МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Обробка зображень із застосуванням технології Open CV**

Виконав: студент 4 курсу, групи 6.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
С.В. Бондаренко
(ініціали та прізвище)
Керівник доцент, к.т.н., доцент В.І. Заяц
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем__
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т. В. Критська
“_01_” __березня_____2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бондаренку Сергію Владиславовичу
(прізвище, ім’я, по батькові)

1.Тема роботи Обробка зображень із застосуванням технології Open CV

керівник роботи Заяц Валерій Іванович, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від від 29.12.2022 №1893-с

2. Строк подання студентом кваліфікаційної роботи 14.06.2023
3. Вихідні дані кваліфікаційної роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення інформаційних систем;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- тестування програмної системи та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	01.03-10.03.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.03-12.03.23	виконано
3	Аналіз існуючих методів рішення	13.03-14.03.23	виконано
4	Аналіз методів розробки	15.03-20.03.23	виконано
5	Аналіз сучасних систем обробки зображень	21.03-26.03.23	виконано
6	Узгодження подальших дій з науковим керівником	27.03-28.03.23	виконано
7	Проектування інформаційної системи	29.03-05.04.23	виконано
8	Програмна реалізація застосунку	06.04-16.04.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	17.04-19.04.23	виконано
10	Реалізація користувацького інтерфейсу для комп'ютерної системи	20.04-01.05.23	виконано
11	Тестування застосунку	02.05-7.05.23	виконано
12	Оформлення звіту	8.06-15.06.23	виконано
13	Оформлення презентації. Отримання рецензій від опонентів.	16.06-20.06.23	виконано

Студент _____ С.В. Бондаренко
(підпис) (прізвище та ініціали)

Керівник роботи _____ В.І. Заяц
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 67

Рисунків – 28

Джерел – 16

Бондаренко С.В. Обробка зображень із застосуванням технології Open CV: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. Керівник В.І. Заяц Запоріжжя : ЗНУ, 2023. 67 с.

Кваліфікаційна робота присвячена вивченню технології OpenCV та її застосування в задачах обробки зображень. У роботі було проведено аналіз можливостей технології та її алгоритмів для обробки зображень. Було досліджено алгоритми Canny та Sobel для визначення контурів об'єктів на зображеннях.

Для реалізації дослідження було використано мову програмування C++ та бібліотеку OpenCV. Були розроблені програмні модулі для визначення контурів об'єктів на зображеннях за допомогою алгоритму Canny та Sobel. Були проведені тести та аналіз результатів роботи програмних модулів.

У роботі було показано, що технологія OpenCV є потужним інструментом для обробки зображень. Алгоритми Canny та Sobel дозволяють ефективно визначати контури об'єктів на зображеннях. Результати тестів підтверджують високу якість роботи програмних модулів, розроблених з використанням технології OpenCV.

Кваліфікаційна робота містить докладний опис процесу створення програмних модулів, дослідження алгоритмів Canny та Sobel, результати тестування та аналіз отриманих результатів. Робота є внеском у розвиток науки та технологій на полі обробки зображень з використанням OpenCV.

Ключові слова*: обробка зображень, OpenCV, знаходження контурів на зображенні, алгоритм Кенні, фільтри зображення, фільтр гауса.*

ABSTRACT

Pages – 67

Drawings – 28

Sources – 16

Bondarenko S.V. Image processing with the use of Open CV technology: bachelor's qualification thesis of the specialty 121 "Software engineering" / Science. Head V.I. Zayats Zaporizhzhia: ZNU, 2023. 67 p.

The qualification work is devoted to the study of OpenCV technology and its application in image processing tasks. The paper analyzed possible technologies and their algorithms for image processing. The Canny and Sobel algorithms for determining object boundaries in images were investigated.

The C++ programming language and the OpenCV library were used to implement the research. Software modules were developed for determining object boundaries in images using the Canny and Sobel algorithm. Tests and analysis of the results of the software modules were carried out.

The work showed that OpenCV technology is a powerful tool for image processing. Canny and Sobel algorithms allow you to effectively determine the boundaries of objects in images. The test results confirm the high quality of the software modules developed using OpenCV technology.

The qualification work contains a report description of the process of creating software modules, Canny and Sobel research algorithms, test results and analysis of the obtained results. The work is beyond the scope of science and technology development in the field of image processing using OpenCV.

Keywords*: image processing, OpenCV, image edge detection, algorithm Canny, image filters, Gaussian filter.*

ЗМІСТ

ВСТУП

Актуальність теми

Тема обробки зображень є важливою в сучасному науково-технічному прогресі. Це пов'язано з швидким зростанням обсягів графічної та фотоінформації в усіх сферах людської діяльності. Обробка зображень здатна значно покращити якість, зменшити обсяг та забезпечити високу швидкість роботи з графічними даними.

На сьогоднішній день, оброблені зображення використовуються в багатьох галузях людської діяльності, включаючи медицину, науку, техніку, архітектуру та дизайн. У медицині, обробка зображень дозволяє діагностувати різні патології та здійснювати точні інтервенції з високим рівнем точності. У науці та техніці, обробка зображень використовується для аналізу складних структур та для розробки нових матеріалів в області нанотехнологій. У дизайні та архітектурі, обробка зображень дозволяє створювати віртуальні моделі будівель та об'єктів, що допомагає планувати будівництво та моделювати різні варіанти дизайну.

Таким чином, тема обробки зображень є актуальною завдяки широкомасштабному застосуванню цієї технології та необхідності в постійному покращенні якості та швидкості роботи з графічною та фотоінформацією. Відповідно, розвиток цієї технології сприятиме подальшому розвитку науково-технічного прогресу та покращить якість життя людей.

Мета дослідження

Розробка програми для редагування зображень із використанням технології OpenCV.

Завдання дослідження

Розглянути особливості та можливості технології OpenCV,. Проаналізувати існуючі підходи та методи створення тих, або інших

функціональних можливостей та обрати відповідні, для створення застосунку редагування та обробки зображень.

Об’єкт дослідження

Об’єктом дослідження є процес розробки застосунку для обробки та редагування зображень та відео.

Предмет дослідження

Предметом дослідження є технології та методи створення застосунку для обробки та редагування зображень та відео.

Методи дослідження

Аналіз і порівняння методів, технологій та їх результатів, для реалізації різних функціональних можливостей обробки зображень, за допомогою OpenCV.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що використання даної технології прискорить та спростить роботу пов’язаною із обробкою зображень. Результати дослідження можуть бути використані для створення різноманітних застосунків , від редакторів зображень до складних систем оцінки якості , або розпізнавання та ідентифікації особи, або об’єкту, що посприяє розвитку галузі і можливості конкурувати на міжнародному ринку.

Глосарій

Контур — зовнішня межа об'єкту на зображенні бінарного кольору (чорний і білий) або зображенні з градаціями кольору, яка відображає межі об'єкту.

Роздільна здатність екрану — це кількість пікселів, яку може відобразити екран у певному напрямку на дюйм.

Колірний простір — це математична модель, яка дозволяє представити кольори у вигляді чисел або векторів.

Конвертування у градації сірого — процес перетворення кольорового зображення в зображення, що містить тільки відтінки сірого кольору.

Апертура — прямокутна область навколо поточного пікселя, що бере участь у перетворенні.

Конкатенація — об'єднання двох і більше зображень в одне зображення.

Колбек функція — функція, яка передається іншій функції як аргумент і викликається згідно з певними умовами або подіями, що відбуваються в програмі.

Трекбар — елемент інтерфейсу користувача, який дозволяє вибирати значення числової величини, переміщуючи повзунок по шкалі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд джерел

Обробка зображень є важливою складовою сучасного науково-технічного прогресу. Технологія обробки зображень стає все більш розвиненою та значущою завдяки росту обсягів графічної та фотографічної інформації в усіх сферах нашої діяльності. Використання зображень забезпечує можливість розв'язання різноманітних завдань від діагностики хвороб до побудови графічних інтерфейсів. У цьому контексті, використання відкритої бібліотеки OpenCV для розробки технології обробки зображень є актуальним предметом дослідження.

OpenCV виступає як бібліотека комп'ютерного зору, що надає функціональність для обробки зображень та відео, які можуть бути використані для вирішення різноманітних задач у багатьох галузях діяльності. Найпоширенішими задачами, що можуть бути вирішені з використанням OpenCV, є визначення облич людей, розпізнавання об'єктів, детекція осіб та розпізнавання руху.

1.2 Огляд методів та алгоритмів обробки зображень.

Під обробкою зображень звичайно розуміють деякий набір математичних перетворень вхідних даних про графічну інформацію. Конкретна форма цих перетворень визначається кінцевою метою задачі.

Багато галузей техніки, що стосуються отримання, обробки, зберігання і передачі інформації, значною мірою орієнтуються нині на розвиток систем, у яких інформація має характер зображень. Зображення, яке можна розглядати як двовимірний сигнал є значно більш повним носієм інформації, ніж звичайний одномірний об’єкт. Разом з тим, вирішення наукових та інженерних задач під час роботи з візуальними даними вимагає особливих

зусиль, що спираються на застосуванні спеціальних методів, оскільки традиційні методи роботи з одновимірними об’єктами мало придатні у цих випадках.

Розглянемо основні методи і алгоритми, які найбільш часто зустрічаються в задачах обробки зображень [1,12,15].

1.2.1 Інтенсивність пікселя

Визначення інтенсивності пікселя є процесом вимірювання яскравості конкретного пікселя на цифровому зображенні. Піксель цифрового зображення має тільки одне значення – колір.

В задачах обробки зображень обсяг обчислень, а отже і час обробки, суттєво скорочується , якщо основною характеристикою пікселя вважати інтенсивність. Інтенсивність пікселя зазвичай визначається за його значенням яскравості, яке може бути представлено числовим значенням в межах визначеної кількісної шкали.

Процес визначення інтенсивності для кожного пікселя полягає в отриманні значення кольору даного пікселя та його конвертації у числове значення.

Наприклад, для зображень у градаціях сірого, значення інтенсивності пікселя може бути подане відносними значеннями від 0 до 255, де 0 представляє чорний колір, а 255 - білий колір.

У кольорових зображеннях інтенсивність кожного пікселя зазвичай розраховується як середнє значення інтенсивності окремих каналів кольору, таких як червоний, зелений та синій.

Інші колірні простори можуть мати різні методи визначення інтенсивності пікселів, що залежать від їх особливостей та властивостей.

Функція типу повертає значення пікселя - колір.

C=GetPixel(x,y).

Інтенсивність кожної складової можна отримати як

I_r = GetRValue(C); I_g = GetGValue(C); I_b = GetBValue(c) .

Тоді загальна інтенсивність для цього пікселя дорівнює

$$I_s = 0{,}299\, I_r + 0{,}587\, I_g + 0{,}114\, I_b$$

1.2.2 Фільтрація зображень

При обробці та аналізі зображень виділяють такі основні етапи: фільтрація, попередня обробка, сегментація, розпізнавання, діагностика. Ефективність наступних етапів обробки зображення безпосередньо залежить від результатів фільтрації та попередньої обробки. Етап фільтрації необхідний для пом'якшення різних дефектів.

Фільтри бувають багатьох типів, включаючи низькочастотні, високочастотні, медіанні, адаптивні та інші типи цифрових фільтрів [8].

Лінійна фільтрація широко використовується в цифровій обробці зображень. Це досягається за рахунок використання швидкого алгоритму згортки. Статична фільтруюча маска, не завжди забезпечує прийнятні результати, бо лінійний фільтр згладжує різницю в яскравості, що ускладнює виявлення контуру.

Нелінійна фільтрація має багато переваг перед лінійною. Згладжування різниці яскравості менше, контури об’єктів можна знайти точніше, а імпульсні поміхи усуваються.

Інший тип фільтрації - адаптивна фільтрація. Цей тип фільтрації має багато переваг, таких як локальна фільтрація, зміна локальної маски фільтра та розміру апертури фільтра.

Методи, які використовуються на етапі попередньої обробки, дуже різноманітні залежно від дослідницького завдання. Це може бути виділення найбільш інформативного фрагмента, його збільшення, корекція кольору, зміна просторової роздільної здатності, зміна контрастної роздільної здатності тощо. Однією з основних дій, які виконуються на етапі попередньої обробки, є зміна контрастності та яскравості зображення. Методи зміни контрастності

та яскравості поділяються на лінійні, нелінійні та адаптивні. З відповідними масками два етапи (етап фільтрації та етап попередньої обробки можна поєднати для збільшення швидкості). Етап попередньої обробки також включає геометричні операції над зображенням. До них відноситься, обертання зображення, масштабування зображення та зменшування зображення.

Розглянемо основні методи фільтрації зображень. Якщо для обчислення значення інтенсивності кожного пікселя враховуються значення сусідніх пікселів в деякому околі, то таке перетворення називається локальним, а сам окіл — вікном, яке є деякою матрицею, яка має називається маскою, фільтром, ядром фільтра, оператором згортки, а самі значення елементів матриці називаються коефіцієнтами.

Маска зазвичай є двовимірним квадратним вікном, яке рухається по зображенню, впливаючи лише на один піксель за раз. Кожне число у фільтрі розглядається як коефіцієнт. Центр маски співпадає з аналізованим пікселем, а коефіцієнти маски множаться на значення інтенсивності пікселів, накритих маскою. Коефіцієнти фільтрації визначають ефективність фільтрації та зовнішній вигляд отриманого зображення. Як правило, маска має квадратну форму розміру 3×3, 5×5 тощо.

Фільтрування зображення I, що має розміри $M \times N$, за допомогою маски розміру $(2m+1) \times (2n+1)$ описується формулою:

$$\tilde{I}_{ij} = \frac{1}{(2n+1)(2m+1)} \sum_{l=-n}^{n} \sum_{k=-m}^{m} w(l,k) I_{i-lj-k},$$

де l та k - індекси елементів маски щодо її центру (у центрі l = k = 0).

Такі перетворення називаються лінійними.

Після обчислення нового значення інтенсивності пікселя $\tilde{I}_{ij}$ вікно w, в якому описана маска фільтра, зсувається і обчислюється інтенсивність наступного пікселя, тому подібні перетворення називається фільтрацією у вікні, що ковзає .

