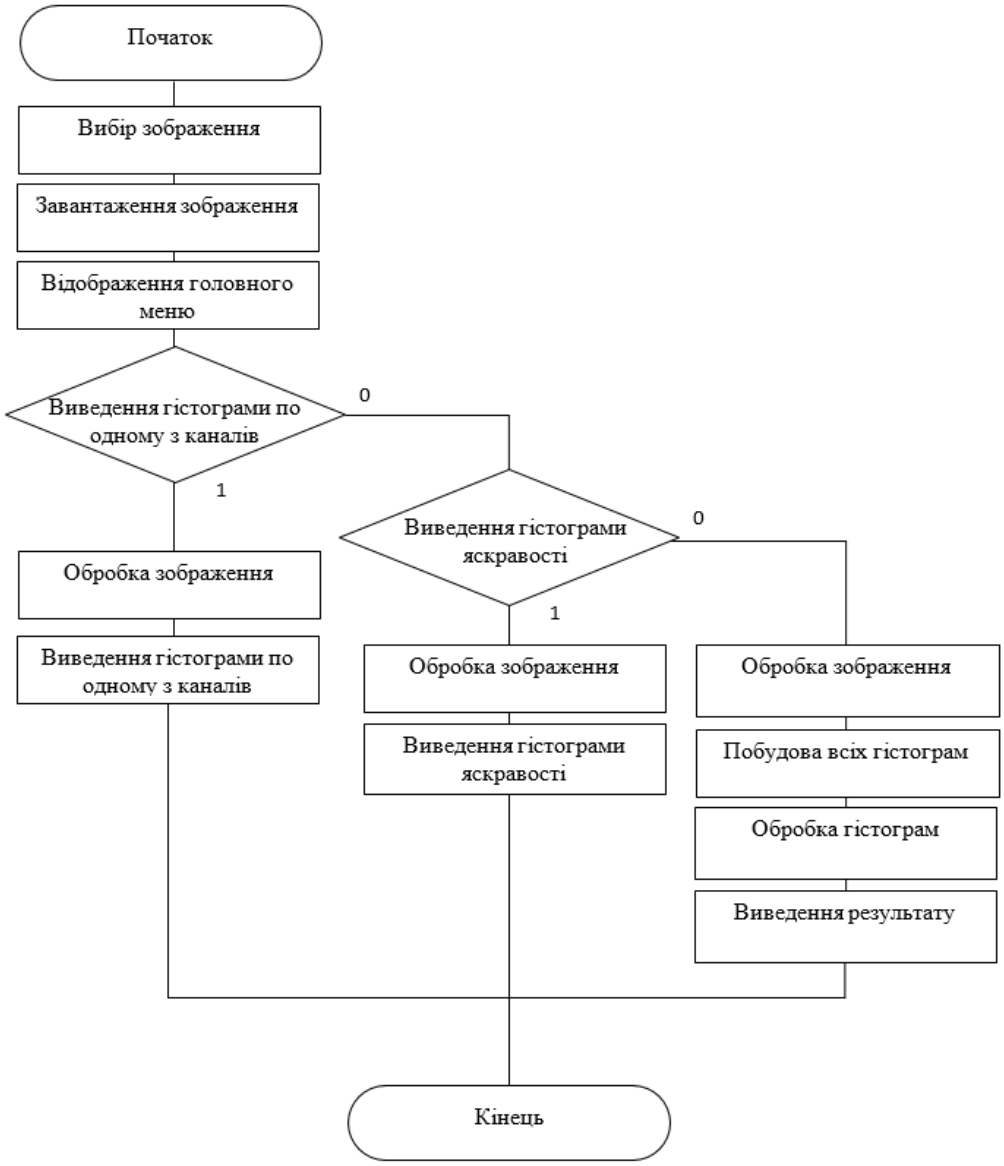

Рисунок 2.3 - Схема алгоритму роботи модуля цифрової обробки зображень

Схему алгоритму розбиття зображення на пікселі зображено на рисунку 2.4. Цей процес є ключовим етапом в обробці зображень, оскільки саме пікселі становлять базові елементи цифрового зображення. Розбиття зображення на пікселі дозволяє комп'ютеру аналізувати, зберігати та маніпулювати зображеннями на основі їхньої піксельної структури

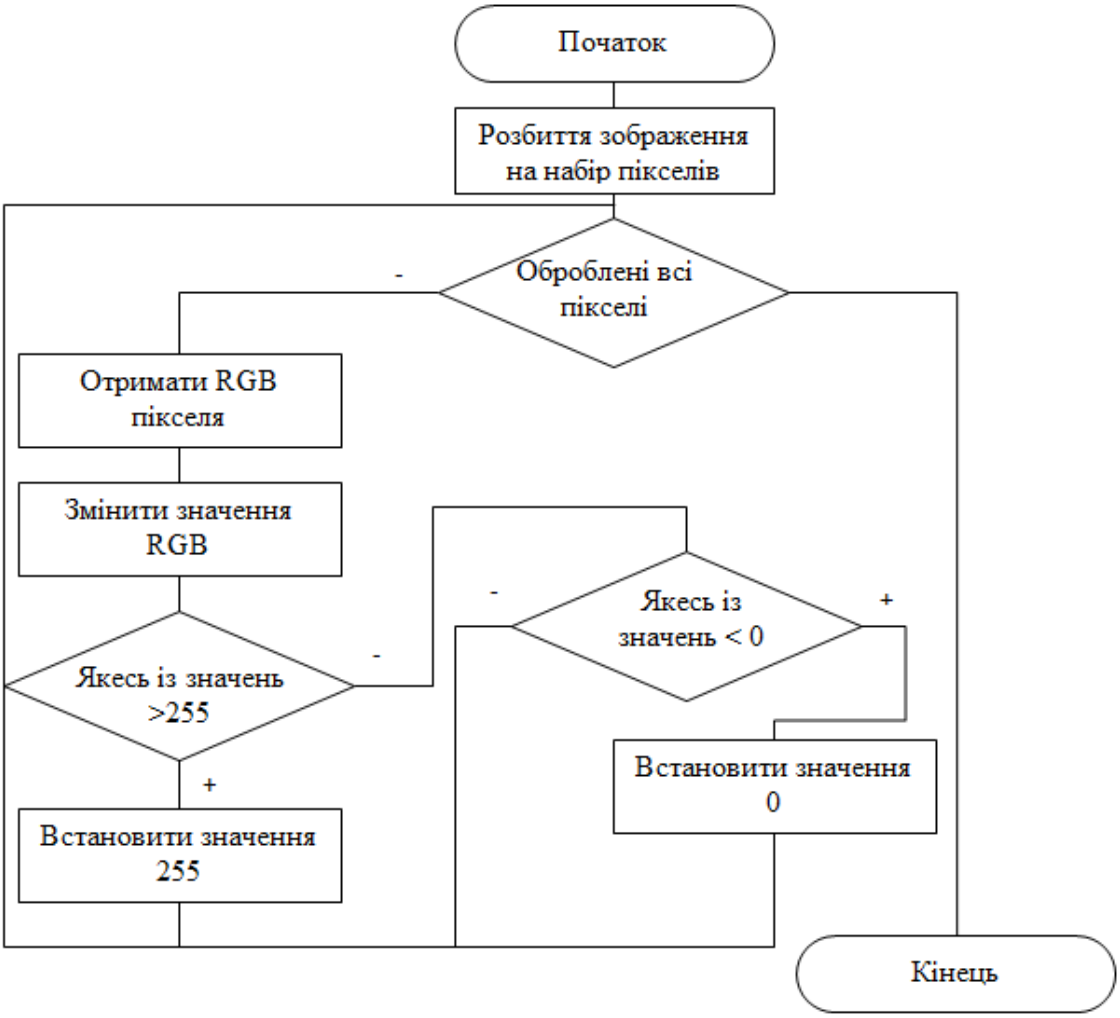

Рисунок 2.4 - Схема алгоритму розбиття зображення на пікселі

2.4 Висновки

У другому розділі було охарактеризовано метод використання гістограм та аналізу спектру зображення, який є оптимальним для цифрової обробки зображення. Даний розділ містить розроблену загальну структурну схему, схему алгоритму функціонування програмного модуля цифрової обробки зображень та схему алгоритму розбиття зображення на пікселі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ЦИФРОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору мови програмування

Мови та технології програмування існують для конкретних цілей, як для розробки сайтів, програмування для Windows, або поєднання обох наведених функцій. Для розробки програмного модуля обробки цифрових зображень розглянемо такі популярні мови програмування, як C #, C ++ та Java, а також наведемо їх основні переваги та недоліки.

Першою розглянемо мову C ++. C ++ - це мова програмування загального призначення, яка сьогодні широко використовується для конкурентного програмування. Вона має обов’язкові, об’єктно-орієнтовані та загальні особливості програмування. C ++ працює на багатьох платформах, таких як Windows, Linux, Unix, Mac тощо. Але є переваги та недоліки C ++. Це одна з найдавніших та найефективніших мов, яка також продовжує домінувати у сфері програмування.

Наведемо кілька переваг мови [28]:

- Програма C ++ має багато переваг, коли вона передбачає програмування. Усі автономні файли програми C ++ повинні використовувати найголовнішу функцію, щоб програма могла запускатися та мотивувати свої функції.
- Програма C ++ може підтримувати об'єднання та структури, що є сумішшю окремих та складених файлів. Вона використовує стандартну програму C ++, згадану як «.cpp», C ++ використовує зарезервоване слово бібліотеки, згадане як «goto», те саме, що Java продовжити або розбити команди.
- Глобальні дані та глобальні функції використовуються в C ++, які не використовуються в багатьох інших мовах високого рівня в галузі ПК, і це є перевагою для мов програмування.

- Програма C ++ використовує програмування з декількома парадигмами, парадигма означає планування програмування, парадигма, що стосується логіки, структури та процедури програми, програма C ++ є багатопарадигмою, означає, що вона відповідає трьом парадигмам: загальній, імперативній, об'єктно-орієнтованій.
- Програма C ++ корисна для низькорівневої мови програмування і дійсно ефективна для загальних цілей, вона ефективно забезпечує продуктивність та пам'ять, пропонує абстракцію високого рівня в мові предметної області.

Недоліки C ++:

- Універсальний код Java для правил ASCII є 16-розрядним, тоді як програма C ++ є лише 8-розрядним, отже, C ++ може бути менш вражаючою мовою програмування, але економить пам'ять.
- Програма C ++ є складною під час дуже великої програми високого рівня, C ++ застосовується для програм, що відповідають певним платформам, зазвичай.
- Програма C ++ не може підтримувати вивіз сміття, вона не підтримує динамічне розподіл пам’яті, вона не захищена, оскільки є покажчиком, функцією друзів та глобальною змінною, і не підтримує вбудовані потоки.

Це лише декілька недоліків мови, однак вони суттєві для розробки програмного модуля обробки цифрових зображень. Тому слід розглянути інші мови програмування.

C # - це об'єктно-орієнтована мова програмування, яку називають «C sharp» [29]. Вона випускається компанією Microsoft під керівництвом Андерса Хейлсберга та його членів команди в рамках ініціативи .Net - і була прийнята Європейською асоціацією виробників комп'ютерів (ECMA) та Міжнародною організацією стандартів (ISO).

