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 ПОШУКУ ДЕФЕКТІВ НА ЗОБРАЖЕННЯХ

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Коржанівський В.Ю.
(прізвище та ініціали)

Керівник: к.ф.-м.н., доцент кафедри КН
Шевчук О.Ф.
(прізвище та ініціали)

«07» 06 2024 р.

Рецензент: к.т.н., доцент кафедри АІІТ
Барабан М.В.
(прізвище та ініціали)

«07» 06 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)

«07» 06 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" 06 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Коржанівському Владиславу Юрійовичу

1. Тема роботи: Програмний модуль пошуку дефектів на зображеннях
керівник роботи: Шевчук Олександр Федорович, к.ф.-м.н., доцент
затверджені наказом вищого навчального закладу "11" 03 2024 року №__

2. Строк подання студентом роботи 27.05.2024

3. Вихідні дані до роботи :
Формати графічних файлів – JPEG,PNG;
Програмне забезпечення – Browser (Google Chrome);
Мова програмування – об'єктно-орієнтована.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
аналіз предметної області та існуючих рішень пошуку дефектів на зображеннях; постановка задачі та формування вимог до програмного модуля; розробка програмного модуля пошуку дефектів на зображеннях; програмна реалізація модуля пошуку дефектів на зображення; висновки; перелік використаних джерел; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
структура програмного модуля пошуку дефектів на зображеннях; схема загального алгоритму функціонування програмного забезпечення; 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 07.03.2024.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Приміт
		початок	закінчення	
1	Обгрунтування доцільності розробки програмного модулю пошуку дефектів на зображеннях	06.05.2024	07.05.2024	
2	Розробка програмного модулю пошуку дефектів на зображеннях	08.05.2024	09.05.2024	
3	Програмна реалізація модулю пошуку дефектів на зображеннях	13.05.2024	14.05.2024	
4	Аналіз функціонування програмного забезпечення	19.05.2024	19.05.2024	
5	Оформлення додатків	23.05.2024	26.05.2024	
6	Розробка інструкції користувача	26.05.2024	27.05.2024	
7	Оформлення матеріалів до захисту БДР	04.06.2024	06.06.2024	

Студент _____ Коржанівський
(підпис) (прізвище та і

Керівник проекту (роботи) _____ Шевчук О.Ф
(підпис) (прізвище та і

АНОТАЦІЯ

УДК 621.374.415

Коржанівський В.Ю. Програмний модуль пошуку дефектів на зображеннях.. Бакалаврська дипломна робота складається з 68 сторінок формату А4, на яких є 17 рисунків, список використаних джерел містить 21 найменування.

В роботі проаналізовано предметну область пошуку дефектів на зображенні. Було проаналізовано переваги та недоліки програм аналогів, що дозволяють проводити пошук дефектів на зображеннях. Було проаналізовано та описано класифікацію дефектів на зображеннях. Також було досліджено можливі методи усунення шуму. Створено алгоритм роботи програмного модуля та UML-діаграми діяльності та прецедентів. Програмно реалізовано на мові Python модуль пошуку дефектів на зображеннях та подальшого їх виправлення.

Ключові слова: дефекти, зображення, нейромережеві технології, аналіз дефектів, зниження шуму, Python.

ABSTRACT

Korzhanivskyi V.Yu. Software module for searching for defects in images. The bachelor's thesis consists of 68 pages of A4 format, on which there are 17 figures, the list of used sources contains 21 names.

The subject area of image defect search is analyzed in the work. The advantages and disadvantages of analog programs that allow searching for defects on images were analyzed. The classification of defects on the images was analyzed and described. Possible noise reduction methods were also investigated. The algorithm of the software module and the UML diagram of activities and precedents have been created. A module for searching for defects in images and their subsequent correction is implemented in Python.

Keywords: defects, images, neural network technologies, defect analysis, noise reduction, Python.

ЗМІСТ

ВСТУП

Актуальність теми. Цифрова фотографія є майже основним джерелом інформації нашого часу, за рахунок її високої пропускної спроможності для людського мозку. Процес обробки фотографій є невід'ємним етапом її сприйняття. Тому досить актуальним є програмне забезпечення засобів обробки цифрових фотографій [1]. На сьогодні існує достатньо велика кількість програмних засобів для обробки фотографій. Одними з найпотужніших і найвідоміших серед них є Adobe Photoshop Lightroom, Adobe Photoshop, Corel PaintShop Pro, Cyberlink Photo Director, ACDSee Photo Editor, GIMP, Paint.NET та інші.

Перераховані засоби є багатофункціональними, проте мають певні недоліки, серед яких: вартість продукту, складність для новачків, довготривалість запуску, нагромадженість інтерфейсу, великий об'єм програми на диску.

Обробка зображень вимагає необхідності роботи з великими наборами даних за короткі періоди часу в режимі реального часу. Традиційні засоби обробки не дозволяють вирішити проблему в повному обсязі. Це пов'язано з проблемою введення-виведення інформації в комп'ютер та послідовними алгоритмами обробки кожної точки зображення. Тому цей шлях, незважаючи на підвищення продуктивності універсальних комп'ютерів, є принципово непридатним. Існуючі традиційні електронні методи запису, зберігання та обробки зображень призводять до неоднорідності та складності комп'ютерних систем загалом. Тому необхідно знайти методи та інструменти, які будуть спрямовані на комплексний підхід до вирішення цієї проблеми.

Ці недоліки спричиняють необхідність розробки програмного модуля обробки цифрових фотографій, зокрема пошуку дефектів на зображенні, що матиме функціонал лише для базових автоматизованих корекцій фотографій, відповідно матиме простіший інтерфейс, займатиме менше місця та швидше запускатиметься. Програму планується випустити, як проект з відкритим програмним кодом, відповідно вона буде безкшотовною.

Об'єкт дослідження - це процес пошуку дефектів на зображеннях.

Предмет дослідження - це програмні засоби пошуку дефектів на зображеннях.

Мета дослідження - розширення функціональних можливостей програмного забезпечення пошуку дефектів на зображеннях.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналіз предметної області пошуку дефектів на зображеннях;
- аналіз методів боротьба з шумом на зображеннях;
- аналіз можливих варіантів штучних нейронних мереж;
- розробка алгоритму для пошуку дефектів на зображеннях;
- програмна реалізація модуля пошуку дефектів на зображеннях;
- тестування розробленого модуля та аналіз результатів.

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЮ ПОШУКУ ДЕФЕКТІВ НА ЗОБРАЖЕННІ

1.1 Аналіз сучасного стану пошуку дефектів на зображенні

На даний момент існує велика кількість обчислювальної техніки, яка зчитує різні показники згідно з її задачами. Наступним етапом після отримання даних є їх розшифровка та аналіз. На цьому етапі виникає проблема обсягу обробки даних, для вирішення якої використовуються спеціалізовані алгоритми, штучні нейронні мережі або навіть. В цьому розділі ми розглянемо сфери застосування обробки зображень, з чого зображення складаються та предметну область комп'ютерного зору.

Набувають поширення системи розпізнавання образів, які мають практичне використання у всіх сферах нашого життя. Головною проблемою таких систем є їх неточність, яка виникає внаслідок поганої якості зображення та його зашумленості. В такому випадку варто звернути увагу на методи, що аналізують та знижують кількість артефактів на зображенні.

Щоб довести актуальність теми, наведемо сфери практичного застосування методів обробки зображень та розпізнавання образів. Серед них:

- геофізичні спостереження. Це спостереження за поверхнею Землі та інших небесних об'єктів. Кількість потенційних споживачів подібної інформації постійно зростає, а перелік сфер застосування орбітальних знімків майже нескінченний. В багатьох випадках виникає потреба в розпізнаванні і класифікації об'єктів в режимі реального часу;

- застосування на транспорті. Розпізнавання автомобілів та їх номерних знаків, слідкування за транспортним рухом;

- розпізнавання літаків. Використовується для управління повітряним рухом та для систем ППО;

- розпізнавання рукописних знаків;

- корекція нерівномірного засвічення зображення;

- сегментація кольорових зображень;
- вимірювання кутів перетину;
- пониження шумів на зображеннях;
- реконструкція зображень;
- відновлення зображень;
- виявлення об'єктів на зображенні;
- витяг даних з тривимірних магніторезонансних зображень;
- ідентифікація предметів та людей;
- аналіз ознак об'єктів;
- виявлення руху у відеопотоці.

Комп'ютерне зір - це одна з найбільш затребуваних областей на сучасному етапі розвитку інформаційних технологій. Під комп'ютерним зором розуміють теорію і технологію створення штучних комп'ютерних систем, які можуть виробляти виявлення, стеження і класифікацію об'єктів. інформацію такі системи отримують із зображень, які можуть бути представлені безліччю форм: відео послідовність, зображення з різних камер або тривимірні дані з пристрою Kinect, відскановані зображення і багато інших форм [1].

Кожне зображення складається з набору пікселів. Піксель - це будівельний блок зображення. Якщо уявити зображення у вигляді сітки, то кожен квадрат в сітці містить один піксель, де точці з координатою (0, 0) відповідає верхній лівий кут зображення. Наприклад, уявімо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків і 300 стовпців. У сукупності в нашому зображенні є 400 * 300 = 120000 пікселів.

У більшості зображень пікселі представлені двома способами: у відтінках сірого і в колірному просторі RGB. У зображеннях у відтінках сірого кожен піксель має значення між 0 і 255, де 0 відповідає чорному, а 255 відповідає білому. А значення між 0 і 255 приймають різні відтінки сірого, де значення ближче до 0 темніші, а значення ближче до 255 світліші.

Кольорові пікселі зазвичай представлені в колірному просторі RGB (red, green, blue - червоний, зелений, синій), де одне значення для червоної компоненти, одне

для зеленої і одне для синьої. Кожна з трьох компонент представлена цілим числом в діапазоні від 0 до 255 включно, яке вказує як «багато» кольору міститься. Виходячи з того, що кожна компонента представлена в діапазоні [0,255], то для того, щоб представити насиченість кожного кольору, нам буде достатньо 8-бітного цілого беззнакового числа. Потім ми об'єднуємо значення всіх трьох компонент в кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонент має дорівнювати 255: (255, 255, 255). Тоді, щоб отримати чорний колір, кожна з компонент має дорівнювати 0: (0, 0, 0).

Область комп'ютерного зору почала інтенсивно вивчатися в 70-х роках минулого століття. Дослідження починалися з розвитку інших галузей науки. Багато з методів рішень завдань комп'ютерного зору все ещѐ знаходяться в стадії фундаментальних досліджень і носять приватний характер для вирішення прикладних задач. Але деякі методи, засновані на інших областях науки, стають все більш загальними. Вони лягають в основу створення великих систем, які можуть вирішувати складні завдання, наприклад, в галузі контролю якості сировини і готової продукції в процесах виробництва на підприємствах текстильної та легкої промисловості. З комп'ютерним зором пов'язані такі галузі науки, як автоматичне планування, розпізнавання образів, навчальні методи. В цьому випадку комп'ютерний зір розглядається як частина області штучного інтелекту. Значна частина комп'ютерного зору має справу з методами, які вимагають досконального розуміння процесу, в якому електромагнітне випромінювання вимірюється якимись датчиком зображення для отримання відеоданих. Це вже відноситься до області фізики, до розділів оптики, фізики твердого тіла, квантової механіки. деякі методи вивчення, розроблені в області комп'ютерного зору, зобов'язані своїм походженням нейробіології, особливо вивчення систем біологічного зору. Також багато методи обробки одновимірних сигналів можуть бути розширені для обробки двовимірних або багатовимірних сигналів в комп'ютерному зорі. Це область обробки сигналів. багато методи обробки зображень ґрунтуються на статистиці, методах оптимізації або геометрії. Деякі з досліджуваних питань можуть бути вивчені з чисто математичної точки зору. Великі роботи ведуться в області практичного

застосування комп'ютерного зору. Досліджуються питання оптимізації роботи алгоритмів для досягнення високої швидкості роботи систем комп'ютерного зору без істотного збільшення споживаних ресурсів. Комп'ютерне зір - це будь-яка форма обробки графічної інформації, яка може бути представлена як статичним зображенням (кадром), так і послідовністю кадрів (відеозаписом). Результатом комп'ютерного зору може бути видозмінене зображення або список значень деяких параметрів зображення (розмір об'єкта, його колір, орієнтація по відношенню до камери, швидкість і т.п.) Можна виділити три основні напрями досліджень і розробок в сфері комп'ютерного зору: розпізнавання образів, аналіз зображення, обробка зображення.

1.2 Огляд та аналіз існуючих рішень в предметній області пошуку дефектів на зображенні

Серед можливих аналогів можна розглянути фоторедактори. Для прикладу візьмемо Adobe Photoshop[2].

Adobe Photoshop — графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень і найвідомішим продуктом фірми Adobe. У наш час Photoshop доступний на платформах Mac OS X/Mac OS і Microsoft Windows.

Можливості. Photoshop головним чином призначений для редагування цифрових фотографій та створення растрової графіки. Особливості Adobe Photoshop полягають у багатому інструментарії для операції створення і обробки зображень, високій якості обробки графічних зображень, зручності й простоті в експлуатації, широких можливостях до автоматизації обробки растрових зображень, які базуються на використанні сценаріїв, механізмах роботи з кольоровими профілями, які допускають їх втілення в файли зображень з метою автоматичної корекції кольорових параметрів при виводі на друк для різних пристроїв, великому наборі команд фільтрації, за допомогою яких можна створювати найрізноманітніші художні ефекти.

Базові інструменти редагування дозволяють змінювати тон, насиченість зображення, обтинати його, накладати фотофільтри, виправляти перспективу тощо. Photoshop підтримує так звані шари — прозорі області зображення, на яких розміщуються елементи фотомонтажу, текст, геометричні фігури. Програма містить інструменти для роботи з текстом і нескладними фігурами, дозволяє малювати робочі контури, задавати текстам і фігурам стилі оформлення. Для роботи з окремими фрагментами зображення передбачені різні типи виділення: за фігурою, в режимі «малювання» зони виділення, за діапазоном кольорів тощо. Існують різноманітні фільтри для деформації та стилізації зображення, такі як фільтри розмиття, імітації різних художніх технік. Photoshop також містить інструменти для цифрового живопису, зокрема набори пензлів. Користувач може змінювати їх розмір, кут нахилу, колір. Підтримується встановлення сторонніх пензлів, стилів, шрифтів, палітр. Розширена версія програми Adobe Photoshop Extended призначена для більш професійного використання, а саме — при створенні фільмів, відео, мультимедійних проектів, тривимірного графічного дизайну та веб-дизайну, для роботи в галузях виробництва, медицини, архітектури, при проведенні наукових досліджень (рис. 1.1).

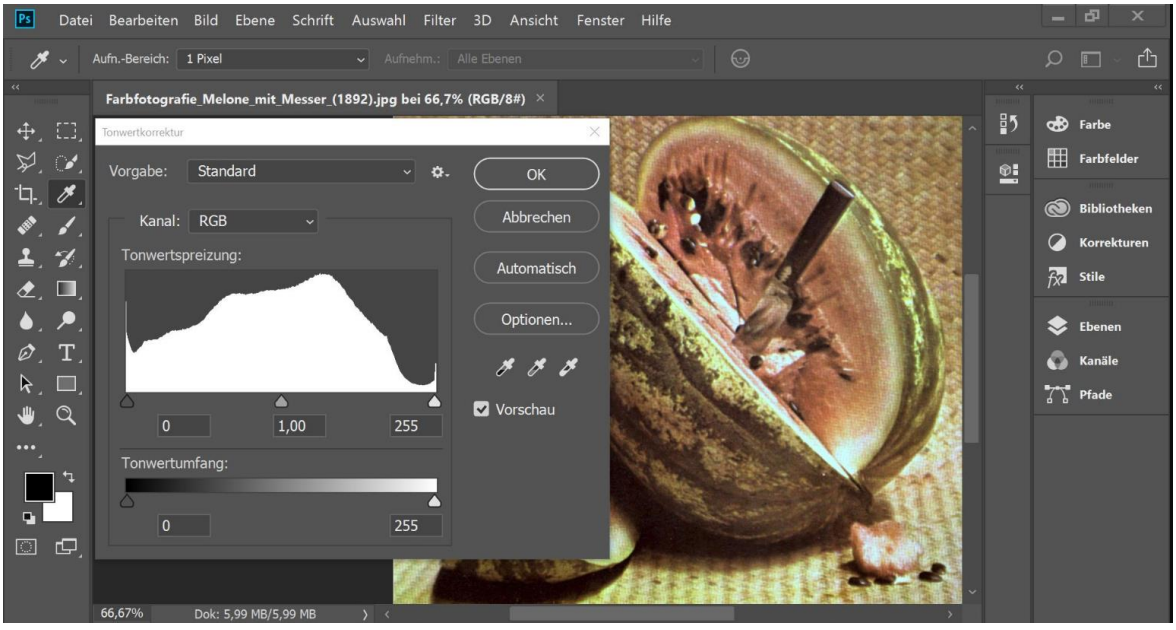

Рисунок 1.1 – Програма Adobe Photoshop

Даний аналог, згідно з поставленими нами вимогами має недостатній аналітичний апарат та недостатню кількість методів шумозниження.