Згортка (англ. convolution) - це операція, що показує «схожість» однієї функції з відбитою та зсунутою копією іншої. У разі роботи із зображеннями згортка - це операція обчислення нового значення заданого пікселя, при якій враховуються значення сусідніх пікселів, що його оточують.

Якщо зображення містить крім основного ідеального зображення деякий випадковий шум типу гаусового шуму, поліпшення зображення можна досягти застосуванням фільтра, що згладжує. Гаусовий шум описується нормальним розподілом з математичним очікуванням, рівним нулю, та середньоквадратичним відхиленням σ. Найпростіший варіант згладжуючого фільтра містить одиниці для кожного пікселя, що потрапляє у вікно. Розмір вікна зазвичай може бути 3x3 чи 5x5.

Під час накладання вікна на зображення виконується обчислення нового значення пікселя, що знаходиться у центрі вікна.

Для цього виконується підсумовування всіх пікселів у вікні та ділення отриманої суми на 9 (вікно 3x3) або 25 (вікно 5x5). Тобто нове значення є середнє арифметичне значення всіх пікселів у вікні.

Якщо застосувати фільтр до пікселів, які лежать на межі зображення, частина фільтра буде виходити за межі області зображення. Тоді навколо зображення можна додати «рамку». Цей процес називається *padding* (Рис. 1).

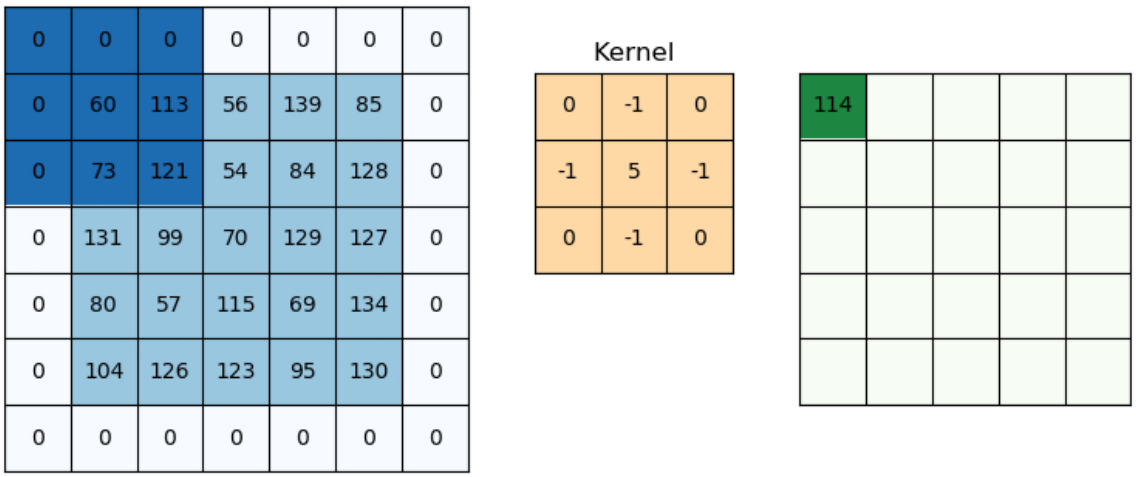

Рис. 1 *Візуалізація педдінгу зображення*

При цьому нові межові пікселі заповнюються за певними правилами. Зокрема, є такі варіанти:

- Наповнення нулями.
- Наповнення постійними значеннями (наприклад, числом 5).
- Наповнення значенням найближчого сусіднього пікселя.
- Наповнення повторенням останнього рядка (стовпця).
- Наповнення останнього рядка першим, або навпаки.

У математиці функції можна класифікувати на два великі класи: лінійні та нелінійні. Функція називається лінійною, якщо має місце

$$f(\alpha x+\beta y)=\alpha f(x)+\beta f(y)$$

для будь-яких змінних x, y і будь-яких коефіцієнтів α, β. Функція, яка не є лінійною, називається нелінійною.

Лінійний фільтр представляє конкретне узагальнення лінійної функції. Одним із найпоширеніших лінійних фільтрів є фільтр усереднення. Наприклад, ядро для фільтра усереднення (Рис. 2) розмірністю 5×5 має вигляд:

$$K = \frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Рис. 2 *Ядро фільтру усереднення 5x5*

1.2.3 Фільтр Гауса

Фільтр Гауса - це стандартний фільтр для зменшення шуму в зображенні за допомогою згладжування контурів

При використанні фільтра Гауса, який відрізняється тим, що вагові коефіцієнти фільтра залежать від відстані до центрального пікселя, виходить найкраща якість згладжування.

Ці коефіцієнти зменшуються зі збільшенням відстані. У цьому випадку найбільший вплив на результат мають ближні пікселі, а найменше – більш віддалені.

Чим більший радіус апертури, тим більше область згладжування. Дисперсія вказує на ширину функції Гауса, якою він використовується для згладжування зображення (Рис. 3). Вона визначає, як далеко розташовані точки від центру отриманих на вихідному зображенні.

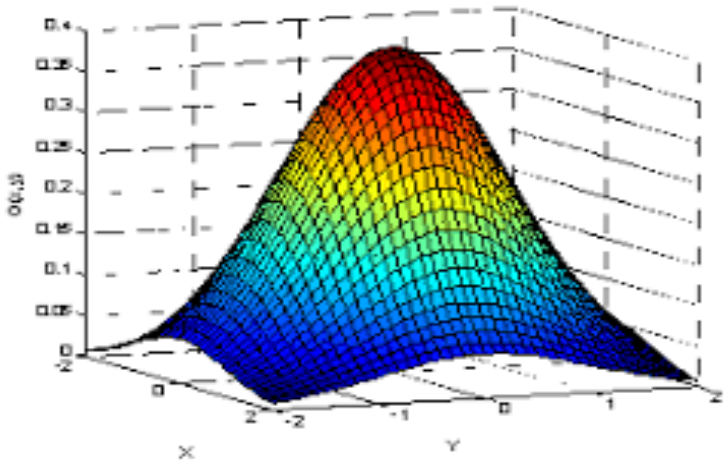

Рис. 3 *Тривимірна візуалізація функції Гауса*

Вагова функція розраховується за формулою:

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Коефіцієнти фільтрів Гауса повинні бути нормовані таким чином, щоб їх сума була рівна одиниці (Рис. 4).

0.0947416	0.118318	0.0947416
0.118318	0.147761	0.118318
0.0947416	0.118318	0.0947416

Рис. 4 *Матриця 3×3 фільтру Гауса* ($\sigma = 1{,}5$)

Головна перевага фільтра Гауса полягає в його здатності до зменшення шуму у зображенні за рахунок виключення нерегулярних елементів. Так, зображення з маленькими дільницями можуть бути згладжені, але це зробить лінії менш виразними та зменшить деталізацію (Рис. 5). На відміну від цього, фільтр Гауса дозволяє зберігати деталі та робити зображення більш реалістичним.

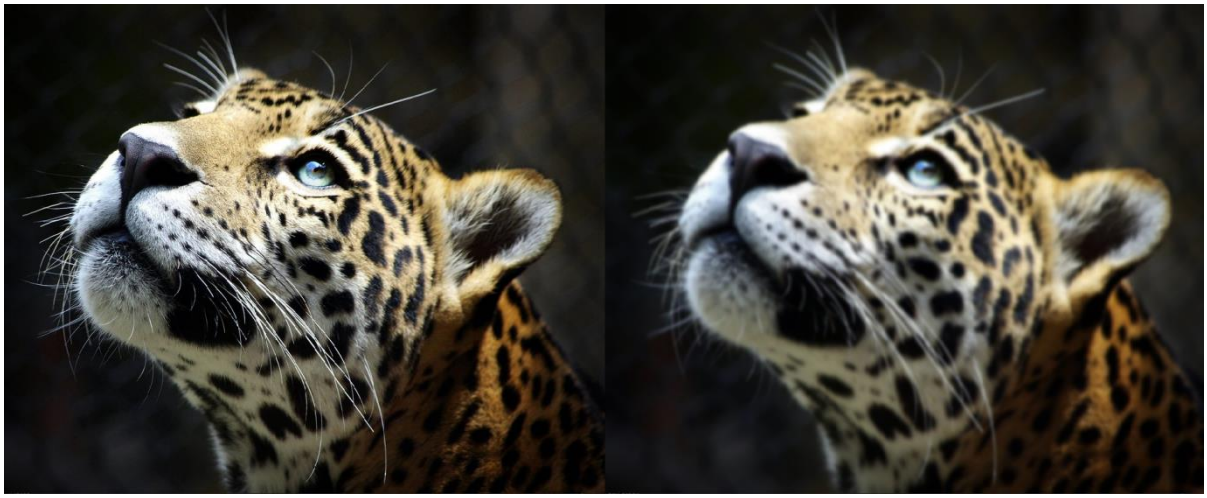

Рис. 5 *Демонстрація фільтру Гауса*

Також, слід зазначити, що фільтр Гауса має певні недоліки, зокрема проблему із заплющенням контурів на зображенні. Це може стати причиною втрати деяких деталей та структур в зображенні

Отже, аналізуючи фільтр Гауса, ми можемо зазначити, що він є ефективним інструментом для зменшення шуму та поліпшення якості зображення. Він має свої недоліки, які слід враховувати при його використанні. Але при правильному застосуванні, фільтр Гауса може покращити якість зображення, зробити його більш реалістичним та зберегти важливі деталі для подальшого аналізу.

1.2.4 Оператор Лапласа

Оператор Лапласа - це математичний оператор, який широко використовується в області обробки зображень та аналізу даних. Оператор Лапласа дозволяє отримати другий похідний значення функції зображення, що дає змогу визначити локальні зміни у градієнтах та контурах на зображенні.

Підхід зводиться до вибору дискретної формулювання другої похідної та побудови фільтра, заснованої на даному формулюванні. Як правило застосовуються ізотропні фільтри які є інваріантними до повороту - поворот зображення і подальше застосування фільтра дає той же результат, що і початкове застосування фільтра з подальшим поворотом результату.

Найпростішим ізотропним оператором є лапласіан

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

Дискретна форма двовимірного лапласіана має вигляд:

$$\nabla^2 f = f_{ij-1} - 2 f_{ij} + f_{ij+1} + f_{i-1j} - 2 f_{ij} - f_{i+1j}.$$

Ця формула може бути записана у формі маски (згортки), яка дає ізотропний результат поворотів на кути, кратні 90° :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (*)$$

Співвідношення (*) можна узагальнити , якщо до формули дискретного лапласіана включити діагональні напрямки. Маска, що відповідає такому визначенню, є ізотропною для поворотів на кути, кратні 45 і має вигляд

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Наступні дві маски також часто використовуються на практиці. Вони основані на визначенні лапласіана, що є «негативним» стосовно того, що вже було розглянуто. Фактично вони дають ідентичний результат, але різницю у знаку має враховуватися при комбінації – операцією складання чи віднімання – зображення, відфільтрованого лапласіаном, з іншим зображенням.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Оператор Лапласа, будучи другою похідною, акцентує увагу на різницях рівнів яскравості на зображенні та пригнічує області з незначними змінами яскравості. Це може призвести до отримання зображення, що містить сіруваті лінії на місцях контурів та інших розривів, які накладаються на темний фон без яскравих особливостей (Рис. 6).

Рис. 6 *Демонстрація фільтру лапласа*

Однак, з відновленням зображення відновлюється фон, зберігши при цьому ефект підвищення різкості, що досягається за допомогою лапласіана. Це можна зробити шляхом додавання лапласіана до вихідного зображення. При цьому необхідно пам'ятати, яке визначення лапласіана було використане.

Якщо під час використання оператора Лапласа відповідне визначення включало негативні центральні коефіцієнти, то для отримання ефекту підвищення різкості необхідно відняти зображення-лапласіан, а не додавати його.

Таким чином, узагальнений алгоритм використання лапласіану для покращення зображень зводиться до наступного:

$$\tilde{I}_{ij} = \begin{cases} I_{ij} - \nabla^2 I_{ij}, w(0,0) < 0 \\ I_{ij} + \nabla^2 I_{ij}, w(0,0) > 0 \end{cases}.$$

Тут w(0,0) – значення центрального коефіцієнта маски лапласіана.

1.2.5 Сегментація

В основному результатом аналізу зображення є якість його сегментації, яка задається виділенням областей із певними характеристиками. Загальний результат виділення деталей залежить від конкретної задачі, тому методи та алгоритми сегментації повинні бути добре узгоджені з особливостями аналізованого зображення. Для кращого результату, можливо потрібно використати послідовні різні методи сегментації.

Відповідно до математичного апарату, методи сегментації класифікуються на морфологічну, порогову та нарощування областей. Сегментація є важливою задачею для аналізу медичних зображень, оскільки має на меті виділення областей, які відповідають різним об'єктам та структурам на зразок біологічної тканини. Однак, вимоги до сегментації в цій області є складним завданням через малий контраст і мінливість властивостей тканин.

До переваг морфологічної сегментації можна віднести її здатність розпізнавати та обробляти області з різною формою і розміром. Порогова сегментація є простою та швидкою методою, яка може бути ефективною для зображень з великим контрастом. Об’єднання (нарощування) областей є хорошою альтернативою при використанні порогової сегментації при вирішенні завдань з обробки медичних зображень.

Недоліки методів сегментації полягають в тому, що немає єдиного універсального методу, який зможе забезпечити ефективну сегментацію всіх типів зображень. Деякі методи можуть не показувати ефективних результатів при роботі з великими зображеннями або в разі внесення шумів у зображення. Також, методи сегментації можуть не забезпечувати належної точності

1.2.6 Виділення контурів на зображенні

Основним інструментом обробки зображень є виявлення контурів, яке здійснюється шляхом пошуку точок на цифровому зображенні, де яскравість різко змінюється, а також розривів. Метою цього процесу є зменшення кількості даних у зображенні зі збереженням структурних властивостей для подальшої обробки. Зазвичай, підходи до виявлення контурів поділяються на методи першого і другого порядку, які використовують перші і другі похідні інтенсивності.

Перший підхід полягає у пошуку максимуму градієнту, тоді як другий заснований на фільтрі Лапласа шукає піксель, в якому значення лапласіана функції інтенсивності змінює знак. Однак, щоб підтвердити наявність контуру, необхідно перевірити, що в цій точці досягається локальний максимум градієнту.

