НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

«До захисту допущено»
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Практичне застосування комп'ютерного зору в робототехніці на прикладі розпізнавання жестів руки»**

Виконала: студентка 4 курсу, групи ФІ-92
Шевченко Юлія Олексіївна

Керівник: асистент кафедри ММАД Яворський О.А. _____

Консультант: _ _____

Рецензент: доцент кафедри ММЗІ, к.ф.-м.н. Яковлєв С.В._____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи моделювання, розпізнавання образів та комп'ютерного зору»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Шевченко Юлія Олексіївна

1. Тема роботи: «*Практичне застосування комп'ютерного зору в робототехніці на прикладі розпізнавання жестів руки*»,

керівник: асистент кафедри ММАД Яворський О.А.,

затверджені наказом по університету №__ від «__» _____ 2023 р.

2. Термін подання студентом роботи: «__» _____ 2023 р.

3. Вихідні дані до роботи: *опубліковані джерела за тематикою дослідження*

4. Зміст роботи: *огляд літератури та публікацій за темою дослідження; дослідження, реалізація та порівняльний аналіз методів детекції контурів; дослідження, реалізація та порівняльний аналіз згорткових нейронних мереж на основі методів для детекції жестів руки; аналіз результатів дослідження.*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *Презентація доповіді*

6. Дата видачі завдання: 10 вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	15-30 листопада 2022 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Листопад-грудень 2022 р.	Виконано
3	Дослідження методів виявлення контурів	Січень 2023 р.	Виконано
4	Збір набору даних та тестування методів	Лютий-березень 2023 р.	Виконано
5	Побудова архітектури нейронної мережі та навчання моделей	Квітень 2023 р.	Виконано
6	Тестування та порівняння навчених моделей	Травень 2023 р.	Виконано
7	Оформлення дипломної роботи	Травень 2023 р.	Виконано

Студент _____ Шевченко Ю.О.

Керівник _____ Яворський О.А.

РЕФЕРАТ

Кваліфікаційна робота містить: 85 стор., 45 рисунків, 3 таблиці, 21 джерело.

Оскільки робототехніка наразі дуже швидко розвивається і знаходить все більше застосувань у таких важливих сферах, як медицина та military-tech актуальним стає дослідження з метою проведення порівняльного аналізу різноманітних методів виявлення контурів для розпізнавання жестів і рухів руки з метою їх подальшої взаємодії між роботами та людьми. В ході роботи було здійснено аналіз методів розпізнавання жестів руки людини, протестовано їх ефективність та швидкодії та застосовано для навчання згорткових нейронних мереж для розпізнавання жестів.

КОМП'ЮТЕРНИЙ ЗІР, РОБОТОТЕХНІКА, РОЗПІЗНАВАННЯ ЖЕСТІВ, МАШИННЕ НАВЧАННЯ, МЕТОДИ РОЗПІЗНАВАННЯ КОНТУРІВ

ABSTRACT

The qualification work contains: 85 pages, 45 figures, 3 tables, 21 sources.

As robotics is currently rapidly developing and finding more applications in such important areas as medicine and military tech, it becomes relevant to conduct research with the aim of a comparative analysis of various methods of contour detection for recognizing hand gestures and movements for their further interaction between robots and humans. In the course of the work, an analysis of the methods of recognizing human hand gestures was carried out, their efficiency and speed were tested, and they were applied for training convolutional neural networks for gesture recognition.

COMPUTER VISION, ROBOTICS, GESTURE DETECTION, MACHINE LEARNING, EDGE DETECTION METHODS

ЗМІСТ

ВСТУП

Актуальність дослідження. В сучасному світі робототехніка швидко розвивається і знаходить все більше застосувань у різних сферах життя. Наприклад, важливими областями є медицина та military-tech, де необхідно досягнення високої точності розпізнавання рухів і жестів роботами. Стандартні методи керування роботами, такі як клавіатура чи джойстик не дають змоги досягти цього, адже даний вид взаємодії не є завжди зручним к своєму використанні і рухи, що необхідно виконати для відсилання необхідних команд не корелюють з тими, що будуть відтворені. Тому, це призводить до того, що необхідно багато часу для навчання та практики. Відповідно, постає питання у створенні більш адекватних та інтуїтивних методів керування роботами. Особливо це буде корисно у сфері медицини для реабілітації пацієнтів з руховими обмеженнями, або при проведенні операцій, або в military-tech для керування маніпулятором для розміновування територій.

Метою дослідження є проведення порівняльного аналізу різноманітних методів виявлення контурів для розпізнавання жестів і рухів руки з метою їх подальшого застосування в системах взаємодії людини та робота у сфері робототехніки. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) дослідити сучасні досягнення на тему поєднання комп'ютерного зору та робототехніки;

3) розглянути й реалізувати методи та алгоритми для виявлення контурів;

4) виконати порівняльний аналіз обраних методів та алгоритмів;

5) виконати тренування моделей для детекції жестів руки;

6) провести порівняльний аналіз підготовлених моделей.

Об'єктом дослідження є проблема ефективного виявлення та

розпізнавання жестів руки в системах робототехніки, використовуючи методи комп'ютерного зору та згорткових нейронних мереж.

Предметом дослідження є оператори градієнта, методи вторинного розподілу, методи морфологічної обробки, методи на основі оптимального фільтру та їх вплив на ефективність розпізнавання жестів руки за допомогою згорткових нейронних мереж в системах робототехніки.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: аналітичний метод, експерементальний, методи машинного навчання, обробки зображень, методи статистичного аналізу, візуалізації даних.

Наукова новизна отриманих результатів полягає в порівняльному аналізі обраних методів обробки зображень для задачі виявлення та розпізнавання жестів руки людини. Отримані результати дозволили визначити вплив на якісні та чисельні результати алгоритмів розпізнавання. Експериментальне підтвердження результатів було отримано за допомогою використання методів машинного навчання (нейронних мереж).

Практичне значення результатів полягає в їх подальшому використанні для покращення систем розпізнавання жестів в робототехніці, що своєю чергою покращує взаємодію людей та роботів. Дослідження, проведене в рамках цієї роботи, може бути корисним для розробників систем комп'ютерного зору під час створення архітектур для певних задач. Також, результати даної роботи можуть слугувати основою для подальших досліджень в області розпізнавання жестів в системах робототехніки та в області комп'ютерного зору в цілому.

1 ТЕОРЕТИЧНИЙ ОГЛЯД

У сучасному світі робототехніка активно інтегрується в різні сфери життя, від промисловості до домашнього використання. Для досягнення більш ефективної взаємодії між роботами та людьми все більше уваги приділяється розробці систем розпізнавання жестів.

Цей розділ присвячений теоретичному огляду основних методів комп'ютерного зору, які використовуються для розпізнавання жестів руки, включаючи оператори Собеля, Прюїтта, Робертса, методи Кенні та Лапласа, методи розширення та ерозії, методи відкриття та закриття, Марра-Хілдрета та Шен-Кастана. Ми розглянемо принципи роботи цих методів, їх застосування та можливі обмеження.

Аналіз цих методів є необхідною складовою для подальшого експериментального дослідження та реалізації системи розпізнавання жестів руки.

1.1 Історія та розвиток методів виявлення контурів

Історія методів виявлення контурів значно еволюціонувала протягом останніх десятиліть, перетворюючись з простих алгоритмів виявлення контурів до більш складних моделей, заснованих на машинному навчанні. Ця еволюція була перш за все обумовлена зростаючою потребою в розумінні та інтерпретації візуальної інформації в різних областях, включаючи робототехніку та розпізнавання жестів.

Ранні методи виявлення контурів ґрунтувалися на простих математичних принципах. Однією з найраніших технік, розроблених Робертом Дудою та Пітером Хартом у 1972 році [1], була перетворення Хафа – метод виявлення недосконалих екземплярів певних форм на зображенні. Перетворення Хафа являти собою фундаментальну зміну у

підході до виявлення контурів, оскільки воно перенесло акцент з окремих пікселів на більш глобальну перспективу, розглядаючи все зображення.

У 1980-х роках дослідники розробили більш витончені алгоритми виявлення контурів, такі як оператори Прюїтта, Собеля та Робертса, які обчислювали величину градієнта та напрямок змін інтенсивності пікселів [1, 2, 3]. Ці методи були ефективними для виявлення контурів на зображеннях з високим контрастом, але вони мали проблеми зі зображеннями зі змінюваною освітленістю або шумом.

Алгоритм виявлення контурів Кенні [5] було запропоновано у 1986 році, його багато хто вважає оптимальним детектором контурів через низький рівень помилок, точну локалізацію краю та однозначну характеристику реакції. Метод Кенні використовує багатоступеневий алгоритм для виявлення широкого спектра контурів на зображеннях, його все ще широко використовують сьогодні.

З появою згорткових нейронних мереж (CNN) у кінці 1990-х та на початку 2000-х, область виявлення контурів зробила значний крок вперед. CNN, з їх здатністю вивчати ієрархічні представлення ознак, показали дивовижний успіх у складних задачах виявлення контурів, включаючи розпізнавання жестів рук у робототехніці.

Область робототехніки стикалася з кількома викликами при впровадженні цих методів виявлення контурів. Наприклад, прості методи, засновані на градієнтах, не є стійкими проти змінних умов освітлення або перекриття, які є поширеними в реальних застосуваннях робототехніки. З іншого боку, хоча методи, засновані на CNN, пропонують покращену точність, вони вимагають великих обсягів позначених тренувальних даних та обчислювальних ресурсів, що може бути непрактичним у всіх застосуваннях робототехніки.

Задача розпізнавання жестів рук є відносно складною у реалізації через велику варіативність форм, орієнтацій та рухів людської руки. Попри ці виклики, прогрес у методах виявлення контурів зробив можливим розробку робототехнічних систем, здатних розпізнавати та

інтерпретувати жести рук, відкриваючи шлях до більш інтуїтивної взаємодії людини та робота.

Сьогодні основна увага дослідників прикута до розробки гібридних систем, які поєднують традиційні методи виявлення контурів з моделями машинного навчання, щоб досягти високої точності, зберігаючи при цьому обчислювальну ефективність. Використовуючи переваги обох підходів, дослідники прагнуть подолати обмеження окремих методів та розширити границі можливого у робототехніці та розпізнаванні жестів. Область виявлення контурів, що постійно розвивається, продовжує відігравати важливу роль у покращенні нашої здатності розуміти та інтерпретувати візуальну інформацію, що значно впливає на робототехніку та численні інші області.

1.2 Оператор Собеля

Оператор Собеля є одним з основних методів виявлення контурів на зображеннях в області комп'ютерного зору. Цей оператор був названий на честь Ірвіна Собеля, американського вченого в галузі комп'ютерних наук, який був одним із його розробників [3].

Оператор використовує дві 3×3 матриці, які називаються ядрами Собеля. Одне ядро виявляє границі (або зміни інтенсивності) по горизонталі, а інше - по вертикалі. Ці ядра допомагають визначити, де знаходяться контури на зображенні шляхом вимірювання величини градієнта. Ці матриці подані нижче:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Ідея оператора Собеля полягає в конволюції зображення з ядром Собеля. Цей процес включає обчислення зваженої суми інтенсивностей пікселів в околі центрального пікселя для кожного пікселя зображення. Якщо результуюча сума велика, то це індикатор наявності контуру.

Означення 1.1. Конволюція (згортка) – математична операція двох функцій $f(t)$ та $g(t)$, що дозволяє отримати таку функцію: $(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$.

Алгоритм дій для оператора Собеля включає наступні кроки:

1) **Нормалізація зображення.** Даний крок відбувається шляхом відображення кольорового зображення у чорно-білий спектр. Якщо зображення вже відтінків сірого, цей крок може бути пропущений.

2) **Ініціалізація ядер Собеля.** Ініціалізуються дві 3×3 матриці для горизонтального (G_x) та вертикального (G_y) градієнтів. Ці ядра складаються з вагованих значень, які показують вплив сусідніх пікселів на обчислення градієнта.