Наступним аналогом буде Mathlab [3].

На сьогоднішній день система Matlab, зокрема пакет прикладних програм Image Processing Toolbox, є найбільш потужним інструментом для моделювання і дослідження методів обробки зображень. Він включає велику кількість вбудованих функцій, що реалізують найбільш поширені методи обробки зображень. Розглянемо основні можливості пакета Image Processing Toolbox.

Геометричні перетворення зображень. До найбільш поширених функцій геометричних перетворень відноситься кадрування зображень (imcrop), зміна розмірів (imresize) і поворот зображення (imrotate).

Аналіз зображень. Для роботи з окремими елементами зображень використовуються такі функції як imhist, impixel, mean2, corr2 і інші. Однією з найбільш важливих характеристик є гістограма розподілу значень інтенсивностей пікселів зображення, яку можна побудувати за допомогою функції imhist.

Поліпшення зображень. Серед вбудованих функцій, які реалізують найбільш відомі методи поліпшення зображень, виділимо наступні - histeq, imadjust та imfilter (fspecial). Гістограма зображення є однією з найбільш інформативних характеристик. На основі аналізу гістограми можна судити про яскравості зображення спотворене, тобто сказати про те, чи є зображення затемненим або освітлення. Відомо, що в ідеалі на цифровому зображенні в рівній кількості повинні бути присутніми пікселі з усіма значеннями яскравості, тобто гістограма повинна бути рівномірною. Перерозподіл яскравостей пікселів на зображенні з метою отримання рівномірної гістограми виконує метод еквалізації, який в системі Matlab реалізований у вигляді функції histeq.

Фільтрація зображень. Пакет Image Processing Toolbox володіє дуже потужним інструментарієм по фільтрації зображень. Серед безлічі вбудованих функцій, які вирішують завдання фільтрації зображень, особливо слід виділити fspecial, ordfilt2, medfilt2.

Сегментація зображень. Серед вбудованих функцій пакету Image Processing Toolbox, які застосовуються при вирішенні завдань сегментації зображень, слід виділити qtdecomp, edge і roicolor. Функція qtdecomp виконує сегментацію зображення методом розділення і аналізу однорідності що не перекриваються блоків зображення.

Морфологічні операції над бінарними зображеннями. Система Matlab володіє досить потужним інструментарієм морфологічної обробки бінарних зображень. Серед основних операцій виділимо наступні - ерозія, нарощування, відкриття, закриття, видалення ізольованих пікселів, побудова скелета зображення і т.п.

Даний аналог дуже гнучкий та ефективний в плані обробки зображень, і єдиною невідповідністю до нашої задачі є його мала автоматизація. Отже, даний засіб можна розглядати як інструмент наших досліджень.

Наступний аналог – сервіс Imatest [4].

Imatest LLC - це компанія, яка виробляє програмне забезпечення для тестування якості зображення, обладнання та діаграми випробувань. Imatest був заснований фотографом / інженером Норманом Кореном у Боулдері, штат Колорадо, у 2004 році для розробки програмного забезпечення для тестування якості зображення цифрової камери.

За допомогою програмного забезпечення Imatest можна проаналізувати різноманітні фактори якості зображення, включаючи гостроту (чіткість зображення), кольорову реакцію, шум зображення, динамічний діапазон, тональну реакцію, відблиск об’єктива, спотворення об’єктива, віньєтування об’єктива, реакцію текстури та кольоровий муар.

До програмних продуктів Imatest належать Imatest Master: для інженерів R&D, Imatest IT: для автоматизації тестів та Imatest Studio: для фотографів.

Тестові діаграми Imatest включають SFRplus, eSFR ISO, SFRreg, розлиті монети та цілі динамічного діапазону з 36 патчами.

Imatest було прийнято в широкому діапазоні галузей, що використовують вбудовані цифрові системи візуалізації, включаючи мобільну візуалізацію (телефони з камерою), автомобільну, медичну візуалізацію, аерокосмічний та машинний зір, а

також публікації / огляди, а також наукові, культурні та науково-дослідні установи . Програмне забезпечення Imatest є посиланням на дослідження, мистецтво та промисловість.

Imatest є членом Міжнародної організації зі стандартизації та Інституту інженерів електротехніки та електроніки, вносячи вклад та застосовуючи стандартизовані методи аналізу якості зображення (рис. 1.2).

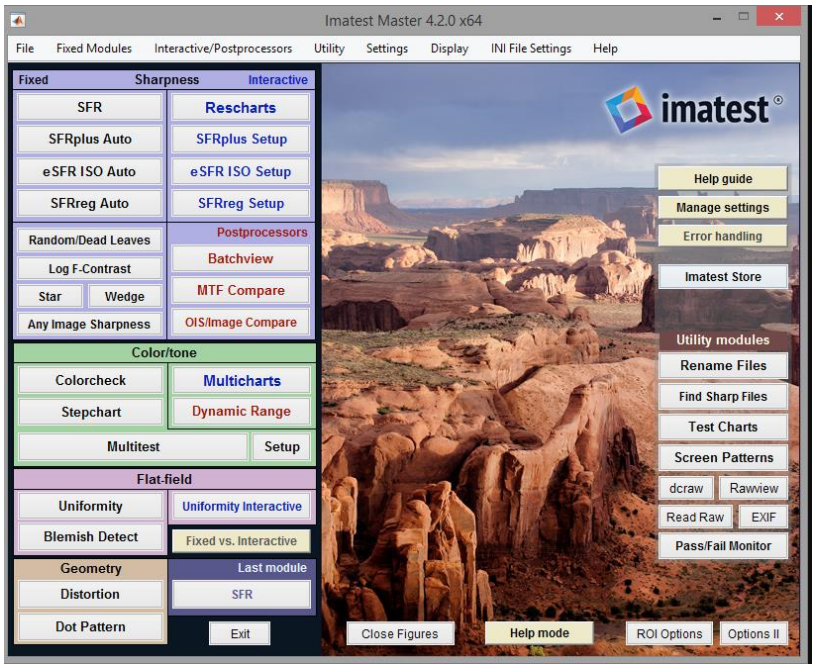

Рисунок 1.2 – Сервіс Imatest

Серед багатьох інших програм Imatest використовувався для регулювання зображень із різним рівнем різкості до стандартного рівня різкості та для порівняння роздільної здатності цих налаштованих зображень. Imatest також перевіряє точність кольорів, лінійність тональної шкали, спотворення зображення, світле падіння та діапазон друку.

Даний аналог є платним, проте має 14-денну безкоштовну версію.

Наступним аналогом є ImageJ [5].

ImageJ - програма з відкритим вихідним кодом для аналізу і обробки зображень. Написана на мові Java співробітниками National Institutes of Health і поширюється без ліцензійних обмежень як суспільне надбання. Відкритий API

дозволяє гнучко нарощувати функціональність за рахунок додаткових плагінів, а вбудована макромова - автоматизувати складні повторювані дії. ImageJ широко застосовується в біомедичних дослідженнях, астрономії, географії та інших дисциплінах, пов'язаних з аналізом зображень, в якості альтернативи пропрієтарного ПЗ.

Модулі сторонніх розробників охоплюють широке коло завдань аналізу і обробки зображень: дозволяють проводити тривимірну візуалізацію в діапазоні від клітин до рентгенологічних зображень, автоматичні порівняння аж до створення автоматизованих систем вивчення, наприклад, в гематології. Архітектура плагінів ImageJ і вбудована в програму система розробки робить цю платформу вельми популярною для роботи і викладання аналізу і обробки зображень.

ImageJ може працювати як онлайн аплет, завантажувати додаток. Використання оригінальну програму можна в усіх операційних системах, для яких існує Java Virtual Machine версії 1.4 або більш пізньої: Microsoft Windows, Mac OS, Mac OS X, Linux і Sharp Zaurus PDA. Вихідний код ImageJ також знаходиться у вільному доступі (рис. 1.3).

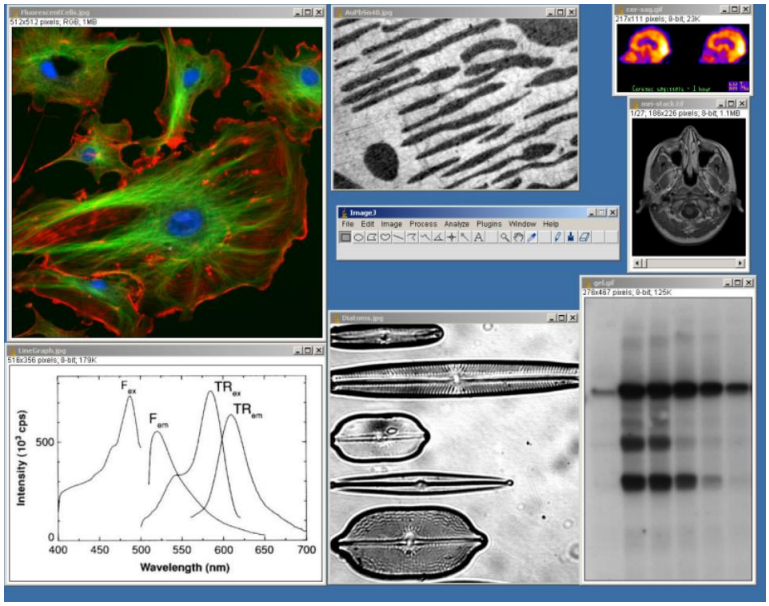

Рисунок 1.3. – Сервіс Imagej

ImageJ дозволяє відображати, редагувати, аналізувати, обробляти, зберігати і друкувати 8-бітові, 16-бітові та 32-бітові зображення. Програма може читати багато форматів зображень, такі як TIFF, PNG, GIF, JPEG, BMP, DICOM, FITS, а також raw формати. ImageJ підтримує стеки - серії зображень, які об'єднані в одному вікні, а багатопотокові трудомісткі операції можуть виконуватися на багатопроцесорних системах в паралельному режимі. У ImageJ можна обчислювати площі, статистичні показники піксельних значень різних виділених областей інтересу на зображеннях, які виділені вручну або за допомогою порогових функцій. Програма може вимірювати відстані і кути. Вона може створювати гістограми щільності і малювати профілі ліній. ImageJ підтримує стандартні функції обробки зображень, такі як логічні і арифметичні операції між зображеннями, маніпуляції з контрастністю, згортки, фур'є-аналіз, підвищення різкості, згладжування, виявлення меж і медіанний фільтр. Програма дозволяє проводити різні геометричні перетворення, такі як масштабування, поворот або віддзеркалення. Програма підтримує будь-яку кількість одночасно використовуваних зображень, обмеження пов'язане тільки з об'ємом доступної пам'яті.

1.3 Формування вимог до розроблюваного програмного забезпечення

Необхідно побудувати систему, що визначає наявність дефектів на зображеннях та здійснює їх аналіз. Ця система повинна бути обладнана простим і зрозумілим інтерфейсом, який дозволить користувачеві легко взаємодіяти з нею. Система повинна мати можливість автоматично розпізнавати різні типи дефектів на зображеннях. Це можуть бути подряпини, тріщини, плями та інші аномалії, які відрізняються від нормального стану зображення. Використання сучасних алгоритмів комп'ютерного зору та машинного навчання для точного визначення та класифікації дефектів. Інтерфейс повинен бути інтуїтивно зрозумілим і зручним для користувачів, незалежно від їх технічної підготовки. Надавати можливість завантаження зображень користувачами для подальшого аналізу. Візуалізація результатів аналізу у зрозумілому форматі, що може включати виділення дефектів

на зображенні, таблиці та графіки з даними про дефекти. Загалом, система повинна не тільки ефективно виявляти та аналізувати дефекти на зображеннях, але й бути зручною та доступною для користувачів, забезпечуючи високу якість і надійність результатів.

На вхід система повинна приймати зображення, на якому відображений дефект.

Програмне забезпечення повинно:

- Приймати зображення на вхід.
- Виявляти та аналізувати зображення.
- Реалізувати декілька методів зниження шумів на зображенні.
- Обирати метод, який застосовувати.
- Редагувати зображення згідно з обраним методом та виводити результат.

1.4 Висновок

В цьому розділі проаналізовано предметну область пошуку дефектів на зображенні. Було проаналізовано переваги та недоліки програм аналогів, що дозволяють проводити пошук дефектів на зображеннях. Було сформовано вимоги до розроблюваного програмного забезпечення та здійснено постановку задачі.

2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ПОШУКУ ДЕФЕКТІВ НА ЗОБРАЖЕННЯХ

2.1 Класифікація дефектів на зображеннях

Для аналізу дефектів, для початку потрібно їх класифікувати. Класифікація дефектів:

- Загальна різкість. Низька різкість, низька текстурованість, низька деталізація, розмитість. Ефект накладеності, муаровий ефект. Оцінка різкості за допомогою градієнтів. Це найпростіший і основний показник чіткості. Хоча існує багато чудових інших методів, навіть цей примітивний алгоритм оцінки виявляє гідне практичне використання. Файл зчитує зображення і ітеративно згладжує його все більше і більше, щоб показати, як зменшується різкість. Метод перевірений на середню та нерізку (різку) фільтрацію.

- Пов'язані з лінзами. Відблиск об'єктива, огріх лінзи, лінзовий туман, хроматична аберація, спотворення, віньєтування.

- Автоекспозиція. Неточність автоматичної експозиції, спалахи в автоекспозиції, повільна конвергенція автоекспозиції, мерехтіння лампи.

Експозиція - кількість актинічного випромінювання, яке містить світлочутливі елементи. Для видимого випромінювання може бути розрахована як добуток освітленості на витримку, протягом якої світло впливає на світлочутливий елемент: матрицю або фотоемульсію.[6]

Для видимого випромінювання експозиція виражається в (люкс-секунда). Термін також вживається стосовно до самого процесу експонування світлочутливого елемента, і в інших областях, пов'язаних з опроміненням світлочутливих шарів: фотолітографії, радіографії і т. П. При експонуванні змінюються фізико-хімічні або електричні властивості світлоприймачів. Наприклад, в галогенідах срібла відбувається відновлення металевого срібла.

У більшості пристроїв для запису зображення експозиція залежить від чинного відносного отвору об'єктива і витримки. Ці значення називаються

експозиційними параметрами. У фотоапаратах витримка регулюється затвором, а в кінокамера - обтюратором. При кінозйомки витримка залежить від частоти зміни кадрів і кута розкриття обтюратора (коефіцієнта обтюрациі), тому експозиція регулюється, головним чином, діафрагмою, що змінює відносний отвір об'єктиву і, в кінцевому підсумку - освітленість. У телекамери і відеокамерах, що оснащувалися вакуумними передають трубками, експозиція могла регулюватися тільки діафрагмою, оскільки витримка завжди точно відповідала тривалості телевізійного поля. Сучасні відеокамери з напівпровідниковими матрицями мають можливість регулювати час зчитування кадру, змінюючи витримку. При фотографуванні експозиція може регулюватися в більш широких межах за рахунок витримки, значення якої можуть вимірюватися хвилинами і годинами, на відміну від кінознімального апарату і відеокамери, що допускають при стандартній кадрової частоті витримку не довше 1/50 секунди.

Крім діафрагми для регулювання освітленості можуть застосовуватися світлофільтри, що поміщаються перед об'єктивом, або за ним. Деякі камери спеціально оснащуються вбудованими нейтрально-сірими фільтрами, в потрібний момент всувають в оптичну систему, іноді між лінзами. Такий спосіб особливо актуальне при кіно- або відеозйомки, компенсуючи труднощі зменшення витримки. У випадках, коли експонування відбувається без застосування об'єктива (наприклад, при контактного друку), освітленість може регулюватися інтенсивністю джерела випромінювання. У деяких процесах, пов'язаних з експонуванням, витримка регулюється часом роботи джерела випромінювання, наприклад, при фотодруку або в фотолітографії. У кінокопіювальних апаратах з безперервним рухом кіноплівки експозиція задається шириною друкованого вікна, і може регулюватися яскравістю друкуючої лампи і швидкістю переміщення плівок. У кінокопіювальних машинах проміжної друку експозиція регулюється за допомогою світлового паспорта.

При фотографуванні із застосуванням електронних спалахів експозиція регулюється діафрагмою об'єктива і тривалістю імпульсу, оскільки його інтенсивність не піддається регулюванню. Найпростіші фотоспалахи, в яких відсутня регулювання тривалості імпульсу, дають можливість управління

експозицією тільки діафрагмою. У деяких сучасних видах обладнання (наприклад, SIMD-матриці, камери світлового поля і Foveon X3) так само, як і в багатошарових плівках, уявлення про експозицію (а також про витримку і діафрагму) можна відносити не тільки до фотоматеріалів або пристрою в цілому, але і до окремих його елементів (шарів) і сполученням елементів.