Головною математичною ідеєю всіх методів виявлення контурів є використання градієнту функції двох змінних для кожної точки зображення. Градієнт це двовимірний вектор, який складається з похідних інтенсивності зображення по горизонталі та вертикалі.

$$G(g_x, g_y) \qquad g_x = \frac{\partial I}{\partial x},\ g_y = \frac{\partial I}{\partial y}.$$

У кожній точці зображення градієнтний вектор має напрямок, що збігається з напрямом найбільшого зростання яскравості, водночас його довжина відповідає величині зміни яскравості.

Модуль градієнта

$$\|G\| = \sqrt{g_x^2 + g_x^2}.$$

Напрямок градієнта

$$\Theta = \arctan \frac{g_y}{g_x}.$$

Це означає, що результатом застосування диференціальних операторів в точці області постійної яскравості буде нульовий вектор, а в точці, що лежить на межі областей різної яскравості - вектор, що перетинає межу у напрямку збільшення яскравості. Напрямок градієнта завжди є перпендикулярним до межі.

1.2.7 Градієнт растрових зображень

Градієнт растрових зображень - це континуальна зміна кольорів та тонів спочатку від одного колірного тону до іншого, а потім до наступного. Градієнт може бути лінійним або радіальним, залежно від того, як точки кольору розміщені на зображенні.

Важливо розуміти, що градієнт - це не просто наступність кольорових тонів, а складний алгоритм, який зберігає кожну точку кольору на зображенні. Градієнт легко створювати за допомогою графічних редакторів та програм обробки зображень.

Один з найважливіших аспектів роботи з градієнтом - це збереження гладкості та контрастності кольорів на зображенні. Також важливо з

естетичної точки зору користуватися відповідними кольоровими схемами, аби затримати погляд глядача та викликати потрібні враження на зображенні.

Для обчислення градієнта застосовуються дискретні диференціальні оператори [14]. Основною проблемою чисельного диференціювання растрових зображень є те, що значення інтенсивності задається у центрах пікселів, а значення градієнтів - у кутах.

Розглянемо два оператора, які дозволили розв’язати цю проблему. Робертс запропонував обчислювати часткові похідні за допомогою центральних різниць в системі координат, нахиленій до оригінальної під кутом $\frac{\pi}{4}$.

Оператор Робертса - це один з перших відомих та застосовуваних диференціальних операторів для обчислення градієнту на растрових зображеннях, запропонований у 1963 році Л. Робертсом [13].

Даний оператор є дискретним та використовується для визначення градієнту на зображенні шляхом обчислення різниці між інтенсивністю двох сусідніх пікселів по діагоналі.

Зазвичай оператор Робертса використовують в квадратній формі розміром 2x2 (Рис. 7) маючи на меті детектувати локальні зміни інтенсивності на малих осередках зображення.

Необхідно відзначити, що оператор Робертса має певні обмеження в застосуванні, оскільки він не забезпечує максимальної точності в обчисленні градієнту на зображенні. Проте, він має свої переваги, такі як швидкодія та чутливість до дрібних змін інтенсивності на зображенні, що робить його придатним для використання в багатьох випадках.

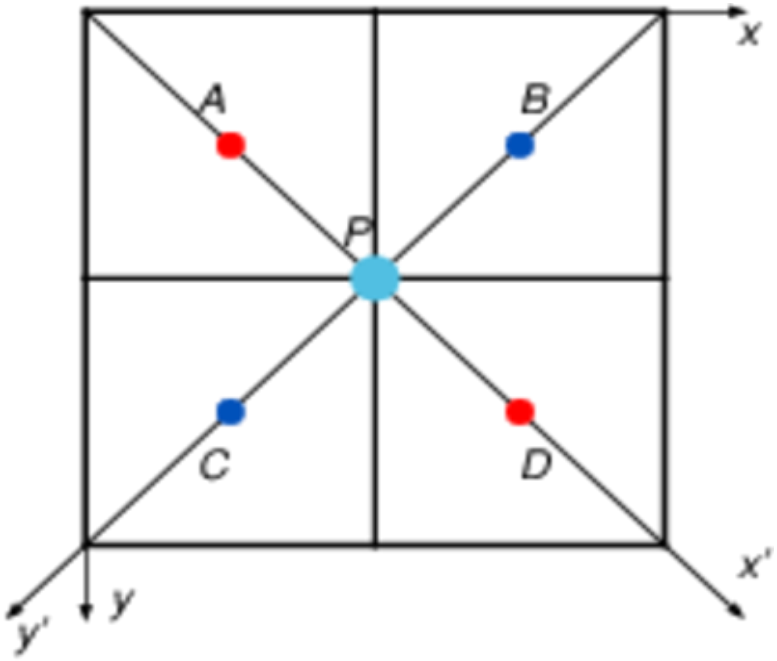

Рис. 7 *Оператор Робертса*

$$I_{x'P} = \frac{I_D - I_A}{\sqrt{2}\Delta} + O(2\Delta^2) \qquad I_{y'P} = \frac{I_C - I_B}{\sqrt{2}\Delta} + O(2\Delta^2).$$

Оператор Робертса**.** Плюси - простота і, як слідство, хороша швидкодія. Мінуси - маленьке ядро, і, як слідство, велика чутливість до шуму.

Оператор Собеля - це диференціальний оператор, що використовується для обчислення градієнту на растрових зображеннях [11]. Оператор Собеля має вигляд як піксельний фільтр розміром 3x3 (Рис. 8), складається з двох фільтрів, що обчислюють градієнти за осями x та y.

Оператор Собеля широко застосовується в обробці зображень для визначення контурів та областей змінних градієнтів на зображенні. Це досягається шляхом обчислення величини градієнту на кожному пікселі шляхом обчислення квадратного кореня суми квадратів градієнтів по осям x та y.

Один з головних переваг оператора Собеля полягає в його здатності виявляти дрібні деталі на зображенні та мінімізувати помилки, пов'язані з шумом на зображенні.

Оператор Собеля також має просту будову, що дозволяє швидко та ефективно виконувати обчислення в багатьох програмах обробки зображень.

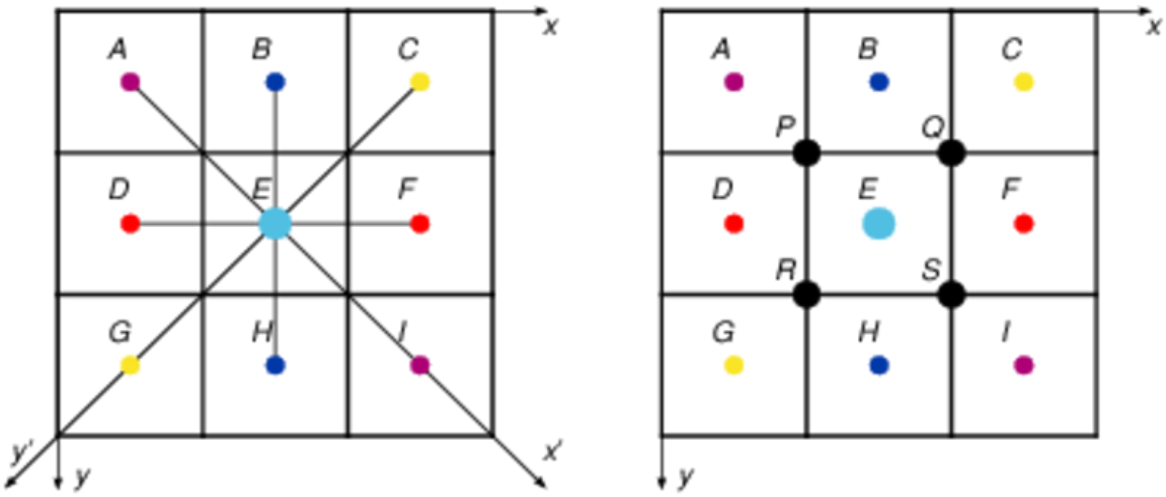

Рис. 8 *Оператор Собеля*

Можемо обчислити градієнт у точці E як центральну різницю

$$I_{xE} = \frac{I_F - I_D}{2\Delta} + O(4\Delta^2),\ I_{yE} = \frac{I_H - I_B}{2\Delta} + O(4\Delta^2) \quad (1)$$

Якщо провести осі координат через точки AI і CG, то можна обчислити центральні різниці в новій координатній системі як

$$I_{x'E} = \frac{I_I - I_A}{2\sqrt{2}\Delta} + O(8\Delta^2),\ I_{y'E} = \frac{I_G - I_C}{2\sqrt{2}\Delta} + O(8\Delta^2) \quad (2)$$

Формули (1) і (2) - це два різних наближених обчислення однієї і тієї ж величини. Щоб зробити обчислення більш ізотропними, треба просто усереднити результати (1) і (2). Особливістю цього усереднення є те, що потрібно знайти середнє від двох векторів, заданих у різних системах координат. Тому спочатку потрібно перерахувати вектор (2) у системі xy за допомогою формул повороту координатної системи на кут $\frac{\pi}{4}$, а потім скласти їх.

Матрична форма операторів Робертса і Собеля

Оператор Робертса:

$$D_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad D_y = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}.$$

Оператор Собеля:

$$D_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad D_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Оператор Собеля.

Переваги: оператор має велике ядро, що дає меншу похибку і робить оператор менш сприйнятливим до шуму.

Недоліки: потрібно більше часу на обчислення, порівняно з оператором Робертса.

1.2.8 Алгоритм Кенні

Знаходження контуру є важливою задачею в обробці зображень та комп'ютерному зорі. Одним з найефективніших методів знаходження контурів є алгоритм Кенні, розроблений Джоном Кенні у 1986 році [3]. Цей алгоритм забезпечує високу точність визначення контурів та знижує кількість шуму на зображенні.

Даний алгоритм дозволяє не тільки визначати граничні пікселі, а й зв'язати їх в граничні лінії. Алгоритм складається з наступних кроків:

-Усунення дрібних деталей шляхом згладжування вихідного зображення за допомогою фільтра Гауса.

-Використання диференціального оператора Собеля для визначення значень напрямку градієнта всіх пікселів, причому результат обчислення округляється з кроком $\frac{\pi}{4}$.

-Аналіз значень модулів градієнта пікселів, розташованих в напрямку вектора градієнту. Якщо значення модуля градієнта досліджуваного пікселя більше, ніж у сусідніх пікселів у цьому напрямку, то він є граничним, у противному випадку - немаксимумом.

-Пікселями контуру оголошуються пікселі, в яких досягається локальний максимум градієнту в напрямку вектора градієнту.

-Виконання подвійної порогової фільтрації граничних пікселів, відібраних на попередньому кроці:

-Якщо значення модуля градієнта вище за поріг t_t , то наявність контуру в пікселі є достовірною.

-Якщо значення модуля градієнта нижче порога t_b, то піксель однозначно не є граничним.

-Якщо значення модуля градієнта лежить в інтервалі $[t_b, t_t]$ то такий піксель вважається неоднозначним.

-Придушення всіх неоднозначних пікселів, не пов'язаних з достовірними пікселями по 8-зв'язності.

-Якщо якийсь фрагмент контуру не містить жодного пікселя, інтенсивність якого більше верхнього порогу,то цей фрагмент вилучається повністю.

-Придушення всіх неоднозначних пікселів, не пов'язаних з достовірними пікселями по 8-зв'язності.

-В якості порогових значень інтенсивності звичайно приймаються

$$t_t \approx 0,7\mathrm{I}_{\max}; \quad t_b \approx 0,5\mathrm{I}_{\max}$$

-Якщо якийсь фрагмент контуру не містить жодного пікселя, інтенсивність якого більше верхнього порогу, то цей фрагмент вилучається повністю.

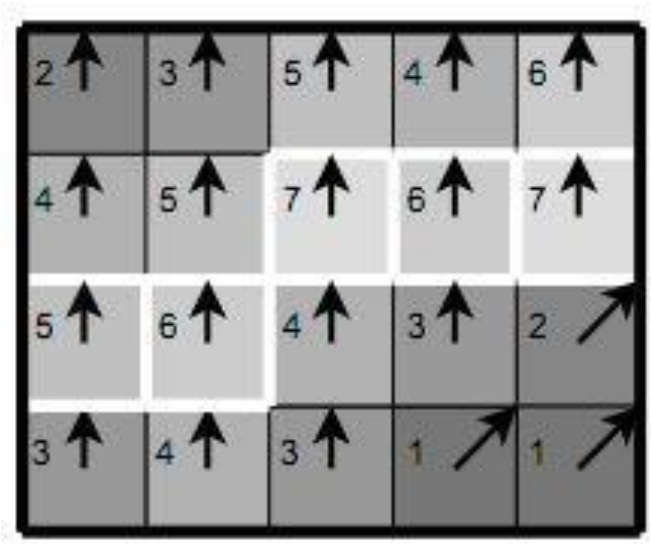

Рис. 9 *Принцип придушення немаксимумів*

Майже всі пікселі в прикладі «мають орієнтацію вгору», тому значення градієнта в цих точках буде порівняно з нижче і вище розташованими пікселями. Обведені білим контуром пікселі залишаться в результуючому зображенні, решта будуть придушені (Рис. 9).

Алгоритм Кенні забезпечує відмінну точність та ефективність визначення контурів на зображенні. Цей алгоритм знайшов своє застосування в багатьох галузях, включаючи медичну діагностику, автоматичну обробку зображень та комп'ютерний зір.

Види редагування зображень:

1. Ретушування зображень - видалення дефектів зображення, плям, подертостей тощо з метою покращення зовнішнього вигляду зображення.

2. Реставрація зображень - відновлення пошкоджених частин зображення, підвищення якості та збільшення роздільної здатності.

3. Обрізання зображень - видалення непотрібних елементів зображення та виділення головного об'єкту.

4. Корекція кольору - забезпечення збалансованої передачі кольорів зображення.

Цілі редагування зображень:

1. Покращення якості зображення - мета більшості обробок зображень, включаючи ретушування, реставрацію та корекцію кольору.

2. Розпізнавання об'єктів у зображеннях - обробка зображення може використовуватися для визначення та розпізнавання об'єктів на зображенні, що дозволяє досягти точнішого аналізу данних.

3. Видалення шумів з зображення - допомагає збільшити якість зображення та отримати точніші дані для аналізу.