C # покращив та оновив різні функції C та C ++, включаючи такі:

- C # має строгі булеві дані змінних типів, таких як bool, тоді як булеві типи змінних C ++ можуть повертатися як цілі числа або покажчики, щоб уникнути поширених помилок програмування.
- C # автоматично управляє пам’яттю віддалених об’єктів через збирач сміття, що скасовує турботи розробника та витоки пам’яті.
- Тип C # є більш безпечним, ніж C ++, і має лише надійні перетворення за замовчуванням (наприклад, цілочисельне розширення), які виконуються під час компіляції або часу виконання.

Синтаксис C # надзвичайно виразний, але також простий і легкий у вивченні. Синтаксис C # буде безпосередньо ідентифікований будь-ким, хто добре знайомий з C або Java. Розробники, які знайомі з будь-якою з цих мов, можуть почати ефективно працювати в C # за дуже короткий час. Синтаксис C # робить більш зрозумілим багато труднощів C і пропонує унікальні функції, такі як нульовий, перерахований, делегований, лямбда-вирази та прямий доступ до пам'яті, які недоступні в Java. C # підтримує стандартні методи та типи, які пропонують кращий захист та продуктивність у своєму роді, та ітератори, які дозволяють реалізаторам збору класів описувати власні поведінки ітерацій, які клієнтський код може легко використовувати. Вирази запитів, інтегровані в мову (LINQ), роблять потужний набір запитів першокласною конструкцією мови.

Серед переваг C # необхідно відзначити:

- C # є суто об'єктно-орієнтованим, але C ++ - це поєднання об'єктно-орієнтованого та орієнтованого на процедури.
- C # є безпечним для типу
- Програмісту не потрібно приділяти особливої уваги таким проблемам, як втрата пам'яті, що є тривожною проблемою для програміста на C ++.
- Концепція збірки добре вирішує питання контролю версій.
- Легка у розробці, багата бібліотека класів робить багато функцій простими для реалізації.
- Крос-платформа. Додаток буде працювати нормально, лише якщо машина встановила .NET Framework.

- Підтримка розподіленої системи.

Недоліками C # можна назвати:

- Програміст не може робити такі речі низького рівня, як безпосередня взаємодія з апаратним забезпеченням за допомогою драйверів та прошивки.
- Він не постачається з незалежним компілятором, який може прямо інтерпретувати максимальні рівні мови основної апаратної архітектури чистого асемблера. Він використовує свій байтовий код та компілятор JIT, який надзвичайно вбудований у фреймворк .Net і є основою структури .Net як посередник між машинним кодом у місці безпосереднього спілкування з апаратним забезпеченням.

Java - це об'єктно-орієнтована мова програмування, випущена Sun Microsystems в 1995 році як ключовий елемент платформи Java. Наразі Oracle працює над цією мовою. Компанія придбала Sun Microsystems у 2009 році. Синтаксис мови дуже схожий на C та C ++. Офіційна реалізація програм Java компілюється в байт-код, який інтерпретується віртуальною машиною при виконанні для певної платформи [30].

Oracle публікує компілятор Java та віртуальну машину Java, які відповідають специфікаціям процесів спільноти Java під загальною публічною ліцензією GNU.

Перевагами Java є:

- Однією з головних переваг Java є незалежність від платформи, на якій працюють програми: код може працювати на Windows, Solaris, Linux, Machintosh та інших. Це дуже важливо, коли програми завантажуються з Інтернету для подальшого запуску в різних операційних системах.

- Синтаксис мови Java подібний до синтаксису мови C ++ і для програмістів, знайомих з C і C ++, що полегшує вивчення.

- Java - це об’єктно-орієнтована мова, яка навіть більша за C ++. Усі сутності в Java є об'єктами, за винятком кількох примітивних типів, таких як Б. Числа. (Об’єктно-орієнтоване програмування полегшує розробку складних проектів.)

- висока надійність. Розробники надали інструменти Java, які виключають можливість створення додатків, які приховують найпоширеніші помилки.

Програми Java перекладаються в байт-код, який виконується віртуальною машиною Java, програмою, яка обробляє байт-код і передає інструкції апаратному забезпеченню як інтерпретатор. Перевага цього методу виконання програм полягає в тому, що байт-код повністю незалежний від операційної системи та обладнання, тому ви можете запускати програми Java на будь-якому обладнанні, для якого існує відповідна віртуальна машина [31].

Java була обрана завдяки тому, що вона дозволяє повністю реалізувати завдання розробки програмного модуля обробки цифрових зображень, вона зручна, а головне, вона просто має організувати користувальницький інтерфейс таким чином, щоб користувачі могли швидко ознайомитися з програмою та почати користуватися нею. Java також пропонує низку інших переваг, включаючи потужні інструменти для роботи з різними типами даних та об'єктів, високу швидкість та можливість підключення зовнішніх ресурсів.

3.2 Обґрунтування вибору середовища програмування

Так як в попередньому підрозділі було виявлено суттєві переваги програмування на Java, слід розглянути середовище програмування, що завдяки своїм перевагам та особливостям зможе сприяти легкій та зручній розробці програмного модуля обробки цифрових зображень.

Таким середовищем програмування є IntelliJ IDEA 2020.

IntelliJ IDEA - це комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP тощо) від JetBrains. Система пропонується у вигляді зменшеної функціональності безкоштовної версії «Community Edition» та повністю функціональної комерційної версії «Ultimate Edition», для яких активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Вихідний код версії спільноти поширюється під ліцензією Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X та Windows [32].

IntelliJ IDEA - це набір інтегрованих високотехнологічних інструментів програмування, що включає інтелектуальний редактор вихідного коду з удосконаленими засобами автоматизації, електроінструменти для рефакторингу

коду, інтегровану навантаження J2EE преміум-класу, інструменти інтеграції та контрольну версію тестового середовища Ant / JUnit, а також унікальний код перевірки. графічний інтерфейс користувача.

Унікальні можливості JetBrains IntelliJ IDEA підтримують рутинну рутину, з часом усувають умови та покращують якість коду, виводячи розробника на новий рівень.

Середовище IntelliJ IDEA, має наступні переваги:

- глибоке розуміння коду - після індексації вихідного коду IntelliJ IDEA пропонує безліч можливостей для швидкої та ефективної розробки: інтелектуальне завершення, аналіз коду в реальному часі та надійний рефакторинг;

- відповідає всім вимогам розробника - усі необхідні інструменти вже доступні, тому немає необхідності шукати та встановлювати плагіни, щоб інтегрувати їх у системи контролю версій та підтримувати популярні мови та фреймворки;

- розумне заповнення коду - хоча основне автозаповнення включає назви класів, методів, полів та ключових слів за обсягом, розумне автозаповнення надає лише елементи коду, які є актуальними в поточному контексті;

- розробка різними мовами - хоча IntelliJ IDEA є в першу чергу IDE для Java, він розуміє та надає інтелектуальну допомогу в написанні коду на SQL, JPQL, HTML, JavaScript та багатьох інших мовах і дозволяє редагувати код, що не є Java, у рядкові літерали коду Java;

- продуктивна робота - IntelliJ IDEA аналізує повторювані завдання під час розробки та автоматизує їх, щоб ви могли зосередитися на загальній картині;

- ергономічне середовище - подбали про те, щоб розробник ні на що не відволікався, усунуто або мінімізовано ризик втрати контексту. IntelliJ IDEA розуміє, що ви хочете зробити, і автоматично запускає відповідні інструменти;

- продуктивність у всьому - IntelliJ IDEA не тільки допомагає писати код в редакторі, але й спрощує роботу в цілому. Ви можете швидко та легко заповнити поле, знайти елемент у списку, відкрити потрібне вікно, змінити налаштування та багато іншого.

3.3 Обґрунтування вибору бібліотек для Java

Так як для програмного модуля обробки цифрових зображень застосовуємо об’єктно-орієнтовану мову програмування Java, опишемо деякі бібліотеки, що використовуються у розробленому програмному модулі.

Для створення графічного інтерфейсу на Java існує кілька бібліотек. Розглянемо бібліотеку Swing, яка є однією з найпопулярніших бібліотек і дозволяє створювати повноцінні віконні додатки на мові Java.

Swing був розроблений, щоб надати більш функціональний набір програмних компонентів для створення графічного інтерфейсу користувача, ніж попередні інструменти AWT. Компоненти Swing підтримують певні модулі зовнішнього вигляду, які динамічно поєднуються. За допомогою них можна імітувати графічний інтерфейс платформи (тобто компонент може бути динамічно пов’язаний з іншим, специфічним для цього типу та поведінки операційної системи). Основним недоліком таких компонентів є відносно повільна робота, хоча це нещодавно не було підтверджено через збільшення потужності персональних комп'ютерів. Позитивна сторона - універсальність інтерфейсу створюваних програм на всіх платформах [33].

Створення нового вікна необхідно робити в окремих файлах, а в основному класі лише запускати потрібне вікно.

Для роботи з бібліотекою необхідно імпортувати класи, що дозволять не тільки створювати графічний дизайн, але також відстежувати різні натискання на кнопки і виконувати дії після натискання.

```
import java.awt. *;
import java.awt.event. *;
import javax.swing. *;
```

Щоб працювати з багатьма методами, наш основний клас повинен наслідувати все від класу з назвою JFrame.

Після виконання успадкування можна отримати доступ до класів для створення полів, кнопок і інших елементів.

У Swing використовуються контейнери «Container» для угруповання декількох елементів в одну форму. Перед створенням контейнера необхідно встановити: назву вікна, його розміри і координати (x, y, w, h), а також встановили операцію при

закритті вікна.

Далі створюється об'єкт на основі класу Container і прописується метод getContentPane (). Даний метод повертає контейнер верхнього рівня, в який поміщаються всі інші об'єкти на вікні.

Для розміщення об'єктів варто вибрати будь-якої шар. Наприклад, можна використаний шар GridLayout, що дозволяє розташовувати об'єкти в форматі сітки або ж таблиці.

Imageio – клас, що містить статичні методи, зручні для того, щоб розташувалися ImageReaders і ImageWriters, і виконувати просте кодування і декодування. Таким чином підтримується читання зображень в форматах GIF, JPEG і PNG, а запис - в форматах JPEG і PNG. Запис зображень у форматі GIF не передбачена.

Працюючи з бібліотеками введення / виведення зображень, можна помітити, що майже вся робота запитується через статичні методи класу ImageIO. Ці статичні методи - все, що потрібно для базового використання. Наприклад, щоб прочитати зображення, його місцезнаходження передається одному з наступних методів read класу ImageIO:

- read(File input);
- read(ImageInputStream stream);
- read(InputStream input);
- read(URL input).