3) **Конволюція зображення з ядрами Собеля.** Проходяться по кожному пікселю зображення і застосовуються ядра Собеля, використовуючи процес конволюції. В результаті ми отримаємо два нові зображення, по одному для кожного напрямку градієнта.

4) **Обчислення величини градієнта.** Для кожного пікселя обчислюється величина градієнта, використовуючи формулу $\sqrt{{G_x}^2 + {G_y}^2}$. Величина градієнта являє собою відстань в просторі градієнтів і допомагає визначити силу контуру.

5) **Бінарне представлення зображення.** На останньому кроці можна застосувати порогове значення до величини градієнта, щоб отримати бінарне зображення контурів. Всі пікселі, значення яких перевищують порогове значення, вважатимуться частиною контуру.

Метод Собеля є ефективним і швидким способом виявлення контурів. Він може виявляти різноманітні типи контурів, включаючи горизонтальні, вертикальні та похилені. Однак, він може бути чутливим до шуму і не завжди здатен розпізнати контури зі слабкими змінами інтенсивності.

(а) Тестове зображення **(б)** Оператор Собеля

Рисунок 1.1 – Результат роботи оператора Собеля.

1.3 Оператор Прюїтта

Оператор Прюїтта [2] – це метод виявлення контурів на зображеннях, що базується на обчисленні градієнта інтенсивності зображення. Він дуже схожий на оператор Собеля і використовує дві 3x3 ядерні матриці для обчислення горизонтального та вертикального напрямків градієнта. Проте, відмінність полягає в тому, що оператор Прюїтта присвоює однакові ваги всім елементам в ядрі, що призводить до меншої чутливості до діагональних границь. Дані матриці представлені нижче:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Алгоритм дій для оператора Прюїтта включає наступні кроки:

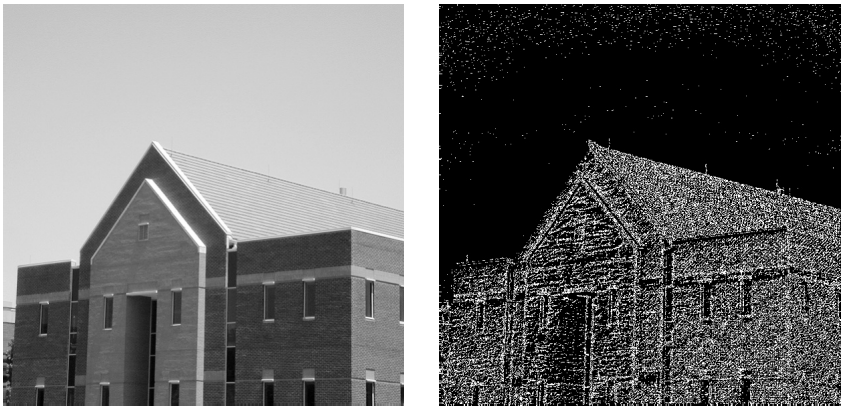
1) **Нормалізація зображення.** Даний крок відбувається шляхом відображення кольорового зображення у чорно-білий спектр. Якщо зображення вже відтінків сірого, цей крок може бути пропущений.

2) **Ініціалізація ядер Прюїтта.** Ініціалізуються дві 3x3 матриці для горизонтального (G_x) та вертикального (G_y) градієнтів. Ці ядра складаються з вагованих значень, які показують вплив сусідніх пікселів на обчислення градієнта.

3) **Конволюція зображення з ядрами Прюїтта.** Проходяться по кожному пікселю зображення і застосовуються ядра Прюїтта, використовуючи процес конволюції. В результаті ми отримаємо два нові зображення, по одному для кожного напрямку градієнта.

4) **Обчислення величини градієнта.** Для кожного пікселя обчислюється величина градієнта, використовуючи формулу $\sqrt{{G_x}^2 + {G_y}^2}$. Величина градієнта являє собою відстань в просторі градієнтів і допомагає визначити силу контуру.

5) **Бінарне представлення зображення.** На останньому кроці можна застосувати порогове значення до величини градієнта, щоб отримати бінарне зображення контурів. Всі пікселі, значення яких перевищують порогове значення, вважатимуться частиною контуру.



(а) Тестове зображення **(б)** Оператор Прюїтта

Рисунок 1.2 – Результат роботи оператора Прюїтта.

Як і оператор Собеля, оператор Прюїтта може бути чутливим до шуму, адже він просто обчислює різницю в інтенсивностях пікселів. Більш складні методи, як-то оператор Кенні або Лапласа, можуть надати кращі результати на шумних зображеннях. Проте, оператор Прюїтта є

простим в реалізації та швидким у виконанні, що робить його корисним для багатьох застосувань.

1.4 Оператор Робертса

Оператор Робертса [4] – це дуже простий, але досить ефективний метод виявлення контурів у зображеннях, особливо важливий у ранніх стадіях розвитку комп'ютерного зору.

Основна ідея оператора Робертса полягає у використанні двох 2×2 конволюційних ядер. Ці ядра розраховані так, щоб вони вимірювали різницю в інтенсивності між двома діагонально сусідніми пікселями. Одне ядро використовується для виявлення границь, що проходять зліва вправо і зверху вниз, а інше – для виявлення границь, що проходять зправа вліво і знизу вгору. Дані матриці представлені нижче:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Алгоритм дій для оператора Робертса включає наступні кроки:

1) **Нормалізація зображення.** Даний крок відбувається шляхом відображення кольорового зображення у чорно-білий спектр. Якщо зображення вже відтінків сірого, цей крок може бути пропущений.

2) **Ініціалізація ядер Робертса.** Ініціалізуються дві 2x2 матриці для горизонтального (G_x) та вертикального (G_y) градієнтів. Ці ядра складаються з вагованих значень, які показують вплив сусідніх пікселів на обчислення градієнта.

3) **Конволюція зображення з ядрами Робертса.** Проходяться по кожному пікселю зображення і застосовуються ядра Робертса, використовуючи процес конволюції. В результаті ми отримаємо два нові зображення, по одному для кожного напрямку градієнта.

4) **Обчислення величини градієнта.** Для кожного пікселя обчислюється величина градієнта, використовуючи формулу $\sqrt{{G_x}^2 + {G_y}^2}$. Величина градієнта являє собою відстань в просторі градієнтів і допомагає визначити силу контуру.

5) **Бінарне представлення зображення.** На останньому кроці можна застосувати порогове значення до величини градієнта, щоб отримати бінарне зображення контурів. Всі пікселі, значення яких перевищують порогове значення, вважатимуться частиною контуру.

(а) Тестове зображення **(б)** Оператор Робертса

Рисунок 1.3 – Результат роботи оператора Робертса.

Хоча оператор Робертса є простим у використанні та швидким для виконання, він може бути чутливим до шуму, оскільки він обчислює різницю лише між двома пікселями. Це також означає, що він може не виявляти більш слабкі контури, які можуть бути виявлені більш розгорнутими методами виявлення контурів, такими як оператор Собеля або оператор Прюїтта.

1.5 Метод Кенні

Метод Кенні – це популярний алгоритм виявлення контурів, який був названий на честь його розробника Джона Ф. Кенні [5]. Він використовується в комп'ютерному зору для виявлення місць на

зображенні, де яскравість зображення різко змінюється або, іншими словами, має високі просторові градієнти.

Метод Кенні включає в себе чотири основні кроки:

1) **Зменшення шуму.** На першому кроці, вхідне зображення згладжується за допомогою Гауссового фільтра. Це робиться для зменшення впливу шуму на подальші процеси. Гауссовий фільтр являє собою фільтр з лінійною відповіддю, який використовується для видалення Гауссового шуму. Фільтр є конволюційним, і він проходить по вхідному зображенню, щоб згладити його.

2) **Пошук градієнта інтенсивності зображення.** Після того, як зображення згладжено, відбувається визначення градієнта інтенсивності. Градієнт можна визначити за допомогою операторів Собеля, Прюїтта або Робертса, але найчастіше використовується саме оператор Собеля. Градієнт визначається окремо для x та y напрямків. Результатом цього кроку є зображення градієнта, що показує величину та напрямок зміни яскравості.

3) **Подавлення немаксимальних значень.** У цьому кроці виконується згладжування градієнта інтенсивності. Ідея полягає в тому, щоб перетворити розмиті контури зображення на тонкі контури. Це досягається шляхом проходження через всі пікселі на зображенні градієнта і видалення пікселів, які не є локальним максимумом.

4) **Подвійне порогове значення і трекінг по границі.** На останньому кроці, визначаються фактичні контури. Це досягається шляхом встановлення двох порогових значень (високого і низького). Якщо величина градієнта перевищує високе порогове значення, вона маркується як сильний край. Якщо величина градієнта менша від нижнього порогового значення, вона відкидається. Якщо величина градієнта знаходиться між двома порогами, тоді вона класифікується як слабкий край, і враховується лише в тому випадку, якщо вона з'єднана з сильним краєм. Це гарантує, що тільки найбільш важливі контури беруться до уваги.

(а) Тестове зображення **(б)** Метод Кенні

Рисунок 1.4 – Результат роботи методу Кенні.

Метод Кенні є одним з найточніших методів виявлення контурів, особливо при роботі з реальними зображеннями (фотографіями). Завдяки використанню подвійного порогового значення, метод дозволяє ефективно розрізняти важливі контури від менш вагомих. А початкове згладжування зображення з допомогою Гауссового фільтра допомагає зменшити вплив шуму. Проте через багатоступеневий процес, метод Кенні може бути обчислювально важким, особливо для великих зображень та правильне встановлення порогових значень може бути складним і вимагає певного досвіду або експериментування.

1.6 Метод Лапласа

Метод Лапласа або Лапласіан для виявлення контурів використовується у візуальному обчислювальному середовищі для ідентифікації областей, де інтенсивність зображення зазнає різких змін. Основна причина, для якої цей метод є корисним, полягає в тому, що контури на зображенні часто відповідають областям, де фізичний об'єкт на сцені змінюється – це може бути місце, де об'єкт закінчується, або де об'єкт переходить від одного типу поверхні до іншого.

Оператор Лапласа є диференціальним оператором, що може бути

використаним для виявлення областей, де інтенсивність зображення змінюється. У випадку двовимірного зображення, він може бути визначений як сума других похідних по x та y координатам, що є еквівалентом конволюції зображення з ядром Лапласіана.

$$\text{Laplacian}(I) = \nabla^2(I) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Алгоритм застосування оператора Лапласа для виявлення контурів включає такі кроки:

1) **Нормалізація зображення.** Зображення перетворюється на відтінки сірого, якщо воно ще не має такого формату.

2) **Застосування Лапласіана.** Застосовується оператор Лапласа, який обчислює другу похідну зображення. Це може бути зроблено шляхом використання фільтра Лапласа або конволюційного ядра Лапласа.

3) **Обчислення другої похідної.** Кожен піксель зображення пропускається через оператор Лапласа, і в кожній точці обчислюється значення другої похідної інтенсивності. Зазвичай, це виконується шляхом застосування ядра до сусідніх пікселів і обчислення різниці між центральним пікселем та сумою вагованих значень сусідніх пікселів.

4) **Фільтрування за порогом.** Отримані значення другої похідної можуть бути порівняні з певним пороговим значенням, щоб відфільтрувати контури з низькою інтенсивністю або шумом. Зазвичай, якщо значення другої похідної перевищують поріг, вони вважаються контурами.

5) **Уточнення контурів.** Після виявлення контурів за допомогою методу Лапласа, можуть бути виконані додаткові кроки для вдосконалення контурів, такі як використання порогового значення для відфільтрування непотрібних контурів або застосування морфологічних операцій для згладжування або з'єднання контурів.

Переваги використання методу Лапласа для виявлення контурів включають його здатність виявляти контури в будь-якому напрямку і його незалежність від орієнтації краю. Цей метод відмінно працює, коли

(а) Тестове зображення **(б)** Метод Лапласа

Рисунок 1.5 – Результат роботи методу Лапласа.

контури об'єктів добре визначені. Однак, як і інші методи виявлення контурів, метод Лапласа може бути чутливим до шуму. Шум може викликати невеликі, але різкі зміни інтенсивності, які Лапласіан може помилково визначити як контури. Також цей метод може не виявити слабкі контури, оскільки вони не приводять до різких змін у другій похідній.