4. Відновлення пошкоджених зображень - реставрація зображення дозволяє відновити пошкоджені частини, що може бути важливо для аналізу інформації на зображенні.

Такі види та цілі обробки зображень відіграють значну роль для інженерів з розробки програмного забезпечення, оскільки вони дозволяють отримати якісні дані для подальшого аналізу та використання.

1.3 Аналіз існуючих програмних засобів обробки зображень

Обробка зображень є важливою складовою сучасного науково-технічного прогресу. Технологія обробки зображень стає все більш розвиненою та значущою завдяки росту обсягів графічної та фотографічної інформації в усіх сферах нашої діяльності. Використання зображень забезпечує можливість розв'язання різноманітних завдань від діагностики хвороб до побудови графічних інтерфейсів. У цьому контексті, використання відкритої бібліотеки OpenCV для розробки технології обробки зображень є актуальним предметом дослідження [2,5,9,10].

OpenCV виступає як бібліотека комп'ютерного зору, що надає функціональність для обробки зображень та відео, які можуть бути

використані для вирішення різноманітних задач у багатьох галузях діяльності. Найпоширенішими задачами, що можуть бути вирішені з використанням OpenCV, є визначення облич людей, розпізнавання об'єктів (Рис. 10), детекція осіб та розпізнавання руху.

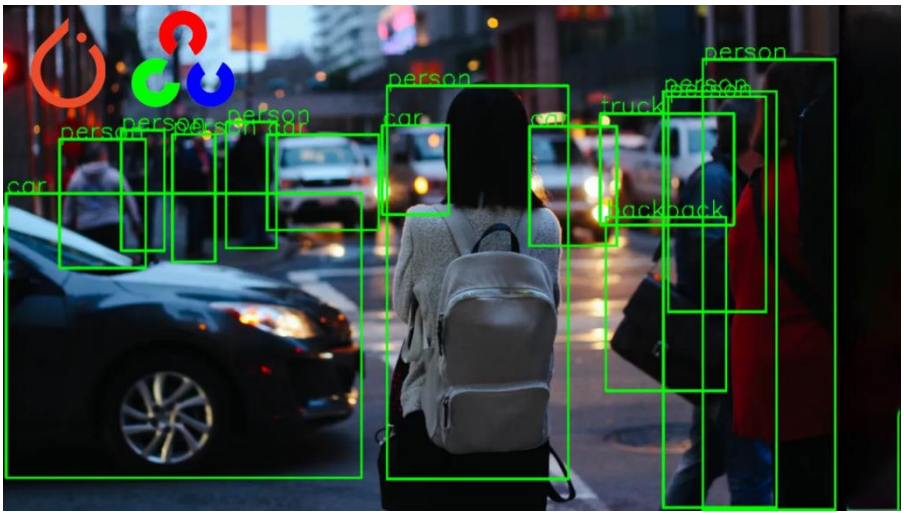

Рис. *10 Розпізнавання об’єктів з використанням OpenCV*

Застосування OpenCV може допомогти розробити технології для зниження шуму (Рис.11) та поліпшення якості зображень, що є важливою задачею для багатьох галузей діяльності, включаючи медицину та науку. Прикладами такої обробки зображень є рентгенівські знімки, мікроскопічні зображення та зображення з космосу.

Рис. *11 Усування шуму з використанням OpenCV*

OpenCV також здатний забезпечити точність (Рис.12) під час обробки зображень. Наприклад, його можна використовувати для детектування та класифікації елементів на зображеннях під час аналізу різних видів даних, включаючи дані медичного обладнання та сканування генетичних матеріалів.

Рис. 12 *Точне знаходження елементів з використанням OpenCV*

Отже, використання відкритої бібліотеки OpenCV для розробки технології обробки зображень є важливим кроком у сучасному науково-технічному прогресі. На сьогоднішній день OpenCV є однією з найпотужніших бібліотек комп'ютерного зору, що надає розробникам можливість створювати якісні та швидкі технології обробки зображень для різноманітних задач у багатьох галузях діяльності.

1.3.1 Бібліотека MATLAB

Розробка технологій обробки зображень є важливою складовою сучасного науково-технічного прогресу. В одному з напрямків розвитку знаходиться відкрита бібліотека комп'ютерного зору OpenCV, яка дозволяє розробникам створювати високоякісні та швидкі системи обробки зображень.

Аналогами OpenCV є інші бібліотеки, такі як MATLAB, ImageJ, SciPy і інші. Однак, лише OpenCV надає потужні засоби для обробки зображень на досить високому рівні, що робить її однією з найефективніших на ринку.

MATLAB (MATrix LABoratory) є переможцем серед програм для чисельного аналізу, який спеціалізується на обробці даних, моделюванні, аналізі, тестуванні та розв'язанні складних проблем. MATLAB включає в себе інтерактивне середовище, яка дозволяє використовувати мову високого рівня для виконання операцій з матрицями, обчислень комплексних чисел, аналізу даних та візуалізації результатів [6].

MATLAB обладнана готовими пакетами для чисельних обчислень, що робить її значно більш придатною для використання в аналізі імовірності та статистики. Також, MATLAB має вбудовані рішення для роботи зі звуком та зображеннями (Рис.13), що дозволяє розв'язувати складні задачі в таких областях, як обробка зображень та обробка сигналів.

У підсумку, MATLAB є могутньою інструментальною програмою, яка є особливо корисною для проведення чисельних розрахунків.

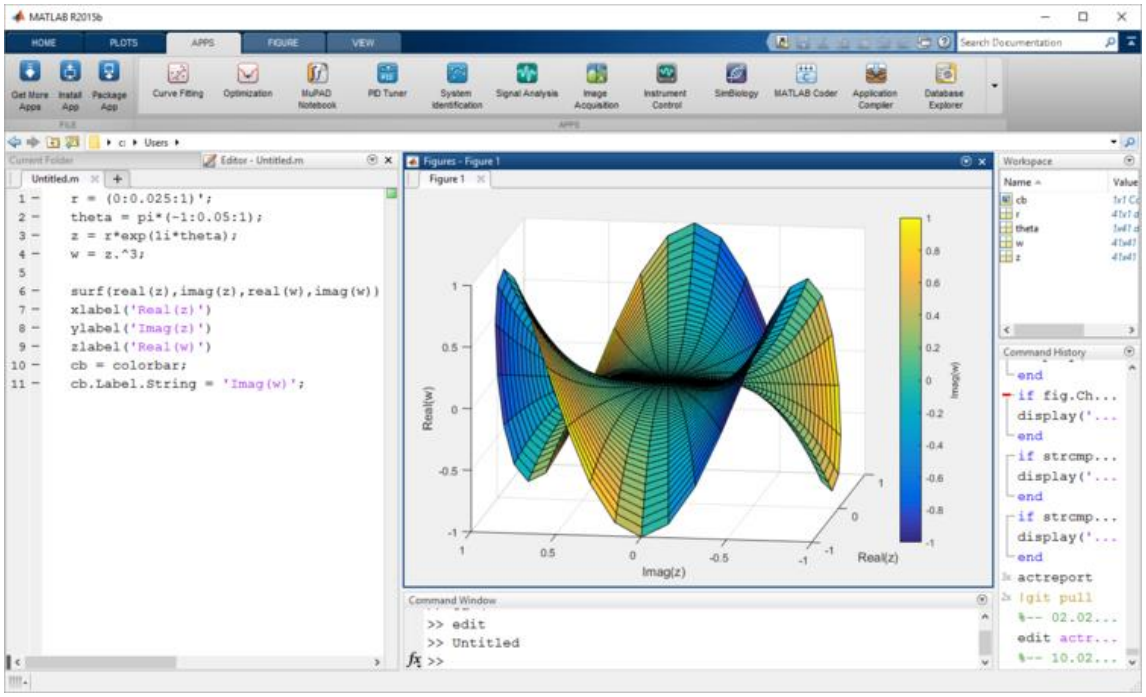

Рис. 13 *Процес роботи у програмі MATLAB*

1.3.2 Інструментарій ImageJ.

ImageJ - це відкритий і безкоштовний інструмент для аналізу та обробки зображень, який розроблений на платформі Java[7,16]. Він може використовуватися для обробки зображень з мікроскопів, магнітно-резонансної томографії, комп'ютерної томографії та інших джерел зображень. ImageJ орієнтований на наукових дослідників, лабораторії та клінічні дослідження у біомедичній галузі.

Однією з найбільших переваг ImageJ є його відкритість та безкоштовність. Це дозволяє дослідникам і лабораторіям які займаються біомедичною тематикою ефективно використовувати його для своїх досліджень, навіть з обмеженим бюджетом. ImageJ містить безліч вбудованих модулів, які дозволяють проводити різноманітні дослідження, від простої фільтрації зображень до складних аналізів даних.

ImageJ є легким та простим у використанні інструментом для аналізу зображень. Його інтерфейс досить простий, що дозволяє користувачам проводити аналіз зображень без складнощів. Також, ImageJ підтримує різноманітні формати зображень та може працювати з великими об'ємами даних (Рис.14).

Однак, ImageJ має й деякі недоліки. Наприклад, його масштабний аналіз вимагає високої обчислювальної здатності, особливо, якщо об'єми даних дуже великі. Також, ImageJ не підтримує роботу з 3D-зображеннями та більш вимірними даними.

У загальному, ImageJ є корисним інструментом для задач аналізу та обробки зображень в наукових дослідженнях, особливо у біомедичній галузі. Він є відкритим та безкоштовним, легким у використанні та має широкі можливості для аналізу зображень. Однак, його обмеження у роботі з великими об'ємами даних та невідповідність за роботу з 3D-зображеннями можуть стати перешкодою для використання в деяких дослідженнях.

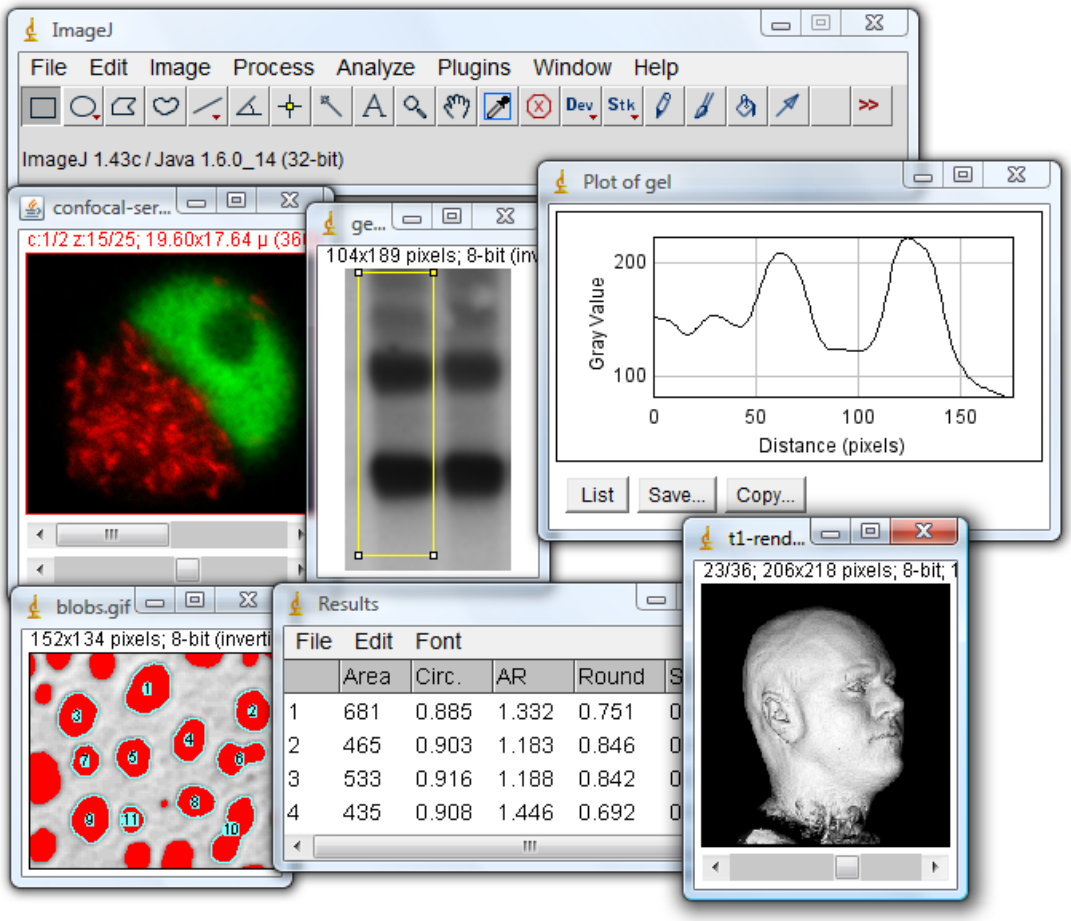

Рис. 14 *Процес роботи у програмі ImageJ*

1.3.3 Технологія PyCuda

Іншим аналогом, який можна порівняти з OpenCV, є PyCuda [4]. PyCuda пропонує аналогічні можливості обробки даних, що і OpenCV, за допомогою технології GPU. Однак, використання PyCuda не дає тієї ж широти функцій з обробки зображень, як OpenCV. Крім того, PyCuda потребує обов'язкового використання GPU, що не може бути доступним на деяких пристроях.