Наступний фрагмент коду містить приклад використання Imageio.

```
import javax.imageio.ImageIO;
import java.io.File;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.IOException;

public class Image {
   public static void main(String[] args) {
      try {

         // Открываем изображение
         File file = Main.img;
         BufferedImage source = ImageIO.read(file);

         BufferedImage result = new BufferedImage(source.getWidth(), source.getHeight(), source.getType());
```

```
            for (int x = 0; x < source.getWidth(); x++) {
               for (int y = 0; y < source.getHeight(); y++) {

                  Color color = new Color(source.getRGB(x, y));

                  int blue = color.getBlue();
                  int red = color.getRed();
                  int green = color.getGreen();

                  int grey = (int) (red * 0.299 + green * 0.587 + blue * 0.114);

                  int newRed = grey;
                  int newGreen = grey;
                  int newBlue = grey;

                  Color newColor = new Color(newRed, newGreen, newBlue);

                  result.setRGB(x, y, newColor.getRGB());
               }
            }
            File output = new File("katana_grey.jpg");
            ImageIO.write(result, "jpg", output);

         } catch (IOException e) {

            System.out.println("Файл не знайдено ");
         }
      }
   }
```

Для побудови гістограм було створено клас Line. Даний клас використовує абстрактні класи Point2D, Line2D та Rectangle2D.

Клас Point2D обчислює точку, що зображає місце в координатному просторі (x, y) [34].

Клас Rectangle2D зображає прямокутник, визначений розташуванням (x, y) і розмірність (w x h) [35].

Клас Line2D зображає відрізок лінії в просторі координат (x, y) [36]. Цей клас, як і всі Java 2D API, використовує систему координат за замовчуванням, яка називається простором користувача, в якій значення осі y збільшуються вниз, а значення осі x збільшуються направо.

Описані класи є лише абстрактними суперкласами усіх об’єктів, що зберігають 2D-координату. Фактичне представлення зберігання координат залишається для підкласу.

Для демонстрації розробки класу Line наведено наступний код

```
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;
public class Line extends Line2D{

	private Point p1;
	private Point p2;
	public Line(){
		p1 = new Point();
		p2 = new Point();
	}

	public Line(double x1, double y1, double x2, double y2){
		p1 = new Point(x1, y1);
		p2 = new Point(x2, y2);
	}

	@Override
	public Rectangle2D getBounds2D() {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public Point2D getP1() {
		// TODO Auto-generated method stub
		return p1;
	}

	@Override
	public Point2D getP2() {
		// TODO Auto-generated method stub
		return p2;
	}

	@Override
	public double getX1() {
		// TODO Auto-generated method stub
		return p1.getX();
	}

	@Override
	public double getX2() {
		// TODO Auto-generated method stub
		return p2.getX();
	}

	@Override
	public double getY1() {
		// TODO Auto-generated method stub
```

```
		return p1.getY();
	}

	@Override
	public double getY2() {
		// TODO Auto-generated method stub
		return p2.getY();
	}

	@Override
	public void setLine(double x1, double y1, double x2, double y2) {
		// TODO Auto-generated method stub
		p1.setLocation(x1, y1);
		p2.setLocation(x2, y2);
	}
}
```

Spring Framework (або коротше Spring) - це універсальний фреймворк з відкритим кодом для платформи Java [20].

Spring пропонує рішення багатьох проблем, з якими стикаються розробники та організації Java, які прагнуть створити інформаційну систему на основі платформи Java. Через його широку функціональність важко визначити основні конструктивні елементи, з яких вона складається. Незважаючи на широку інтеграцію, Spring не повністю інтегрований з платформою Java Enterprise, що є великою причиною його популярності.

Spring, мабуть, найвідоміший як джерело вдосконалень (функціональних можливостей), необхідних для ефективної розробки складних бізнес-додатків, які виходять за рамки високопродуктивних програмних моделей, що переважали в галузі в минулому. Ще однією перевагою є те, що раніше невикористані функції були введені в переважаючі сьогодні методи розробки навіть за межами платформи Java.

З цією технологією пов’язано два цікаві моменти.

По-перше, на відміну від багатьох інших платформ (таких як Apache Struts, які створюють лише Інтернет), Spring можна використовувати для побудови будь-яких програм Java (наприклад, окремих, веб-програм або Java Enterprise Edition (JEE)).

По-друге, легка вага не має нічого спільного з кількістю класів чи розміром розподілу, але визначає принцип всієї філософії стрибків - мінімальний вплив. Платформа Spring є легкою, оскільки вона максимально використовує свої основні функції.

3.4 Програмна реалізація модуля цифрової обробки зображень

На рисунку 3.1 зображено загальну UML-діаграму класів програмного модуля обробки цифрових зображень.

Клас Main – основний клас програми що здійснює доступ до всіх інших функціональних класів.

У класі MainG відбувається відображення початкової активності програми, що містить кнопку для вибору зображення.

Клас Men відповідає за створення головного меню, що реалізовує перехід до кожного з пунктів обробки.

З допомогою класу Gist відбувається побудова гістограм для обробки зображення.

Клас GraphicsPanel – відповідає відображення гістограм.

Клас Point використовує абстрактний клас Point2D, що обчислює точку для представлення місця в координатному просторі (x, y).

Клас Line відповідає за використання абстрактних класів Point2D, Line2D та Rectangle2D для побудови гістограм.

Клас Processing відповідає за зміну яскравості зображення.

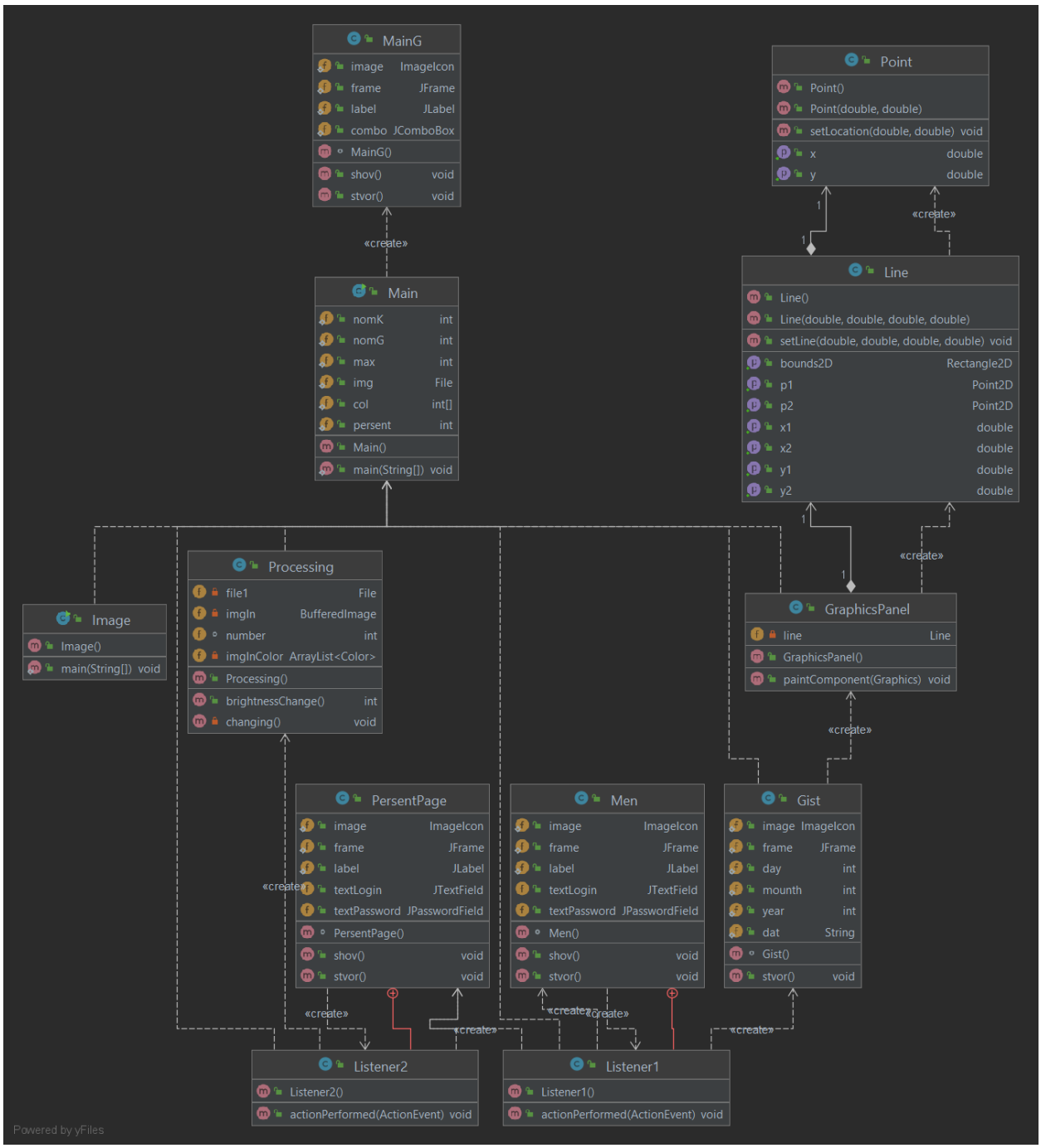

Рисунок 3.1 – Загальна UML-діаграма класів

3.5 Тестування та аналіз результатів роботи програмного модуля цифрової обробки зображень

Програмний модуль цифрової обробки зображень було протестовано. Було проведено 300 запусків модуля, протестовано можливості його роботи для подальшої оцінки його роботи.

Спершу відкривається вікно початкової активності (рис. 3.2). Воно містить лише одну кнопку «Обрати зображення».

Необхідно обрати картинку з допомогою провідника (рис. 3.3) та натиснути кнопку «Open».

Після обрання зображення користувач натискає кнопку «Open». Як результат відкривається головне меню обробки зображення, що дозволяє обрати один із варіантів обробки зображення: «Гістограма по червоному каналу», «Гістограма по зеленому каналу», «Гістограма по синьому каналу», «Гістограма яскравості», «Змінити яскравість» (рис. 3.4).