- Awb(баланс білого). Неточність в awb, спалахи в awb повільна конвергенція.

Баланс білого кольору (також коротко званий баланс білого) - один з параметрів методу передачі кольорового зображення, що визначає відповідність кольорової гами зображення об'єкта колірній гамі об'єкту зйомки [7].

Зазвичай вживається як змінна характеристика фотографічного процесу, фотоматеріалу, систем кольорового друку і копіювання, телевізійних систем і пристроїв відтворення графічної інформації (наприклад, моніторів).

Баланс білого, корекція балансу білого, настройка білої точки або корекція - технологія корекції кольорів зображення об'єкта до тих кольорів, в яких людина бачить об'єкт в природних умовах (об'єктивний підхід), або до тих кольорів, які представляються найбільш привабливими (суб'єктивний підхід). Аналог біологічного механізму - кольоростійкість.

- Фільтрування шумів. Сліди/гостинг, висока зашумленність.

Рівень шуму є важливим параметром для багатьох програм обробки зображень. Наприклад, продуктивність алгоритму шумозаглушення зображення може значно погіршитися через погану оцінку рівня шуму. Більшість існуючих алгоритмів шумозаглушення просто припускають, що рівень шуму відомий, що значною мірою перешкоджає їм практичному використанню. Більше того, навіть при заданому справжньому рівні шуму, ці алгоритми шумозаглушення все ще не можуть досягти найкращої продуктивності, особливо для сцен із насиченою текстурою. У цій роботі ми пропонуємо алгоритм оцінки рівня шуму на основі виправлення та пропонуємо налаштовувати параметр рівня шуму відповідно до складності сцени. Існує підхід, що включає процес вибору патчів низького рангу без високочастотних компонентів з одного шумного зображення. Вибір базується на

градієнтах патчів та їх статистиці. Потім рівень шуму оцінюється за обраними плямами за допомогою аналізу основних компонентів. Оскільки справжній рівень шуму не завжди забезпечує найкращі характеристики алгоритмів шумозаглушення, ми додатково налаштовуємо параметр рівня шуму для сліпого шуму. Експерименти демонструють, що як точність, так і стабільність перевершують сучасний алгоритм оцінки рівня шуму для різних сцен та рівнів шуму.

- Колір [8]. Помилка кольору/точність сатурації, артефакти кольору, поганий контраст/гамма.

Контраст - в оптиці (сенситометрії і фотометрії) різниця в характеристиках різних ділянок зображення, здатність фотографічного матеріалу або оптичної системи відтворювати цю різницю, а також характеристика чутливості очі (зорової системи) щодо яскравості і кольору.

Контрастність (також в різних контекстах вживається і саме слово контраст і коефіцієнт контрасту) - ступінь контрасту, найчастіше виражається безрозмірною величиною, відношенням або логарифмом відносності.

2.2 Можливості попередньої обробки для покращення зображення

Шумопониження під час відтворення служить для поліпшення візуального сприйняття, проте можливе застосування в медицині для збільшення чіткості зображення на рентгенограмах, як передобробка для подальшого розпізнавання і в інших випадках.

Цифровий шум може виявлятися в двох формах:

- шум світності (градації сірого) - зображення виглядає зернистим або плямистим.

- кольоровий шум - зазвичай у вигляді кольорових артефактів на зображенні.

При цифровій обробці зображень використовується просторове шумопонижения.

Адаптивна фільтрація - лінійне усереднення пікселів по сусідніх. Найпростіша ідея видалення шуму - усереднювати значення пікселів в просторової околиці. Для

кожного пікселя аналізуються сусідні для нього пікселі, які розташовуються в деякому прямокутному вікні навколо цього пікселя. Чим більше узятий розмір вікна, тим сильніше відбувається усереднення. Найпростіший варіант фільтрації - в якості нового значення центрального пікселя брати середнє арифметичне всіх тих його сусідів, значення яких відрізняється від значення центрального не більше ніж на деякий поріг. Чим більше величина цього порога, тим сильніше відбувається усереднення [9].

Замість середнього арифметичного сусідів можна брати їх зважену суму, де ваговий коефіцієнт кожного сусіднього пікселя залежить або від відстані в пікселях від нього до центрально пікселя, або від різниці їх значень.

Ці алгоритми дуже прості, але вони не дають позитивного результату.

Цікава модифікація цього методу була запропонована Де Хаан. Він запропонував в якості значення центрального пікселя також брати зважену суму сусідніх пікселів, тільки сусідів брати не підряд, а через один або два пікселя. Стверджується, що при такому підході вдається придушити низькочастотний шум, який помітніше на око, ніж високочастотний.

Медіанна фільтрація - це стандартний спосіб придушення імпульсного шуму. Для кожного пікселя в деякому його оточенні (вікні) шукається медіанне значення і присвоюється цьому пікселю. Визначення медіанного значення: якщо масив пікселів впорядкувати за їх значенням, медіаною буде серединний елемент цього масиву. Розмір вікна відповідно повинен бути непарною, щоб цей серединний елемент існував. Однак в чистому вигляді медіанний фільтр розмиває дрібні деталі, величина яких менше розміру вікна для пошуку медіани, тому на практиці практично не використовується [10].

Математична морфологія - шумозаглушення можна також здійснювати з використанням двох основних морфологічних операцій: звуження (erosion) і розширення (dilation), а також їх комбінацій - закриття (closing) і розкриття (opening). Розкриття (спочатку звуження, потім розширення) прибирає виступи на межі об'єктів, а закриття (спочатку розширення, потім звуження) заповнює отвори всередині і на межі [11].

Можна запропонувати наступний алгоритм. Спочатку по вихідного зображення I обчислюється нове зображення I ', рівне напівсума відкриття-закриття і закриття-відкриття вихідного зображення. Отримаємо згладжені зображення, яке не містить шуму. Тоді зображення D, однакову різниці I і I ', буде містити весь шум і всі ті деталі вихідного зображення, розмір яких менше розміру структурного елементу, застосованого при морфологічних операціях. Припускаючи, що амплітуда у шуму менше, ніж у деталей, обнулив в D все значення, менші деякого порога, і знову складемо з I '.

Зображення, оброблені цим методом, виглядають дещо штучним, тому для обробки фотореалістичних зображень він не підходить, хоча, наприклад, для анімації може виявитися дуже корисним.

Розмивання Гауса - це метод фільтрації зображення за допомогою функції Гауса, який призводить до розмивання зображення. Даний ефект широко використовується в графічних програмах, як правило, для зменшення зашумленності зображенні і зниження деталізації. Візуальний ефект цієї фільтрації розмивання аналогічний погляду на зображення крізь напівпрозорий екран, істотно відрізняючись від ефекту боке, який можна отримати за допомогою несфокусованого об'єктива або тіні об'єкта при звичайному освітленні.

Розмивання Гауса - це згортка зображення з функцією де параметр s задає ступінь розмиття, а параметр A забезпечує нормування. Фактично, це те ж усереднення, тільки піксель змішується з оточуючими за певним законом, заданому функцією Гауса. Матричний фільтр, порахований за вказаною формулою, називається гауссіаном; чим більше його розмір, тим сильніше розмиття (при фіксованому s). Поблизу кордонів (контурів на зображенні) такий фільтр застосовувати не можна, щоб не змазати деталі зображення. Як наслідок уздовж кордонів залишається зашумлений контур.

Можна трохи модифікувати цей метод для кращої адаптації до кордонів: шукати в кожному вікні найкраще напрямок розмиття (наявність кордону), обчислюючи похідні за напрямками, і застосовуючи в даному вікні спрямований

Гауссіан уздовж знайденої кордону. В результаті розмиття буде проводитися вздовж кордонів зображення, і багатого на перешкоди контуру не буде.

Методи на основі дискретного вейвлет-перетворення. Вейвлет-перетворення - це інструмент багатомасштабного аналізу. Стосовно до області шумозаглушення воно дозволяє видаляти шум з зображення, не зачіпаючи значно межі і деталі. Також воно дозволяє ефективно гасити шуми зі спектрами, відмінними від білого [12].

Звичайне пряме одномірне дискретне вейвлет-перетворення (ДВП) - це ітераційне застосування низькочастотного і високочастотного фільтрів з подальшим видаленням кожного другого елементу (проріджуванням) до низькочастотного сигналу, що отримується на виході. В результаті низькочастотної фільтрації виходить наближення вихідного сигналу, в результаті високочастотної - деталізує інформація про вихідний сигнал, а отримані значення високочастотного сигналу називаються вейвлет-коефіцієнтами. Зворотне ДВП складається з ітераційного застосування зворотних фільтрів до високочастотних і низькочастотних коефіцієнтами з відновленими другими елементами (їх значення приймаються за 0) та їх складання. Пряме перетворення називається аналізом, а зворотне - синтезом. Пара фільтрів, що беруть участь в перетворенні - вейвлетного базисом.

Метод головних компонент (МГК) - дозволяє виділити структуру в багатовимірному масиві даних і застосовується в основному для розпізнавання або для стиснення зображень. В області шумозаглушення цей підхід є досить новим і мало дослідженим. Працює він найкраще для зображень з білим гауссівським шумом [13].

Метод головних компонент полягає в знаходженні таких базисних векторів досліджуваного багатовимірного простору, які б найкращим чином відображали розташування деяких вихідних даних в цьому просторі (характеризували ці дані). Коротко опишемо алгоритм побудови таких векторів.

Нехай задано N-мірний простір і M векторів в ньому. Під набором даних будемо розуміти вектор, складений з і-ї координати всіх векторів. Отже, нехай у нас є N наборів даних, відповідних N вимірам. Спочатку ми будуємо матрицю ковариаций для цих наборів і шукаємо її власні вектора і власні значення.

Ортонормований базис з N власних векторів і буде шуканим базисом, максимально наближає вихідні дані, іншими словами, що відображає їх структуру в N-вимірному просторі. Причому, чим більше власне значення, тим сильніше відображає характер розташування даних відповідний власний вектор.

Власний вектор, відповідний максимальному власному значенню, називається головною компонентою і визначає основний напрямок, уздовж якого розташовані дані. З іншого боку, власні вектора, для яких відповідне власне значення мало, практично ніяк не відображають характер розташування даних. Виключивши з нового базису власні вектора з найменшими власними значеннями, можна зменшити розмірність вихідних даних без особливих втрат для них. Таким чином, МГК забезпечує найкраще наближення вихідних даних мінімальним числом базисних векторів. На зменшенні розмірності і грунтуються алгоритми стиснення. Тепер розглянемо, як застосувати цей метод для придушення шуму.

Спочатку все зображення розбивається на блоки. Блоки обробляються незалежно, тому вони повинні розташовуватися з невеликим перекриттям, щоб уникнути артефактів блочності при їх стикування. Пікселі блоку NxN утворюють вектор довжини N2.

Далі, навколо кожного блоку розглядається деякий вікно, з якого вибираються всілякі блоки того ж розміру, що і центральний. Вектора, відповідні цим блокам укупі з центральним, утворюють вихідний набір даних, для якого застосовується описаний вище алгоритм МГК. Будується матриця коваріацій, шукаються власні вектора і власні значення, і потім проводиться сортування векторів по спадаючій власних значень. Шумозаглушення здійснюється шляхом зменшення довжин власних векторів і / або їх обнулення в залежності від величини відповідних власних значень. Приклад адаптивного алгоритму придушення величин власних векторів можна подивитися в.

Відновлення вихідного зображення з урахуванням зроблених змін проводиться в два етапи: спочатку вихідні дані наводяться до нового базису з власних векторів (якщо частина власних векторів була обнулена, то в результаті вийдуть дані меншої розмірності), потім назад до колишнього базису. При цьому

якщо власні вектора не змінювалися, відновлені дані будуть точно збігатися з вихідними.

Отже, як же зменшення довжин власних векторів впливає на придушення шуму? Справа в тому, то знайдені таким способом вектора будуть відповідати основним напрямам візерунка (кордонів, деталей) на даній області зображення. Оскільки шум розподілений по зображенню хаотично, без будь-якої системи, йому будуть відповідати власні вектора з маленькими власними значеннями. Пригнічуючи їх, ми будемо зменшувати амплітуду значень пікселів в цьому напрямку, а отже, придушувати і шум. Саме тому метод головних компонент добре справляється лише з білим шумом: у шуму не повинно бути ніякої структури, інакше МГК прийме його за візерунок і не придушить.

Найкраще метод головних компонент працює на текстурованих областях, навіть якщо текстура дуже дрібна. Після придушення шуму методом головних компонент текстура залишиться як і раніше чіткої, чого не можна сказати про всіх перерахованих вище методах.

Анізотропна дифузія - основна ідея даного підходу полягає в наступному. Яскравість кожного пікселя інтерпретується як значення температури в даній точці зображення, таким чином, всі зображення представляється у вигляді карти температур. Шумозаглушення проводиться шляхом вирівнювання температур (фактично, інтенсивностей пікселів) за допомогою моделювання процесу теплопереносу.

Так як на кордонах деталей зображення, як правило, відбувається значна зміна яскравості, коефіцієнт теплопровідності, який визначається за вказаною вище формулою, буде невеликим і, як наслідок, перенесення <тепла> через кордон буде мінімальним. Іншими словами, розмиття кордонів не відбудеться. На рівних галасливих областях, навпаки, зміна яскравості пікселів незначно, тому такі області будуть добре згладжені.

За своєю суттю даний метод дуже нагадує гауссівське розмиття (усереднення по сусідах), тому докладніше розглядати його не будемо.

2.3 Використання нейронної мережі для вирішення задачі пошуку дефектів на зображеннях

Нейрон - це обчислювальна одиниця, яка отримує інформацію, виробляє над нею прості обчислення і передає її далі. Сигнали, що надходять на вхід нейрона від інших нейронів, називаються синапсами. Кожен синапс характеризується вагою, завдяки якому вхідна інформація змінюється при передачі від одного нейрона до іншого. Чим більше вага синапсу, тим більший вплив на результат надає інформація, яка надходить з відповідного йому нейрона [14].

Функція активації - це функція, що здійснює нормалізацію вихідних даних і видає дійсне число в потрібному діапазоні. В вихідних шарах можливе використання простої лінійної функції $y = z$ для вирішення завдання регресії, коли нейронна мережа повинна повертати довільне дійсне число. Для успішного процесу навчання функція активації повинна бути диференційована.

Нейрони об'єднуються в шари, які утворюють нейронну мережу. Повнозв'язна нейронна мережа складається з вхідного шару, який отримує інформацію, одного або декількох прихованих шарів, які її обробляють, і вихідного шару, який видає підсумковий результат роботи мережі. При ініціалізації нейронної мережі значення ваг розставляються в випадковому порядку. Процес навчання штучної нейронної мережі спрямований на виявлення закономірностей, характерних для вхідних даних. Під час тренування мережі значення ваг змінюються таким чином, щоб вона видавала необхідні результати для тренувального набору вхідних даних зі збереженням узагальнюючої здатності для самостійної роботи з новими даними.

Згорткові нейронні мережі шукають найкращі результати в областях розподілення образів за порівнянням з багатошаровим персептроном, так як вони містять компоненти, спеціально створені для роботи із зображеннями. Завдякия цим компонентам згорткові нейронні мережі навчають дворівневу топологію зображень та кольорову схему, а також забезпечують часткову стійкість до змін величини, поворотам та іншим пошукам. Згорткова нейронна мережа є з різних видів шарів: згорткових шарів, шарів підвибірки и повнозв’язних шарів.

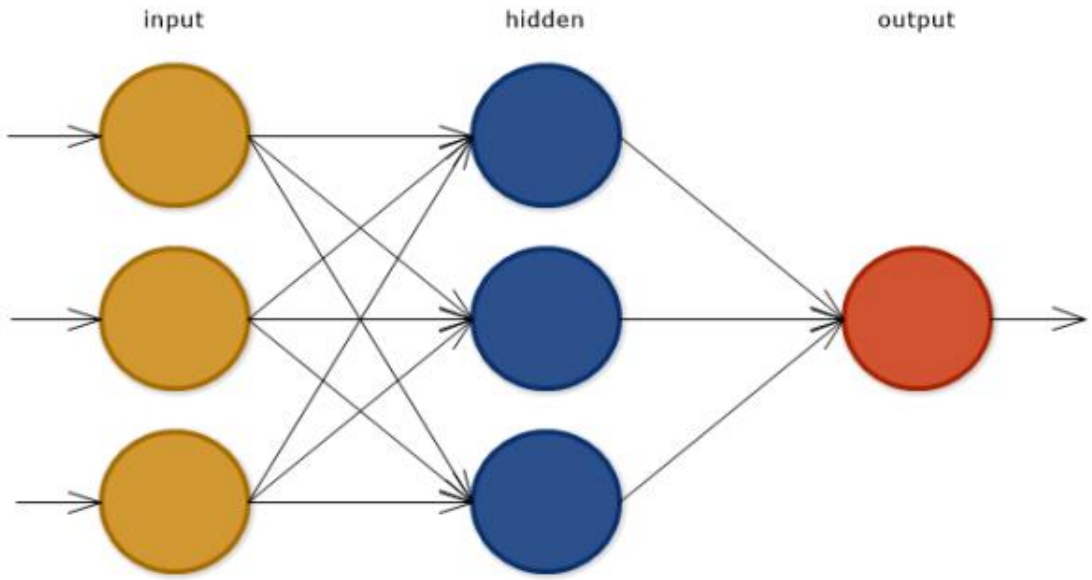

Рисунок 2.1. – Схема повнозв’язної нейронної мережі

Шари згортки та підвибірки створені для обробки зображень та виведення інформації з них. Вони формують вхідну векторну ознаку для багатошарового персептрона, який формує образотворчі зображення із зображеннями та видає необхідний результат.