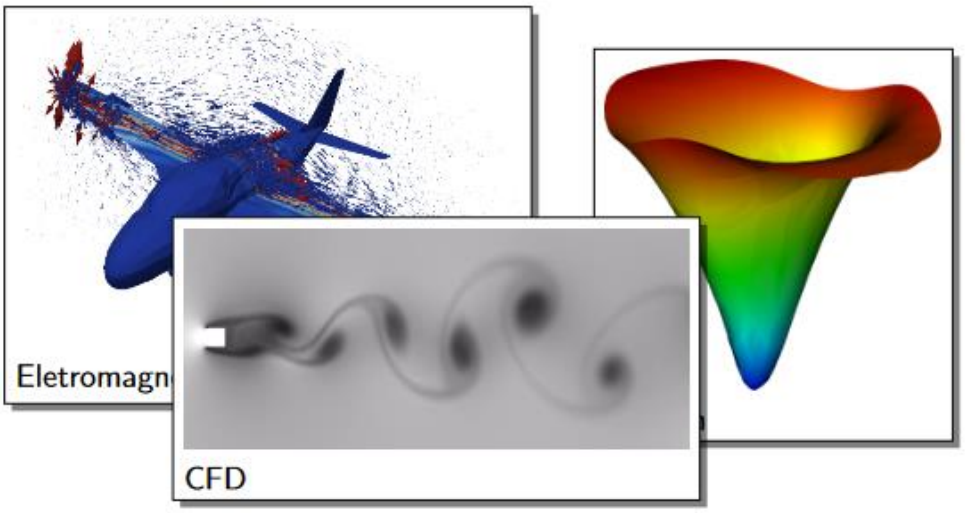

Рис. 15 *Результат роботи з використанням PyCuda*

PyCuda є високопродуктивним інструментарієм для обчислення на відеокартах, побудованим на базі бібліотеки CUDA від NVIDIA, яка дозволяє розробникам використовувати пакетні функції CUDA на мові програмування Python. Технологія основана на використанні можливостей відеокарти для розпаралеленого обчислення, що дозволяє значно прискорити обчислення в ряді задач, зокрема, в тих, що потребують довгої обробки великих обсягів даних.

Використовуючи PyCuda, розробники можуть реалізувати широкий спектр задач обробки даних (Рис.15), від простих математичних операцій до складних алгоритмів машинного навчання. Технологія дозволяє працювати з багатовимірними масивами даних і матрицями, а також використовувати різні оптимізаційні техніки, що робить її вельми ефективною у виконанні обчислювально важких операцій.

PyCuda має безкоштовний відкритий код та широку спільноту розробників, що активно розвивається і приносить нові функції і можливості. Технологія дозволяє розробникам максимально використовувати можливості їхніх відеокарт, що робить її цікавим варіантом для проектів, які потребують виконання значної кількості обчислень.

В цілому, PyCuda є ефективним інструментом для виконання обчислювальних завдань, зокрема тих, що стосуються машинного навчання та аналізу даних, тому він може бути використаний у різноманітних проектах, що потребують швидкого та ефективного обчислення.

Що до ImageJ та MATLAB. Обидві бібліотеки є широко використовуваними засобами для обробки зображень в академічних дослідженнях та промисловості. MATLAB надає користувачеві розширені можливості по роботі з масивами даних, в той час, як ImageJ спеціалізується на аналізі даних мікроскопічного рівня.

Однак, OpenCV в порівнянні з цими бібліотеками дозволяє розробникам більш легкі та прості інтерфейси для роботи з зображеннями, та дозволяє відносно легко інтегрувати сторонні програмні модулі. Багато з існуючих систем безпосередньо підтримують OpenCV, що дозволяє полегшити роботу з нею для розробників.

Отже, при порівнянні з іншими засобами OpenCV є однією з найбільш потужних бібліотек обробки зображень для усіх сфер діяльності. Завдяки широкому функціоналу, простому інтерфейсу і свого роду інтеграції до середовища роботи, OpenCV використовується у багатьох проектах, які стосуються обробки зображень на високому рівні

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Технологія OpenCV

OpenCV (Open Source Computer Vision Library) - бібліотека відкритого коду, яка включає в себе набір алгоритмів комп'ютерного зору та обробки зображень. Ця бібліотека забезпечує надзвичайно потужні та ефективні засоби для обробки та аналізу зображень, що зробило її невід'ємною частиною в сучасних проектах з обробки зображень.

OpenCV розроблена на мові програмування C++ та має широкий спектр функцій, що дозволяє виконувати різні завдання з обробки зображень: від простого зчитування та запису зображень до складного аналізу та розпізнавання об'єктів на зображеннях.

Основні риси OpenCV такі:

1. Підтримка багатьох платформ: OpenCV може використовуватися на Windows, Linux, macOS та Android.

2. Велика підтримка форматів зображень: OpenCV підтримує багато різних форматів зображень, таких як JPG, PNG, BMP, TIFF і т.д.

3. Гнучкість: OpenCV надає користувачам велику гнучкість вибору методів та алгоритмів, що використовуються для обробки зображень.

4. Ефективність: OpenCV забезпечує швидку та ефективну обробку зображень завдяки оптимізації використання ресурсів системи.

5. Відкритий код: OpenCV піддається розширенням та модифікаціям, що дозволяє видатними спеціалістам у галузі комп'ютерного зору створювати власні алгоритми та методи.

OpenCV застосовується в різних галузях, таких як: медицина, промисловість, наука та інженерія, автомобільна промисловість, логістика та інші. OpenCV дозволяє досягати високого рівня точності та ефективності

Інтеграція з різними бібліотеками, такими як NumPy, Python дозволяє OpenCV виконувати структурований аналіз. Щоб ідентифікувати форму

зображення та його різні характеристики, використовується векторний простір і виконуються математичні операції над цими функціями.

Ліцензія OpenCV дозволяє використовувати будь-яку версію OpenCV, що розповсюджується під цією ліцензією, для будь-яких цілей, включаючи комерційну та некомерційну експлуатацію. Крім того, ліцензія дозволяє модифікувати вихідний код OpenCV та розповсюджувати модифікований код за умови, що будь-які зміни, внесені до вихідного коду OpenCV, повинні бути вказані та відзначені у вихідному коді модифікованого продукту.

Ліцензія OpenCV також дозволяє використовувати сторонні бібліотеки, які були розповсюджені на умовах відкритої ліцензії, разом з OpenCV. Крім того, ліцензія дозволяє поєднувати OpenCV з іншими продуктами та використовувати OpenCV у складі комерційних продуктів.

У загальному, ліцензія OpenCV дозволяє вільне використання та розповсюдження програмного забезпечення для обробки зображень та комп'ютерного зору, зберігаючи при цьому авторські права. Ця ліцензія забезпечує велику свободу використання OpenCV без необхідності оплачувати ліцензійні внески або інші обмеження.

Основні модулі OpenCV наведені нижче:

1. Core — це основний модуль OpenCV, який надає функції для роботи з матрицями, які є базовими структурами даних.

2. Imgproc — це модуль для обробки зображень та більшість алгоритмів, що пов’язані з ними. Він надає функції для зміни розміру та зміни формату зображень, висвітлення та зміни контрастності, виявлення контурів та багатьох інших.

3. Video — модуль OpenCV, призначений для опрацювання відео та аудіо з використанням різних технологій.

4. Highgui — модуль для роботи з вікнами та графікою. Він надає функції для створення та керування вікнами, для читання та записування зображень та відео.

5. Machine Learning — модуль, що містить багато алгоритмів для машинного навчання, такі як розпізнавання об’єктів та класифікація.

6. Features2D — модуль, який містить алгоритми для знаходження ключових точок та дескрипторів на зображеннях. Він також дозволяє знаходити відповідності між дескрипторами на двох зображеннях.

7. Objdetect — модуль, присвячений розпізнаванню об’єктів та використанню каскадів визначення ознак для детекції об’єктів.

8. Calib3d — модуль, що містить функції для калібрування камер та роботи з відеостереосистемами.

У цьому переліку основних модулів OpenCV не вичерпується існуючий функціонал бібліотеки. OpenCV має величезну базу функцій та інструментів, завдяки чому вона знаходить широке застосування для вирішення завдань у галузях комп’ютерного зору та обробки зображень.

2.2 OpenCV Mat

Mat — це основний клас для обробки та зберігання даних , який складається із двох частин: заголовок матриці (містить таку інформацію, як розмір матриці, метод, що використовується для зберігання, за якою адресою зберігається матриця тощо) і вказівник на матрицю, що містить значення пікселів (з будь-якою розмірністю залежно від обраного методу зберігання). Розмір заголовка матриці є постійним, однак розмір самої матриці може відрізнятися від зображення до зображення і зазвичай більший в рази.

Клас Mat представляє n-вимірний щільний числовий одноканальний або багатоканальний масив (Рис.16). Його можна використовувати для зберігання дійсних або комплексних векторів і матриць, зображень у градаціях сірого або кольорових зображень, воксельних об’ємів, векторних полів, хмар точок, тензорів, гістограм (хоча гістограми з дуже великими розмірами краще зберігати в SparseMat).

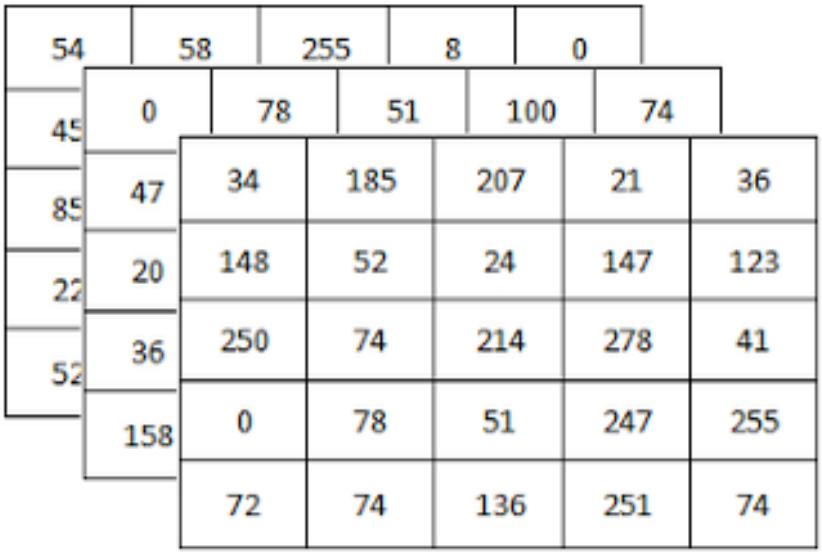

Рис. 16 *Візуалізація збереження даних в 3-х канальній матриці*

Отже, макет даних у Mat повністю сумісний із типами CvMat, IplImage та CvMatND із OpenCV 1.x. Він також сумісний із більшістю типів щільних масивів зі стандартних наборів інструментів і SDK, таких як Numpy (ndarray), Win32 (незалежні растрові зображення пристрою) та інші. Завдяки цій сумісності можна створити заголовок Mat для призначених користувачем даних і обробляти їх на місці за допомогою функцій OpenCV.

2.3 OpenCV базові операції із зображенням

Зображення представлені у вигляді матриць, (або масив матриць, як у випадку кольорового зображення), операції над матрицями також стосуються зображень. Ви можете змінити пікселі зображення, виділяти окремі ділянки зображення тощо. Так само змініть значення пікселя на Виділіть окремі області зображення. ми теж можемо це зробити На зображенні також можна використовувати розділення каналів зображення, або склеювання, тощо. Якщо до всіх елементів зображення додати число, то збільшується інтенсивність (зображення стає світлішим), при відніманні - зменшується. Ви також можете додати саме зображення. При цьому використовується формула:

$$g(x)=(1-\alpha)f0(x)+\alpha f1(x).$$

Для додавання зображень використовується функція cv2.addWeighted, яка додає зображення за формулою

$$img=\alpha\cdot img1+\beta\cdot img2+\gamma.$$

Фільтрація в OpenCV має функцію cv2.filter2D. Додатковий параметр borderType відповідає за педдінг цієї функції.

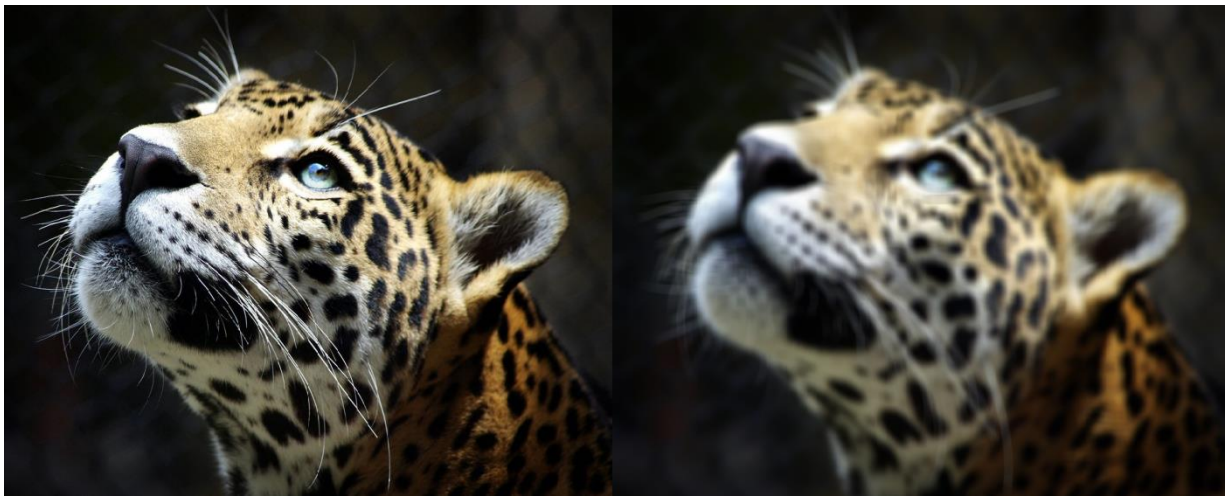

Рис. 17 *Демонстрація фільтру усереднення*

Цю саму операцію (фільтр усереднення) можна виконати за допомогою функцій cv2.blur або cv2.boxFilter. Ви також повинні одночасно встановити розмір фільтра.

Перевагами усереднювальних фільтрів є:

- Позбавлення шуму.
- Робить зображення яскравішим і покращує якість.

Недоліки:

- Розмиті контури зображення.
- Зменшує розкид значень пікселів.

На відміну від усереднюючого фільтра, медіанний фільтр зіставляє кожен піксель із середнім значенням його сусідів. Відповідна функція — cv2.medianBlur.