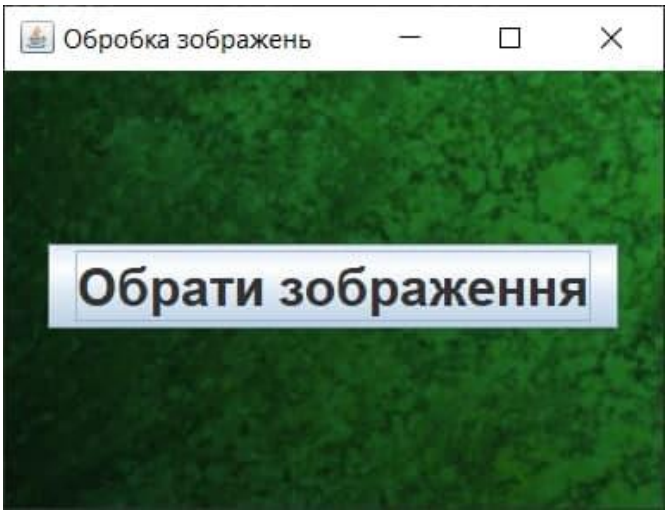

Рисунок 3.2 – Вікно початкової активності

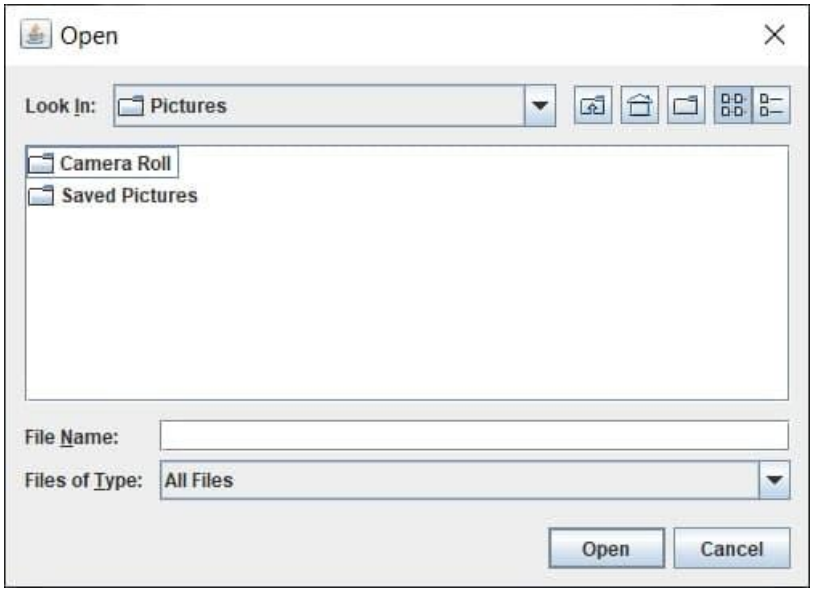

Рисунок 3.3 – Вибір зображення

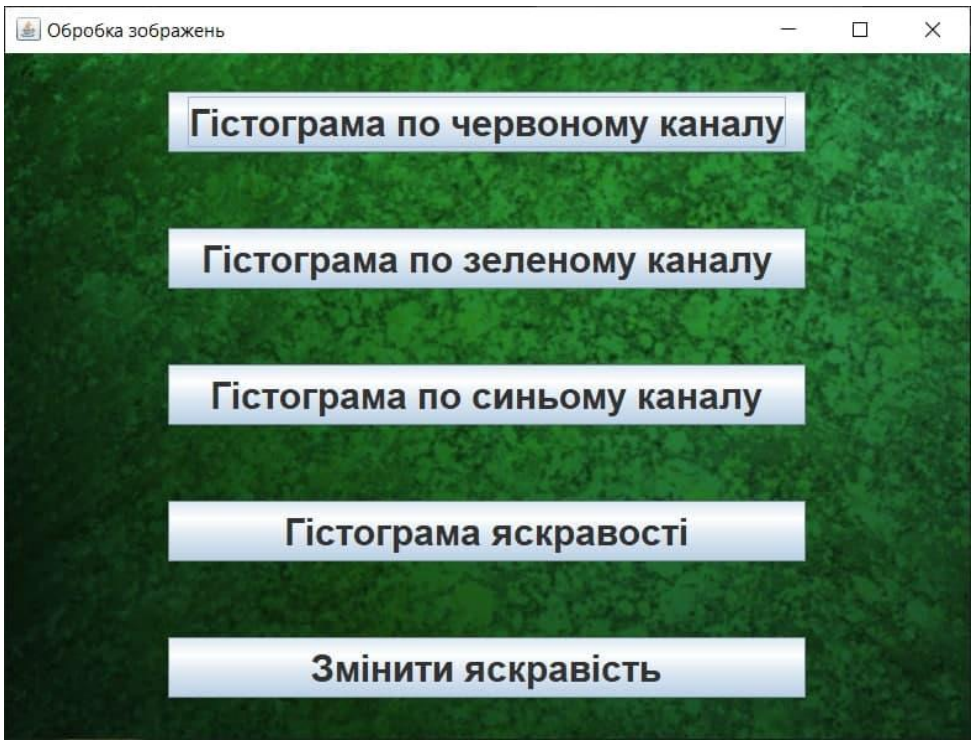

Рисунок 3.4 – Меню аналізу зображення

Вибравши пункт «Гістограма яскравості» відбувається перехід до вікна з відповідною гістограмою (рис. 3.5). Вибравши пункт «Гістограма по зеленому каналу» відбувається перехід до вікна з гістограмою, зображеною на рисунку 3.6.

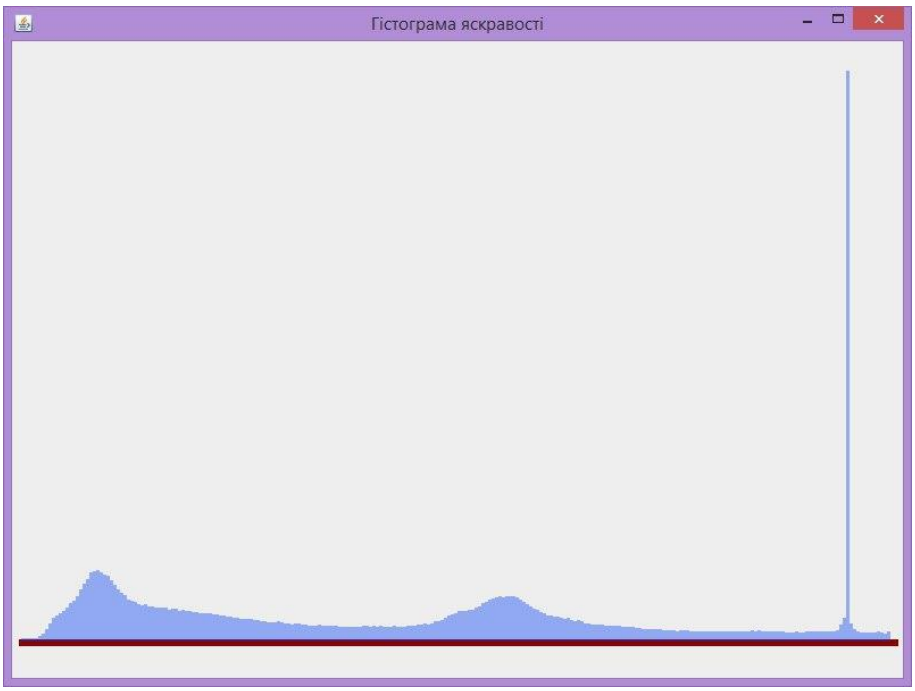

Рисунок 3.5 – Гістограма яскравості тестового зображення

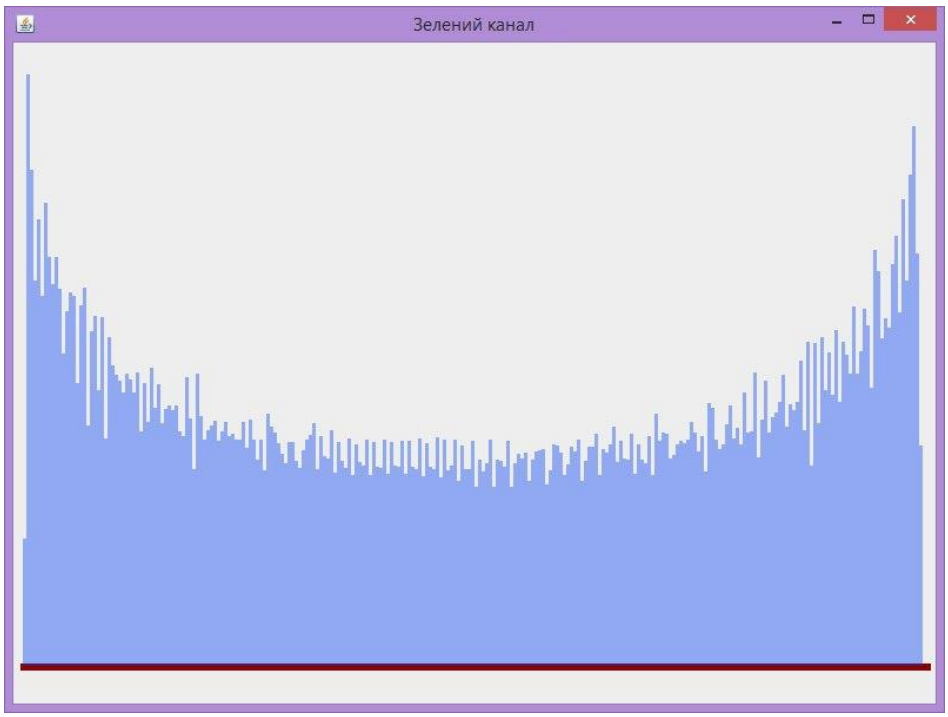

Рисунок 3.6 – Гістограма по зеленому каналу

Вибравши пункт «Змінити яскравість» відбувається перехід до вікна для вводу числового значення зміни яскравості (рис. 3.7). Користувачу необхідно ввести числове значення та натиснути кнопку «Далі».

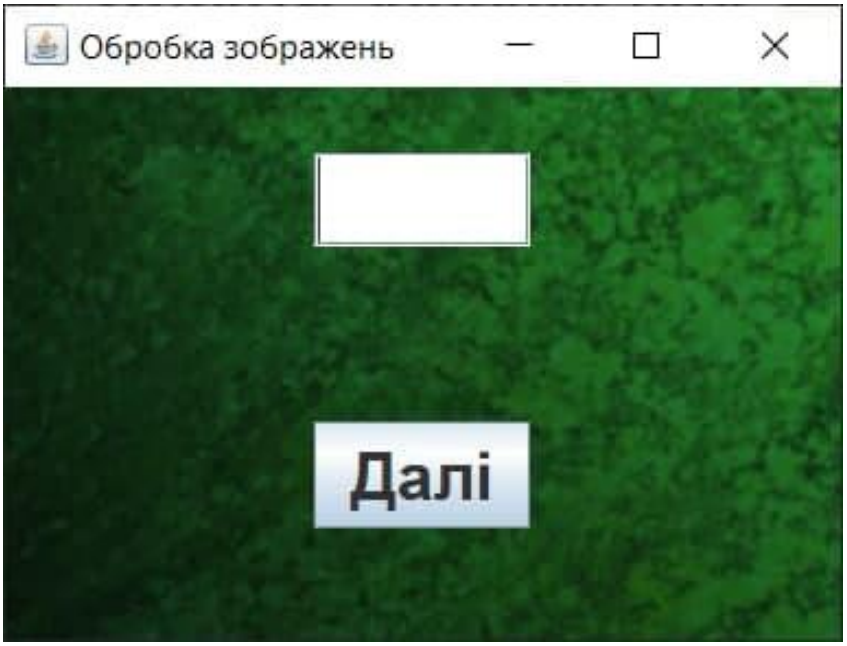

Рисунок 3.7 – Вікно зміни яскравості

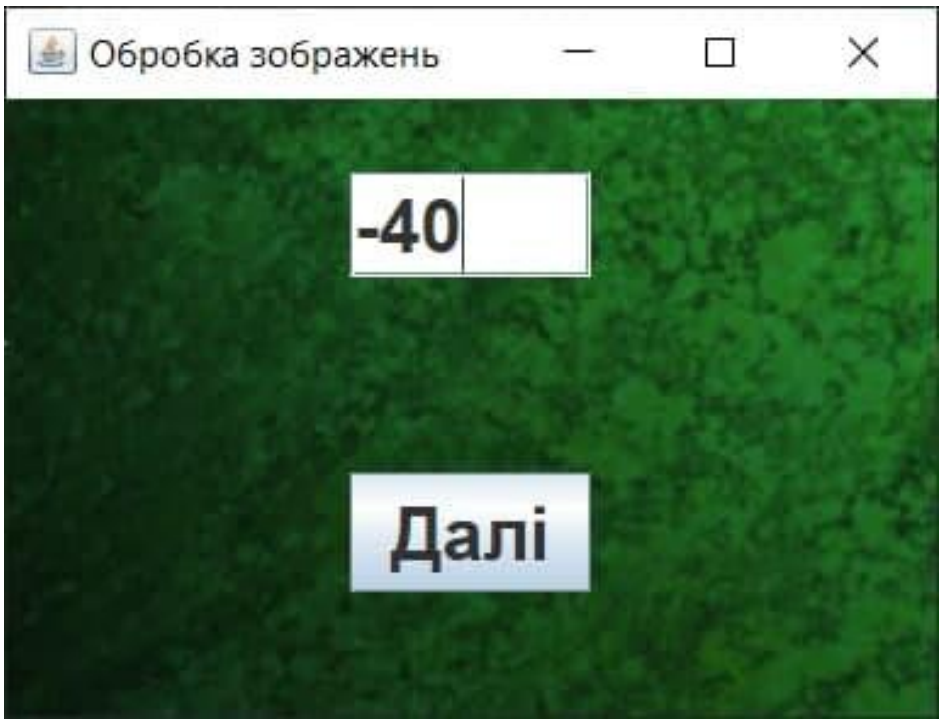

Рисунок 3.8 – Введення яскравості

Таблиця 3.1 містить порівняльні результати роботи розробленого додатку з програмами-аналогами IC Measure, Jmicrovision та ChaosPro. Розроблений додаток має вищу швидкість ніж розглянуті програми-аналоги.