Для вирішення поставленого завдання необхідно побудувати модель машинного навчання. Ця модель повинна розділяти зображення на дефектні, тобто вирішувати завдання класифікації. Для успішної класифікації необхідно підібрати архітектуру моделі і навчити її на вихідних даних. Вихідні дані подаються у вигляді двох векторів однакового розміру X і Y, де X містить зображення, а Y - мітки класів, Кожному елементу вибірки X відповідає один елемент вибірки Y. Для більшої стійкості моделі була проведена нормалізація зображень, а саме поділ значень пікселів на 255.

2.4 Розробка алгоритму пошуку дефектів на зображеннях

Алгоритм пошуку дефектів на зображеннях зазвичай включає кілька основних етапів: попередня обробка зображень, виявлення контурів або аномалій, класифікація дефектів та кінцева обробка результатів. Нижче наведений загальний алгоритм, який можна використовувати для цієї задачі:

1. Попередня обробка зображень. Цей етап включає підготовку зображень для подальшого аналізу, включаючи нормалізацію, фільтрацію та корекцію контрасту. Перетворення кольорових зображень у градації сірого, якщо зображення кольорове, щоб зменшити складність аналізу. Застосування фільтрів для зменшення шуму, наприклад, Гауссове розмиття. Підвищення контрасту зображення для кращого виділення дефектів (наприклад, використовуючи гістограмне вирівнювання).

2. Виявлення контурів або аномалій. На цьому етапі використовуються методи виділення контурів або аномалій на зображенні. Метод порогового значення застосовує порогове значення для виділення контурів об'єктів. Метод Canny використовується для виявлення контурів.

3. Класифікація дефектів. На цьому етапі використовуються моделі машинного навчання або інші методи для класифікації виявлених дефектів. Використання CNN (Convolutional Neural Networks) для класифікації різних типів дефектів. Використання традиційних методів класифікації SVM або інші класифікатори.

4. Кінцева обробка результатів. На цьому етапі обробляються результати аналізу для виведення користувачеві. Накладання контурів на оригінальне зображення: виділення дефектів на зображенні.

Схема алгоритму пошуку дефектів на зображеннях наведена на рисунку 2.2.

Діаграма діяльності програмного модуля пошуку дефектів на зображеннях наведена на рисунку 2.3.

UML-діаграма прецедентів програмного модуля пошуку дефектів на зображеннях наведена на рисунку 2.4.

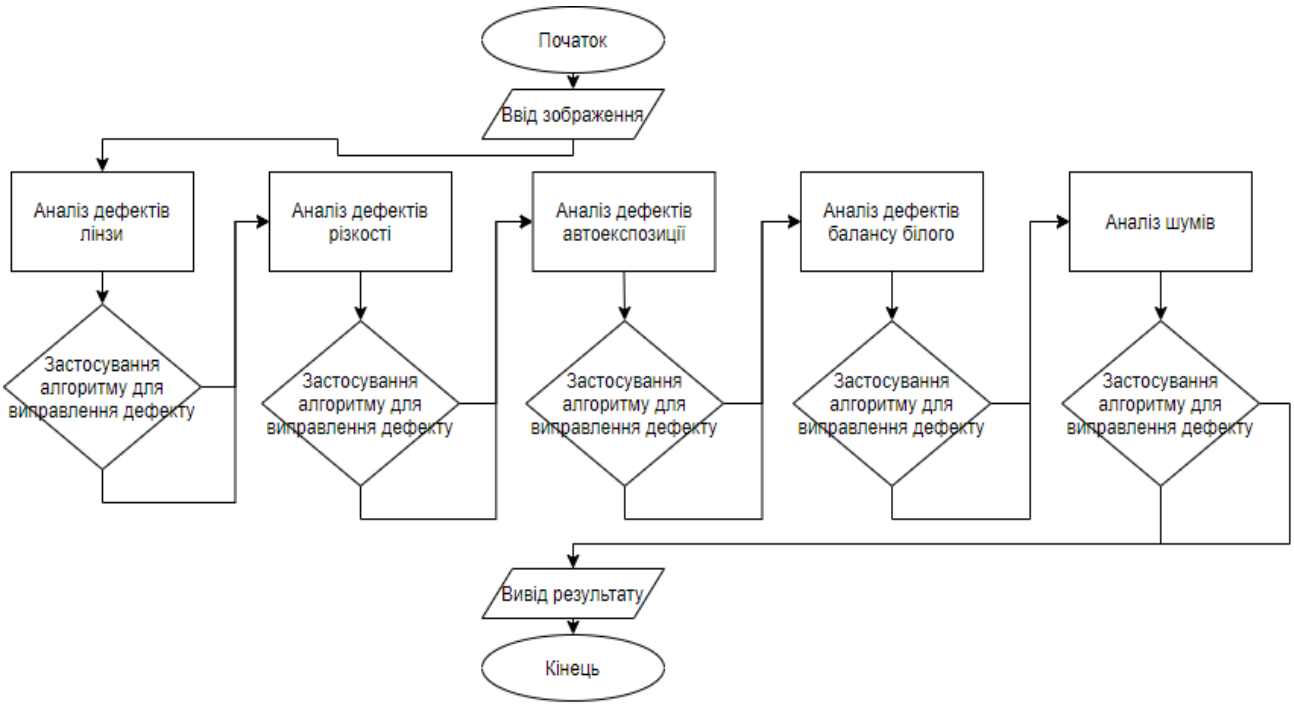

Рисунок 2.2 – Схема алгоритму роботи програмного модуля пошуку дефектів на зображеннях

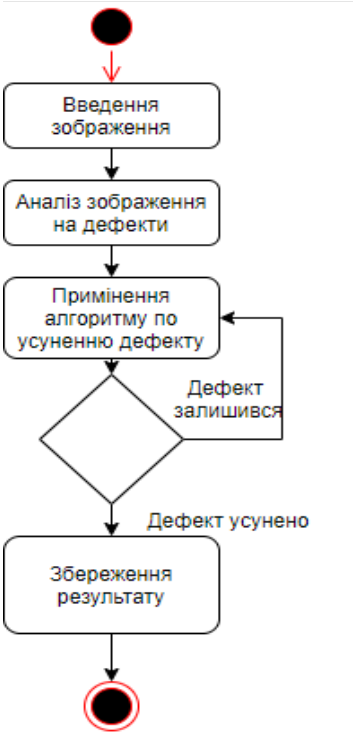

Рисунок 2.3 - Діаграма діяльності модуля пошуку та аналізу дефектів на зображеннях

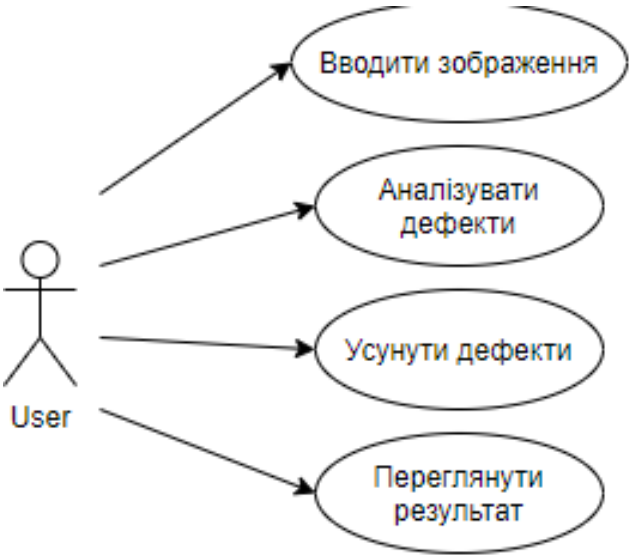

Рисунок 2.4 - UML-діаграма прецедентів

2.5 Висновок

В ході виконання даного розділу було проаналізовано та описано класифікацію дефектів на зображеннях. Також було досліджено можливі методи усунення шуму та інших шумів. Було розглянуто багатошарову згорткову нейронну мережу, як можливе вирішення проблеми аналізу дефектів. Було створено алгоритм роботи програмного модуля та UML-діаграми діяльності та прецедентів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ПОШУКУ ДЕФЕКТІВ НА ЗОБРАЖЕННЯХ

3.1 Обґрунтування вибору мови програмування

Порівняно мови програмування: C++, C# та Python.

C++, безсумнівно, є однією з найдавніших та найефективніших мов програмування, яка все ще продовжує домінувати у сфері програмування.[15]

Переваги C++:

1. Переносимість. C++ пропонує функцію переносимості або незалежності від платформи, що дозволяє користувачеві легко запускати одну і ту ж програму на різних операційних системах або інтерфейсах.

Припустимо, ви пишете програму в ОС LINUX і з явних причин перемикаєтесь на ОС Windows, ви зможете запускати ту саму програму у вікнах без помилок. Ця функція виявляється дуже зручною для програміста.

2. Об’єктно-орієнтованість. Однією з найбільших переваг C ++ є особливість об’єктно-орієнтованого програмування, що включає такі поняття, як класи, успадкування, поліморфізм, абстракція даних та інкапсуляція, які дозволяють повторно використовувати код і роблять програму ще більш надійною.

Мало того, це допомагає нам вирішувати реальні проблеми, розглядаючи дані як об’єкт. Мові C бракувало цієї функції, і тому вона була створена, що виявилося дуже важливим.

Ця особливість породила численні перспективи роботи та технології. Зачаровує той факт, що C ++ був створений комбінуванням функцій не тільки з C, але і Simula 67, першої об'єктно-орієнтованої мови програмування.

3. Мультипарадигма

C++ - мова програмування з багатьма парадигмами. Термін "парадигма" стосується стилю програмування. Він включає логіку, структуру та процедуру програми. Загальна, імперативна та об’єктно-орієнтована - це три парадигми C ++.

Спробуємо тепер зрозуміти, що означає загальне програмування. Загальне програмування стосується використання однієї ідеї, яка служить для кількох цілей. Імперативне програмування, навпаки, стосується використання операторів, які змінюють стан програми.

4. Маніпуляції на низькому рівні

Оскільки C++ тісно пов'язаний із C, яка є процедурною мовою, тісно пов'язаною з машинною мовою, C++ дозволяє низькорівневі маніпуляції з даними на певному рівні. Вбудовані системи та компілятор створюються за допомогою C++.

5. Управління пам'яттю

C++ надає програмісту повний контроль над управлінням пам'яттю. Це можна розглядати і як актив, і як зобов'язання, оскільки це збільшує відповідальність користувача за управління пам'яттю, а не за тим, щоб нею керував збирач сміття. Ця концепція реалізована за допомогою DMA (динамічне розподіл пам'яті) за допомогою покажчиків.

6. Підтримка великої громади

C++ має велику спільноту, яка підтримує його, пропонуючи онлайн-курси та лекції, як платні, так і безоплатні. Статистично кажучи, C++ є шостим найбільш часто використовуваним і відстежуваним тегом на StackOverflow та GitHub.

7. Сумісність з C

C++ майже повністю сумісний з C. Практично кожна безпроблемна програма C є дійсною програмою C++. Залежно від використовуваного компілятора, кожна програма C++ може працювати на файлі з розширенням .cpp.

8. Масштабованість

Масштабованість означає здатність програми масштабуватись. Це означає, що програма C++ може працювати як в малому, так і в великому масштабі даних. Ми також можемо створювати додатки, що вимагають великих ресурсів.

Недоліки C++

1. Використання покажчиків

Вказівники на C/C++ є відносно важким поняттям, яке споживає багато пам'яті. Неправильне використання покажчиків, як дикі вказівники, може призвести до аварійного завершення роботи системи або поведінки аномально.

2. Питання безпеки

Хоча об'єктно-орієнтоване програмування забезпечує значну безпеку даних, що обробляються, порівняно з іншими мовами програмування, які не є об'єктно-орієнтованими, як-от C, певні проблеми безпеки все ще існують через наявність дружніх функцій, глобальних змінних та покажчиків.

3. Відсутність збирача сміття

Як обговорювалося раніше, C++ надає користувачеві повний контроль над керуванням пам’яттю комп’ютера за допомогою DMA. У C++ відсутня функція збирача сміття для автоматичного фільтрування непотрібних даних.

4. Відсутність вбудованої нитки

C++ не підтримує жодних вбудованих потоків. Потоки - це відносно нова концепція в C++, якої спочатку не було. Тепер C++ здатний підтримувати лямбда-функції [15].

C #, розроблений у 2000 році Андерсом Хейлсбергом у Microsoft, C# - це сучасна об'єктно-орієнтована мова програмування. Це мова загального призначення, яка в основному розроблена для розробки програм на платформі Microsoft і вимагає платформи .NET для Windows [16].

Завдяки широкій підтримці Microsoft, C # швидко зростав з часу свого створення. Він особливо використовується для створення настільних додатків та ігор Windows. C # також використовується для розробки веб-додатків і стає все більш популярним і для мобільних розробок.

Але, як і інші мови програмування, мова програмування C # має свої плюси і мінуси.

Плюси мови програмування на C #

Мова програмування C # добре інтегрується з Windows. Спеціальні конфігурації для запуску програми C # у вашому середовищі Windows не потрібні.

Будь то веб-програма, служба Windows або програма для настільних ПК, програми C # легко встановлюються в мережі.

Легко знайти додаткових розробників для мови C #, незалежно від того, на контрактній або штатній основі. Оскільки C # є однією з поширених мов, яку вивчають програмісти, для розвитку вашого бізнесу можна легко знайти додаткових розробників. Крім того, ця мова програмування тісно пов'язана з java, тому розробник може працювати над обома з них одночасно.

C # - це компільована мова, що означає, що код, що зберігається на сервері, є у двійковій формі. Хакер не має автоматично доступу до вашого вихідного коду, оскільки він є у двійковій формі, в той час як, у разі поширених мов, таких як PHP, хакер отримує доступ до вихідного коду, який потім може надати йому доступ до паролів бази даних.

Хоча скомпільований код може виявитись вигідним, але він має деякі недоліки. Працювати з цим досить складно, оскільки код потрібно компілювати щоразу, коли ви вносите навіть незначні зміни. Одна зміна вашого коду змушує користувача перекомпілювати цілу програму та розгорнути її знову. Це часто призводить до додаткових помилок, якщо незначні зміни не ретельно перевірені.

Корпорація Майкрософт припинила підтримку старих платформ .NET після кількох оновлень ОС. Оскільки C # є частиною фреймворку .NET, сервер повинен запускати свою програму, має бути у Windows. Багато нових компаній працюють із серверами Linux, оскільки це набагато дешевше середовище. Для запуску програми .NET потрібен хостинг Windows. Якщо ваша компанія використовує робочі станції та сервери Windows, інтегрувати .NET найпростіше.Python - інтерпретована мова програмування високого рівня, загального призначення. Створена Гідо ван Россумом та вперше випущена в 1991 році, філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню значного пробілу. Його мовні конструкції та об'єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та масштабних проектів.

Python став однією з найбільш швидкозростаючих і найпопулярніших мов програмування у світі. Python універсальний, його легко використовувати та

розробляти. Більше того, у ньому дуже жива громада. Ви можете легко знайти підтримку у найкращих умів у цій галузі. Однак є кілька недоліків Python, на які слід звернути увагу, наприклад, обмеження швидкості [17].

Ось декілька найважливіших переваг Python:

Універсальний, простий у використанні та швидкий у розробці. Python зосереджується на читабельності коду. Мова універсальна, акуратна, проста у використанні та вивченні, читабельна та добре структурована.

Грегорі Решетняк, архітектор програмного забезпечення Nokia, каже: - Я та інші користуємось Python як для швидкого створення сценаріїв, так і для розробки корпоративного програмного забезпечення для компаній Fortune 500. Це потужність - це гнучкість та простота використання в обох випадках. Крива навчання дуже м'яка, а мова багатофункціональна. Python динамічно набирається, що робить його зручним і швидшим для розробки, забезпечуючи REPL, а також середовища, подібні до ноутбуків, таких як Jupyter. Останнє швидко стає фактичним робочим середовищем для науковців даних.