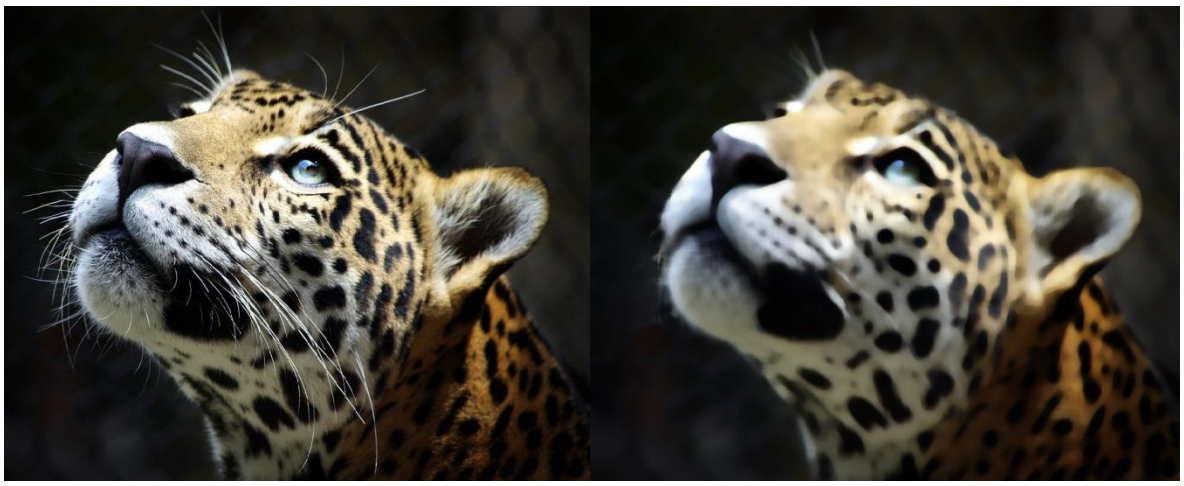

Рис. 18 *Демонстрація медіанного фільтру*

Фільтр «максимум» замінює кожен піксель максимальним значенням його сусідів. Функція OpenCV — cv2.dilate.

“Мінімальний” фільтр замінює кожен піксель мінімальним значенням у його околиці. Функція OpenCV — cv2.erode.

Над зображеннями можна також проводити побітові логічні операції та змінювати колірні моделі.

Колірна модель - це формальна система для представлення кольорів за допомогою чисел, символів або візуальних компонентів. Кожна колірна модель описує кольори за допомогою різних характеристик, таких як відтінок, насиченість та яскравість. Колірні моделі використовуються в різних областях, включаючи графіку, фотографію, друк і наукові дослідження.

Набір кольорів у колірній моделі разом із тим, як інтерпретувати ці дані (наприклад, завдання визначення умов для відтворення та/або відображення, тобто методи реалізації), визначає колірний простір.

Колірну модель також слід розуміти як спосіб представлення колірних гам у дискретній формі для представлення в обчислювальних і цифрових системах.

OpenCV в основному має справу з трьома кольоровими моделями: градації сірого (чорно-біле зображення), BGR і HSV (або HSB). Розглянемо ці моделі докладніше (Рис.19).

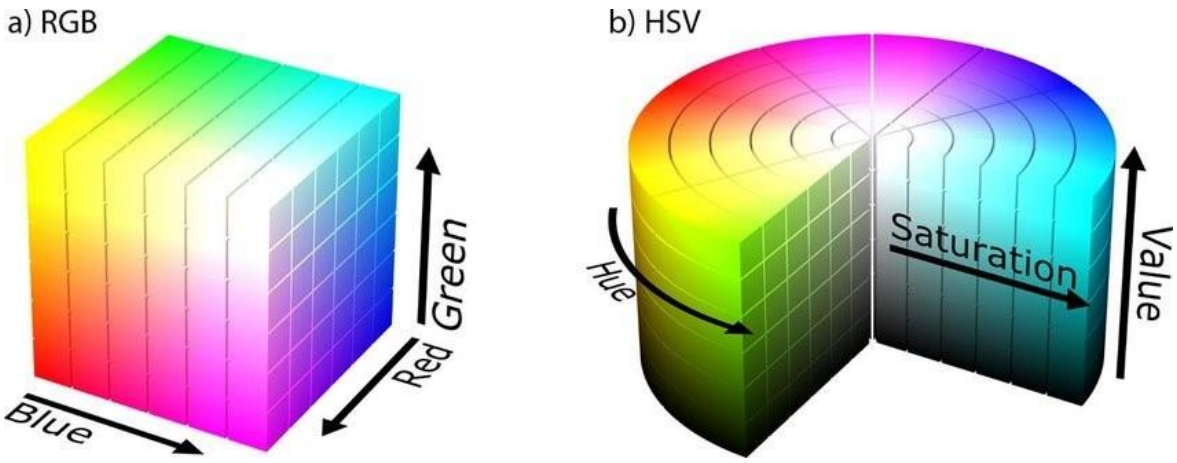

Рис. 19 *Об'ємна візуалізація колірних моделей RGB та HSV*

Grayscale — це модель, яка сумує всі кольори Інформація про відтінки сірого, тобто індивідуальна яскравість частина зображення. Ця модель дуже корисна при зборі інформації якщо яскравості достатньо (наприклад, при розпізнаванні обличчя). Кожен піксель зазвичай представлений 8 бітами: числом від 0 (чорний) до 255 (білий).

BGR — в цій моделі представлений кожен піксель як 3 значення. Кожне значення відповідає за окремий канал: синій, зелений, червоний. У цій моделі (0,0,0) позначає чорний колір, (255,0,0) — синій, (0,255,0) — зелений, (0,0,255) — червоний.

HSV (також HSB) — це колірна модель на основі трьох характеристик кольору: тон (Hue), насиченість (Saturation) і значення (Value) кольору, це також називається яскравістю (Brightness) у випадку HSB. Тон змінюється в проміжку [0,179], насиченість у [0,255] та значення кольору у проміжку [0,255]. Різні програми використовують різні шкали, тому може знадобитися нормалізація значення при обробці зображення в OpenCV.

.

3 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОБРОБКИ ЗОБРАЖЕНЬ

3.1 Вибір інструментарію розробки

OpenCV є однією з найбільш популярних та відкритих бібліотек для обробки зображень та комп'ютерного зору. Вона підтримує багато мов програмування, зокрема C++, Python та Java.

Мова програмування C++ є досить популярною серед розробників OpenCV. Це пов'язано з тим, що C++ є мовою низького рівня, яка забезпечує швидкість та ефективність виконання коду. Крім того, OpenCV написана на C++ та має багато прикладів коду для цієї мови, що робить її найпопулярнішим вибором для багатьох розробників.

Python також є дуже популярною мовою програмування для розробки зображень та комп'ютерного зору, включаючи використання OpenCV. Python є високорівневою мовою програмування, що робить її значно більш зручною для швидкого тестування та прототипування. Крім того, Python є більш доступною мовою програмування з погляду вивчення та розробки.

Java є третьою мовою програмування, яку підтримує OpenCV. Java є мовою програмування високого рівня та має добру підтримку в багатьох інших розробчих середовищах.

Перевагою використання кожної мови програмування є те, що вони дозволяють розробникам створювати різні за складністю та обсягом програми за допомогою OpenCV. Проте, кожна з вищезазначених мов має свої переваги та недоліки.

Переваги використання C++ включають:

-Високу ефективність та швидкість виконання коду, що робить її ідеальним вибором для більш вимогливих за ресурсами програм;

-Широку кількість керування оперативною пам'яттю, що дозволяє розробникам більш точно керувати ресурсами;

-Можливість використання широкого спектру додаткових бібліотек та інструментів.

Недоліки використання C++ включають:

-Більш складний синтаксис порівняно з іншими мовами програмування, що може збільшити час розробки;

-Складніше відлагодження коду.

Переваги використання Python включають:

-Простіший та більш доступний синтаксис, що дозволяє швидше розробляти та тестувати програми;

-Широке співробітництво з бібліотеками для наукових обчислень та машинного навчання, що робить її ідеальним вибором для розробки додатків, які включають машинне навчання.

Недоліки використання Python включають:

-Менша ефективність та швидкість виконання порівняно з C++;

-Менша контрольованість пам'яті порівняно з C++.

Переваги використання Java включають:

-Велику кількість бібліотек та інструментів, що дозволяє швидше розробляти програми;

-Простоту відлагодження коду.

Недоліки використання Java включають:

-Меншу ефективність та швидкість виконання порівняно з C++;

-Меншу контрольованість пам'яті порівняно з C++.

Для розробки застосунку було обрано мову програмування C++, в середовищі розробки Microsoft Visual Studio 2019, окрім перерахованих переваг та недоліків на вибір мови програмування також вплинув попередній досвід розробки на мові C++, зокрема у середовищі Microsoft Visual Studio 2019.

Для цілей розробки було використано основний модуль бібліотеки OpenCV. Основний модуль OpenCV включає в себе функції для роботи з зображеннями, такі як зчитування та запис зображень, обрізання, маскування,

перетворення, згладжування, детекція контурів і просторовий фільтринг. Він також містить інструменти для роботи з відео, такі як зчитування та запис відео, відтворення відео з файлів та потоків, обрізання, зміна розміру та інші функції.

3.2 Функціональні вимоги

Застосунок повинен забезпечувати :

Введення даних – можливість завантажити зображення за своїм вибором для подальшої обробки.

Можливість обрати серед різних функцій застосунку , потрібну для користувача на даний момент.

Застосунок повинен мати змогу відображати зображення, для перевірки результату.

3.3 Опис використаних класів та методів OpenCV

В даному підрозділі наведено опис основних класів, функцій і методів розробленої програми

Перш за все треба розглянути найважливіше, а саме читання та збереження зображення. OpenCV зберігає зображення під час зчитування у спеціальному класі *Mat*, що є сумісним з багатьма засобами обробки зображень, не враховуючи OpenCV, але в даному випадку будуть використовуватися базові засоби читання та збереження зображень , які пропонує бібліотека OpenCV.

Для читання зображень використовує функцію *imread.*

imread завантажує зображення з указаного файлу та повертає його, зберігаючи в результуючу матрицю, яку треба попередньо створити. Якщо зображення неможливо прочитати (через відсутність файлу, неправильні

дозволи, непідтримуваний або недійсний формат), функція повертає порожню матрицю (Mat::data==NULL).

Функція визначає тип зображення за вмістом, а не за розширенням файлу. У випадку кольорових зображень декодовані зображення матимуть канали, збережені в порядку B G R.

- Формати файлів які підтримує клас Mat:
- Растрові зображення Windows - .bmp, .dib
- Файли JPEG - .jpeg, .jpg, .jpe
- Файли JPEG 2000 - .jp2
- Портативна мережева графіка - .png
- WebP - .webp
- Портативний формат зображення - .pbm, .pgm, .ppm .pxm, .pnm
- Сонячні растри - .sr, .ras
- Файли TIFF - .tiff, .tif
- Файли зображень OpenEXR - .exr
- Radiance HDR - .hdr, .pic
- Растрові та векторні геопросторові дані, що підтримуються GDAL

Клас *imread* приймає два параметри, ім'я файлу та спосіб читання файлу.

Filename – ім’я файлу , який потрібно загрузити. Може приймати шлях до файлу, або, якщо файл знаходиться у папці з застосунком, можна використати назву файла, без вказання повного шляху до нього.

Flags – прапор що може нести у собі інформацію про режим зчитування файлу. Для базових дій із зображенням найчастіше використовується три варіації параметру:

IMREAD_UNCHANGED або -1 – вказує що зображення після зчитування не зміниться, та збереже усі канали, наприклад альфа канал, якщо він є, тобто зображення не буде обрізане , якщо на ньому є прозорі частини, а буде зчитано у повному розмірі.

IMREAD_GRAYSCALE або 0 – вказує що зображення буде зчитане в градаціях сірого. Під час використання IMREAD_GRAYSCALE буде

використано внутрішнє перетворення відтінків кодека, якщо воно доступне. Результати можуть відрізнятися від результатів cvtColor.

IMREAD_COLOR або 1 – вказує що зчитане зображення буде конвертовано у 3 канальне BGR кольорове зображення.

imshow Відображає зображення у вказаному вікні. Функція imshow відображає зображення у вказаному вікні. Якщо вікно було створено з прапорцем WINDOW_AUTOSIZE, зображення відображається в оригінальному розмірі, однак воно все ще обмежене роздільною здатністю екрана. В іншому випадку зображення масштабується відповідно до вікна.

За цією функцією має слідувати виклик waitKey або pollKey для виконання робочих завдань графічного інтерфейсу користувача, необхідних для фактичного показу даного зображення та змусити вікно реагувати на події миші та клавіатури. Інакше зображення не відображатиметься, і вікно може заблокуватися. Наприклад, waitKey(0) відображатиме вікно нескінченно до будь-якого натискання клавіші (це підходить для відображення зображення). waitKey(25) відобразить кадр і зачекає приблизно 25 мс для натискання клавіші (придатне для покадрового відображення відео). Щоб видалити вікно, використовуйте destroyWindow. *imshow* Може приймати два аргументи:

-*winname* – ім’я вікна в якому буде відображене зображення.

-*Mat* – Зображення яке буде відображено.

Деякі аспекти відображення залежать від вікна у якому зображення буде показане. Якщо вікно не було створено до функції, передбачається створення вікна з WINDOW_AUTOSIZE. Для створення спеціального типу вікна використовується функція *namedWindow*. Ця функція приймає два аргументи, ім’я вікна яке буде створене, та прапор , який визначає який тип вікна буде створено.

Якщо вам потрібно показати зображення, яке перевищує роздільну здатність екрана, вам потрібно буде викликати namedWindow("", WINDOW_NORMAL) .

Imwrite – Зберігає зображення у вказаний файл. Формат зображення вибирається на основі розширення імені файлу. Якщо формат зображення не підтримується, зображення буде перетворено на 8-бітне беззнакове (CV_8U) і збережено таким чином. Приймає 3 атрибути:

-*filename* – Ім'я файлу.

-*img* (Mat або вектор Mat) – Зображення або декілька зображень для збереження.

-*params* – Специфічні для формату параметри.

cvtColor – Перетворює зображення з одного колірного простору в інший. Ця функція перетворює вхідне зображення з одного колірного простору в інший. У разі перетворення в колірний простір RGB порядок каналів має бути вказано явно (RGB або BGR). Зверніть увагу, що стандартний формат кольору в OpenCV часто називають RGB, але насправді це BGR (байти перевернуті).

Аргументи:

-*Src* – вхідне зображення.

-*dst* – вихідне зображення того самого розміру та глибини, що й src.

-*code* – код перетворення простору кольорів

-*dstCn* – кількість каналів у образі призначення; якщо параметр дорівнює 0, кількість каналів виводиться автоматично з src і коду.

blur – Усереднюючий фільтр. Розмиває зображення за допомогою нормалізованого прямокутного фільтра. Функція згладжує зображення за допомогою ядра. Аргументи:

-*src* – вхідне зображення.