Таблиця 3.2 – Порівняльний аналіз роботи програм-аналогів та розробленого додатку

Програма	Швидкість обробки
Розроблений додаток	0,5 секунди
Jmicrovision	0,9 секунди
IC Measure	0,9 секунди
ChaosPro	1,3 секунди

Отже, із таблиці 3.1 випливає, що розроблений програмний модуль цифрової обробки зображень має на 0,5 с вищу швидкодію обробки, що означає досягнення мети.

3.6 Висновки

В третьому розділі обґрунтовано використання мови програмування Java, розглянуто її особливості, переваги та недоліки. Обґрунтовано вибір середовища програмування та наведено переваги роботи в ньому, обґрунтовано використання наведених бібліотек мови програмування Java, використаних для написання розробленого програмного модуля цифрової обробки зображень.

В даному розділі розроблено UML-діаграму класів програми. Програмно реалізовано та протестовано модуль цифрової обробки зображень.

Проведене тестування показало підвищення швидкості обробки зображення порівняно з програмами аналогами на 0,5с, що довело досягнення мети.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи було розроблено програмний модуль цифрової обробки зображень.

В роботі проаналізовані сучасні технології цифрової обробки зображень. Обґрунтовано застосування основних етапів цифрової обробки зображень. Здійснено порівняльну характеристику методів, що застосовуються для обробки зображень. Проаналізовано переваги та недоліки програм аналогів для цифрової обробки зображень. Здійснено постановку задачі. Розроблено загальну структурну схему, схему алгоритму функціонування програмного модуля цифрової обробки зображень та схему алгоритму розбиття зображення на пікселі. Розроблено UML-діаграму класів програми. Здійснено програмну реалізацію модуля цифрової обробки зображень мовою програмування Java в середовищі IntelliJ IDEA.

Проведене тестування показало підвищення швидкості обробки зображення порівняно з програмами аналогами на 0,5с.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Інженерна та комп’ютерна графіка : підруч. для студ. ВНЗ / В. Є. Михайленко, В. В. Ванін, С. М. Ковальов. – 6-те вид. – К. : Каравела, 2020. – 360 с. – (Українська книга)
2. Ющик О. Основи цифрової обробки зображень [Текст] : навч. посіб. для студ. вид.-поліграф. спец. вищ. навч. закл. III-IV рівня акредитації / О. Ющик. Львів : УАД, 2018. – 184 с. : іл.
3. Романюк О. Н.. Комп'ютерна графіка. Навчальний посібник – Вінниця: ВДТУ, 2015.
4. Кобилін О. А. Методи цифрової обробки зображень: навч. посібник./ О. А. Кобилін, І. С.Творошенко. — Харків: ХНУРЕ, 2021. — 124 с.
5. Dash L., Chatterji B.N. Adaptive contrast enhancement and de-enhancement // Pattern Recognition, 1992. – V. 24. – № 4. – P.289–302.
6. Тотосько О. В. Цифрова обробка сигналів та зображень / О. В. Тотосько, П. Д. Стухляк. — Тернопіль: ТНТУ, 2016. — 140 с.
7. Russ John / The image processing handbook / John C. Russ . – Boca Raton; Ann Arbor; London etc. : CRC, 1992 . – 445 с. – ISBN 0-8493-4233-3.
8. Munteanu C., Rosa A. Gray-Scale Image Enhancement as an Automatic Process Driven by Evolution // IEEE Transactions on Systems, Man, and Cybernetics. – 2004. – V. 34. – P. 1292–1298.
9. Солтис І. В. Опрацювання графічної інформації / І. В. Солтис, О. В. Дуболазов, Р. М. Бесага. — Чернівці: Чернівецький нац. ун-тет, 2022. — 124 с.
10. Müller Meinard. Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications / Meinard Müller. — Springer, 2015. — 487p.
11. Paylidis, George. Mixed Raster Content: Segmentation, Compression, Transmission. / George Paylidis. — Springer, 2018 . — 354 p.
12. Каїро А. Функціональне мистецтво: вступ до інфографіки та візуалізації /переклад з англ. Л. Белея за ред. Р. Скакуна. Львів: Видавництво Українського католицького університету 2018. – 350с.

13.А. С. Василюк, Н. І. Мельникова. Комп'ютерна графіка: навч. посіб. для студентів напряму підгот. 6.040303 «Систем. аналіз». — Львів: Вид-во Львів. політехніки, 2016. — 308 с.

14.Веселовська Г. В. Комп'ютерна графіка: Навчальний посібник для вузів. — Херсон: ОЛДІ-плюс, 2014. — 582 с.

15. Інженерна комп'ютерна графіка: навч. посіб. для студ. вищ. навч. закл., Р. А. Шмиг, В. М. Боярчук, І. М. Добрянський, В. М. Барабаш ; за ред. Р. А. Шмига ; М-во освіти і науки, молоді та спорту України. — Л. : Укр. бестселер, 2022. — 600 с.

16. Основи комп'ютерної графіки: курс лекцій / О. Я. Різник ; М-во освіти і науки, молоді та спорту України, Нац. ун-т «Львів. політехніка». — Л. : Вид-во Львів. політехніки, 2019. — 220 с.

17. Програмування комп‘ютерної графіки та мультимедійні засоби : навч. посіб. / Л. М. Журавчак, О. М. Левченко. – Львів : Львівська політехніка, 2019. – 276 с.

18. Мартиненко Т. В., Розробка графічних інтерфейсів у середовищі Visual Basic 6.0. / Мартиненко Т. В., Савкова О. Й., Турупалов В. В., Фонотов А. М. Навч. посіб. – Львів: “Магнолія 2006”, 2021. – 312 с..

19..Фрактальна графіка [Електронний ресурс]. Режим доступу: https://sites.google.com/a/bpedk.com.ua/informatika/temi/fraktalna-grafika (дата звернення 12.03.2024). – Назва з екрана. 12.Інструменти фрактальної графіки [Електронний ресурс]. Режим доступу: https://sites.google.com/site/namalujsvoumriu25/home/fraktalna-grafika

20. Безкоштовний курс «Візуалізація даних» [Електронний ресурс]. Режим доступу: https://courses.prometheus.org.ua/courses/IRF/DV101/2016_T3/about

21. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний модуль цифрової обробки зображень

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 91,79% Схожість 8,21%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Пінтя Д.В.

Керівник роботи Шевчук О.Ф.

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧА

Спершу відкривається вікно початкової активності (рис. Б.1). Воно містить лише одну кнопку «Обрати зображення».

Необхідно обрати картинку з допомогою провідника (рис. Б.2) та натиснути кнопку «Open».

Після обрання зображення користувач натискає кнопку «Open». Як результат відкривається головне меню обробки зображення, що дозволяє обрати один із варіантів обробки зображення: «Гістограма по червоному каналу», «Гістограма по зеленому каналу», «Гістограма по синьому каналу», «Гістограма яскравості», «Змінити яскравість» (рис. Б.3).

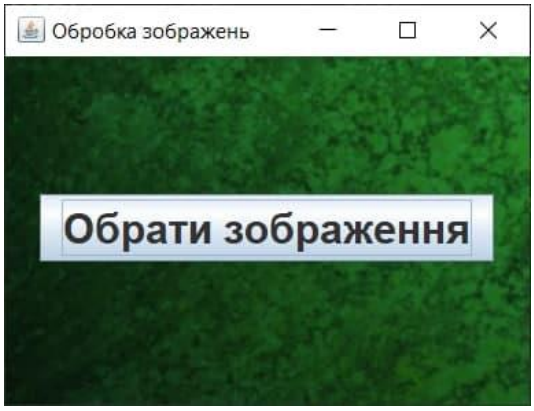

Рисунок Б.1 – Вікно початкової активності

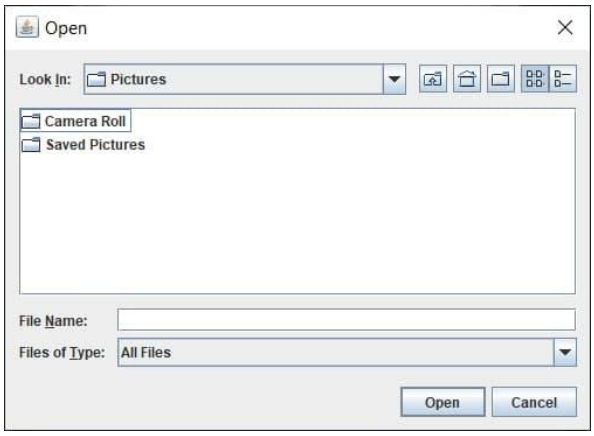

Рисунок Б.2 – Вибір зображення

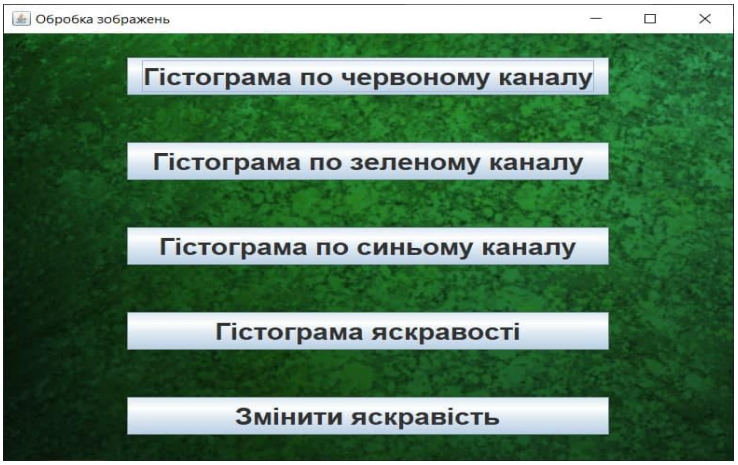

Рисунок Б.3 – Меню аналізу зображення

Вибравши пункт «Гістограма яскравості» відбувається перехід до вікна з відповідною гістограмою (рис. Б.4). Вибравши пункт «Гістограма по зеленому каналу» відбувається перехід до вікна з гістограмою, зображеною на рисунку Б.5.