Завдяки гнучкості Python легко провести аналіз дослідницьких даних - в основному шукати голки в копиці сіна, коли ви не впевнені, що це за голка. Python дозволяє використовувати найкращі з різних парадигм програмування. Він об'єктно-орієнтований, але також активно застосовує функції функціонального програмування.

Відкритий код із яскравою спільнотою. Ви можете завантажити Python безкоштовно і написати код за лічені хвилини. Розробка за допомогою Python - це без проблем.

Більше того, спільнота програмістів Python є однією з найкращих у світі - вона дуже велика та активна. Деякі з найкращих IT-інтелектуалів у світі беруть участь як у самій мові, так і на форумах підтримки.

Має всі бібліотеки, які ви можете собі уявити. Ви можете знайти бібліотеку для всього, що ви могли собі уявити: від веб-розробки, розробки ігор до машинного навчання. Відмінно підходить для прототипів - ви можете робити більше, використовуючи менше коду.

Як вже було згадано раніше, Python легко вивчити і швидко розвиватися. Ви можете зробити більше, використовуючи менше коду, а це означає, що ви можете створювати прототипи та тестувати ідеї набагато швидше в Python, ніж в інших мовах. Це означає, що використання Python не лише економить багато часу, але й зменшує витрати вашої компанії.

Продуктивність. Ще однією перевагою Python є те, що ця мова програмування може збільшити продуктивність. Його функції інтеграції та можливості управління можуть підвищити продуктивність програм. У порівнянні з іншими мовами Python є більш продуктивним, ніж Java, оскільки він динамічно набирається та є лаконічнішим.

Обмеження швидкості. Python - це інтерпретована мова, тому ви можете виявити, що вона працює повільніше, ніж деякі інші популярні мови. Але якщо швидкість не є найважливішим фактором для вашого проекту, то Python буде вам чудово.

Проблеми з потоками. Потоки не дуже хороші в Python через глобальний інтерпретатор (GIL). GIL - це просто мьютекс, що дозволяє виконувати лише одному потоку одночасно. Як результат, багатопотокові програми, пов'язані з процесором, можуть бути повільнішими, ніж однопотокові, - каже Матеуш Опала, керівник машинного навчання в Netguru. На щастя, є рішення цієї проблеми. - Нам потрібно впроваджувати багатопроцесорні програми замість багатопотокових. Це те, що ми часто робимо для обробки даних.

Не є рідним для мобільного середовища. Python не є рідним для мобільного середовища, і деякі програмісти розглядають його як слабку мову для мобільних обчислень. Android та iOS не підтримують Python як офіційну мову програмування.

І все-таки Python можна легко використовувати для мобільних цілей, але для цього потрібні певні додаткові зусилля.

Грегорі Решетняк пояснює: - Існує ряд бібліотек, які забезпечують можливість розробки як для Android, так і для iOS за допомогою Python. Найбільш помітним прикладом може бути фреймворк Kivy, який дозволяє використовувати той самий API для створення додатків не тільки для мобільних додатків, але й програмного

забезпечення, призначеного для роботи на Windows, Linux та Raspberry PI. Це безпрецедентний різновид, і його також просто використовувати!

Споживання пам'яті. Слід врахувати, що споживання пам'яті Python дуже велике. З цієї причини це може бути не найкращим вибором для завдань із великою пам’яттю. Це може бути проблематично, коли велика кількість об'єктів активна в оперативній пам'яті.

Простота - проблема чи унікальна особливість? Деякі програмісти кажуть, що перевага Python - простота - це також його слабке місце. Але чи справді це так?

Отже, найбільше для вирішення нашої задачі підходить саме Python, адже він має різноманітні бібліотеки, зокрема для роботи з графікою, обробкою зображень та нейронними мережами.

3.2 Обгрунтування вибору середовища програмування

Visual Studio Code - редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб-і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під пропрієтарною ліцензією.[18]

Visual Studio Code заснований на Electron і реалізується через веб-редактор Monaco, розроблений для Visual Studio Online.

3.3 Використання додаткових бібліотек для пошуку дефектів на зображеннях

TensorFlow - відкрита програмна бібліотека. TensorFlow полегшує початківцям та експертам створення моделей машинного навчання. TensorFlow пропонує

сукупність робочих процесів для розробки та навчання моделей за допомогою Python, JavaScript або Swift, а також для легкого розгортання у хмарі, за попереднім доступом, у браузері чи на пристрої, незалежно від того, якою мовою ви користуєтесь.

Matplotlib - це всебічна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python. Matplotlib - бібліотека на мові програмування Python для візуалізації даних двовимірної (2D) графіки (3D графіка також підтримується). Одержувані зображення можуть бути використані в якості ілюстрацій в публікаціях [19].

Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy і IPython надає можливості, подібні MATLAB. В даний час пакет працює з декількома графічними бібліотеками, включаючи wxWindows і PyGTK.

Пакет підтримує багато видів графіків і діаграм:

- графіки (line plot);
- діаграми розкиду (scatter plot);
- стовпчасті діаграми (bar chart) і гістограми (histogram);
- кругові діаграми (pie chart);
- стовбур-лист діаграми (stem plot);
- контурні графіки (contour plot);
- поля градієнтів (quiver);
- спектральні діаграми (spectrogram);

Користувач може вказати осі координат, грати, додати написи і пояснення, використовувати логарифмічну шкалу або полярні координати.

Нескладні тривимірні графіки можна будувати за допомогою набору інструментів (toolkit) mplot3d. Є й інші набори інструментів: для картографії, для роботи з Excel, утиліти для GTK і інші.

За допомогою Matplotlib можна робити і анімовані зображення [18]

- OpenCV - це open source бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. Широко використовується в таких мовах як C, C ++, Python і Java.[20]

Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору. Алгоритми OpenCV застосовують у таких сферах:

- Аналіз та обробка зображень;
- Системи з розпізнавання обличчя;
- Ідентифікації об'єктів;
- Розпізнавання жестів на відео;
- Відстежування переміщення камери;
- Побудова 3D моделей об'єктів;
- Створення 3D хмар точок зі стерео камер;
- Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю;
- Система взаємодії людини з комп'ютером;
- Пошуку схожих зображень із бази даних;
- Усування ефекту червоних очей при фотозйомці зі спалахом;
- Стеження за рухом очей;
- Аналіз руху;
- Ідентифікація об'єктів;
- Сегментація зображення;
- Трекінґ відео;
- Розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності.

NumPy - це open-source модуль для python, який надає загальні математичні і числові операції у вигляді пре-скомпільовані, швидких функцій. Вони об'єднуються в високорівневі пакети. Вони забезпечують функціонал, який можна порівняти з функціоналом MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з великими масивами і матрицями. SciPy (Scientific Python) розширює функціонал numpy величезною колекцією корисних алгоритмів, таких як мінімізація, перетворення Фур'є, регресія, і інші прикладні математичні техніки [19]

Головною особливістю numpy є об'єкт array. Масиви схожі зі списками в python, виключаючи той факт, що елементи масиву повинні мати однаковий тип даних, як float і int. З масивами можна проводити числові операції з великим обсягом інформації в рази швидше і, головне, набагато ефективніше ніж зі списками.

Pillow це форк PIL (Python Image Library), яка з'явилася завдяки підтримці Алекса Кларка та інших учасників. Заснована на коді PIL, а потім перетворилася в поліпшену, сучасну версію. Надає підтримку при відкритті, управлінні і збереженні багатьох форматів зображення. Багато що працює так само, як і в оригінальній PIL.

Scikit-image (раніше scikits.image) - це бібліотека для обробки зображень з відкритим кодом для мови програмування Python. Він включає алгоритми сегментації, геометричних перетворень, маніпулювання кольоровим простором, аналіз, фільтрацію, морфологію, виявлення об’єктів тощо. Розроблений для взаємодії з числовими та науковими бібліотеками Python NumPy та SciPy.

Проект scikit-image розпочався як scikits.image, автора Стефана ван дер Вальта. Його назва походить від уявлення про те, що це "SciKit" (SciPy Toolkit), окремо розроблене та розподілене стороннім розширенням SciPy. Пізніше оригінальна база кодів була широко перероблена іншими розробниками. У листопаді 2012 р. Scikit-image також брав участь у Google Summer of Code.

Scikit-image в основному написаний на Python, а деякі основні алгоритми написані на Cython для досягнення продуктивності.

Imutils. Набір зручних функцій для полегшення основних функцій обробки зображень, таких як переклад, обертання, зміна розміру, скелетування, відображення зображень Matplotlib, сортування контурів, виявлення країв та спрощується завдяки OpenCV та Python 2.7 і Python 3.

Модуль easygui - це набір з двох десятків функцій-методів, що полегшують в скриптах "Пайтона" створення елементів графічного інтерфейсу. Графічний інтерфейс в "Пайтон" реалізується за допомогою бібліотек TKinter і WxPython, які для початківця можуть виявитися недостатньо легкими. Методи easygui є як би обгорткою до функцій TKinter, в результаті чого елементи графічного інтерфейсу

легко і невимушено викликаються в одну-дві строчки. Це дозволяє на перших порах початківцям не особливо заморочуватися вивченням цієї самої бібліотеки TKinter. Також easygui буде хороший і для чорнових начерків скриптів з графічним інтерфейсом. Але все-таки повноцінного графічного додатку на "Пайтон" за допомогою цього модуля ви не створите.

Tkinter. В Python є досить багато GUI фреймворків (graphical user interface), однак тільки Tkinter вбудований в стандартну бібліотеку мови. У Tkinter є кілька переваг. Він багатоплатформовий, тому один і той же код можна використовувати на Windows, macOS і Linux.

Візуальні елементи відображаються через власні елементи поточної операційної системи, тому додатки, створені за допомогою Tkinter, виглядають так, як ніби вони належать тій платформі, на якій вони працюють.

Хоча Tkinter є популярним GUI фреймворком на Python, у нього є свої недоліки. Один з них полягає в тому, що графічні інтерфейси, створені з використанням Tkinter, виглядають застарілими. Якщо вам потрібен сучасний, яскравий інтерфейс, то Tkinter може виявитися не зовсім тим, для цього є PyQt5 який розвивається сильніше в даному плані.

Проте, в плані використання, Tkinter є відносно легким у порівнянні з іншими бібліотеками. Це відмінний вибір для створення GUI додатків в Python, особливо якщо сучасний вигляд не в пріоритеті для програми, а велику роль грає функціональність і кросплатформна швидкість.

3.4 Тестування розробленого програмного модуля пошуку дефектів на зображеннях

Спочатку запускаємо програму (рис. 3.1).

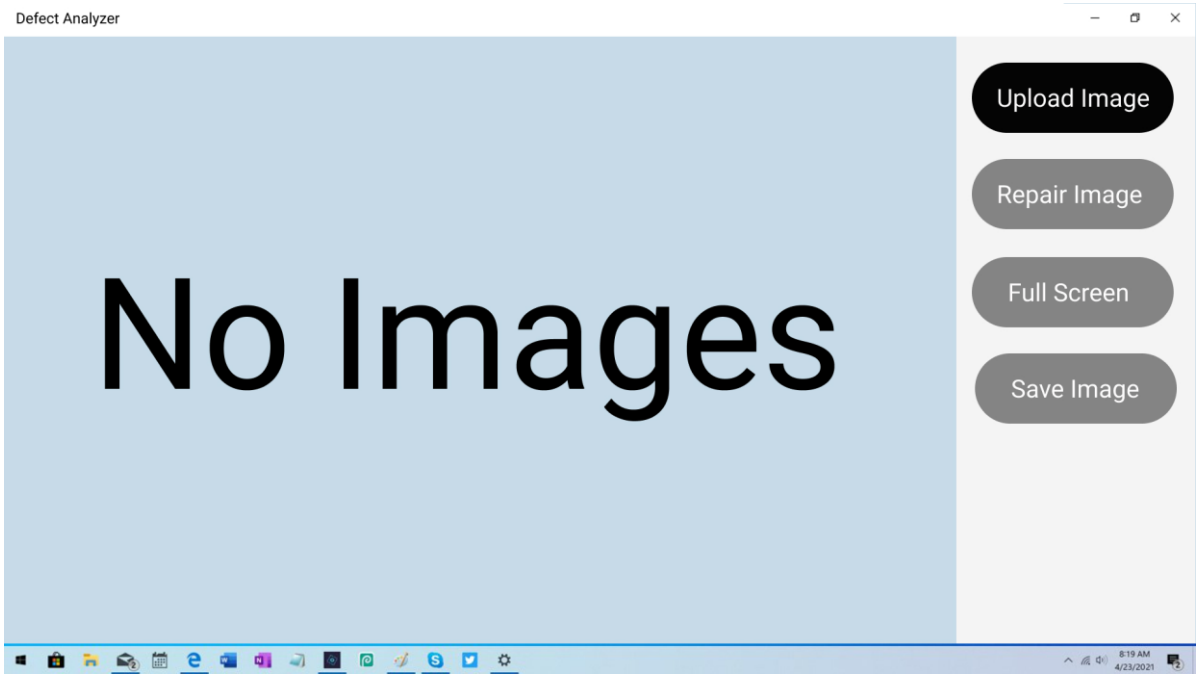

Рисунок 3.1 – Початковий екран програми

Вибираємо зображення з файлової системи, натиснувши Upload Image (рис. 3.2).

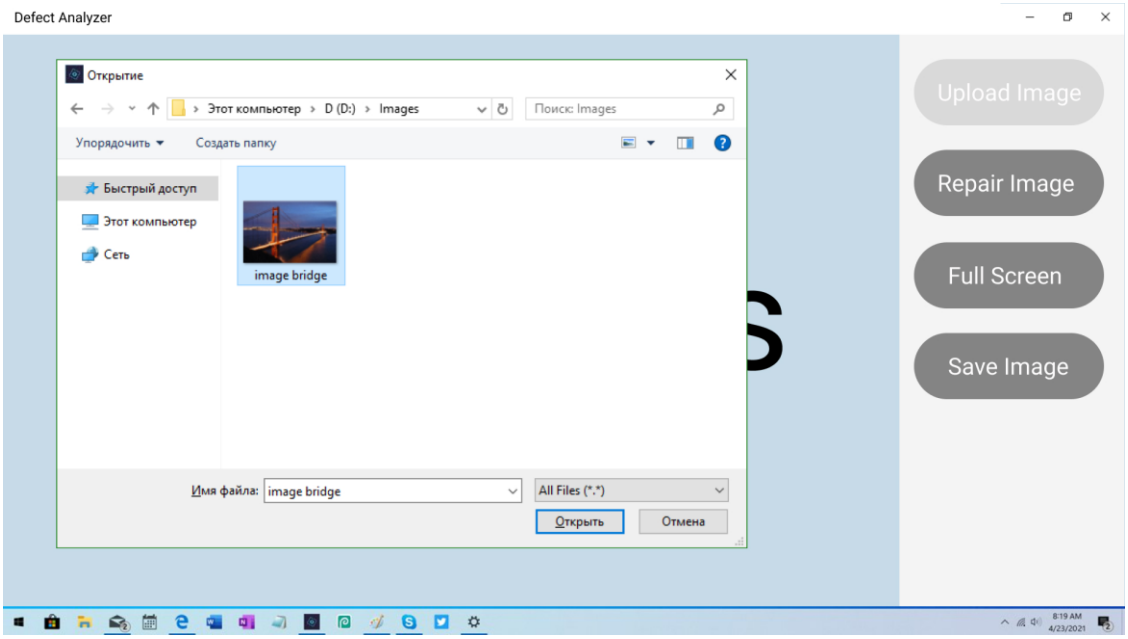

Рисунок 3.2 – Вибір зображення для аналізу дефектів

Натискуємо Repair Image для пошуку та усунення дефектів на обраному зображенні (рис. 3.3).