Метод Лапласа виявлення контурів має широкий спектр застосувань у комп'ютерному зорі, включаючи сегментацію зображень, розпізнавання об'єктів, розпізнавання тексту та обробку зображень у медичних дослідженнях.

1.7 Метод розширення та ерозії

Метод розширення та ерозії є одним з основних етапів обробки зображень, який використовується для згладжування та виділення об'єктів, а також для виявлення контурів. Цей метод базується на морфологічних операціях, які виконуються на бінарних зображеннях або на зображеннях, які були попередньо бінаризовані.

Метод розширення:

– Розширення – це морфологічна операція, яка розширює об'єкти

на зображенні шляхом додавання пікселів, що оточують кожен об'єкт. Це виконується шляхом проходження всіх пікселів зображення та визначення найвищого значення в області, яка визначена структурним елементом (ядром).

– Структурний елемент (ядро) – це малий шаблон, який використовується для визначення сусідніх пікселів для кожної точки зображення. Вона може мати різні форми, такі як квадрат, коло або кривуватий шаблон.

– Під час розширення кожен піксель на зображенні замінюється на найвище значення серед пікселів, що знаходяться в межах структурного елемента. Це призводить до розширення об'єктів та заповнення невеликих пропусків або вигинів у контурах.

Метод ерозії:

– Ерозія – це морфологічна операція, яка зменшує об'єкти на зображенні шляхом видалення пікселів на основі структурного елемента. Вона також виконується шляхом проходження всіх пікселів зображення та визначення найнижчого значення в області, визначеній структурним елементом.

– Під час ерозії кожен піксель на зображенні замінюється на найнижче значення серед пікселів, що знаходяться в межах структурного елемента. Це призводить до зменшення об'єктів та видалення невеликих виступів або шуму.

Застосування методу розширення та ерозії:

– **Виявлення контурів.** Метод розширення та ерозії може використовуватися для виділення контурів об'єктів на зображенні. Застосування послідовної ерозії та розширення дозволяє виявити зовнішні та внутрішні контури об'єктів.

– **Згладжування та видалення шуму.** Метод розширення та ерозії може використовуватися для згладжування зображення та видалення невеликого шуму. Послідовна ерозія та розширення може згладити контури об'єктів та видалити дрібний шум, не суттєво

впливаючи на форму та структуру об'єктів.

– **Виділення об'єктів.** Метод розширення та ерозії може використовуватися для виділення окремих об'єктів на зображенні. Застосування відповідного структурного елемента дозволяє розрізняти об'єкти та виділити їх на тлі.

(а) Тестове зображення **(б)** Метод розширення та ерозії

Рисунок 1.6 – Результат роботи методу розширення та ерозії.

Метод розширення та ерозії є потужним і гнучким інструментом в обробці зображень, який знаходить широке застосування в багатьох задачах комп'ютерного зору, включаючи сегментацію зображень, виявлення контурів та обробку зображень з метою покращення якості та виділення цікавих об'єктів.

1.8 Метод відкриття та закриття

Метод відкриття та закриття є одним із фундаментальних етапів морфологічної обробки зображень. Він використовується для згладжування, видалення шуму, відокремлення об'єктів та замикання розривів в контурах об'єктів на бінарних або бінаризованих зображеннях.

Метод відкриття:

– Відкриття є комбінацією двох морфологічних операцій: ерозії та розширення. Перш за все, виконується ерозія зображення, яка видаляє

невеликі об'єкти та тонкі виступи, згладжує контури та видаляє шум. Потім застосовується розширення до отриманого зображення, що допомагає відновити розміри та форму об'єктів, які не були повністю видалені ерозією.

– Під час відкриття кожен піксель на зображенні проходить через ерозію та розширення з використанням певного структурного елемента. Цей процес дозволяє згладити контури, видалити дрібний шум та малі об'єкти.

Метод закриття:

– Закриття також є комбінацією двох морфологічних операцій: розширення та ерозії. Перш за все, виконується розширення зображення, що допомагає замкнути розриви в контурах об'єктів, заповнити невеликі дірки та відновити пропущені пікселі. Потім застосовується ерозія до отриманого зображення, яка згладжує контури об'єктів та видаляє надлишкові виступи.

– Під час закриття кожен піксель на зображенні проходить через розширення та ерозію з використанням структурного елемента. Цей процес допомагає відновити форму та розміри об'єктів, заповнити дірки та замкнути розриви в контурах.

Застосування методу відкриття та закриття:

– **Видалення шуму.** Метод відкриття використовується для видалення дрібного шуму та артефактів на зображенні. Ерозія допомагає видалити шумові пікселі, а розширення відновлює розміри об'єктів.

– **Замикання розривів.** Метод закриття використовується для замикання розривів в контурах об'єктів. Розширення допомагає замкнути розриви та заповнити пропущені пікселі, а ерозія згладжує контури об'єктів.

– **Відокремлення об'єктів.** Відкриття та закриття можуть бути використані для відокремлення окремих об'єктів на зображенні. Вони допомагають видалити непотрібні деталі та об'єднати близькі пікселі.

Метод відкриття та закриття є важливим інструментом у

(а) Тестове зображення **(б)** Метод відкриття та закриття

Рисунок 1.7 – Результат роботи методу відкриття та закриття.

морфологічній обробці зображень. Використовуючи комбінацію ерозії та розширення, він дозволяє згладжувати контури, видаляти шум, відокремлювати об'єкти та замикати розриви в контурах. Цей метод є потужним інструментом для покращення якості та виділення об'єктів на зображеннях.

1.9 Метод Марра-Хілдрета

Метод детекції контурів Марра-Хілдрета (Marr-Hildreth Edge Detector) є одним із класичних методів обробки зображень, який базується на використанні фільтрації за Гаусом та лапласіана. Цей метод був розроблений Девідом Марром та Ейданом Хілдретом і вперше був представлений в 1980 році [9].

Основна ідея методу Марра-Хілдрета полягає в поєднанні двох етапів: спочатку здійснюється згладжування зображення за допомогою фільтра Гауса для видалення шуму та згладжування деталей, а потім застосовується оператор Лапласіана для виявлення контурів на згладженому зображенні.

Детальний опис методу Марра-Хілдрета:

1) **Згладжування за Гаусом.** Спочатку на вхідне зображення

застосовується фільтр Гауса. Фільтр Гауса використовується для створення гаусівського розмиття зображення. Це допомагає видалити шум та згладити деталі на зображенні. Гаусівське розмиття можна виконати шляхом прокручування ядра Гауса над усіма пікселями зображення та обчислення зваженого середнього значення.

2) **Обчислення Лапласіана.** Після згладжування за Гаусом, на отримане зображення застосовується оператор Лапласіана. Цей оператор є оператором другої похідної, який використовується для виявлення змін інтенсивності на зображенні. В результаті застосування Лапласіана, отримується зображення, на якому контури виражені як точки максимальних або мінімальних значень другої похідної.

3) **Виявлення контурів.** Отримане зображення після застосування Лапласіана містить контурні точки, які можна вважати розрізними елементами на зображенні. Для визначення фактичних контурів, необхідно провести порогову обробку, де встановлюється межове значення, яке визначає, які точки вважати контурними. Зазвичай, точки, значення яких перевищує поріг, вважаються контурними точками.

4) **Масштабування та локалізація контурів.** Після виявлення контурів, їх можна масштабувати та локалізувати для поліпшення точності та зручності використання. Це можна зробити шляхом застосування фільтра Non Maximum Suppression, де вибирається лише одна точка з локального максимуму на кожному контурі, а інші точки пригнічуються.

5) **Параметризація контурів.** Коли контури виявлені та локалізовані, їх можна параметризувати для подальшого аналізу та використання. Наприклад, контури можуть бути представлені як послідовність точок або складнішою кривою, такою як крива Безьє або сегментована лінія.

Метод детекції контурів Марра-Хілдрета є потужним інструментом для виявлення контурів на зображеннях. Використання фільтрації за Гаусом та оператора лапласіана дозволяє здійснити згладжування,

(а) Тестове зображення **(б)** Метод Марра-Хілдрета

Рисунок 1.8 – Результат роботи методу Марра-Хілдрета.

виявлення та локалізацію контурів. Цей метод знайшов широке застосування у багатьох областях, включаючи комп'ютерне зорове сприйняття, обробку зображень та комп'ютерну графіку.

1.10 Метод Шен-Кастана

Метод детекції контурів Шен-Кастана (Shen-Castan Edge Detector), також відомий як ISEF (Improved Step Edge Filter), є одним із методів обробки зображень, призначеним для точного виявлення контурів. Цей метод був розроблений Чи-Ху Шеном та Едвіном Кастаном і вперше представлений у 1992 році [10]. Основна особливість методу Шен-Кастана полягає в використанні коефіцієнтів згладжування, які обчислюються згідно з локальними особливостями зображення. Цей підхід дозволяє виокремлювати контури на зображенні з високою точністю та зниженою чутливістю до шуму.

Детальний опис методу Шен-Кастана:

1) **Згладжування зображення.** Початковим кроком є згладжування зображення для зниження шуму та видалення непотрібних деталей. Зазвичай, для цього використовують фільтр низькочастотного згладжування, наприклад, фільтр Гауса.

2) **Обчислення першої та другої похідних.** Після згладжування зображення обчислюються перша та друга похідні. Перша похідна використовується для визначення місцезнаходження контурів, а друга похідна використовується для визначення їхньої інтенсивності.

3) **Оцінка згладжування.** Застосовується метод оцінки згладжування, який використовується для визначення оптимальних коефіцієнтів згладжування в залежності від локальних особливостей зображення. Це дозволяє врахувати різні структури та властивості контурів, такі як їхні ширина та форма.

4) **Виявлення контурів.** Після оцінки згладжування використовуються порогові значення для визначення наявності контурів на зображенні. Локальні максимуми та мінімуми інтенсивності використовуються для визначення місцезнаходження контурів.

5) **Виявлення контурів.** Після виявлення контурів виконується їхня локалізація, що дозволяє точніше визначити положення та форму контурів на зображенні.

(а) Тестове зображення **(б)** Метод Шен-Кастана

Рисунок 1.9 – Результат роботи методу Шен-Кастана.

Метод Шен-Кастана є ефективним інструментом для виявлення контурів на зображенні з високою точністю та низькою чутливістю до шуму. Він знайшов застосування у багатьох галузях, таких як комп'ютерне зорове сприйняття, обробка зображень, медичне зображення

та багато інших.

Висновки до розділу 1

У сучасному світі робототехніки, розуміння та впровадження ефективного розпізнавання жестів руками залишається великим викликом. Чим більше ми заглиблюємося в епоху автоматизації та штучного інтелекту, тим більшою стає важливість цього завдання. Зростаюча кількість робототехнічних систем, які охоплюють все від виробничої промисловості до побутових обов'язків, від медицини до military-tech, підкреслює потребу у поліпшенні взаємодії між роботами та людьми.

В даному розділі було проаналізовано різні техніки виявлення контурів, які є ключовими для розпізнавання жестів рук в робототехніці. Були розглянуті такі методи, як оператор Собеля, Прюїтта, Робертса, методи Кенні, Лапласа, розширення та ерозії, відкриття та закриття, метод Марра-Хілдрета та метод Шен-Кастана. Були показані основні принципи їх роботи та обмеження, та були наведені базові приклади вихідних зображень, що піддалися обробці за допомогою цих методів.

Було вказано, що оператори Собеля, Прюїтта та Робертса часто використовуються для виявлення контурів на зображеннях, але вони проявляють обмеження, коли стикаються з низькоконтрастними або високошумовими середовищами. На оброблених зображеннях присутньо багато шумів і контури не завжди є чітко визначеними. На противагу ним, методи Кенні та Лапласа використовують різні алгоритми для виявлення контурів, що демонструють більшу ефективність при обробці висококонтрастних зображень.