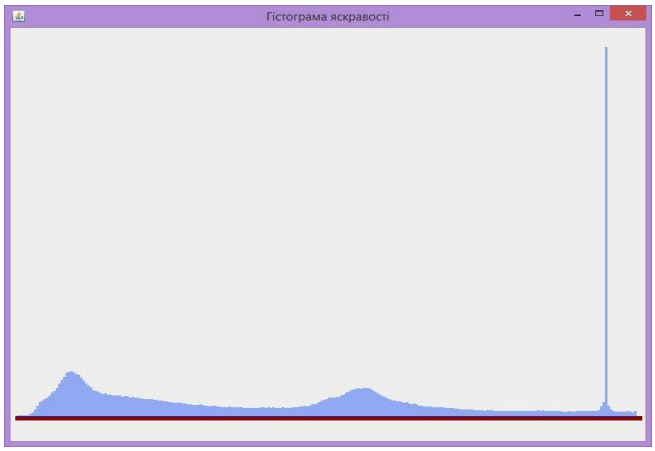

Рисунок Б.4 – Гістограма яскравості тестового зображення

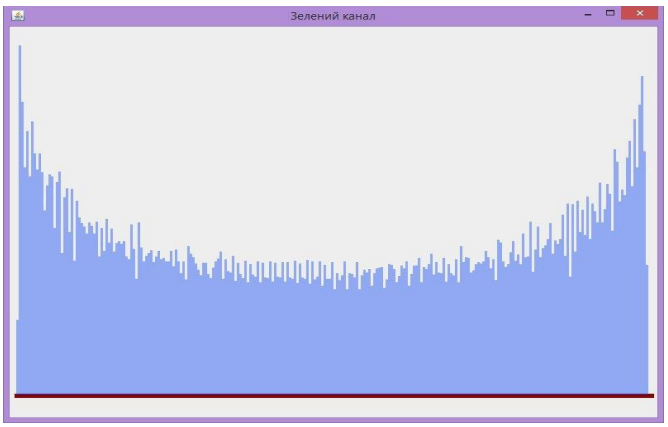

Рисунок Б.5 – Гістограма по зеленому каналу

Вибравши пункт «Змінити яскравість» відбувається перехід до вікна для вводу числового значення зміни яскравості (рис. Б.6). Користувачу необхідно ввести числове значення та натиснути кнопку «Далі».

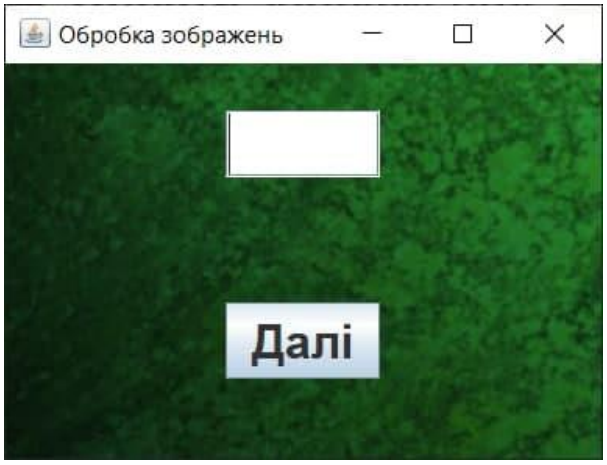

Рисунок Б.6 – Вікно зміни яскравості

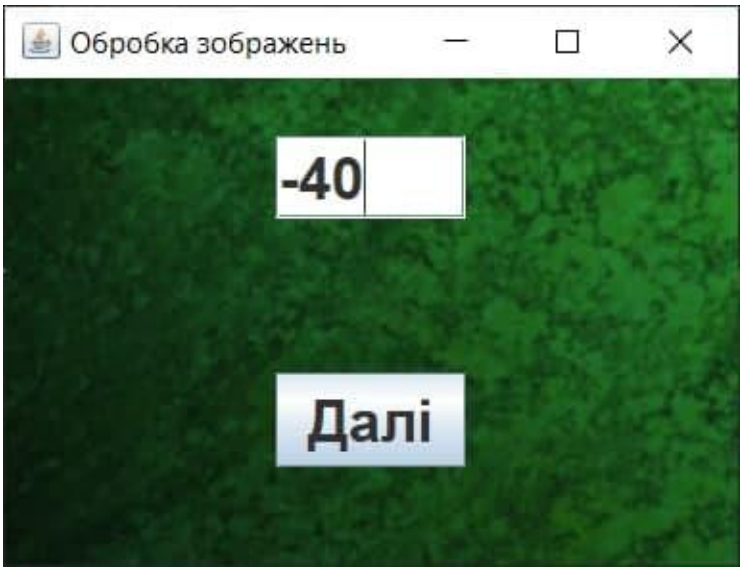

Рисунок Б.7 – Введення яскравості

ДОДАТОК В
ЛІСТИНГ ПРОГРАМИ

```
#include <vcl.h>
```

Клас Main

```
import java.io.File;

public class Main {

	public static int nomK, nomG, max;
	public static File img;
	public static int col[] = new int[256];
	public static int persent;

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		MainG maG = new MainG();

		maG.stvor();

	}

}
```

Клас MainG

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.swing.*;

public class MainG {
	public static ImageIcon image;
	public static JFrame frame;
	public static JLabel label;

	public static JComboBox combo;

	// this.EXIT_ON_CLOSE;
	MainG() {
		image = new ImageIcon(getClass().getResource("f1.jpg"));
	}
```

```
	public void stvor() {
		frame = new JFrame();
		frame.setTitle("Обробка зображень");
		frame.setSize(400, 300);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLocationRelativeTo(null);

		Font font = new Font("TimesRoman", Font.BOLD, 30);
		Font font1 = new Font("TimesRoman", Font.BOLD, 24);
		// zaput.setBackground(Color.RED);
		label = new JLabel();
		label.setLayout(new GridBagLayout());
		label.setVisible(true);
		label.setBackground(Color.BLUE);
		label.setIcon(image);

		JButton btnAuto = new JButton();

		btnAuto.setText("Обрати зображення");
		btnAuto.setFont(font);
		label.add(btnAuto, new GridBagConstraints(0, 1, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
				GridBagConstraints.HORIZONTAL, new Insets(100, 10, 10, 10), 0, 0));

		btnAuto.addActionListener(new Listener());

		frame.add(label);
		frame.setVisible(true);

	}

	public void shov() {
		frame.setVisible(false);
		frame.remove(label);

	}

	public class Listener implements ActionListener {

		@Override
		public void actionPerformed(ActionEvent JCom) {
			JFileChooser fileChooser = new JFileChooser();
			fileChooser.showDialog(label, "Open");

			Main.img=fileChooser.getSelectedFile();
			shov();

			Men m = new Men();
			m.stvor();
		}

	}
```

```
}
```

Клас Men

```
import java.awt.Color;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class Men {
	public static ImageIcon image;
	public static JFrame frame;
	public static JLabel label;

	public JTextField textLogin;
	public JPasswordField textPassword;

	// this.EXIT_ON_CLOSE;
	Men() {
		image = new ImageIcon(getClass().getResource("f1.jpg"));
	}

	public void stvor() {
		frame = new JFrame();
		frame.setTitle("Обробка зображень");
		frame.setSize(800, 600);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLocationRelativeTo(null);

		Font font = new Font("TimesRoman", Font.BOLD, 30);
		Font font1 = new Font("TimesRoman", Font.BOLD, 24);
		// zaput.setBackground(Color.RED);
		label = new JLabel();
		label.setLayout(new GridBagLayout());
		label.setVisible(true);
		label.setBackground(Color.BLUE);
		label.setIcon(image);

		JButton btnAuto = new JButton();
		JButton btnReg = new JButton();
		btnAuto.setText("Гістограма по червоному каналу");
		btnAuto.setFont(font);
```

```
            label.add(btnAuto, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
                    GridBagConstraints.HORIZONTAL, new Insets(30, 10, 10, 10), 0, 0));

            btnReg.setText("Гістограма по зеленому каналу");
            btnReg.setFont(font);
            label.add(btnReg, new GridBagConstraints(0, 1, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
                    GridBagConstraints.HORIZONTAL, new Insets(30, 10, 10, 10), 0, 0));

            JButton btnReg1 = new JButton();
            btnReg1.setText("Гістограма по синьому каналу");
            btnReg1.setFont(font);
            label.add(btnReg1, new GridBagConstraints(0, 2, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
                    GridBagConstraints.HORIZONTAL, new Insets(30, 10, 10, 10), 0, 0));

            JButton btnReg2 = new JButton();
            btnReg2.setText("Гістограма яскравості");
            btnReg2.setFont(font);
            label.add(btnReg2, new GridBagConstraints(0, 3, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
                    GridBagConstraints.HORIZONTAL, new Insets(30, 10, 10, 10), 0, 0));

            JButton btnReg3 = new JButton();
            btnReg3.setText("Змінити яскравість");
            btnReg3.setFont(font);
            label.add(btnReg3, new GridBagConstraints(0, 4, 1, 1, 0.0, 0.9,
GridBagConstraints.NORTH,
                    GridBagConstraints.HORIZONTAL, new Insets(30, 10, 10, 10), 0, 0));
            btnAuto.addActionListener(new Listener1());
            btnReg.addActionListener(new Listener1());
            btnReg1.addActionListener(new Listener1());
            btnReg2.addActionListener(new Listener1());
            btnReg3.addActionListener(new Listener1());

            frame.add(label);
            frame.setVisible(true);

        }

        public void shov() {
            frame.setVisible(false);
            frame.remove(label);

        }

        public class Listener1 implements ActionListener {

            @Override
            public void actionPerformed(ActionEvent JCom) {
                String button = ((JButton) JCom.getSource()).getText();
```

```
                    if (button.equals("Гістограма по червоному каналу")) {
                        Main.nomG = 1;
                        Men m = new Men();
                        m.shov();
                        Gist g = new Gist();
                        g.stvor();
                    } else {
                        if (button.equals("Гістограма по зеленому каналу")) {
                            Main.nomG = 2;
                            Men m = new Men();
                            m.shov();
                            Gist g = new Gist();
                            g.stvor();
                        } else {
                            if (button.equals("Гістограма по синьому каналу")) {
                                Main.nomG = 3;
                                Men m = new Men();
                                m.shov();
                                Gist g = new Gist();
                                g.stvor();
                            } else {
                                if (button.equals("Гістограма яскравості")) {
                                    Main.nomG = 4;
                                    Men m = new Men();
                                    m.shov();
                                    Gist g = new Gist();
                                    g.stvor();
                                } else {
                                    shov();
                                    new PersentPage().stvor();
//                                  Змінити яскравість
                                }
                            }
                        }
                    }
                }
            }
        }
```

Клас Gist

```
import java.awt.Color;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;

public class Gist {
	public static ImageIcon image;
	public static JFrame frame;
	public static int day, mounth, year;
	public static String dat;