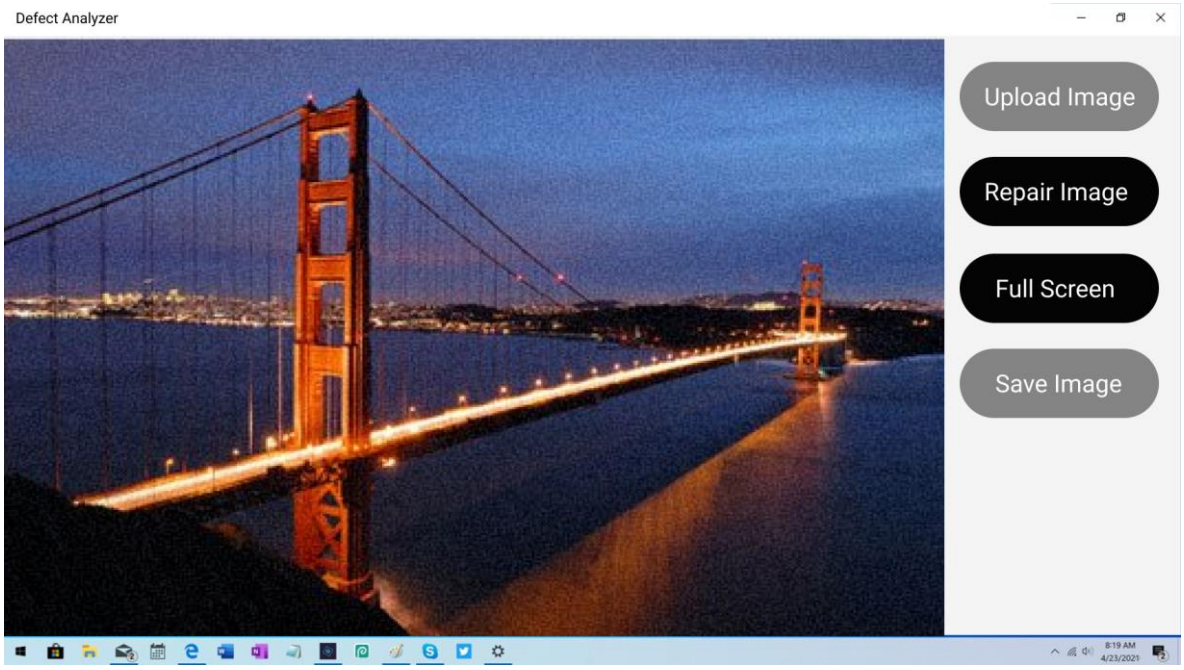

Рисунок 3.3 – Дефектне зображення

Далі зберігаємо зображення, натиснувши Save Image (рис. 3.4, 3.5).

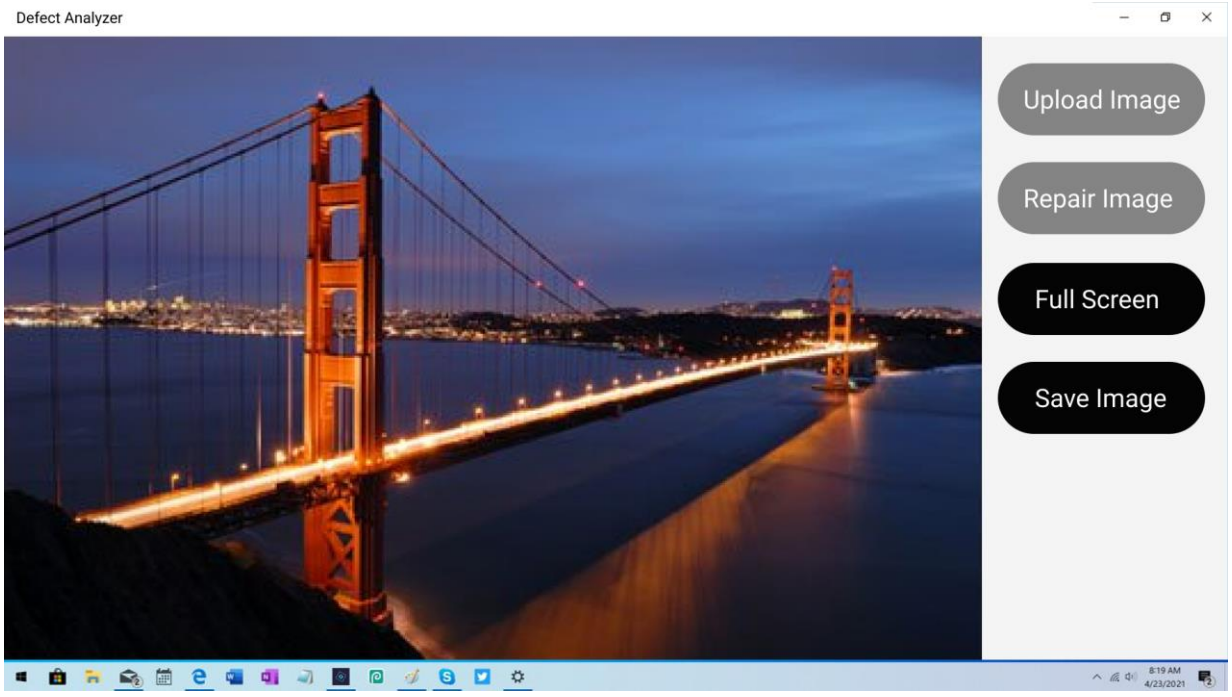

Рисунок 3.4 – Виправлене зображення

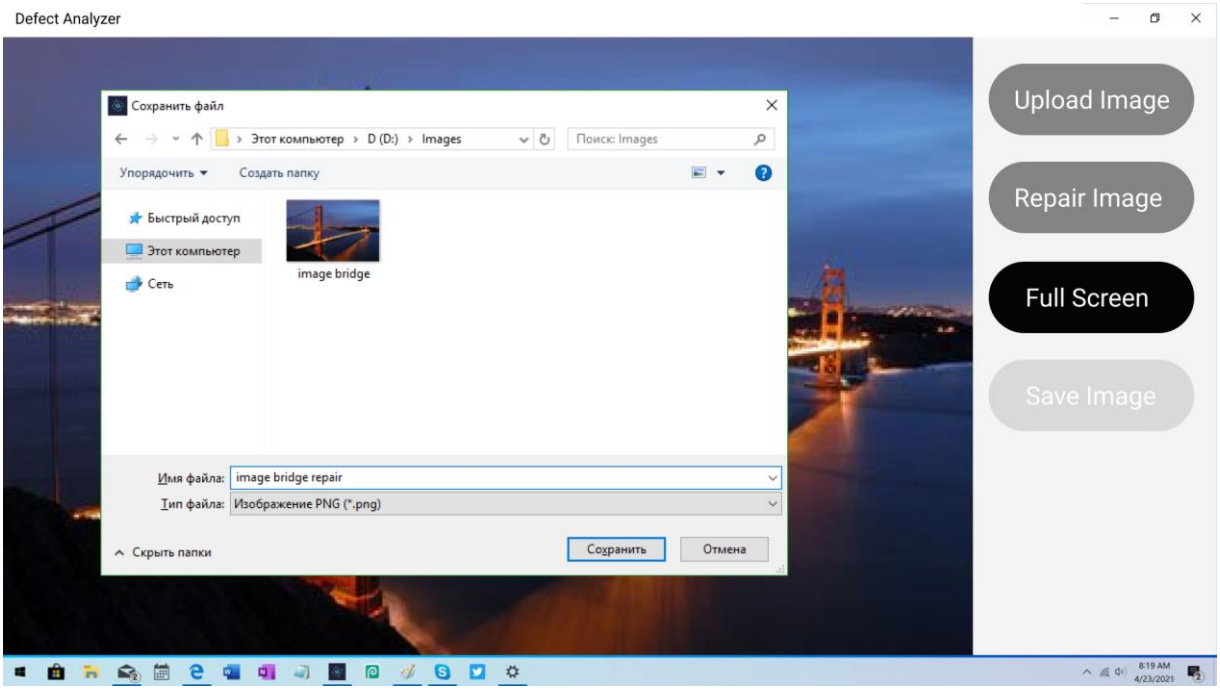

Рисунок 3.5 – Збереження зображення

Можна відобразити виправлене зображення в повноекранному режимі (рис. 3.6).

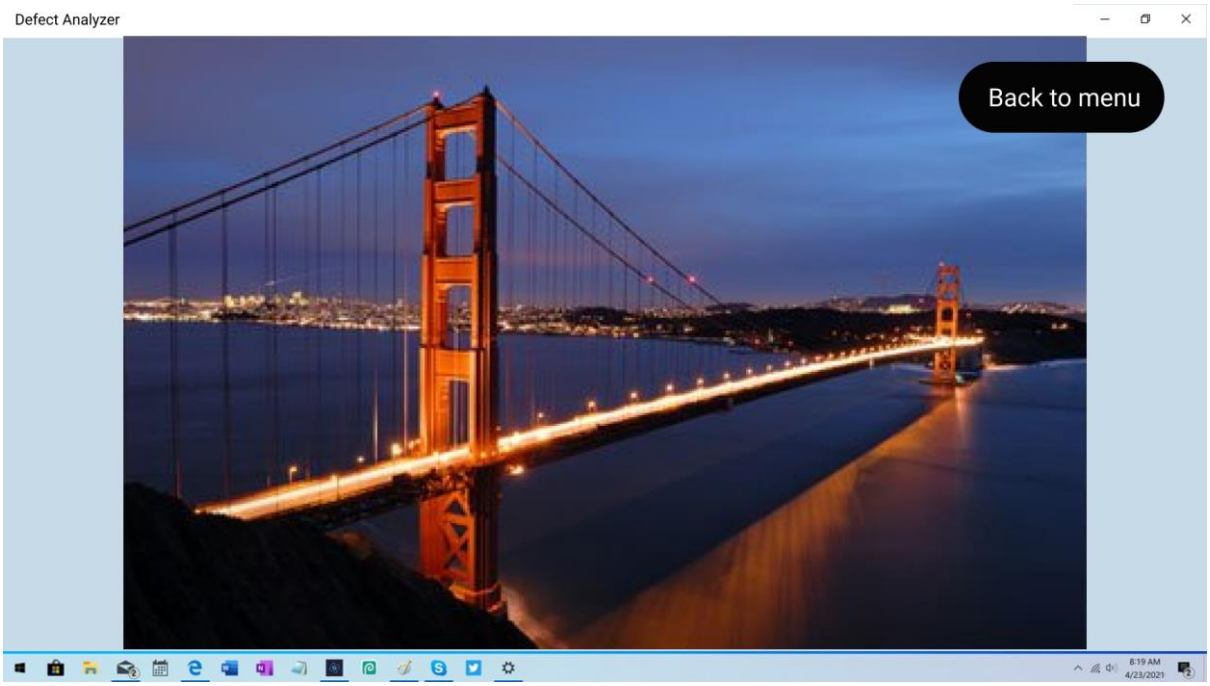

Рисунок 3.6 – Повноекранний режим програми

Розроблений програмний модуль пошуку дефектів на зображеннях має функціонал, наведений в таблиці 3.1, який було порівняно із функціоналом найпопулярніших аналогів для обробки зображень.

Таблиця 3.1 – Порівняльна характеристика розробленого ПЗ

Назва програми	Виправлення дефектів лінзи	Виправлення дефектів різкості	Виправлення дефектів автоекспозиції	Аналіз шумів
ImaTest	+	+	-	-
ImageJ	-	+	-	-
Розроблений модуль	+	+	+	+

Отже, розроблений модуль пошуку дефектів на зображеннях було протестовано на відповідність меті дослідження. Як видно з таблиці 3.1 функціонал програмного забезпечення є розширеним по відношенню до існуючих сучасних рішень та відповідає завданню.

3.5 Висновок

В цьому розділі було детально розглянуто мови програмування, їх переваги та недоліки та було обрано одну – Python. Було обгрунтовано використання додаткових бібліотек, необхідних для нашої задачі. Програмно реалізовано модуль пошуку дефектів на зображеннях та подальшого їх виправлення. В результаті тестування розробленого програмного забезпечення доведено розширення функціональних можливостей по відношенню до програм аналогів.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи реалізовано програмний модуль пошуку дефектів на зображеннях.

В роботі проаналізовано предметну область пошуку дефектів на зображенні. Було проаналізовано переваги та недоліки програм аналогів, що дозволяють проводити пошук дефектів на зображеннях. Було сформовано вимоги до розроблюваного програмного забезпечення та здійснено постановку задачі. Було проаналізовано та описано класифікацію дефектів на зображеннях. Також було досліджено можливі методи усунення шуму та інших шумів. Було розглянуто багатошарову згорткову нейронну мережу, як можливе вирішення проблеми аналізу дефектів. Було створено алгоритм роботи програмного модуля та UML-діаграми діяльності та прецедентів. В роботі було обгрунтовано використання додаткових бібліотек, необхідних для нашої задачі. Програмно реалізовано модуль пошуку дефектів на зображеннях та подальшого їх виправлення. В результаті тестування розробленого програмного забезпечення доведено розширення функціональних можливостей по відношенню до програм аналогів.

Отже всі поставлені задачі виконано, мету роботи досягнуто.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Computer Vision: Algorithms and Applications. Richard Szeliski. 2010.
2. Adobe Photoshop [Електронний ресурс]. Режим доступу до матеріалу: https://www.adobe.com/ua/products/photoshop.html
3. Mathlab [Електронний ресурс]. Режим доступу до матеріалу: https://www.mathworks.com/products/matlab.html
4. Imatest [Електронний ресурс]. Режим доступу до матеріалу: https://www.imatest.com/
5. Imagej [Електронний ресурс]. Режим доступу до матеріалу: https://imagej.nih.gov/ij/
6. G. de Haan, T. G. Kwaaitaal-Spassova and O.A. Ojo "Automatic 2-D and 3-D noise filtering for high-quality television receivers". International Workshop on HDTV'94,4-B-2,1994-10
7. Szymon Graboskwi, Wojciech Bienieck "A two-pass median-like filter for impulse noise removal in multi-channel images". KOSYR 2003, str. 195-200
8. Richard Alan Peters II "A New Algorithm for Image Noise Reduction using Mathematical Morphology". IEEE Transactions on Image Processing, Volume 4, Number 3, pp. 554-568, May 1995
9. Miroslav Vrankic, Damir Sersic "Image Denoising Based on Adaptive Quincunx Wavelets". In Proc. of the 2004 IEEE 6th Workshop on Multimedia Signal Processing, (MMSP 2004), Siena, Italy, September 29 - October 01, 2004, pp. 251-254
10. D. Darian Muresan, Thomas W. Parks "Adaptive principal components and image denoising". IEEE ICIP 2003
11. Visual studio code [Електронний ресурс]. Режим доступу до матеріалу: https://code.visualstudio.com/
12. Matplotlib. [Електронний ресурс]. Режим доступу до матеріалу:: matplotlib.org
13. OpenCV Documentation. [Електронний ресурс]. Режим доступу до матеріалу:: docs.opencv.org/3.0-beta/

14. NumPy. [Електронний ресурс]. Режим доступу до матеріалу:: www.numpy.org
15. Озеранський В.С., Сиротін О.А, «Розробка інформаційної технології розпізнавання об'єктів у відеопотоці» в Матеріали конференції«Молодь в науці: дослідження, проблеми, перспективи (МН2019)», Вінниця, 2019. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-mn/index/pages/view/zbirn2019.
16. Python documentation – [Електронний ресурс] – https://www.python.org/doc/
17. Image Classification on ImageNet – [Електронний ресурс] – Режим доступу: https://paperswithcode.com/sota/image-classification-on-imagenet
18. Eric Rothstein Morris, Ralf C. Staudemeyer, Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks, 2019 – 42c.
19. ML.NET – Machine Learning made for .NET [Електронний ресурс] – Режим доступу: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/.
20. Подоба Віталій. У яких випадках мова програмування Python є правильним вибором? – [Електронний ресурс] – Режим доступу: http://www.vitaliypodoba.com/2015/06/python-application/
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 пошуку дефектів на зображеннях

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

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

Показники звіту подібності Unicheck

Оригінальність 80,3% Схожість 19,7%

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 Коржанівський В.Ю.

Керівник роботи Шевчук О.Ф.

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧА

Спочатку запускаємо програму (рис. Б.1).

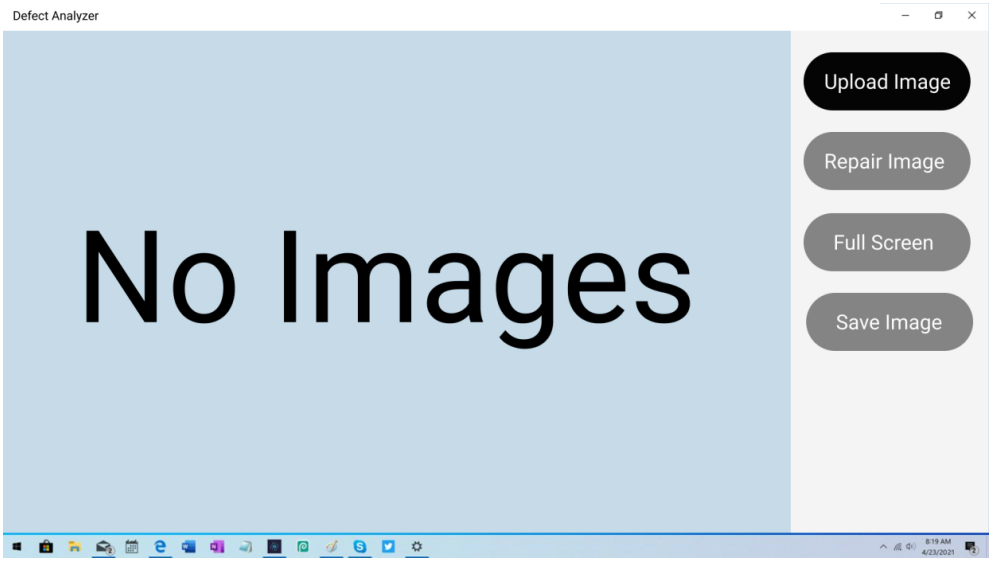

Рисунок 3.1 – Початковий екран програми

Вибираємо зображення з файлової системи, натиснувши Upload Image (рис. Б.2).