Методи розширення та ерозії, разом з процедурами відкриття та закриття, використовують морфологічні операції для обробки зображень. Вони продемонстрували ефективність в зниженні шуму та видобутку важливих деталей зображення. Водночас методи Марра-Хілдрета та

Шен-Кастана застосовують більш складні алгоритми, що виявилися корисними при обробці зображень з різними рівнями деталізації.

Всупереч цим ефективним методам, складності розпізнавання жестів рук в робототехніці залишаються значною перешкодою і натепер. Людські руки представляють величезні варіації у формі, орієнтації та рухах, що робить інтерпретацію жестів та рухів непростим завданням. Отже, подальший розвиток даної технології передбачає змішування цих технік з машинним навчанням та глибоким навчанням, з метою підвищення точності розпізнавання та інтерпретації жестів.

2 ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ МЕТОДІВ ВИЯВЛЕННЯ КОНТУРІВ

У сучасному світі робототехніки та автоматизації, що швидко розвивається, здатність машин сприймати, розуміти та реагувати на навколишнє середовище є ключем до їхньої операційної ефективності. Дуже важливо, аби дані системи могли працювати точно та швидко реагувати на зміни.

В даному розділі будуть розглянуті методи виявлення контурів, їх параметри та загальну ефективність, яку вони можуть пропонувати в контексті розпізнавання жестів та рухів рук людини у робототехніці.

Спочатку буде описана підготовка до даного експерименту, та умови, в яких він проводитиметься. Потім ми перейдемо до практичного застосування обраних методів виявлення контурів. Цей процес передбачає розробку експерименту, який застосовує ці техніки до різноманітного набору зображень, демонструючи їхню поведінку в різних сценаріях, що імітують реальні умови.

Крім того, ми прагнемо оцінити кожен метод на основі різних метрик продуктивності. Ключові параметри для кожного методу будуть ретельно налаштовані та оптимізовані, а їхній вплив на загальну продуктивність буде відомо записаний та проаналізований. Результатом цієї оцінки буде порівняльна оцінка, яка висвітлює сильні та слабкі сторони кожної техніки в різних ситуаціях.

Нарешті, ми узагальнимо наші висновки, щоб зрозуміти загальну картину щодо найбільш потужних методів виявлення країв для розпізнавання жестів рук. Це не тільки виявить окрему дієздатність цих методів, але й виявить будь-які синергії, які можуть існувати, коли ці техніки використовуються разом.

2.1 Підготовка до дослідження методів

Коли ми занурюємося у світ робототехніки та комп'ютерного зору, важливо оснастити себе відповідними інструментами та методологіями. Тому, для реалізації завдань даної роботи було обрано мову програмування Python для експериментального дослідження методів.

Підхід до реалізації буде систематичним та орієнтованим на деталі, що забезпечує ефективне впровадження кожного методу виявлення контурів. Кожний алгоритм буде створено на Python, використовуючи можливості бібліотек, таких як OpenCV та NumPy, для задач обробки зображень. Реалізація буде включати необхідні кроки для передобробки, виявлення контурів та післяобробки для кожного методу. Код буде належним чином задокументовано та модульно організовано, забезпечуючи можливість обслуговування та масштабування. Загалом, дана робота виконана на таких версіях програмного забезпечення:

– Python: version 3.9.7
– Keras: version 2.12.0
– Matplotlib: version 3.7.1
– NumPy: version 1.23.5
– OpenCV: version 4.7.0.72
– Pickle: version 0.7.5
– Scikit-learn: version 1.2.2
– TensorFlow: version 2.12.0

Враховуючи велику обчислювальну інтенсивність завдань обробки зображень, потужна конфігурація обладнання є необхідною. В даній роботі всі методи будуть імплементуватися та досліджуватися на персональному комп'ютері з такою конфігурацією:

– Восьмиядерний Intel Core i7-9700F (3.0 - 4.7 ГГц)
– RAM 32 ГБ
– HDD 2 ТБ + SSD 480 ГБ

– nVidia GeForce GTX 1660 Super

Набір даних, що був використаний у цьому дослідженні, було зібрано за допомогою автоматизованого скрипту, який знімав зображення з камери відповідно до попередньо визначених міток. Мітки представляли різні комбінації пальців, які рука може виконувати, в загальному вони складають 32 комбінації. Кожен палець був представлений як 0 (якщо зігнутий) і 1 (якщо розігнутий). Наприклад, мітка '10101' представляла формування рук, при якому перший, третій та п'ятий пальці були підняті, а другий та четвертий - ні. Перша цифра у даній мітці відповідає за стан великого пальця, друга – за вказівний і т.д.

Щоб створити збалансований набір даних, було гарантовано, що кожна комбінація пальців була рівномірно представлена. Скрипт було запрограмовано зібрати 200 зображень для кожної мітки, що забезпечувало однакове розподілення по всіх комбінаціях пальців.

Після завершення збору даних, набір даних пройшов етап підготовки. На цьому етапі було важливо забезпечити, що дані були у відповідному форматі для наступних методів виявлення країв.

Процес підготовки включав такі етапи попередньої обробки, як зміна розміру зображення, перетворення в градації сірого та нормалізацію. Зміна розміру зображення була необхідною, щоб усі зображення були в однаковому форматі для обробки, в той час, як перетворення в градації сірого було виконано, оскільки колірна інформація не була важливою для процесу виявлення країв.

Нормалізація була важливим етапом для того, щоб значення інтенсивності пікселів у всіх зображеннях були на схожому масштабі, що було корисним для ефективності алгоритмів виявлення країв. Цей процес включав перетворення значень інтенсивності пікселів в межах від 0 до 1.

Збір збалансованого та репрезентативного набору даних є основою для надійних та відповідних результатів дослідження. Автоматизований скрипт, який ми використовували, дозволив нам зібрати велику кількість зображень структурованим та рівномірним способом, охоплюючи всі

(а) ’00000’ **(б)** ’01111’ **(в)** ’11001’

Рисунок 2.1 – Приклади зображень датасету та відповідні мітки.

можливі комбінації пальців.

Процес підготовки відіграв вирішальну роль у тому, щоб наш набір даних був у відповідному форматі та стані для методів виявлення країв. Ці кроки є необхідними для мінімізації потенційних помилок під час реалізації алгоритму та досягнення надійних результатів. Відповідно, метод, який ми використовували для збору та підготовки набору даних, створив міцну основу для наших подальших аналізів та висновків дослідження.

Після реалізації методів виявлення контурів, ми проведемо серію тестів, щоб проаналізувати їх ефективність у розпізнаванні жестів рук. Ці тести допоможуть у тонкій настройці параметрів для кожного методу, покращуючи їх продуктивність, і вибір найбільш ефективних для включення в робототехнічні системи.

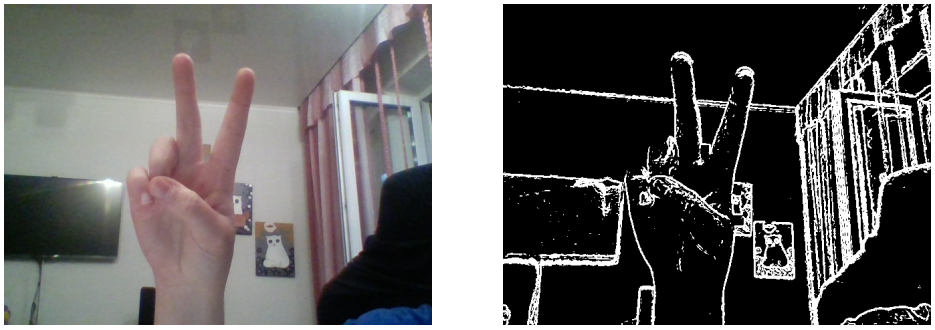
2.2 Дослідження оператору Собеля

Оператор Собеля, дискретний оператор диференціювання, обчислює наближення до градієнта функції інтенсивності зображення. Базуючись на концепції скінченних різниць, він надає простий, але ефективний підхід до виявлення країв.

Впровадження оператора Собеля у Python включає використання функції cv2.Sobel(), наданої бібліотекою OpenCV. Ця функція вимагає параметрів, таких як джерело зображення, глибину даних та порядок похідної в напрямках x та y.

Після застосування оператора Собеля до нашого набору даних ми помітили його ефективність у виявленні країв. Оператор ефективно виділяв переходи від руки до фону на зображеннях, ефективно захоплюючи межі жестів рук. Проте було також зазначено, що оператор Собеля досить чутливий до шуму. У випадках, коли зображення мало високочастотний шум, оператор часто плутав шум з краями, що призводило до помилкового виявлення краю.

На обраному прикладі були проаналізовані різні параметри для реалізації оператору Собеля. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при threshold=16. Тому для подальших експериментів буде використовуватися саме таке значення.



(а) Тестове зображення **(б)** Оператор Собеля

Рисунок 2.2 – Результат роботи оператору Собеля.

Роблячи висновок, можна сказати, що оператор Собеля виявився ефективним методом для виявлення краю при розпізнаванні жестів рук, здатним ефективно захоплювати краї в горизонтальному та вертикальному напрямках.

Основні переваги оператора Собеля включають його простоту та здатність акцентувати краї, що робить його цінним інструментом на початкових етапах обробки зображень. Більш того, його легкість впровадження в Python, особливо за допомогою бібліотеки OpenCV, підсилює його привабливість для використання в бізнесі.

Однак в оператора Собеля є і деякі недоліки. Його висока чутливість до шуму може призвести до помилкового виявлення краю. Більш того, він може не захопити всі відповідні краї на зображеннях з низьким контрастом або тих, що мають складні фони. На нашому прикладі ми можемо спостерігати факт того, що даний оператор виявив також купу зайвих фонових об'єктів і повернув нам зашумлене зображення. І навпаки, в місцях зображення, де яскравість пальців була близькою до яскравості стіни, оператор не виявив контурів.

По суті, хоча оператор Собеля є фундаментальним інструментом у виявленні краю, його ефективність залежить від контексту. Його використання потребує сполучення з відповідними методами зменшення шуму і доповнення іншими методами виявлення краю в сценаріях, де він недостатній.

2.3 Дослідження оператору Прюїтта

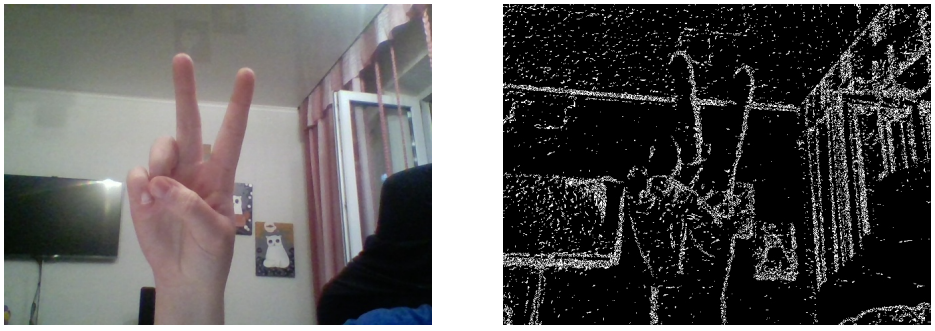
Оператор Прюїтта, який є ще одним популярним методом для виявлення контурів, оцінює градієнт інтенсивності зображення в кожному пікселі. Аналогічно до оператора Собеля, він використовує два ядра розміром 3×3, одне для виявлення контурів у напрямку x та інше у напрямку y. Ці ядра розроблені таким чином, щоб максимально реагувати на вертикальні та горизонтальні контури відносно сітки пікселів.

Для реалізації оператора Прюїтта в Python, ми знову використовуємо бібліотеку OpenCV, зокрема функцію cv2.filter2D(). Після перетворення нашого зображення на відтінки сірого, ми створюємо ядра Прюїтта і застосовуємо їх до зображення за допомогою цієї функції, яка реалізує згортку зображення з ядром.

Однак, подібно до оператора Собеля, оператор Прюїтта також показав високу чутливість до шуму високої частоти, що призводило до

виявлення помилкових контурів на шумних зображеннях.

На обраному прикладі були проаналізовані різні параметри для реалізації оператору Прюїтта. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при threshold=180. Тому для подальших експериментів буде використовуватися саме таке значення.