-*dst* – вихідне зображення того самого розміру та типу, що й src.

-*ksize* – розмір ядра.

-*anchor* – якірна опорна точка; значення за замовчуванням Point(-1,-1) означає, що прив’язка знаходиться в центрі ядра.

-*borderType* – режим межі, який використовується для екстраполяції пікселів за межі зображення.

medianBlur – Розмиває зображення за допомогою медіанного фільтра. Функція згладжує зображення за допомогою медіанного фільтра з діафрагмою ksiize×ksize. Кожен канал багатоканального зображення обробляється незалежно. Атрибути:

src – вхідне зображення.

dst – вихідне зображення того самого розміру та типу, що й src.

ksize – лінійний розмір діафрагми; воно має бути непарним і більше 1

GaussianBlur – Розмиває зображення за допомогою фільтра Гауса.

Атрибути:

src – вхідне зображення.

dst – вихідне зображення того самого розміру та типу, що й src.

ksize – гаусівський розмір ядра. Ширина ядра і висота ядра можуть відрізнятися, але обидва вони мають бути додатними та непарними. Або вони можуть дорівнювати нулю, і тоді вони обчислюються з сигми.

sigmaX – стандартне відхилення ядра Гауса в напрямку X.

sigmaY – стандартне відхилення ядра Гауса в напрямку Y; якщо sigmaY дорівнює нулю, він дорівнює sigmaX, якщо обидві сигми є нулями, вони обчислюються відповідно з ширини ядра і висоти ядра.

Canny – Знаходить контури на зображенні за допомогою алгоритму Canny. Функція знаходить контури у вхідному зображенні та позначає їх у краях вихідної карти за допомогою алгоритму Canny. Найменше значення між threshold1 і threshold2 використовується для зв’язування контурів. Найбільше значення використовується для знаходження початкових сегментів сильних ребер.

Атрибути:

- *image* – вхідне зображення.
- *edges* – вихідне зображення того самого розміру та типу, що й src.
- *threshold1* – перший поріг для процедури гістерезису.
- *threshold2* – другий поріг для процедури гістерезису.
- *apertureSize* – розмір діафрагми для оператора Sobel.
- *L2gradient* – прапор, який вказує, чи слід використовувати точнішу норму для обчислення градієнта зображення величини чи норми за замовчуванням достатньо.

3.4 Проектування застосунку

Після розглядання засобів та можливостей бібліотеки OpenCV, було спроектовано прототип застосунку для обробки зображень, а саме накладання фільтрів різних типів, усереднюючий фільтр, медіанний фільтр, фільтр Гауса, та фільтр збільшення чіткості зображення.

Серед інших функцій знаходження контурів зображення методом Кенні. Наступна діаграма (Рис.20) демонструє можливості програми, розробленої у середовищі Microsoft Visual Studio 2019, і передбачає використання консолі відлагодження Visual Studio в якості найпростішого інтерфейсу, для забезпечення користувача можливістю орієнтуватися та обирати необхідні функції програми.

Діаграма демонструє можливість програми попередньо надати вибір користувачу, якою функцією він буде користуватися: накладенням фільтру на зображення, або знаходження контурів.

У випадку зі знаходженням контурів, програма виводить оброблене зображення на екран і дає можливість налаштувати алгоритм Кенні, бо для кожного зображення необхідні свої параметри.

У випадку накладення фільтру, програма запропонує обрати тип фільтра, яким користувач хоче скористуватися, а вже потім виведе результат у вікно.

Після того як результат був отриманий , програма дає змогу вийти, або повторити процес.

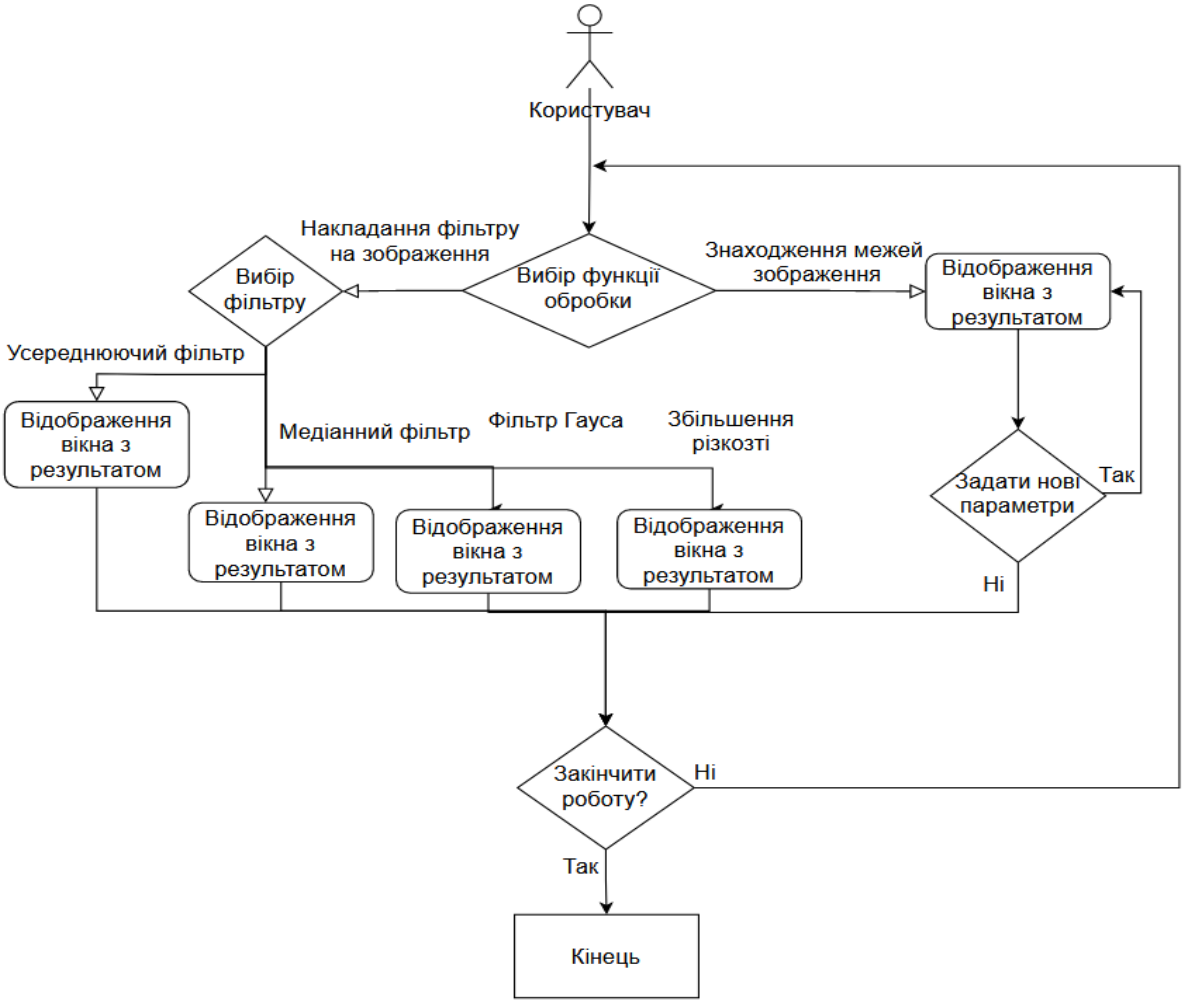

Рис. 20 *Діаграма UseCase*

3.5 Описання створених методів, та змінних

Глобальні змінні:

image_original — змінна типу Mat, використовується для збереження оригіналу зображення.

image_processed — змінна типу Mat, використовується для збереження обробленого зображення, для подальшої можливості збереження.

maxCannyThresholdHigh — змінна типу int, використовується в якості атрибуту для функції createTrackbar.

maxCannyThresholdLow — змінна типу int, яка також використовується в якості атрибуту для функції createTrackbar.

CannyHigh — змінна типу int, зберігає числове значення верхнього порогу для алгоритму Canny.

CannyLow — змінна типу int, зберігає числове значення нижнього порогу для алгоритму Canny.

windowName — змінна типу string, зберігає назву вікна , у якому відображається результат обробки зображення.

trackbarNameThresholdMin — змінна типу string, зберігає назву трекбару який відповідає за нижній поріг.

trackbarNameThresholdMax — змінна типу string, зберігає назву трекбару який відповідає за верхній поріг.

Змінні методу main

image_blured — змінна типу Mat, зберігає зображення результат фільтрування.

image_concat — змінна типу Mat, зберігає зображення результат конкатування.

kernel3 — змінна типу Mat, зберігає створене вручну ядро, яке використовується для фільтру збільшення різкості зображення.

image_path — змінна типу string, зберігає шлях до файлу зображення , або назву файлу.

functionChoiceFlag — змінна типу int, зберігає число, в якості вибору функції.

filterChoiceFlag — змінна типу int, зберігає число в якості вибору фільтру.

kernelSize — змінна типу int, зберігає число в якості розміру ядра.

saveFlag — змінна типу int, зберігає число в якості вибору чи зберігати зображення.

continueFlag — змінна типу int, зберігає число в якості вибору чи слід завершити роботу застосунку.

Змінні методу detectEdges

image_grayscaled — змінна типу Mat, зберігає зображення результат конвертації у градації сірого.

image_blurred — змінна типу Mat, зберігає зображення результат фільтрування за гаусом.

image_edges_canny — змінна типу Mat, зберігає зображення результат обробки алгоритмом Кенні.

Створені методи:

detectEdges — метод що перетворює оригінал зображення на карту контурів, за допомогою алгоритму Кенні.

3.6 Розробка застосунку

Для зчитування зображення, перш за все треба вказати програмі який файл зчитувати. Для цього використовується функція imread, в яку в якості атрибута передається шлях до файлу, або його назва. В якості атрибуту прапора зчитування, використовується конвертація у формат триканального кольорового зображення. Також для можливості передачі шляху до файлу в якості атрибута була створена локальна змінна image_path, типу string.

Була створена глобальна змінна image_original, для збереження оригіналу зображення, для можливості обробляти її у різних функціях. Також була створена глобальна змінна image_processed того ж типу.

Після зчитування зображення функція повертає його і записує у глобальну змінну image_original, це зображення і буде в подальшому оброблятися. Далі змінна проходить перевірка на наявність в ній зображення, і при його відсутності інформує користувача про неможливість відкрити зображення (Лістинг 1).

Лістинг 1

```
image_original = imread(image_path,1);
    if (image_original.empty) {
    cout << "\nCant load image " << image_path << endl;
    }
```

Якщо зображення пройшло перевірку та коректно зчиталося, тоді наступним кроком буде надання користувачу вибору, як обробити зображення. У консолі з'являється повідомлення про можливість вибору функцій застосунку, та перелік варіантів і яку відповідь потрібно надати в консоль, для продовження (Рис.21). Відповідь користувача записується у змінну functionChoiceFlag (Лістинг 2).

Лістинг 2

```
cout << "Choose next step" << endl
<< "'1' for Edge Detection" << endl
<< "'2' for Image Filtering" << endl
<< "Input your choice: ";
cin >> functionChoiceFlag;
```

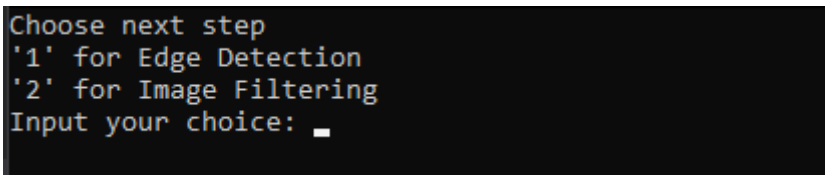

Рис. 21 *Вибір функції застосунку в консолі*

Після того як користувач обрав функцію, якою хоче скористатися, програма перевіряє його вибір за допомогою оператора switch, який передбачає 4 випадки, і виконує ту частину коду яка відповідає вибору користувача. При введенні користувачем варіанту який не передбачається для виконання, програма повідомляє про помилку та дає можливість зробити запит заново (Рис.22) (Лістинг 3).

Лістинг 3

```
default:cout << "\nWrong input value" << endl;
goto functionChoice;
break;
```

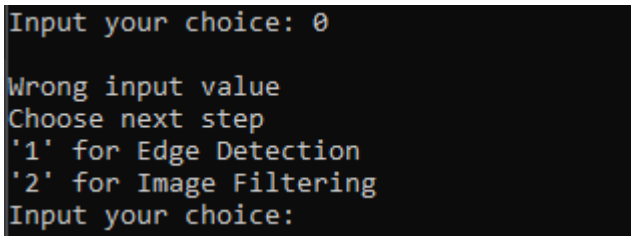

Рис. 22 *Повторний запит при помилці в консолі*

У випадку якщо користувач вдало обрав функцію знаходження контурів зображення, програма виконує наступний код (Лістинг 4).

Лістинг 4

```
cout << "Edge detection" << endl;
namedWindow(windowName, WINDOW_AUTOSIZE);
imshow(windowName, image_original);
```

```
    createTrackbar(trackbarNameThresholdMax,    windowName,
&CannyHigh, maxCannyThresholdHigh, detectEdges);
    createTrackbar(trackbarNameThresholdMin,    windowName,
&CannyLow, maxCannyThresholdLow, detectEdges);
    waitKey(0);
    destroyAllWindows;
```

Програма повідомляє про обрану функцію, створює вікно для відображення результату обробки, методи createTrackbar створюють у вікні трекбари (Рис.23) для налаштування алгоритму Кенні. Трекбари у своїх аргументах приймають колбек функцію detectEdges, яка відповідає за знаходження контурів зображення.

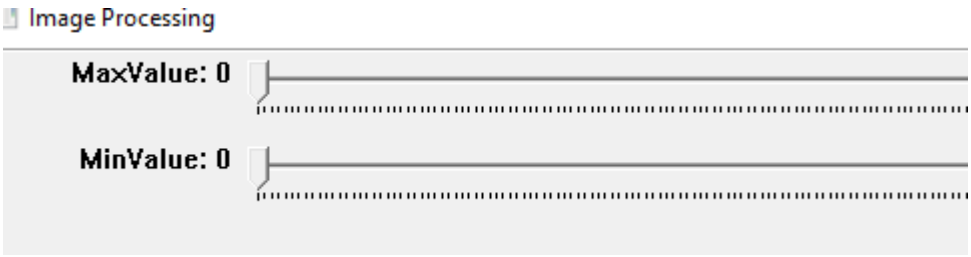

Рис. 23 *Створені трекбари для налаштування алгоритму*

Трекбари були створені для того щоб надати можливість налаштовувати алгоритм Кенні користувачеві власноруч, тому що для кожного зображення потрібні різні параметри, через різну якість та наявність різних шумів на зображеннях. У наступному лістингу описаний метод detectEdges (Лістинг 5).