	Gist() {
		image = new ImageIcon(getClass().getResource("f1.jpg"));
		// setTitle("Прогнозування завантаженості сервера");
		// setSize(800, 600);
		// setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		// setLocationRelativeTo(null);
		// setVisible(true);
	}

	public void stvor() {
		frame = new JFrame();
		String s="";
		if(Main.nomG==1){
			s="Червоний канал";
		}
		if(Main.nomG==2){
			s="Зелений канал";
		}
		if(Main.nomG==3){
			s="Синій канал";
		}
		if(Main.nomG==4){
			s="Гістограма яскравості";
		}
		frame.setTitle(s);

		frame.setSize(800, 600);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setLocationRelativeTo(null);
		frame.setLayout(new GridBagLayout());
		try {

			// Открываем изображение
```

```
                File file = Main.img;
                BufferedImage source = ImageIO.read(file);

                BufferedImage result = new BufferedImage(source.getWidth(), source.getHeight(),
source.getType());

                for (int x = 0; x < source.getWidth(); x++) {
                        for (int y = 0; y < source.getHeight(); y++) {

                                Color color = new Color(source.getRGB(x, y));

                                int blue = color.getBlue();
                                int red = color.getRed();
                                int green = color.getGreen();
                                if (Main.nomG == 1) {
                                        Main.col[red]++;
                                }
                                if (Main.nomG == 2) {
                                        Main.col[green]++;
                                }
                                if (Main.nomG == 3) {
                                        Main.col[blue]++;
                                }
                                if (Main.nomG == 4) {
                                        int y1 = (int) (0.2126 * red + 0.7152 * green + 0.0722 *
blue);
                                        Main.col[y1]++;
                                }

                        }
                }
                Main.max = 0;
                for (int i = 0; i < 255; i++) {
                        if (Main.col[i] > Main.max) {
                                Main.max = Main.col[i];
                        }
                }

        } catch (IOException e) {

                System.out.println("Файл не знайдено");
        }

        frame.setVisible(true);

        GraphicsPanel graphicsPanel = new GraphicsPanel();
```

```
		frame.add(graphicsPanel, new GridBagConstraints(0, 0, 1, 1, 1, 1,
GridBagConstraints.CENTER,
				GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));
		frame.setVisible(true);

	}

}
```

Клас GraphicsPanel

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.geom.RoundRectangle2D.Double;
import java.util.Random;

import javax.swing.JPanel;

import org.w3c.dom.css.Rect;

public class GraphicsPanel extends JPanel {

	private Line line;

	public GraphicsPanel() {
		line = new Line(25, 100, 30, 20);
	}

	@Override
	public void paintComponent(Graphics g) {
		Graphics2D g2 = (Graphics2D) g;
		// g2.draw(line);
		Color newColor = new Color(139, 0, 0);
		g2.setStroke(new BasicStroke(6.0f));
		g2.setColor(newColor);
		g2.drawLine(9, 530, 777, 530);

		Font font1 = new Font("TimesRoman", Font.BOLD, 14);
		Font font = new Font("TimesRoman", Font.BOLD, 10);
		Font font2 = new Font("TimesRoman", Font.BOLD, 18);

		g2.setStroke(new BasicStroke(3.0f));
		Color newColor1 = new Color(0, 58, 250, 100);
		g2.setColor(newColor1);
		for (int i = 0; i < 255; i++) {
			float h = 500f / (float) Main.max * (float) Main.col[i];
```

```
                int hh = (int) h;
                g2.drawLine(9 + i * 3, 527, 9 + i * 3, 527 - hh);
                Main.col[i]=0;
            }
            // g2.setStroke(new BasicStroke(6.0f));
            /*
             * g2.setColor(newColor1); g2.drawLine(30, 505, 750, 505);
             * g2.drawLine(30, 480, 750, 480); g2.drawLine(30, 455, 750, 455);
             * g2.drawLine(30, 430, 750, 430); g2.drawLine(30, 405, 750, 405);
             * g2.drawLine(30, 380, 750, 380); g2.drawLine(30, 355, 750, 355);
             * g2.drawLine(30, 330, 750, 330); g2.drawLine(30, 305, 750, 305);
             * g2.drawLine(30, 280, 750, 280); g2.drawLine(30, 255, 750, 255);
             * g2.drawLine(30, 230, 750, 230); g2.drawLine(30, 205, 750, 205);
             * g2.drawLine(30, 180, 750, 180); g2.drawLine(30, 155, 750, 155);
             * g2.drawLine(30, 130, 750, 130); g2.drawLine(30, 105, 750, 105);
             * g2.drawLine(30, 80, 750, 80); g2.drawLine(30, 55, 750, 55);
             * g2.setColor(Color.BLUE);
             */

            // g2.drawRect(30, 50, 250, 250);

            // g2.drawLine(750, 530, 750, 30);

        }

}
```

Клас Line

```
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

public class Line extends Line2D{

        private Point p1;
        private Point p2;

        public Line(){
                p1 = new Point();
                p2 = new Point();
        }

        public Line(double x1, double y1, double x2, double y2){
                p1 = new Point(x1, y1);
                p2 = new Point(x2, y2);
        }

        @Override
        public Rectangle2D getBounds2D() {
                // TODO Auto-generated method stub
```

```
            return null;
        }

        @Override
        public Point2D getP1() {
            // TODO Auto-generated method stub
            return p1;
        }

        @Override
        public Point2D getP2() {
            // TODO Auto-generated method stub
            return p2;
        }

        @Override
        public double getX1() {
            // TODO Auto-generated method stub
            return p1.getX();
        }

        @Override
        public double getX2() {
            // TODO Auto-generated method stub
            return p2.getX();
        }

        @Override
        public double getY1() {
            // TODO Auto-generated method stub
            return p1.getY();
        }

        @Override
        public double getY2() {
            // TODO Auto-generated method stub
            return p2.getY();
        }

        @Override
        public void setLine(double x1, double y1, double x2, double y2) {
            // TODO Auto-generated method stub
            p1.setLocation(x1, y1);
            p2.setLocation(x2, y2);

        }

}
```

Клас Point

```
import java.awt.geom.Point2D;

public class Point extends Point2D{
	private double x;
	private double y;

	public Point() {
		// TODO Auto-generated constructor stub
	}

	public Point(double x, double y) {
		this.x=x;
		this.y=y;
	}

	@Override
	public double getX() {
		// TODO Auto-generated method stub
		return x;
	}

	@Override
	public double getY() {
		// TODO Auto-generated method stub
		return y;
	}

	@Override
	public void setLocation(double arg0, double arg1) {
		// TODO Auto-generated method stub
		this.x=x;
		this.y=y;
	}

}
```