(а) Тестове зображення **(б)** Оператор Прюїтта

Рисунок 2.3 – Результат роботи оператору Прюїтта.

Висновком є те, що оператор Прюїтта служить непоганим інструментом для виявлення контурів на зображеннях, особливо у сценаріях, де акцент ставиться на горизонтальні та вертикальні контури. Проте саме в умовах нашої задачі даний оператор показав багато зайвих шумів а також неточні контури, отже для цього методу виявлення контурів рекомендується застосовувати додаткову обробку зображень, що створює додаткові операції, взаємодії та витрати часу.

До його переваг належать простота та легкість впровадження, особливо з Python та OpenCV. Він пропонує швидке виявлення контурів, що робить його цінним інструментом на початкових етапах обробки зображень.

Недоліки оператора Прюїтта аналогічні тим, що в оператора Собеля: висока чутливість до шуму та потенційна недостатність у виявленні контурів на зображеннях з низьким контрастом або складними фонами.

Підсумовуючи, оператор Прюїтта є потужним інструментом для виявлення контурів, з відповідними застереженнями, схожими на ті, що в оператора Собеля. Найкраще його комбінувати з відповідними техніками зниження шуму та, можливо, доповнювати іншими методами виявлення контурів для підвищення його ефективності.

2.4 Дослідження оператору Робертса

Оператор Робертса – це простий, швидкий для обчислення, цифровий диференціальний оператор, який використовується для виявлення країв на зображеннях. Він використовує 2×2 згорткове ядро для розрахунку суми квадратів різниці між діагонально сусідніми пікселями.

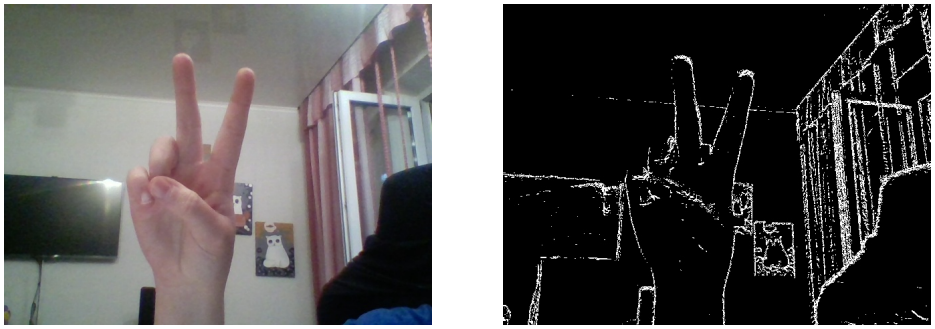
Для впровадження оператора Робертса в Python ми використовуємо функцію cv2.filter2D() бібліотеки OpenCV. Після перетворення зображення в сірі відтінки, ми створюємо ядра Робертса та згортаємо зображення з цими ядрами, використовуючи функцію filter2D().

Після застосування оператора Робертса до нашого набору даних, він показав ефективність у виокремленні країв жестів рук. Однак також було відзначено, що цей оператор має меншу чутливість до шуму порівняно з операторами Собеля та Прюїтта.

Однак оператор Робертса іноді не впорався з підсвітленням дрібних деталей, оскільки він використовує лише 2×2 ядро і враховує тільки діагонально сусідні пікселі, що може призвести до пропуску деяких країв.

На обраному прикладі були проаналізовані різні параметри для реалізації оператору Робертса. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при threshold=120. Тому для подальших експериментів буде використовуватися саме таке значення.

Підсумовуючи, оператор Робертса надає простий і швидкий метод для виявлення країв. Його основні переваги полягають у його простоті та меншій чутливості до шуму.



(а) Тестове зображення **(б)** Оператор Робертса

Рисунок 2.4 – Результат роботи оператору Робертса.

Серед переваг велику роль відіграє простота оператора Робертса, оскільки він вимагає менше обчислень, що робить його швидшим для виконання. Його нечутливість до шуму також є перевагою порівняно з операторами Собеля та Прюїтта.

Однак в оператора є й свої недоліки. Використання 2×2 ядра може призвести до меншої деталізації на карті країв, оскільки він може пропустити деякі дрібніші деталі. Крім того, він може не працювати так ефективно на зображеннях з низьким контрастом або зображеннях зі складними фонами.

В цілому, оператор Робертса – це ефективний інструмент для виявлення країв у сценаріях, де пріоритетними є обчислювальна простота та швидкість. Однак він може вимагати доповнення більш складними методами, коли потрібна більша деталізація.

2.5 Дослідження методу Кенні

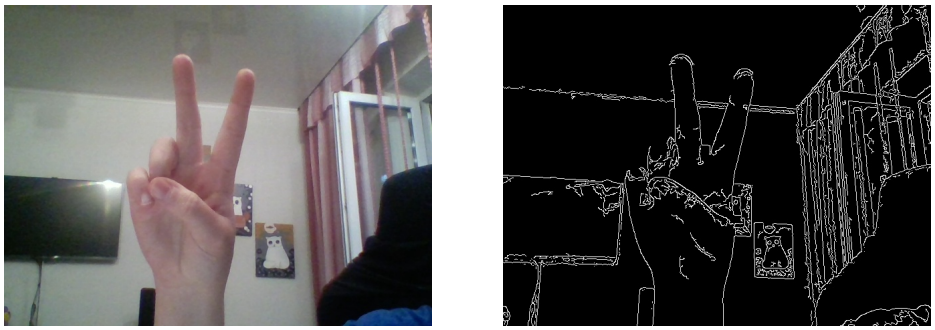
У Python реалізація методу Кенні є прямолінійною за допомогою бібліотеки OpenCV. Конкретно використовується функція cv2.Canny(), яка потребує джерело зображення, нижнє і верхнє порогові значення як параметри. Нижній і верхній пороги використовуються на етапах подвійного порогу і відстеження країв за допомогою гістерезису.

При застосуванні методу Кенні до нашого набору даних, ми

з'ясували, що він надає високу точність виявлення країв. Алгоритм ефективно виділяє краї жестів рук, при цьому успішно пригнічуючи шум, що значно знижує випадки помилкового виявлення країв.

Одним з помітних спостережень було те, що вибір нижнього та верхнього порогів значно впливав на результати. Вибір відповідних значень для цих параметрів був критичним для отримання точного виявлення країв.

На обраному прикладі були проаналізовані різні параметри для реалізації методу Кенні. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при $threshold_1 = 20, threshold_2 = 60$. Тому для подальших експериментів буде використовуватися саме таке значення.



(а) Тестове зображення **(б)** Метод Кенні

Рисунок 2.5 – Результат роботи методу Кенні.

Результат полягає в тому, що метод виявлення країв Кенні виступає як гарна техніка для виявлення країв. Його підхід, який включає згладжування, розрахунок градієнта інтенсивності, подавлення не максимальних значень і гістерезис, робить його особливо ефективним. Саме на нашому прикладі ми можемо побачити певні недоліки у вигляді наявності зайвих задетектованих об'єктів, а також, в місцях зі схожиим рівнем освітленості, невиявлені контури.

Основні переваги методу Кенні включають високу точність, низьку частоту помилок та добру локалізацію. Багатоетапний процес успішно

розв'язує проблему шуму, роблячи метод менш схильним до помилкових виявлень. Крім того, відстеження країв знижує ймовірність розриву країв.

Однак метод Кенні також представляє кілька недоліків. Вибір порогів є важливим аспектом, і невідповідні значення можуть призвести до поганих результатів. Крім того, багатоетапний алгоритм може бути обчислювально-інтенсивним, особливо для великих зображень високої роздільної здатності.

В основі, метод Кенні, не зважаючи на обчислювальну складність, є ефективним та надійним інструментом для виявлення країв, особливо в сценаріях, де пріоритетними є точність та шум.

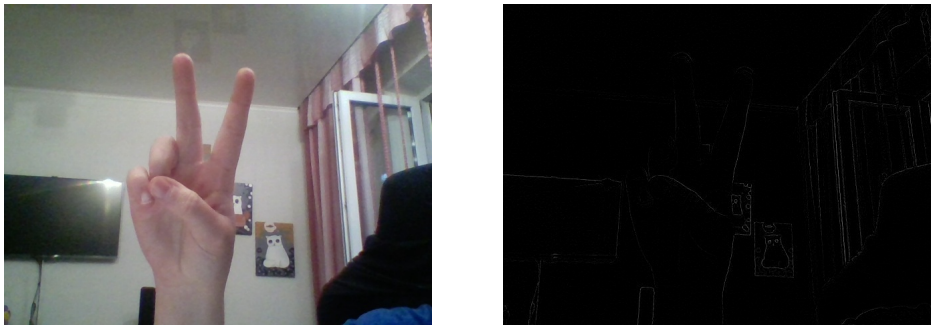
2.6 Дослідження методу Лапласа

Метод виявлення країв за допомогою Лапласіану – це метод другого порядку, який використовується для виявлення країв на зображеннях. На відміну від методів першого порядку, таких як оператори Собеля, Прюїтта та Робертса, які обчислюють градієнт інтенсивності, метод Лапласа обчислює зміну градієнта, що дозволяє йому виявляти як кінці країв, так і центр країв.

Для впровадження методу Лапласа в Python ми використовуємо бібліотеку OpenCV, а точніше функцію cv2.Laplacian(). Ця функція обчислює Лапласіан зображення, враховуючи конкретний розмір ядра.

При застосуванні методу Лапласа до нашого набору даних, ми спостерігали, що він був не дуже ефективним у виявленні країв саме в наших умовах. Через його чутливість до шуму, метод часто виділяв хибні краї на шумних зображеннях. І через різну освітленість об'єктів, він досить слабо виділяв контури, іноді взагалі майже не виділяючи їх.

Одним з ключових висновків було те, що вибір розміру ядра значно впливає на результати. Більші ядра надають ширший діапазон виявлення країв, тоді як менші ядра пропонують більш локалізоване виявлення країв.



(а) Тестове зображення **(б)** Метод Лапласа

Рисунок 2.6 – Результат роботи методу Лапласа.

Результат полягає в тому, що метод Лапласа, в деяких випадках, може бути ефективним інструментом для виявлення країв, особливо у сценаріях, де важливе виявлення як кінців країв, так і центру країв. Проте саме в наших умовах для нашої задачі він показав себе з неефективним результатом. Дуже слабко виявлені контури і розриви у них впливають на те, що на зображенні складно може бути щось розібрати.

Основним недоліком методу Лапласа є його висока чутливість до шуму, яка часто призводить до виявлення хибних країв. Крім того, в оператора Лапласіана відсутній асоційований з ним напрямок, на відміну від операторів Собеля, Прюїтта або Робертса. Цей відсутній напрямок може іноді призводити до труднощів у тлумаченні карти країв, особливо на зображеннях зі складними структурами.

2.7 Дослідження методу розширення та ерозії

Розширення та ерозія є базовими морфологічними операціями, що використовуються в обробці зображень. Ці методи дуже ефективні для зменшення шуму, ізоляції окремих елементів та об'єднання розрізних елементів зображення.

Розширення додає пікселі до меж об'єктів на зображенні, в той час, як ерозія видаляє пікселі на межах об'єктів. Комбінація цих двох може

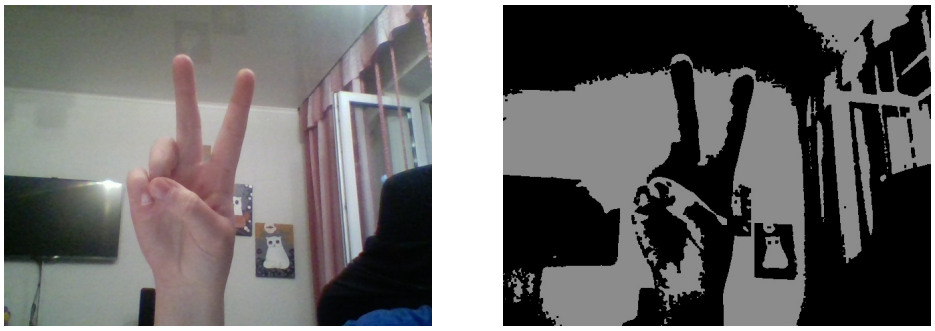
привести до ефективного виявлення країв.