Лістинг 5

```
void detectEdges(int, void*)
{
Mat image_grayscaled;
Mat image_blurred;
Mat image_edges_canny;
if (CannyLow == 0)
```

```
    CannyLow = 1;
    if (CannyHigh == 0)
    CannyHigh = 1;
    cvtColor(image_original,image_grayscaled,
COLOR_BGR2GRAY);
    GaussianBlur(image_grayscaled,image_blurred,Size2i(5,
5), 0);
    Canny(image_blurred, image_edges_canny, CannyLow,
CannyHigh, 3, false);
    imshow(windowName, image_edges_canny);
    image_processed = image_edges_canny;
    }
```

Перш за все значення трекбарів порівнюються з нулем, і якщо вони дорівнюють нулю, то вони прирівнюються 1, щоб запобігти помилок програми, так як ці значення повинні бути додатніми.

Після перевірки, відбувається конвертація оригінального зображення у градації сірого за допомогою методу cvtColor та результат зберігається у локальну змінну image_grayscaled, це робиться щоб збільшити точність алгоритму, так як наявність декількох каналів кольору, може впливати на фінальний результат.

Наступним кроком є застосування фільтру гауса на отримане зображення за допомогою функції GaussianBlur, результат обробки зберігається у локальну змінну image_blurred.

Фільтрація зображення потрібна щоб усунути шуми на зображенні, тим самим збільшуючи якість фінального результату.

Остання обробка зображення відбувається за допомогою методу Canny, метод використовує параметри CannyLow, та CannyHigh які можуть бути налаштовані трекбарами створеними раніше.

Оброблене зображення відображується методом imshow, і записується у глобальну змінну image_processed, щоб потім зберегти це зображення на пристрої при необхідності.

Метод detectEdges можна викликати вручну, але тоді знадобиться кожний раз змінювати значення для налаштування фільтру, при цьому завершуючи роботу застосунку, та змінюючи значення в коді програми.

Щоб цього запобігти було створено трекбари які використовують колбек функцію, що дозволяє запобігти виходу із програми.

Трекбари викликають колбек функцію кожного разу як користувач змінює значення, таким чином використовуючи трекбар можна налаштовувати точність алгоритму для конкретного зображення у реальному часі (Рис 24).

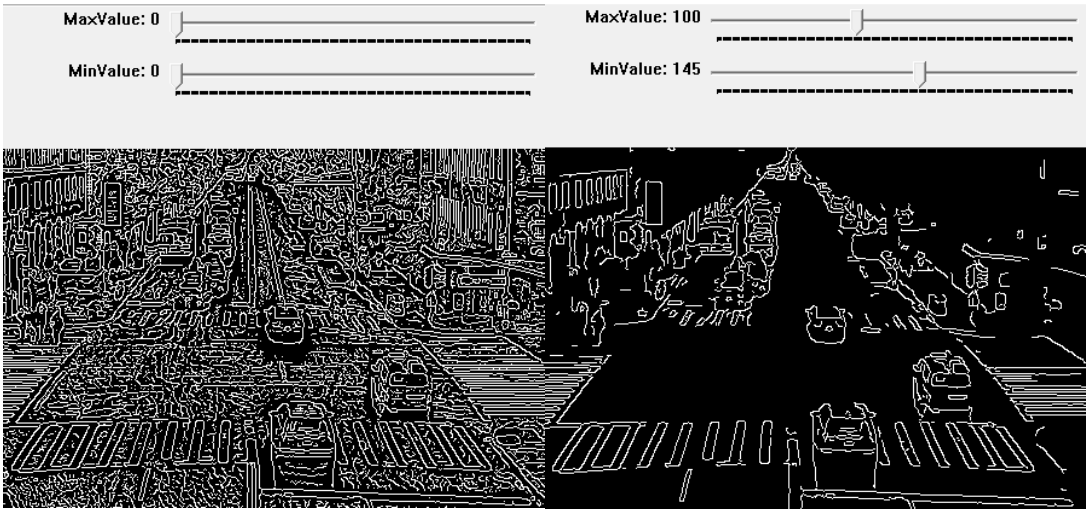

Рис. 24 *Використання трекбарів для налаштування алгоритму*

Якщо користувач обрав замість розпізнання контурів зображення, функцію накладення фільтру на зображення, то перед тим як видати результат, програма повинна отримати ще 2 значення. Спочатку користувач повинен обрати який саме фільтр він хоче використати. Так само як і у випадку з вибором функції, з'явиться список варіантів вибору та назви фільтрів (Рис.25).

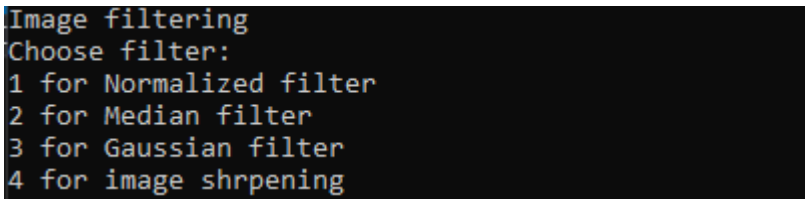

Рис. 25 *Вибір фільтру у вікні консолі*

Після того як користувач введе свій вибір, програма запише його вибір у змінну filterChoiceFlag (Лістинг 6).

Лістинг 6

```
cout << "Choose filter: " << endl << "1 for Normalized filter" << endl << "2 for Median filter" << endl << "3 for Gaussian filter" << endl << "4 for image shrpening" << endl;
cin >> filterChoiceFlag;
```

Одразу після вибору фільтру буде потрібно ввести число яке відповідатиме розміру ядра фільтра, програма попередить про те що фільтри медіанний та гауса використовують тільки непарні ядра (Рис.26), тому і число має бути введене непарне, при виборі фільтру збільшення різкості зображення, цей запит ігнорується, так як цей фільтр використовує спеціальне ядро, задане вручну. Також при введенні від’ємного числа, програма повідомить про помилку і дозволить ввести число заново (Лістинг 7).

Лістинг 7

```
if (filterChoiceFlag != 4) {
kernelInput:
cout << "Enter kernel size\n number from 1 to n \nFor Median, Gausiian fiter number must be odd" << endl;
cin >> kernelSize;
if (kernelSize < 1) {
cout << "Wrong kernel input" << endl;
```

```
goto kernelInput;
}}
```

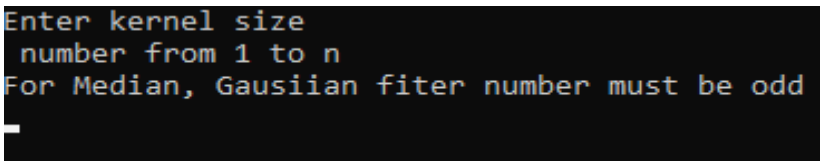

Рис. 26 *Введення розміру ядра у вікні консолі*

Після отримання необхідних програмі даних, вона перевіряє їх на збіги із описаними варіантами в операторі switch (Лістинг 8), кожний випадок має свою функцію фільтрування та повідомлення в консоль про обраний фільтр.

У випадку усереднюючого фільтру, співпадає число 1, і використовує метод blur, та наданий раніше розмір ядра фільтру, в якості аргументу.

Медіанний фільтр відповідає числу 2 та використовує метод medianBlur, але при використанні ядра фільтру воно повинне бути непарним.

Фільтр гауса відповідає числу 3, і використовує метод GaussianBlur, так само як і медіанний фільтр потребує непарне ядро.

Фільтр збільшення різкості відповідає числу 4, використовує метод filter2D, ядро у цьому фільтрі задане вручну і завчасно, тому визначити розмір ядра користувачу можливості не дається.

Лістинг 8

```
switch (filterChoiceFlag)
{
case 1:cout << "Normalized filter" << endl;
blur(image_original, image_blured, Size(kernelSize,
kernelSize));
break;
case 2:cout << "Median filter" << endl;
medianBlur(image_original, image_blured, (kernelSize,
kernelSize));
```

```
    break;
    case 3: cout << "Gaussian filter" << endl;
    GaussianBlur(image_original, image_blured,
Size(kernelSize, kernelSize), 0, 0);
    break;
    case 4:cout << "Image sharpening" << endl;
    filter2D(image_original, image_blured, -1, kernel3,
Point(-1, -1), 0, BORDER_DEFAULT);
    break;
    default: cout << "\nWrong input value" << endl;
    goto filterChoice;
    break;
    }
```

Для демонстрації результату використовується метод hconcat, який горизонтально конкатує зображення, створюючи зручну можливість порівняти оригінальне зображення та зображення з фільтром.

Оброблене зображення виводиться у вікно після чого записується у глобальну змінну image_processed, для можливості збереження (Лістинг 9).

Треба звернути увагу що записується зображення не конкатоване, а лише те, що було оброблене за допомогою фільтру, конкатування проводиться лише для демонстрації результату (Рис.27).

Лістинг 9

```
hconcat(image_original, image_blured, image_concat);
imshow("Processed Image", image_concat);
image_processed = image_blured;
```

Рис. 27 *Результат накладання фільтру та конкатування.*

Після того як користувач обробив зображення потрібно його зберегти, для цього програма повідомляє користувача про можливість збереження зображення (Рис.28), його вибір зберігається у змінну saveFlag, і у випадку підтвердження користувача, використовує метод imwrite для запису зображення на пристрій (Лістинг 10).

Лістинг 10

```
cout << "Do you want to save image? \n'1' for Yes\n'0' for No" << endl;
cin >> saveFlag;
if (saveFlag == 1) {
imwrite("saved_image.jpg", image_processed);
cout << "Image saved";}
```

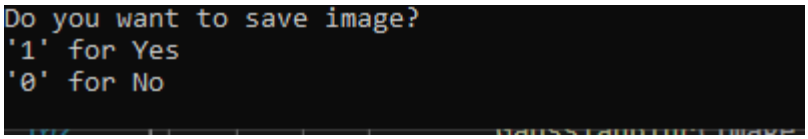

Рис. 28 *Запит про збереження зображення у вікні консолі*..

На останок програма запитує користувача, чи бажає він продовжити роботу у застосунку, при відмові, робота завершується, а при підтвердженні програма повертається до початку (Лістинг 11).

Лістинг 11

```
    cout << "Exit or continue? \n'1' to continue\n'0' to exit" << endl; cin >> continueFlag;
    if (continueFlag == 1)
    {
    goto continueChoice;
    }
```

ВИСНОВКИ

1.У роботі було проведено аналіз можливостей технології та її алгоритмів для обробки зображень. Було досліджено алгоритми Canny та Sobel для визначення контурів об'єктів на зображеннях, алогритми накладання фільтрів на зображення.

2.Виконано огляд існуючих програмних засобів реалізації цих задач. Порівняно їх можливості та підходи до обробки зображень та їх роботи із алгоритмами Canny та Sobel, фільтрації зображень. Обрано середу розробки та мову програмування, для подальшої розробки.

3.Після аналізу існуючих можливостей реалізації було визначено набір задач, і обрано технологію OpenCv, як засіб їх програмної реалізації.

4.Розроблено технічне завдання та виконане планування робіт. В технічному завданні були визначені вимоги до функцій алгоритмів та реалізації їх в програмі та їх взаємодії з користувачем.

5. Проведено проектування застосунку для обробки зображень, побудована діаграма для більш наочного представлення проекту майбутнього застосунку.

6.Створена програма із реалізацією поставлених при плануванні задач для обробки зображень засобами OpenCv. Застосунок та його функціональна частина відповідає проекту, та відповідає вимогам поставлених задач. Була перевірена на ряді задач та показала очікувані при проектуванні результати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Путятін Є.П., Гороховатський В.О., Матат О.О. Методи та алгоритми комп'ютерного зору: навч. посіб. – ТОВ «Компанія СМІТ», – 2006. – 236 с.

2. Bradski G., Gary. Kaehler A.,Learning OpenCV. Computer Vision with the OpenCV Library. O'Reilly Media, Inc. 2008. 556 p.

3.Canny J. Computational approach to edge detection // IEEE Transactions on pattern analysis and machine intelligence. – 1986. – № 6. – pp. 679–698.

4.Cheng J., Grossman M., McKercher T. Professional CUDA C Programming. –Indianapolis: John Wiley & Sons, Inc., 2014. – 528 p

5.García G., Suarez O, Aranda J, Tercero J, Gracia I., Enano N. Learning Image Processing with OpenCV. Packt Publishing, 2015. 232 p.

6.Gonzalez R. C., Woods R. E. Digital Image Processing. Prentice Hall, 2002. 813 p MatLab

7. ImageJ plugins (with source code). URL: http://rsb.info.nih.gov/ij/plugins/index.html.

8.Jähne B., Scharr H., Körkel S. Principles of filter design // Handbook of Computer Vision and Applications. Academic Press. – 1999. – 206 p.

9.Joshi P., Escrivá D., Godoy V. OpenCV By Example. Packt Publishing Ltd., 2016. 306p.

10.Laganiere R. OpenCV 3 Computer Vision Application Programming Cookbook. Packt, 2017. 573 p.

11.Pingle, K.K., Visual Perception by a Computer, Automatic Interpretation and Classification of Images,, Academic Press, New York, 1969, pp. 277-284.(sobel)

12.Pratt W.Digital Image Processing. : Wiley-Interscience; 2007. 812 p.

13. Roberts L. Machine Perception of 3-D Solids, Optical and Electro-optical Information Processing, MIT Press 1965

14.Scharr H. Optimal operators in digital image processing. University of Heidelberg, Germany, 2000, pp. 1-178

15.Sonka M., Hlavac V., Boyle R.Image Processing, Analysis, and Machine Vision. , 2007. 872 p.

16. The ImageJ API documentation. URL: http://rsb.info.nih.gov/ij/docs/api/index.html.