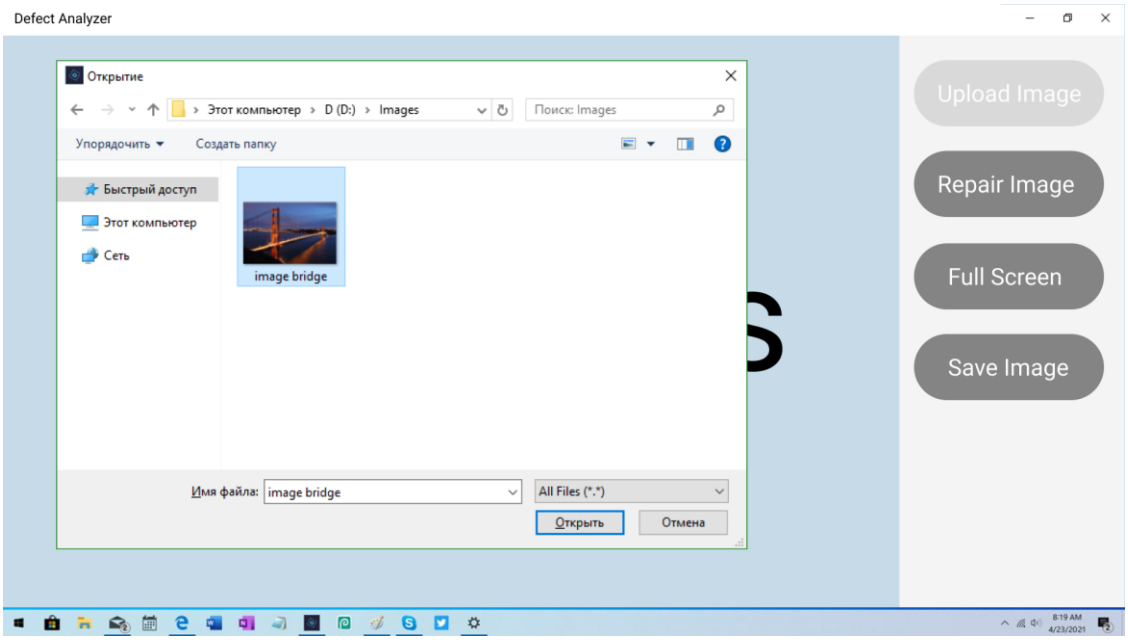

Рисунок Б.2 – Вибір зображення для аналізу дефектів

Натискуємо Repair Image для пошуку та усунення дефектів на обраному зображенні (рис. Б.3).

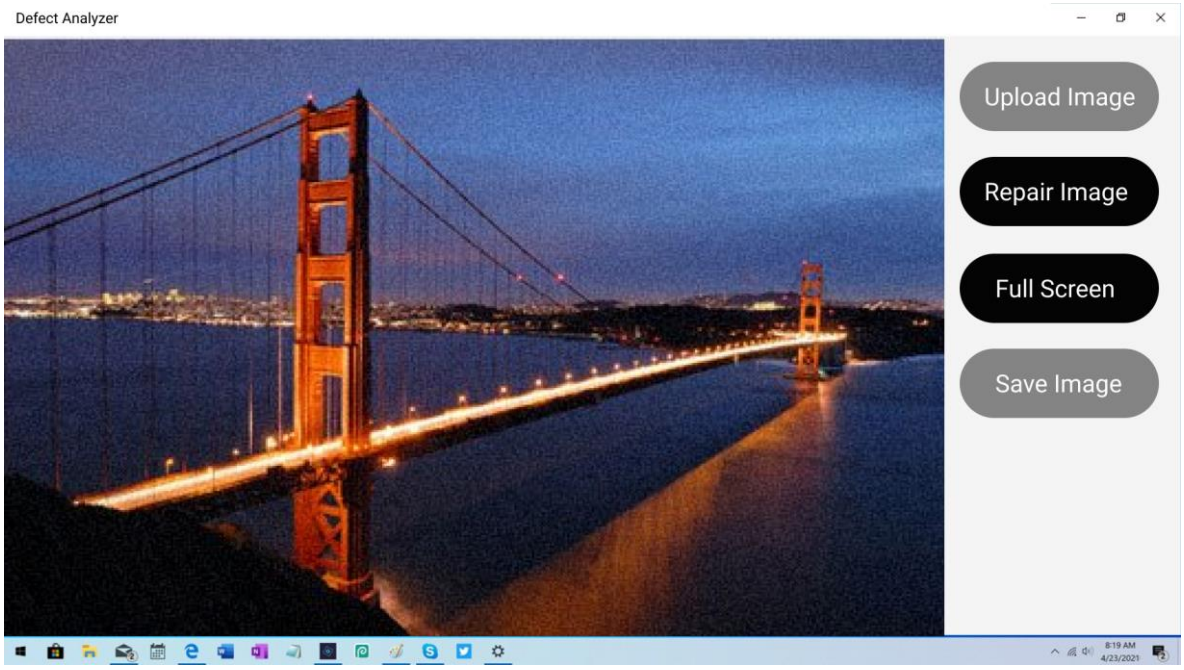

Рисунок Б.3 – Дефектне зображення

Далі зберігаємо зображення, натиснувши Save Image (рис. Б.4, Б.5).

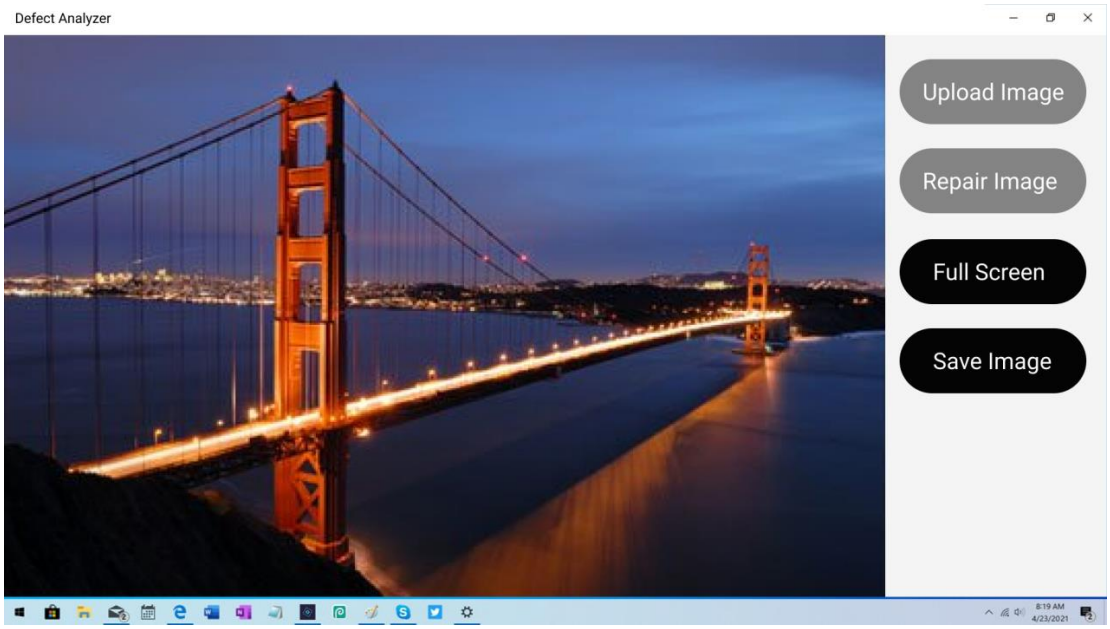

Рисунок Б.4 – Виправлене зображення

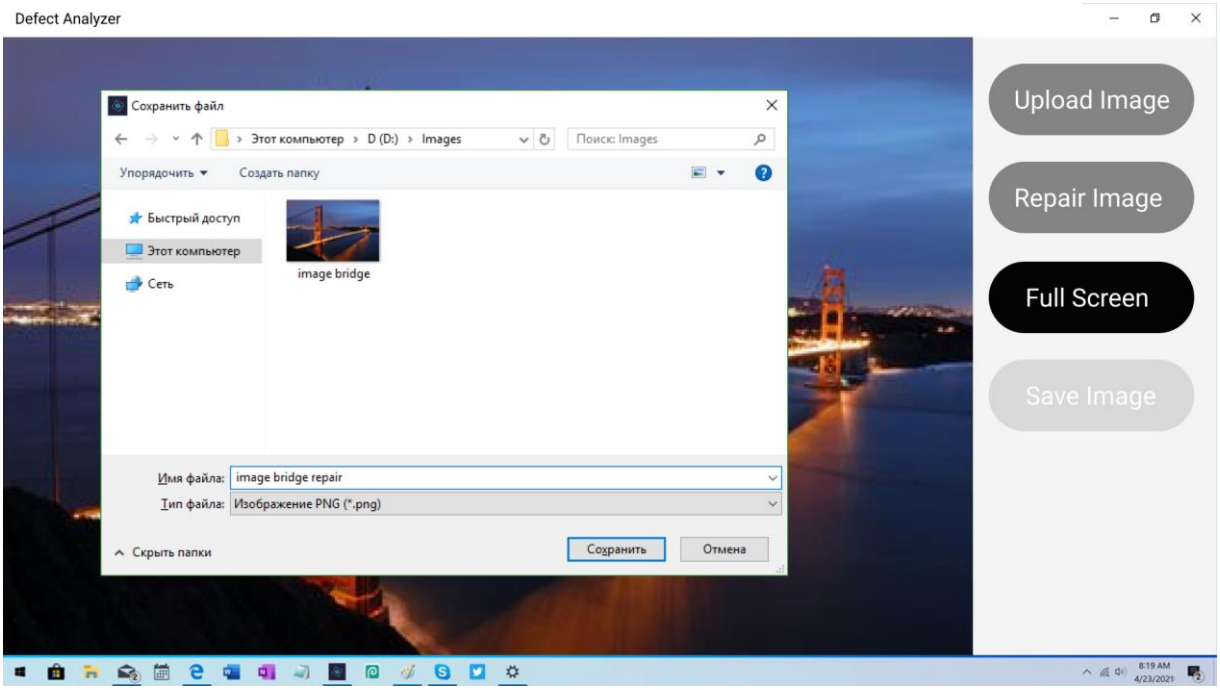

Рисунок Б.5 – Збереження зображення

Можна відобразити виправлене зображення в повноекранному режимі (рис. Б.6).