У Python бібліотека OpenCV пропонує морфологічні перетворення за допомогою функцій cv2.dilate() та cv2.erode(). У нашій реалізації ми спочатку застосовуємо операцію ерозії до оригінального зображення, а потім операцію розширення. Віднімаючи зображення ерозії від зображення розширення, ми отримуємо зображення, яке висвітлює краї об'єктів.

Застосовуючи методи розширення та ерозії до нашого набору даних, ми спостерігали, що ці методи були дуже ефективні при підсиленні та виявленні країв жестів рук, навіть на шумних зображеннях. Однак вибір структурного елементу та його розміру, що використовуються при операціях розширення та ерозії, значно впливав на результати.

Одне ключове спостереження полягало в тому, що більші структурні елементи призводили до більш вираженого ефекту розширення та ерозії, що іноді могло призвести до втрати деталей у результатах виявлення країв.

На обраному прикладі були проаналізовані різні параметри для реалізації методу розширення та ерозії. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при $threshold_1 = 132, threshold_2 = 140$. Тому для подальших експериментів буде використовуватися саме таке значення.



(а) Тестове зображення **(б)** Метод розширення та ерозії

Рисунок 2.7 – Результат роботи методу розширення та ерозії.

Серед переваг, розширення та ерозія сожуть бути ефективні у

зменшенні шуму і можуть ізолювати або об'єднувати окремі елементи на зображенні, залежно від конкретних вимог. Це робить їх особливо корисними на етапі попередньої обробки перед застосуванням більш складних методів аналізу зображень.

Однак, ці методи також мають свої недоліки. Вибір структурного елементу та його розміру може значно впливати на результати, а невірні вибори можуть призвести до поганих результатів виявлення країв або втрати деталей.

По суті, хоча розширення та ерозія надають потужні інструменти для зменшення шуму та виявлення країв, потрібно уважно ставитися до вибору структурних елементів та їх розмірів. Попри потенційні обчислювальні витрати, ці методи пропонують цінні етапи попередньої обробки для більш складних завдань аналізу зображень.

2.8 Дослідження методу відкриття та закриття

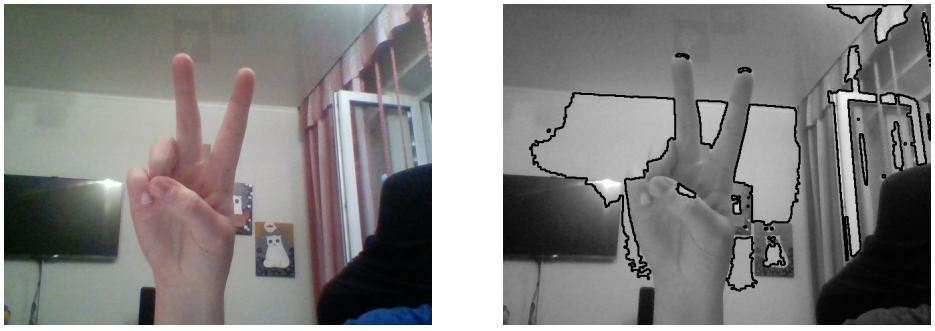
Відкриття та закриття – це морфологічні перетворення, що використовуються в обробці зображень з метою зменшення шуму та розділення об'єктів. Відкриття – це операція, яка включає ерозію, за якою слід розширення, корисна для видалення шуму та розділення об'єктів, що знаходяться близько один до одного. Закриття, яке є розширенням, за яким слідує ерозія, корисне для закриття малих отворів на передньому плані та з'єднання розрізних структур на зображенні.

В Python бібліотека OpenCV надає функцію cv2.morphologyEx() для застосування цих перетворень. Для реалізації відкриття та закриття функція вимагає оригінального зображення, типу операції (або відкриття, або закриття) та структурного елементу в якості параметрів.

Після застосування методів відкриття та закриття до нашого набору даних, ми з'ясували, що ці методи можуть досить ефективно виділяти краї в умовах шуму або коли об'єкти були тісно розташовані. Проте саме в випадку нашого експерименту даний метод показав себе не дуже

ефективно, адже рівень освітленості досить сильно впливав на те, що рука і фоновий колір сприймалися одним контуром, а не 2 окремих, як то має бути при ідеальному визначенні контурів

Одним з ключових спостережень було те, що більші структурні елементи могли призвести до більш виражених ефектів відкриття та закриття, але з ризиком втрати деталей у процесі виявлення країв.



(а) Тестове зображення **(б)** Метод відкриття та закриття

Рисунок 2.8 – Результат роботи методу відкриття та закриття.

На завершення, перетворення відкриття та закриття в деяких ситуаціях можуть бути хорошими інструментами для зменшення шуму та виявлення країв.

Основними перевагами цих методів є їх здатність здійснювати зменшення шуму та розділення об'єктів за один крок, а також гнучкість у виборі розміру структурного елементу, який може дозволити різні рівні виявлення країв.

Проте, ці методи мають декілька обмежень. Вибір структурного елементу та його розміру може значно вплинути на результати, потенційно призводячи до втрати деталей. Крім того, оскільки ці операції включають як ерозію, так і розширення, вони можуть бути обчислювально-дорогими, особливо для більших зображень.

Підсумовуючи, хоча відкриття та закриття пропонують потужні інструменти для виявлення країв, потрібно приділити увагу вибору структурних елементів та їх розмірів. Попри потенційні обчислювальні

витрати, ці методи пропонують цінні вступні етапи для більш складних задач аналізу зображень.

2.9 Дослідження методу Марра-Хілдрета

Метод виявлення країв Марра-Хілдрета – це класичний метод в області обробки зображень. Цей метод використовує функцію Лапласіана гауссівської (LoG) як фільтр для згортки з зображенням. Ідея полягає в тому, щоб виявити точки з нульовими перетинами на зображенні, які представляють краї.

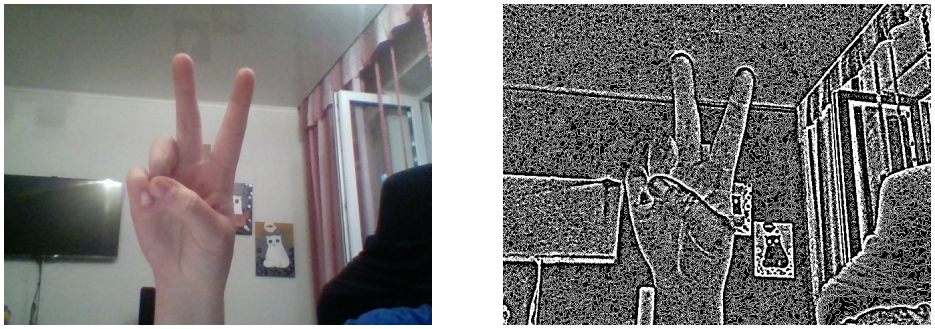
Реалізація методу Марра-Хілдрета в Python включає кілька кроків. Спочатку до зображення застосовується фільтр гауса для зменшення шуму. Далі обчислюється Лапласіан зі згладженого зображення, і, нарешті, визначаються точки з нульовими перетинами для визначення країв.

Після застосування методу Марра-Хілдрета до нашого набору даних ми відзначили його надійність у точному виявленні країв. Однак, як і інші методи другого порядку, він був чутливий до шуму, і тому потребував оптимального вибору розміру та стандартного відхилення Гауссового фільтра для ефективних результатів.

Одним з ключових висновків було те, що метод Марра-Хілдрета, попри його чутливість до шуму, був відмінним у виявленні країв, де зміна градієнта є максимальною, що є теоретичним визначенням краю.

На обраному прикладі були проаналізовані різні параметри для реалізації методу Марра-Хілдрета. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при $\sigma = 1.2, threshold = 10$. Тому для подальших експериментів буде використовуватися саме таке значення.

Результатом є те, що метод Марра-Хілдрета є потужним інструментом для виявлення країв, особливо у сценаріях, коли важливо точне розташування краю.



(а) Тестове зображення **(б)** Метод Марра-Хілдрета

Рисунок 2.9 – Результат роботи методу Марра-Хілдрета.

Серед його переваг, метод Марра-Хілдрета є точним і надійним, і він надає формальне визначення краю (максимальна зміна градієнта). Щобільше, розмір Гауссового фільтра та стандартне відхилення можна регулювати відповідно до конкретних вимог, що додає гнучкості до методу.

Однак, метод також має свої недоліки. Він чутливий до шуму і вимагає оптимального вибору розміру та стандартного відхилення Гауссового фільтра. Крім того, він обчислювально-дорожчий через використання двоетапного процесу (згладжування Гауссовим фільтром, за яким слідує Лапласіан) порівняно з іншими методами виявлення країв.

В сутності, хоча метод Марра-Хілдрета надає точний та надійний механізм виявлення країв, при виборі методу виявлення країв слід враховувати вибір параметрів Гауссового фільтра та його обчислювальні витрати.

2.10 Дослідження методу Шен-Кастана

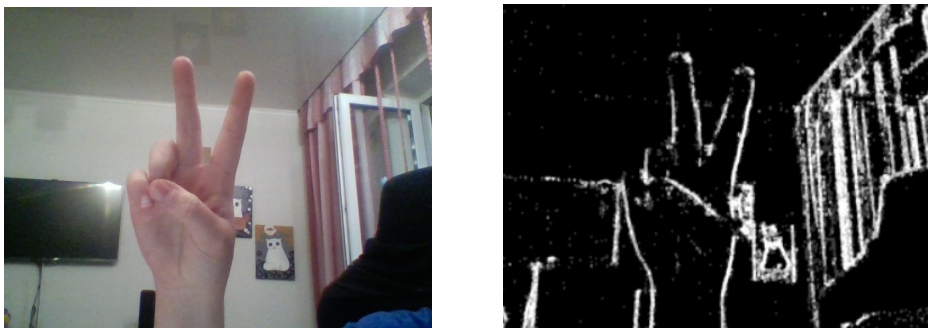
Метод виявлення країв Шен-Кастана, що є впливовим методом у галузі обробки зображень, використовує ізотропну похідну гауссівської згладжувальної функції. Цей метод дозволяє більш точно локалізувати краї та покращує робастність порівняно з більш традиційними методами виявлення країв.

Щодо реалізації даного методу, цей процес включає кілька етапів. Спочатку зображення згладжується за допомогою гауссівського фільтра для зниження шуму. Потім до згладженого зображення застосовується ізотропний диференційний фільтр, за яким слідує процедура подавлення не максимумів. Фінальний етап – застосування гістерезисного порогу для ізоляції точок краю.

При застосуванні методу Шен-Кастана до нашого набору даних, було зауважено, що ця техніка добре виступала, навіть у шумних умовах, що демонструє можливості виявлення країв та імунітет до шуму, які притаманні цьому методу. Однак вибір параметрів гауссівського фільтра та значень порогових значень гістерезису значно впливав на продуктивність виявлення країв.

Важливе спостереження полягало в тому, що метод Шен-Кастана відмінно ідентифікував краї точно та надійно, навіть на фоні шуму зображення.

На обраному прикладі були проаналізовані різні параметри для реалізації методу Шен-Кастана. Проте, саме в даних умовах для задачі розпізнавання жестів та рухів руки себе найкраще показав метод при $r = 2, threshold = 0.035$. Тому для подальших експериментів буде використовуватися саме таке значення.



(а) Тестове зображення **(б)** Метод Шен-Кастана

Рисунок 2.10 – Результат роботи методу Шен-Кастана.

Результатом є те, що метод Шен-Кастана – це надійний та

ефективний інструмент для виявлення країв, особливо в умовах зі значним шумом.

Серед його переваг, даний метод забезпечує точну локалізацію країв і відмінний імунітет до шуму. Його здатність коригувати параметри гауссівського фільтра та гістерезисний поріг додатково підвищує його гнучкість та адаптивність.