ДОДАТОК Г
ГРАФІЧНА ЧАСТИНА

Рисунок Г.1 – Основні стадії цифрової обробки зображень

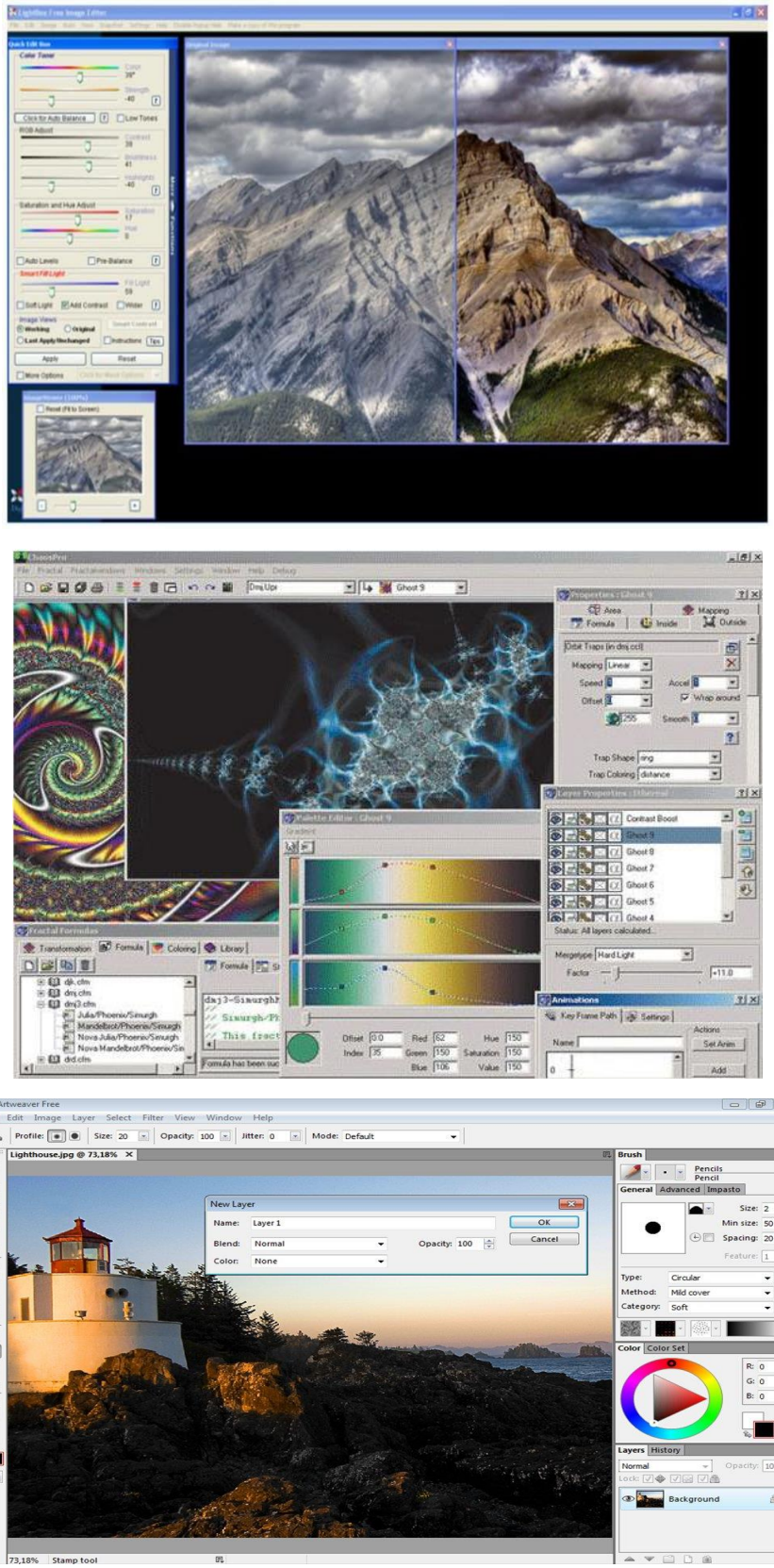

Рисунок Г.2 – Програми аналоги цифрової обробки зображень

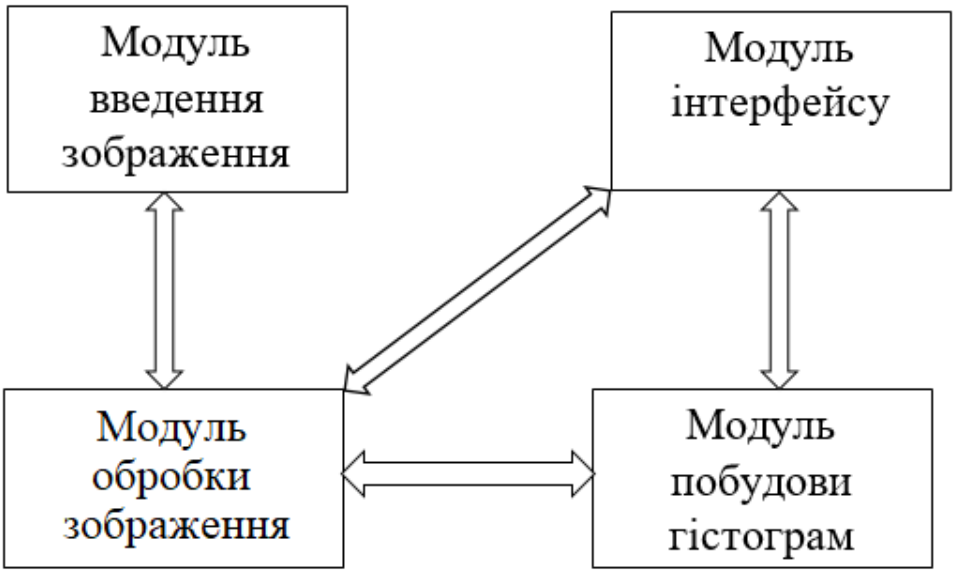

Рисунок Г.3 – Структура програмного модуля цифрової обробки зображень

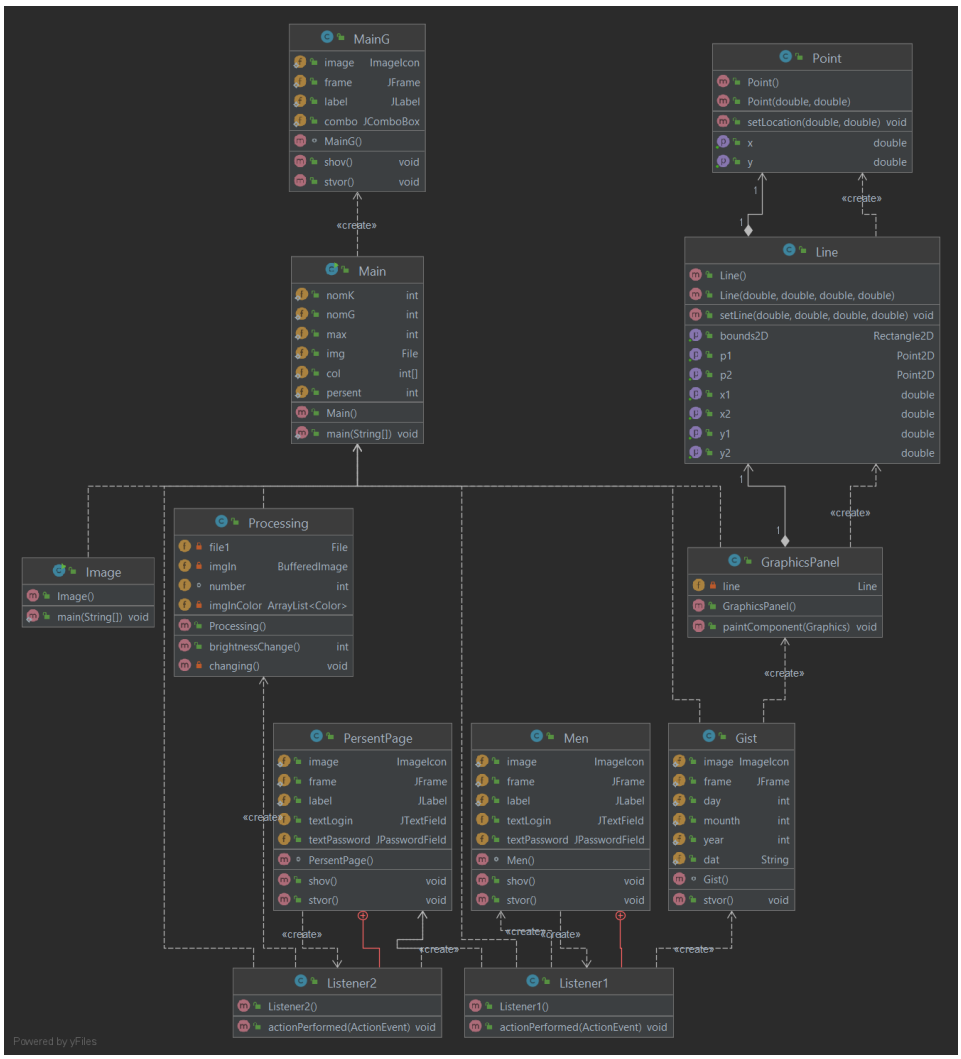

Рисунок Г.4 – UML-діаграма класів програмного модуля цифрової обробки зображень

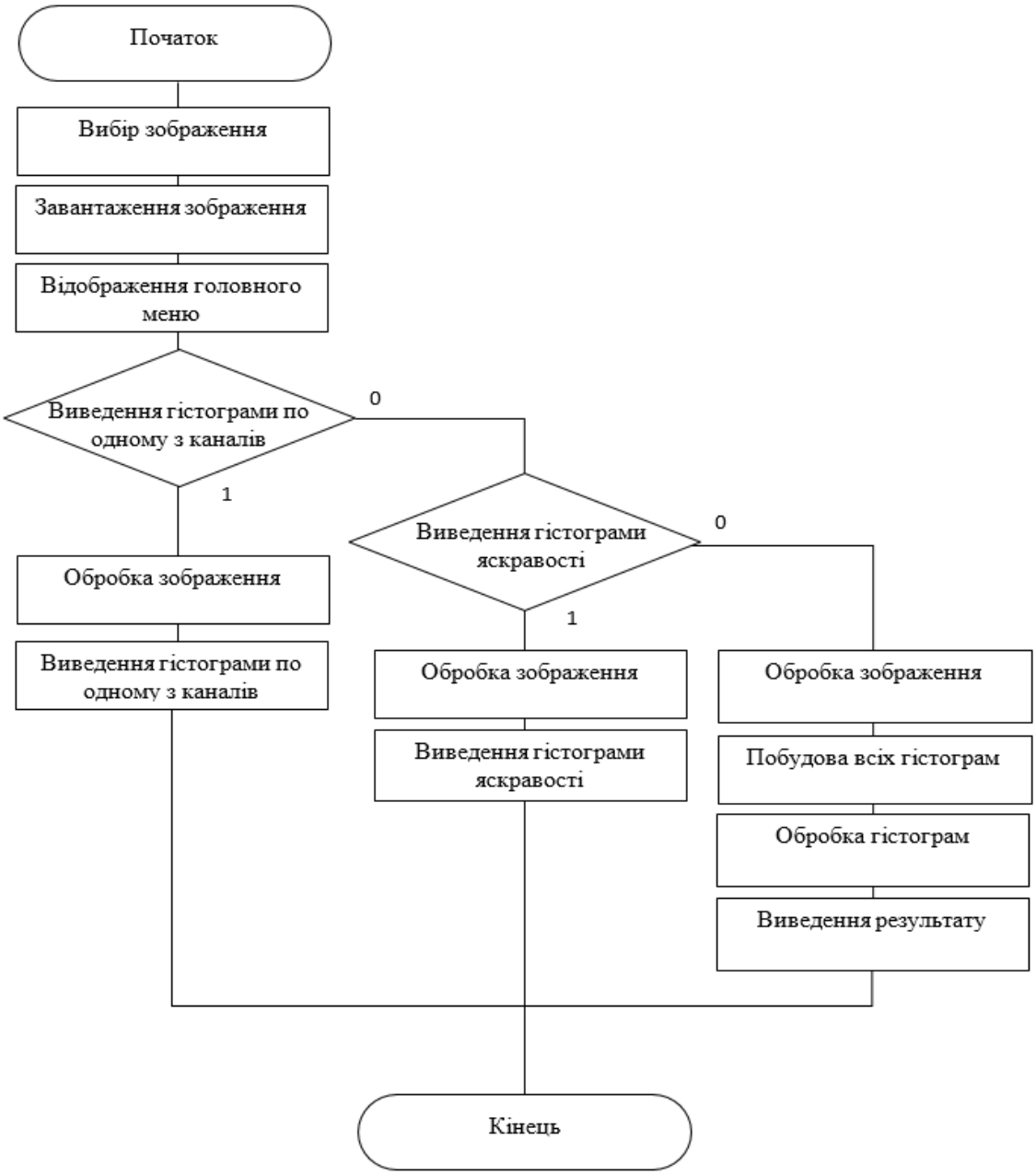

Рисунок Г.5 – Схема алгоритму роботи програмного модуля цифрової обробки зображень

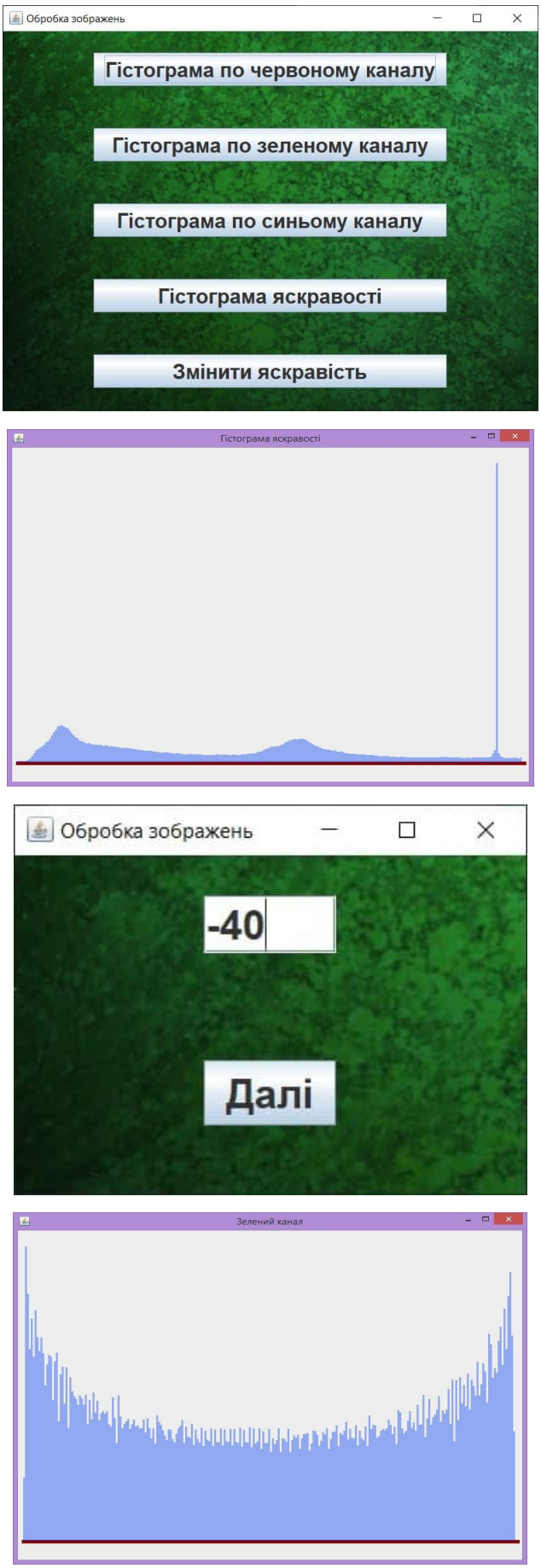

Рисунок Г.6 – Головні вікна програмного модуля цифрової обробки зображень