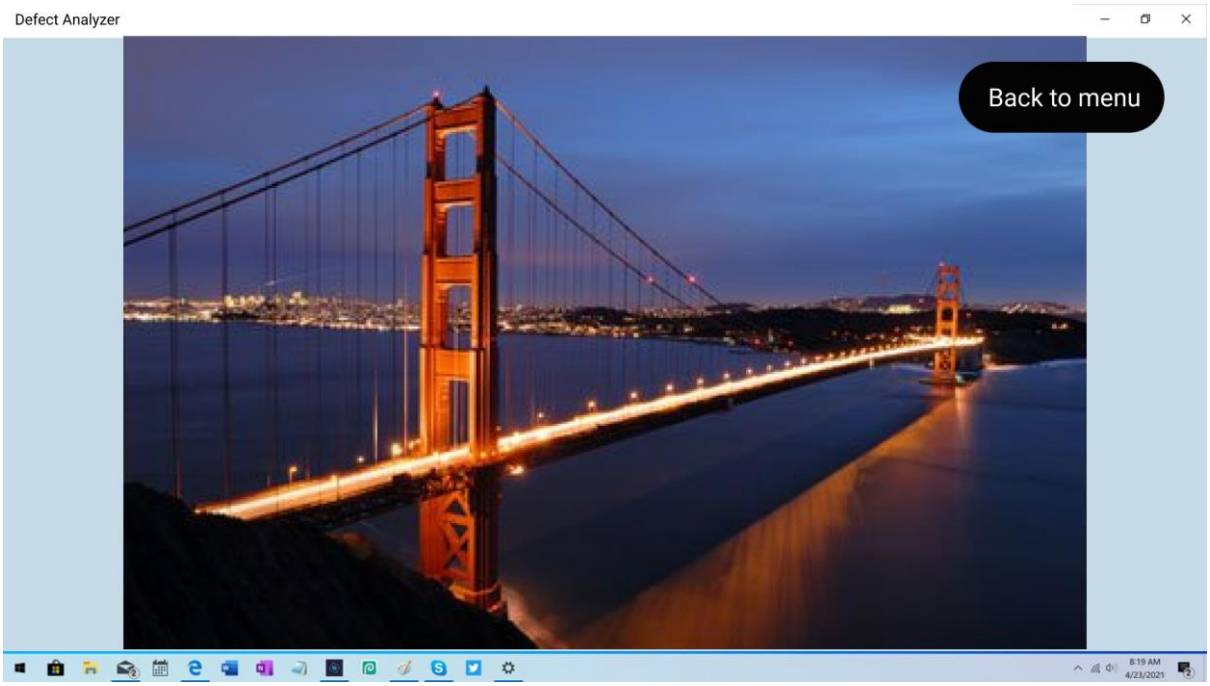

Рисунок Б.6 – Повноекранний режим програми

ДОДАТОК В
ЛІСТИНГ ПРОГРАМИ

```
import cv2
import numpy
import imutils
import skimage
from easygui import *
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import filedialog as fd
from skimage.metrics import structural_similarity

def func():
        print("Select your Reference Image")
        path1a = fd.askopenfilename()
        print("Select your Test Image")
        path1b = fd.askopenfilename()

        imageA = cv2.imread(path1a)
        imageB = cv2.imread(path1b)

        imageA = cv2.resize(imageA,(1200,600),interpolation=cv2.INTER_LINEAR)
        imageB = cv2.resize(imageB,(1200,600),interpolation=cv2.INTER_LINEAR)

        grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
        grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)


        # Structural Similarity Index (SSIM)
        (score, diff) = structural_similarity(grayA, grayB, full=True)
        diff = (diff*255).astype("uint8")
        print("SSIM: {}".format(score))

        thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)

        #loop over the contours after threshold process
        p = 0
        for c in cnts:
```

```
                p=p+1
                (x, y, w, h) = cv2.boundingRect(c)
                #cv2.rectangle(imageA, (x, y), (x+w, y+h), (0, 0, 255), 1)
                cv2.rectangle(imageB, (x, y), (x+w, y+h), (0, 0, 255), 1)
        print("Total number of defects in the image:",p)
        print("\n\n")

        kernel = numpy.ones((5, 5), numpy.uint8)

        #Dilation Morphology --> used to expand the original image
        dilation = cv2.dilate(thresh,kernel,iterations = 1)

        #Erosion Morphology --> used to shrink down the original image
        erosion = cv2.erode(thresh,kernel,iterations = 1)

        #Gradient is used to get the difference between Erosion and Dilation
        gradient = cv2.morphologyEx(thresh, cv2.MORPH_GRADIENT, kernel)

        #Opening --> Erosion followed by Dilation
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)

        #Closing --> Dilation followed by Erosion
        closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)


        cv2.imshow("Displaying Defects", imageB)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

while 1:
  #disp = "img.jpg"
  msg="PCB DEFECT DETECTION"
  choices = ["Detect","Exit"]
  reply   = buttonbox(msg, choices=choices)
  if reply ==choices[0]:
      func()
  if reply ==choices[1]:
      print("Thank you!")
      quit()
import numpy as np
import matplotlib.pyplot as plt

from skimage.transform import hough_line
from skimage.draw import line
```

```
img = np.zeros((100, 150), dtype=bool)
img[30, :] = 1
img[:, 65] = 1
img[35:45, 35:50] = 1
rr, cc = line(60, 130, 80, 10)
img[rr, cc] = 1
img += np.random.random(img.shape) > 0.95

out, angles, d = hough_line(img)

fix, axes = plt.subplots(1, 2, figsize=(7, 4))

axes[0].imshow(img, cmap=plt.cm.gray)
axes[0].set_title('Input image')

angle_step = 0.5 * np.rad2deg(np.diff(angles).mean())
d_step = 0.5 * np.diff(d).mean()
bounds = (np.rad2deg(angles[0]) - angle_step,
        np.rad2deg(angles[-1]) + angle_step,
        d[-1] + d_step, d[0] - d_step)

axes[1].imshow(out, cmap=plt.cm.bone, extent=bounds)
axes[1].set_title('Hough transform')
axes[1].set_xlabel('Angle (degree)')
axes[1].set_ylabel('Distance (pixel)')

plt.tight_layout()
plt.show()
import os

import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid import AxesGrid

from skimage.io import MultiImage
from skimage import data_dir

# Load the multi-layer image
fname = os.path.join(data_dir, 'multipage.tif')
img = MultiImage(fname)

# Create an image grid
fig = plt.figure()
grid = AxesGrid(fig,
```

```
            rect=(1, 1, 1),
            nrows_ncols=(1, 2),
            axes_pad=0.1)

# Plot the layers on the image grid
for i, frame in enumerate(img):
    grid[i].imshow(frame, cmap=plt.cm.gray)
    grid[i].set_xlabel('Frame %s' % i)
    grid[i].set_xticks([])
    grid[i].set_yticks([])

plt.show()
import numpy as np

from ..color.colorconv import rgb2gray, rgba2rgb
from ..util.dtype import dtype_range, dtype_limits
from .._shared.utils import warn


__all__ = ['histogram', 'cumulative_distribution', 'equalize_hist',
           'rescale_intensity', 'adjust_gamma', 'adjust_log', 'adjust_sigmoid']


DTYPE_RANGE = dtype_range.copy()
DTYPE_RANGE.update((d.__name__, limits) for d, limits in dtype_range.items())
DTYPE_RANGE.update({'uint10': (0, 2 ** 10 - 1),
                    'uint12': (0, 2 ** 12 - 1),
                    'uint14': (0, 2 ** 14 - 1),
                    'bool': dtype_range[bool],
                    'float': dtype_range[np.float64]})


def _offset_array(arr, low_boundary, high_boundary):
    """Offset the array to get the lowest value at 0 if negative."""
    if low_boundary < 0:
        offset = low_boundary
        dyn_range = high_boundary - low_boundary
        # get smallest dtype that can hold both minimum and offset maximum
        offset_dtype = np.promote_types(np.min_scalar_type(dyn_range),
                                        np.min_scalar_type(low_boundary))
        if arr.dtype != offset_dtype:
            # prevent overflow errors when offsetting
            arr = arr.astype(offset_dtype)
        arr = arr - offset
```

```
    else:
        offset = 0
    return arr, offset


def _bincount_histogram(image, source_range):
if source_range not in ['image', 'dtype']:
        raise ValueError('Incorrect value for `source_range` argument: {}'.format(source_range))
    if source_range == 'image':
        image_min = int(image.min().astype(np.int64))
        image_max = int(image.max().astype(np.int64))
    elif source_range == 'dtype':
        image_min, image_max = dtype_limits(image, clip_negative=False)
    image, offset = _offset_array(image, image_min, image_max)
    hist = np.bincount(image.ravel(), minlength=image_max - image_min + 1)
    bin_centers = np.arange(image_min, image_max + 1)
    if source_range == 'image':
        idx = max(image_min, 0)
        hist = hist[idx:]
    return hist, bin_centers


def histogram(image, nbins=256, source_range='image', normalize=False):
    """Return histogram of image.
sh = image.shape
    if len(sh) == 3 and sh[-1] < 4:
        warn("This might be a color image. The histogram will be "
             "computed on the flattened image. You can instead "
             "apply this function to each color channel.")

    image = image.flatten()
    # For integer types, histogramming with bincount is more efficient.
    if np.issubdtype(image.dtype, np.integer):
        hist, bin_centers = _bincount_histogram(image, source_range)
    else:
        if source_range == 'image':
            hist_range = None
        elif source_range == 'dtype':
            hist_range = dtype_limits(image, clip_negative=False)
        else:
            ValueError('Wrong value for the `source_range` argument')
        hist, bin_edges = np.histogram(image, bins=nbins, range=hist_range)
        bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2.
```

```
    if normalize:
        hist = hist / np.sum(hist)
    return hist, bin_centers


def cumulative_distribution(image, nbins=256):
if mask is not None:
        mask = np.array(mask, dtype=bool)
        cdf, bin_centers = cumulative_distribution(image[mask], nbins)
    else:
        cdf, bin_centers = cumulative_distribution(image, nbins)
    out = np.interp(image.flat, bin_centers, cdf)
    return out.reshape(image.shape)


def intensity_range(image, range_values='image', clip_negative=False):
  if range_values == 'dtype':
        range_values = image.dtype.type

    if range_values == 'image':
        i_min = np.min(image)
        i_max = np.max(image)
    elif range_values in DTYPE_RANGE:
        i_min, i_max = DTYPE_RANGE[range_values]
        if clip_negative:
            i_min = 0
    else:
        i_min, i_max = range_values
    return i_min, i_max


def _output_dtype(dtype_or_range):
if type(dtype_or_range) in [list, tuple, np.ndarray]:
        # pair of values: always return float.
        return float
    if type(dtype_or_range) == type:
        # already a type: return it
        return dtype_or_range
    if dtype_or_range in DTYPE_RANGE:
        # string key in DTYPE_RANGE dictionary
        try:
            # if it's a canonical numpy dtype, convert
            return np.dtype(dtype_or_range).type
        except TypeError:  # uint10, uint12, uint14
```

```
            # otherwise, return uint16
            return np.uint16
        else:
            raise ValueError(
                'Incorrect value for out_range, should be a valid image data '
                f'type or a pair of values, got {dtype_or_range}.'
            )


def rescale_intensity(image, in_range='image', out_range='dtype'):
"""
    if out_range in ['dtype', 'image']:
        out_dtype = _output_dtype(image.dtype.type)
    else:
        out_dtype = _output_dtype(out_range)

    imin, imax = map(float, intensity_range(image, in_range))
    omin, omax = map(float, intensity_range(image, out_range,
                                            clip_negative=(imin >= 0)))
    if np.any(np.isnan([imin, imax, omin, omax])):
        warn(
            "One or more intensity levels are NaN. Rescaling will broadcast "
            "NaN to the full image. Provide intensity levels yourself to "
            "avoid this. E.g. with np.nanmin(image), np.nanmax(image).",
            stacklevel=2
        )
    image = np.clip(image, imin, imax)
    if imin != imax:
        image = (image - imin) / (imax - imin)
        return np.asarray(image * (omax - omin) + omin, dtype=out_dtype)
    else:
        return np.clip(image, omin, omax).astype(out_dtype)

def _assert_non_negative(image):
    if np.any(image < 0):
        raise ValueError('Image Correction methods work correctly only on '
                         'images with non-negative values. Use '
                         'skimage.exposure.rescale_intensity.')
def _adjust_gamma_u8(image, gamma, gain):
    """LUT based implmentation of gamma adjustement.
    """
    lut = (255 * gain * (np.linspace(0, 1, 256) ** gamma)).astype('uint8')
    return lut[image]def adjust_gamma(image, gamma=1, gain=1):
```

ДОДАТОК Г
ГРАФІЧНА ЧАСТИНА

ПРОГРАМНИЙ МОДУЛЬ ПОШУКУ ДЕФЕКТІВ НА ЗОБРАЖЕННЯХ

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Коржанівський В.Ю.
(прізвище та ініціали)

Керівник: к.ф.-м.н., доцент кафедри КН
Шевчук О.Ф.
(прізвище та ініціали)

«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

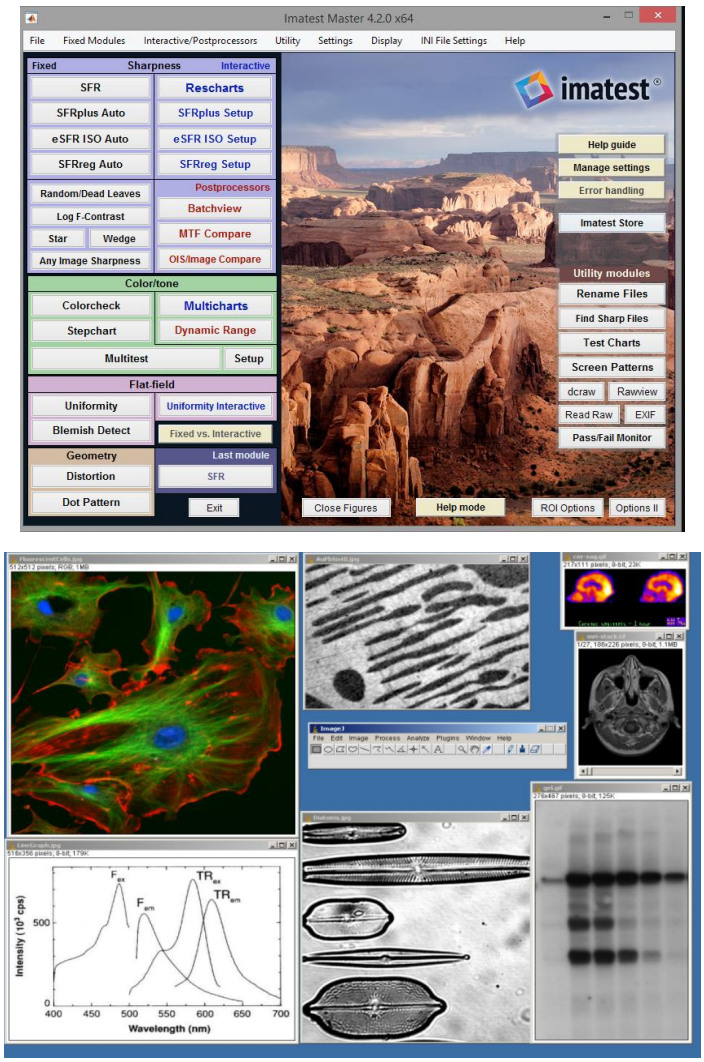

Рисунок Г.1 – Програми аналоги для пошуку дефектів на зображеннях

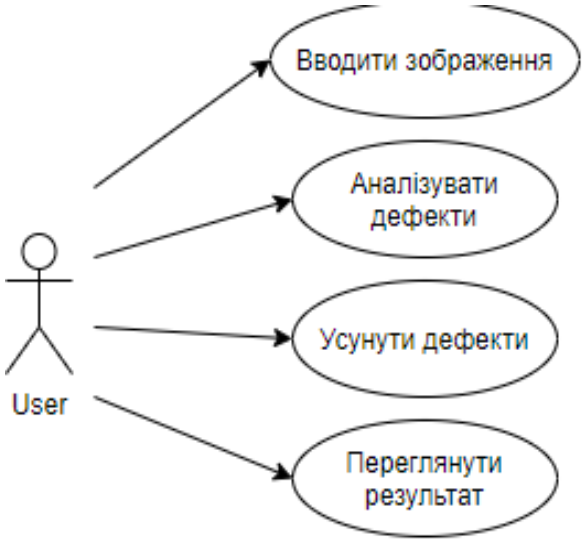

Рисунок Г.2 – UML-діаграма прецедентів

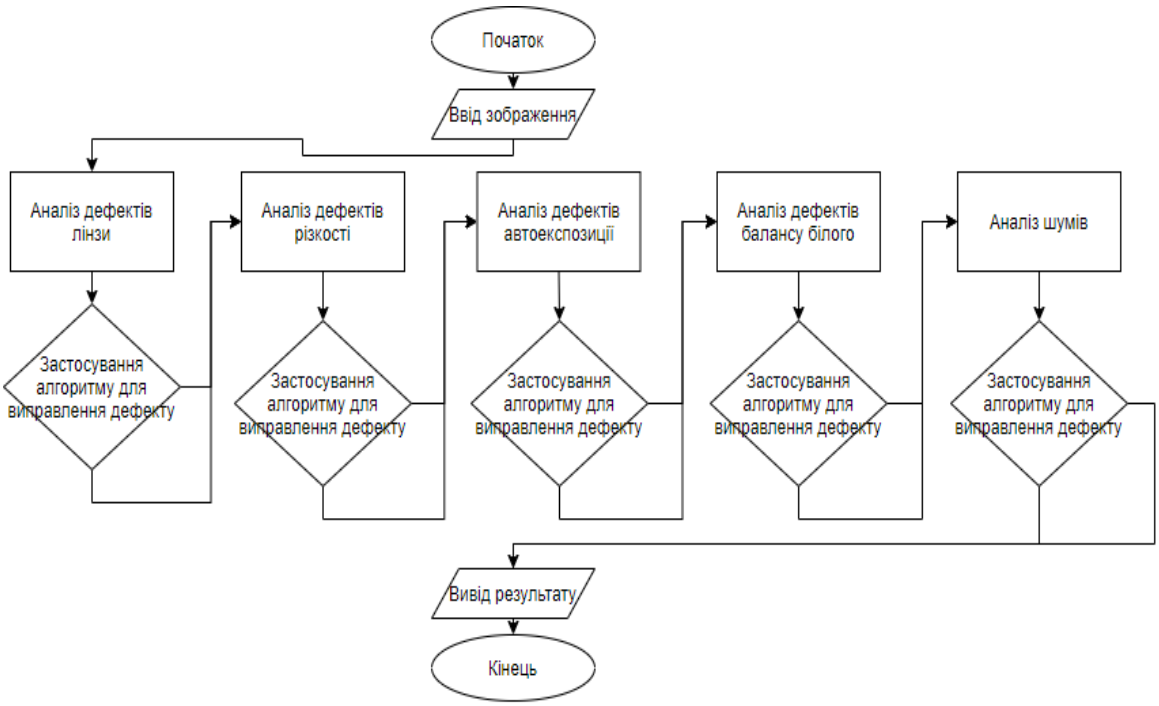

Рисунок Г.3 – Схема алгоритму роботи програмного модуля пошуку дефектів на зображеннях

Рисунок Г.4 – Діаграма діяльності модуля пошуку та аналізу дефектів на зображеннях

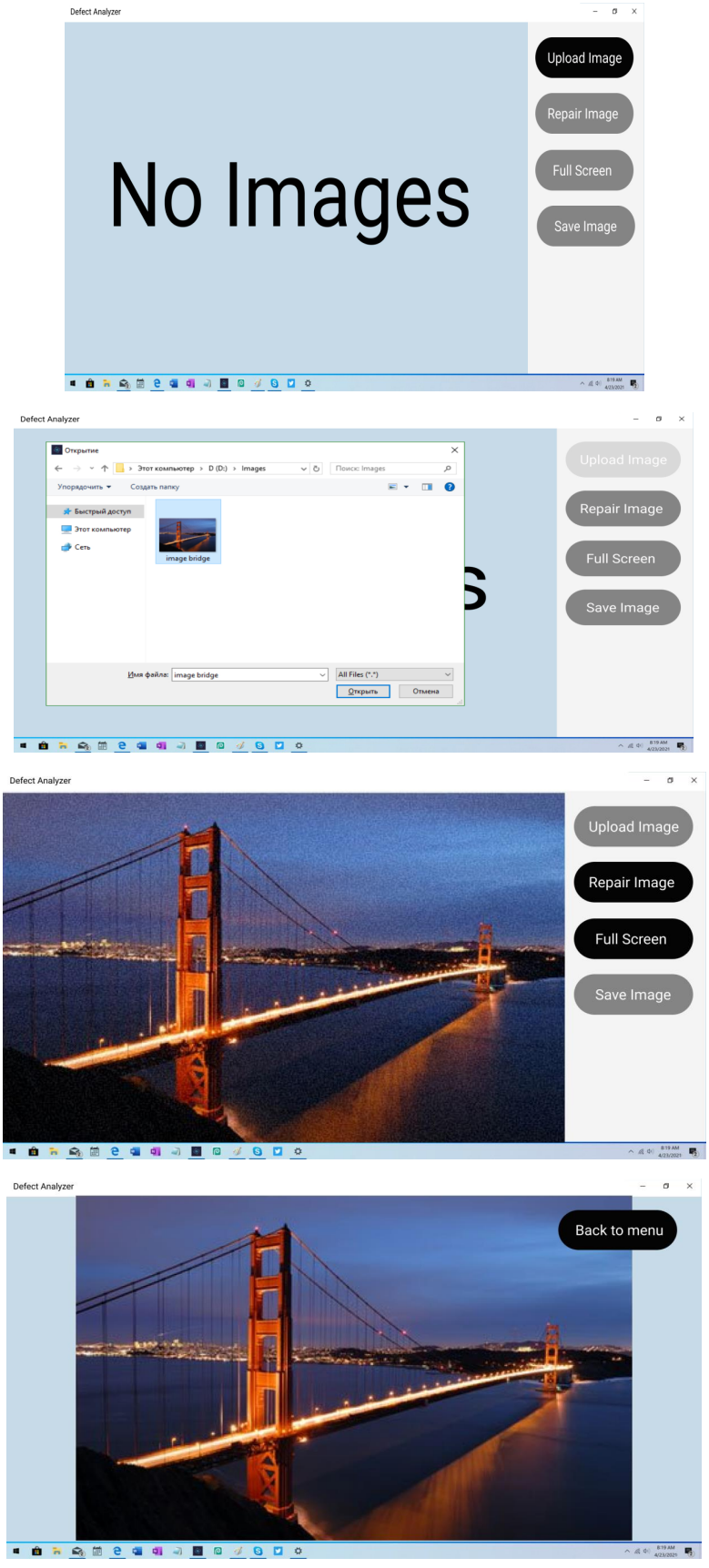

Рисунок Г.5 – Інтерфейс користувача програмного модуля пошуку дефектів на зображеннях