Однак, метод не позбавлений викликів. Підбір оптимальних параметрів для гауссівського фільтра та гістерезисного порогу вимагає обережності, оскільки неоптимальний вибір може привести до гірших результатів виявлення краю. Щобільше, метод Шен-Кастана може бути більш обчислювально-витратним через його багатоетапний процес, порівняно з простішими методами виявлення країв.

В сутності, хоча метод Шен-Кастана пропонує надійне та точне виявлення краю, необхідно враховувати обчислювальні витрати та обережний вибір параметрів гауссівського фільтра та гістерезисного порогу при виборі методу виявлення краю. Попри потенційні обчислювальні виклики, цей метод робить цінний внесок в передобробку для більш складних завдань аналізу зображень.

2.11 Порівняльний аналіз методів

Проведено порівняльний аналіз висвітлених методів виявлення країв, використовуючи одне і те ж зображення для кожного методу, щоб оцінити їхню ефективність на основі часу обробки та розміру отриманого зображення. Ці параметри є критичними в робототехніці, де важливими є швидкість виконання та використання ресурсів. Нижче наведена таблиця, де можна побачити точні дані проведеного експерименту.

Після проведення детального порівняльного аналізу, можемо зробити наступні висновки щодо різних методів виявлення країв, заснованих на часі обробки та розмірі результівного зображення.

Таблиця 2.1 – Порівняння методів на час їх роботи та необхідний обсяг пам'яті.

Метод	Час обробки зображення	Розмір зображення	Нормування
Оператор Собеля	4.957 ms	113 kB	22.796 kB/ms
Оператор Прюїтта	10.941 ms	194 kB	17.731 kB/ms
Оператор Робертса	10.943 ms	96 kB	8.773 kB/ms
Метод Кенні	0.997 ms	99 kB	99.298 kB/ms
Метод Лапласа	0.997 ms	58 kB	58.175 kB/ms
Метод розширення та ерозії	0.996 ms	52 kB	52.209 kB/ms
Метод відкриття та закриття	18.973 ms	82 kB	4.322 kB/ms
Метод Марра-Хілдрета	3.991 ms	350 kB	87.697 kB/ms
Метод Шен-Кастана	5.014 ms	16 kB	3.191 kB/ms

Оператор Собеля демонструє добре співвідношення між часом обробки та розміром зображення, але він не є найшвидшим, ні найменшим за розміром зображення. Цей метод може бути оптимальним вибором для застосувань, де потрібен розумний компроміс між цими двома факторами. При цьому візуально метод визначив контури досить чітко.

Оператори Прюїтта і Робертса мають схожий час обробки, який відносно високий порівняно з іншими методами. Їх результівні розміри зображень досить відрізняються, при цьому Прюїтт генерує значно більше зображення. Це свідчить про те, що ці методи можуть бути використані в застосуваннях, де час обробки не є критичним фактором, а варіативність розміру зображення може бути врахована. Візуально дані методи змогли відносно чітко визначити контури, проте через вплив шумів помітна зернистість контурів.

Методи Кенні та Лапласа, а також метод розширення та ерозії, демонструють дивовижну швидкість обробки, кожен з них займає менше мілісекунди на обробку зображення. Однак метод Кенні дає більший розмір зображення, ніж методи Лапласа та розширення та ерозії, що може бути важливим для застосувань, де обмежені простір для зберігання або пропускна здатність. При цьому, судячи з візуальної складової, метод Кенні в даних умовах показав досить чіткий результат

визначених контурів, що підтверджує те, що даний метод може бути ефективним саме в даній задачі. В той час як метод Лапласа, а також метод розширення та ерозії показали не дуже точні результати, або навіть незадовільні.

Метод відкриття та закриття має найвищий час обробки, але не дає найбільшого розміру зображення, що свідчить про те, що він може не бути найкращим вибором для реальних застосувань, але може бути корисним, коли точність важливіша за швидкість. Проте, через вплив шумів, різну освітленість, в даних умовах цей метод іноді досить неточно визначав контури, вважаючи, що рука та задній фон це один контур, хоча їх мало б бути два.

Метод Марра-Хілдрета, хоч і не є найшвидшим, генерує найбільше зображення, що вказує на те, що він, можливо, краще підходить для сценаріїв, де розмір зображення не є критичним фактором, а потрібен вищий рівень деталізації. При цьому чітко можна розрізнити контури, які визначив даний метод.

Метод Шен-Кастана демонструє добрий час обробки та дає найменше зображення, що робить його відмінним вибором для застосувань, що вимагають швидкої обробки та мінімального зберігання. Щодо візуальної складової, то попри шуми на зображенні метод показав себе досить добре і досить ефективно розрізняв жести рук на зображеннях.

Висновки до розділу 2

Після детального вивчення та порівняння різних методів виявлення контурів, можна зробити кілька важливих висновків, особливо з огляду на їх потенційне застосування в робототехніці для визначення жестів руки людини.

Метод Собеля. Цей метод показав себе добре з точки зору балансу між часом обробки та розміром зображення. Він може бути

корисним для систем, які не вимагають екстремально високої швидкості обробки, але потребують доброї точності детекції контурів.

Метод Кенні. Попри те, що цей метод дає більший розмір зображення, він відрізняється дивовижною швидкістю обробки. Це може бути особливо важливо для робототехнічних систем, які потребують миттєвого відгуку на людські жести.

Метод Шен-Кастана. Цей метод, є надзвичайно швидким і дає дуже маленькі зображення, що робить його відмінним для систем, де обмежені ресурси обчислювальної потужності та пам'яті.

На основі цих висновків можна стверджувати, що найбільш придатні для застосування в робототехніці для визначення жестів руки людини є методи Собеля, Кенні та Шен-Кастана, проте вибір конкретного методу в кожному випадку повинен визначатися відповідно до конкретних обставин та вимог.

Додатково, необхідно врахувати, що в подальшій реалізації будуть тестуватися всі розглянуті методи незалежно від їх поточних результатів. Причина цього полягає в тому, що в контексті взаємодії зі згортковими нейронними мережами, характеристики кожного методу можуть по-різному впливати на процес навчання моделей, а отже, на ефективність їх роботи. Це може відбутися через те, що нейронні мережі добре працюють з різноманітними даними й в залежності від характеристик вхідних даних, різні методи можуть сприяти кращому навчанню мережі.

Також важливо пам'ятати, що окремі особливості роботи кожного методу можуть виявитися корисними в певних специфічних сценаріях використання, які могли б не бути виявлені під час загального порівняльного аналізу.

Отже, наступним кроком у цьому дослідженні буде детальне тестування всіх розглянутих методів у контексті їх використання для навчання згорткових нейронних мереж і визначення, які з них найбільше підходять для цієї конкретної задачі.

3 ЗАСТОСУВАННЯ МЕТОДІВ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ

3.1 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) є категорією нейронних мереж, які виявилися дуже ефективними в таких областях, як розпізнавання та класифікація зображень. Вони були надихнуті організацією зорової кори тварин і є варіаціями багатошарових перцептронів, розроблених для використання мінімального обсягу попередньої обробки.

Розробка згорткових нейронних мереж (CNN) була сильно натхненна біологічними процесами та структурою людського мозку, зокрема зоровим кортексом, що відповідає за наші здатності розпізнавання зображень. Учені 1950-х та 1960-х років, такі як Девід Г. Хюбель і Торстен Візель, проводили нейрофізіологічні експерименти на зоровому кортексі котів та мавп [12]. їхні висновки показали, як конкретні ділянки мозку переважно реагують на певні форми та патерни, що стало раннім уявленням про те, що пізніше стало концепцією карт ознак у CNN.

Перша модель згорткової нейронної мережі була запропонована Куніхіко Фукусіма в 1980 році, відома як "Неокогнітрон". Попри те, що ця піонерська архітектура впровадила основні принципи згорткових шарів, вона не мала таких можливостей навчання, як у сучасних CNN.

У 1989 році Ян Лекун та його команда, натхненні "Неокогнітроном"Фукусіми [13], розробили LeNet-1, першу згорткову мережу з 7 рівнями, яка використовувалася для класифікації цифр для зчитування поштових індексів на листах. LeNet-1пізніше було вдосконалено і перетворено в модель LeNet-5, яка мала поліпшений дизайн і значно вплинула на сучасну архітектуру CNN. LeNet-5 успішно виконувала завдання розпізнавання цифр на чеках і стала важливим етапом в розвитку CNN.

З'явлення потужних графічних процесорів, наявність великих баз зображень та вдосконалені методи навчання у 2000-х роках призвели до відродження CNN. У 2012 році був досягнутий значний прорив зі створенням AlexNet Алекса Кріжевського, Ільї Суцкевера і Джеффрі Хінтона [14]. AlexNet значно перевершив всі попередні моделі в конкурсі ImageNet і поставив CNN на передній план в галузі розпізнавання зображень, де вони займають центральне місце до цього часу.

CNN - це послідовність шарів, і кожен шар перетворює один обсяг активацій на інший за допомогою диференційованої функції. Вона складається з вхідного та вихідного шарів, а також кількох прихованих шарів. Приховані шари CNN зазвичай складаються з набору згорткових шарів, шарів пулінгу, повністю з'єднаних шарів і шарів нормалізації.

Основним будівельним блоком CNN є згортковий шар, де мережа застосовує кілька фільтрів до вхідних даних. Цей процес схожий на застосування різних фільтрів до зображення (наприклад, у програмному забезпеченні для редагування зображень) для створення різних варіантів початкового зображення. Кожен фільтр виділяє різні ознаки зображення, такі як краї, криві тощо.

Основним аспектом згорткових шарів є їхній спеціальний дизайн для обробки даних з сітчастою топологією (такою, як у зображення, яке має ширину, висоту та кольорові канали) та інваріантність до зсувів. Це означає, що один і той самий фільтр використовується по всьому зображенню, розпізнаючи патерни, незалежно від їхнього положення.

Після кожного згорткового шару часто використовується шар пулінгу, який зменшує просторовий розмір представлення, зменшуючи кількість параметрів та обчислень в мережі та, отже, контролюючи перенавчання. Шар пулінгу працює незалежно на кожному глибинному зрізі вхідних даних і змінює його просторовий розмір.

Повністю з'єднаний шар бере всі нейрони з попереднього шару (зазвичай шар пулінгу) та з'єднує їх з кожним окремим нейроном. Повністю з'єднаний шар навчається розпізнавати глобальні патерни в

просторі ознак вхідних даних, на відміну від локальних патернів, вивчених згортковими шарами.

Нарешті, шари нормалізації (також відомі як функції активації), такі як uU (Rectified Linear Unit), вводять нелінійність у мережу, перетворюючи згорнуті ознаки на невід'ємні значення, покращуючи ефективність процесу навчання та продуктивність мережі.

Останнім кроком у CNN є використання функції втрати на вихідному шарі, яка вимірює помилку в передбаченнях мережі, і використання алгоритму оптимізації, такого як градієнтний спуск, для оновлення ваг у мережі таким чином, щоб мінімізувати цю помилку.

Завдяки повторним проходженням прямого поширення (обчислення виходу з входу та ваг) та зворотного поширення (оновлення ваг для зменшення помилки) через мережу під час навчання на наборі даних, мережа налаштовує свої ваги таким чином, щоб вивчити відображення вхідних зображень на правильні класифікаційні мітки. Ця потужна здатність вивчати виділяти найбільш характерні ознаки з сирих вхідних зображень робить CNN потужним інструментом для складних завдань візуального розпізнавання.

Можливі переваги CNN для розпізнавання жестів руки в робототехніці:

– **Висока точність.** CNN довели свою надзвичайну успішність у завданнях класифікації зображень, часто досягаючи високої точності.

– **Зменшена попередня обробка.** CNN автоматично вчаться та витягують особливості, що виключає потребу в ручному витягуванні особливостей.

– **Стійкість до зміщення та обертання.** Через архітектуру CNN, вони є незмінними до локального зміщення та обертання, що є важливим для розпізнавання жестів, які можуть виконуватися в різних орієнтаціях.

– **Ієрархічне вивчення особливостей.** CNN вчать особливості на різних рівнях абстракції, що робить їх дуже ефективними для складних завдань, таких як розпізнавання жестів.

Можливі недоліки CNN для розпізнавання жестів руки в робототехніці:

– **Потреба у великому наборі даних.** CNN зазвичай вимагають багато даних для ефективного тренування та уникнення перенавчання.

– **Висока обчислювальна вартість.** CNN є обчислювально-витратними, що може сповільнити процес тренування і може бути непридатним для деяких реальних застосувань.

– **Інтерпретація.** Часто важко зрозуміти, чому CNN робить конкретні прогнози, що може бути проблематичним у ситуаціях, коли потрібна інтерпретованість.

Попри виклики, CNN виявилися незамінним інструментом у розпізнаванні жестів руки для робототехніки через їх здатність справлятися зі складними візуальними завданнями. Далі буде детально розглянуто методику використання CNN для нашого конкретного застосування в розпізнаванні жестів. Також буде досліджено інтеграцію технік виявлення контурів з CNN для потенційного покращення точності та ефективності системи розпізнавання жестів руки, і буде проведено глибоке тестування кожного методу в контексті тренування згорткових нейронних мереж для визначення того, які методи найкраще підтримують оперативні потреби нашого робототехнічного застосування.

3.2 Побудова архітектури CNN

Процес створення CNN для класифікації рухів руки в робототехніці включає кілька кроків. Використання бібліотек TensorFlow та Keras допомагає зручно та ефективно реалізувати це завдання.

Першим кроком є підготовка набору даних для навчання та валідації моделі. Використовується об'єкт ImageDataGenerator, який нормалізує значення пікселів зображень та застосовує різноманітні перетворення до даних. Наприклад, рандомізоване обертання, зсуви та відображення зображень горизонтально. Це допомагає збільшити

різноманітність даних та покращити загальну роботу моделі.

Після цього створюються генератори даних для навчання та валідації. Вони використовуються для зчитування та підготовки даних з відповідних директорій. Важливо вказати розмір зображень та режим класифікації ('categorical').

Далі визначається архітектура моделі. Архітектура цієї CNN включає кілька шарів згортки, пулінгу та повністю з'єднаних шарів.

1) Перший шар Conv2D з 32 фільтрами розміром (3, 3) та функцією активації ReLU отримує на вхід зображення розміром (224, 224, 3). Цей шар виконує операцію згортки на вхідних даних і застосовує нелінійність через функцію активації.

2) Після цього застосовується шар MaxPooling2D з розміром (2, 2), який зменшує розмір представлення шляхом вибору максимального значення з кожного пулінгового вікна.

3) Слідом за цим повторюється процес згортки та пулінгу з другим шаром Conv2D з 64 фільтрами та третім шаром Conv2D зі 128 фільтрами. Ці шари дозволяють моделі вивчати більш складні ознаки на різних рівнях абстракції.

4) Після останнього шару пулінгу MaxPooling2D ми використовуємо шар Flatten, який перетворює двовимірний вихід з попередніх шарів в одновимірний вектор, готовий для введення в повністю з'єднані шари.

5) Наступним шаром є повністю з'єднаний шар Dense зі 128 нейронами та функцією активації 'relu'. Він вивчає глобальні патерни в просторі ознак і допомагає моделі дійти до класифікації.

6) Завершальний повністю з'єднаний шар Dense має кількість нейронів, що відповідає кількості класів, яку ми хочемо передбачати. Функція активації 'softmax' використовується для отримання імовірностей для кожного класу.

Наступним кроком є визначення функції втрати та оптимізатора для моделі. У цьому випадку використовується функція втрати 'categorical crossentropy' та оптимізатор 'adam'. Нарешті, тренуємо модель

на даних з використанням генераторів даних для навчання та валідації за допомогою методу fit. Кількість епох і інші параметри тренування встановлюються відповідно до потреб нашої задачі.

Ця архітектура CNN дозволяє моделі вивчати різні рівні абстракції та ознаки зображень для класифікації рухів руки в робототехніці. Застосування згорткових шарів, пулінгу та повністю з'єднаних шарів дозволяє моделі автоматично вилучати та розпізнавати релевантні ознаки зображень, забезпечуючи високу точність класифікації.

3.3 Порівняльний аналіз роботи CNN на основі методів

Проведено порівняльний аналіз натренованих згорткових нейронних мереж на основі обговорених методів виявлення країв, щоб оцінити їхню ефективність на основі коректності розпізнавання жестів руки людини. Адже досягання максимальної точності є критично важливим завданням. Нижче наведена таблиця, де можна побачити точні дані проведеного експерименту. Всі тренування моделей проводилися на основі однієї архітектури та в абсолютно однакових умовах. Кількість епох становить 50, розбиття на тренувальну та тестову вибірку становить 70% до 30/

З таблиці порівняння методів розпізнавання контурів, можна зробити наступні висновки:

1) Найвищу точність досягнуто з використанням методу Шен-Кастана, де Accuracy score становить 95,78%. Цей метод продемонстрував найкращу продуктивність серед усіх розглянутих методів.

2) Метод Марра-Хілдрета також показав високий рівень точності, з Accuracy score на рівні 91,12%.

3) Метод розширення та ерозії дав впевнений результат з точністю 88,13%, що робить його привабливим для використання в системі розпізнавання жестів руки.

4) Оператор Собеля, Оператор Прюїтта та Метод Кенні показали

Таблиця 3.1 – Порівняння методів на час їх роботи та необхідний обсяг пам'яті.

Метод	Accuracy score
Оператор Собеля	68,44%
Оператор Прюїтта	63,56%
Оператор Робертса	63,19%
Метод Кенні	64,31%
Метод Лапласа	35,06%
Метод розширення та ерозії	88,13%
Метод відкриття та закриття	24,37%
Метод Марра-Хілдрета	91,12%
Метод Шен-Кастана	95,78%

прийнятну точність в діапазоні від 63,56% до 68,44%. Вони можуть бути ефективними з точки зору ресурсів та швидкодії, але вимагають покращення для підвищення точності.

5) Метод Лапласа та Метод відкриття та закриття продемонстрували нижчу точність, з Accuracy score від 24,37% до 35,06%. Ці методи можуть бути менш ефективними для задачі розпізнавання жестів руки в робототехніці.

Таблиця 3.2 – Зведені результати дослідження методів та натернованих моделей.

Метод	Час обробки зображення	Розмір зображення	Accuracy score
Оператор Собеля	4.957 ms	113 kB	68,44%
Оператор Прюїтта	10.941 ms	194 kB	63,56%
Оператор Робертса	10.943 ms	96 kB	63,19%
Метод Кенні	0.997 ms	99 kB	64,31%
Метод Лапласа	0.997 ms	58 kB	35,06%
Метод розширення та ерозії	0.996 ms	52 kB	88,13%
Метод відкриття та закриття	18.973 ms	82 kB	24,37%
Метод Марра-Хілдрета	3.991 ms	350 kB	91,12%
Метод Шен-Кастана	5.014 ms	16 kB	95,78%

Загалом, з таблиці видно, що метод Марра-Хілдрета та метод Шен-Кастана є найкращими варіантами для розпізнавання жестів руки в робототехніці з точки зору ефективності натренованих моделей в ході даного дослідження, оскільки вони досягають високої точності. Однак, вибір оптимального методу також може залежати від інших факторів, таких як обчислювальні вимоги та складність реалізації.

Висновки до розділу 3

Було розглянуто використання згорткових нейронних мереж (CNN) для розпізнавання жестів руки в робототехніці. Було проведено дослідження різних методів і оцінено їх ефективність на основі точності моделей.

Найвищий рівень точності досягнуто з використанням методу Шен-Кастана, де показник точності становить 95,78%. Цей метод продемонстрував найкращу продуктивність серед усіх розглянутих методів. Метод Марра-Хілдрета також показав високий рівень точності, з показником точності на рівні 91,12%. Оператор Собеля, Оператор Прюїтта та Метод Кенні показали прийнятну точність в діапазоні від 63,56% до 68,44%. Вони можуть бути ефективними з точки зору ресурсів та швидкодії, але вимагають покращення для підвищення точності.

Загалом, з отриманих даних можна спостерігати, що метод Марра-Хілдрета та метод Шен-Кастана є найкращими варіантами для розпізнавання жестів руки в робототехніці з точки зору ефективності натренованих моделей в ході даного дослідження, оскільки вони досягають високої точності. Однак, вибір оптимального методу також може залежати від інших факторів, таких як обчислювальні вимоги та складність реалізації.

ВИСНОВКИ

В ході виконання даної дипломної роботи було досягнуто ряду важливих результатів, які відображають значний внесок в область практичного застосування комп'ютерного зору в робототехніці, зокрема в детекції жестів руки.

Аналіз літератури показав, що різні методи мають свої переваги та недоліки, а також обмеження в застосуванні, тому в різних ситуаціях методи можуть показати себе абсолютно по різному, що приводить до того, що для кожної конкретної задачі треба індивідуально підбирати основу роботи. Також, згорткові нейронні мережі демонструють високий потенціал у розпізнаванні образів і можуть ефективно використовуватися для детекції жестів руки.

На основі огляду було протестовано та досліджено обрані методи саме в умовах нашої задачі – розпізнавання жестів руки. Було розглянуто їх особливості, а також порівняно часові та просторові результати ефективності їх роботи

Також, на основі огляду було розроблено модель використання CNN для детекції жестів руки в робототехніці. Було розглянуто основні принципи роботи CNN, їх структуру та особливості навчання, а також способи оцінки їхньої продуктивності.

Практичне використання CNN для детекції жестів руки було продемонстровано на прикладі дослідження різних методів. Було оцінено ефективність кожного методу за допомогою показника Accuracy score. Метод Шен-Кастана та метод Марра-Хілдрета продемонстрували найвищу точність, що демонструє їх придатність для розпізнавання жестів руки в робототехніці.

Проаналізовано вплив використання різних методів детекції країв на точність моделей згорткових нейронних мереж. Аналіз показав, що вибір методу детекції країв може значно впливати на точність моделі та вимагає

додаткових досліджень для оптимального вибору.

В результаті проведеного дослідження можна зробити висновок, що комп'ютерний зір та згорткові нейронні мережі мають великий потенціал для використання в робототехніці для детекції жестів руки. Однак, важливо розуміти, що успіх в реалізації таких систем залежить від багатьох факторів, включаючи правильний вибір методу детекції країв, обробку даних та тренування моделі. Ця робота показала, що дослідження в цій області мають велику перспективу та відкривають нові можливості для розвитку робототехніки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Duda, R. O., Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 11-15.

2. Prewitt, J. M. S. (1970). Object enhancement and extraction. Picture processing and Psychopictorics, 15-19.

3. Sobel, I., Feldman, G. (1968). A 3x3 isotropic gradient operator for image processing, 271-272.

4. Roberts, L. G. (1965). Machine perception of three-dimensional solids. In Optical and Electro-optical Information Processing. MIT Press, 159-197

5. Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, 679-698.

6. Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.

7. Davies, E. R. (2012). Computer and machine vision: theory, algorithms, practicalities. Elsevier.

8. Parker J.R. (2011). Algorithms for Image Processing and Computer Vision. Second Edition. Wiley Publishing, Inc

9. Marr, D., Hildreth, E. (1980). Theory of Edge Detection. Proceedings of the Royal Society of London. Series B, Biological Sciences, 187–217.

10. Shen, J., Castan, S. (1992). An optimal linear operator for step edge detection. CVGIP: Graphical Models and Image Processing, 112-133.

11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 2278-2324.

12. Hubel, D. H., Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology, 574–591.

13. Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 193–202.

14. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems.

15. Rosebrock, A. (2017). Deep Learning for Computer Vision with Python. PyImageSearch.

16. Nixon, M. S., Aguado, A. S. (2012). Feature Extraction and Image Processing for Computer Vision. Academic Press.

17. Jan Erik Solem (2012). Programming Computer Vision with Python: Tools and algorithms for analyzing images. O'Reilly Media.

18. Sonka, M., Hlavac, V., Boyle, R. (2014). Image Processing, Analysis, and Machine Vision. Cengage Learning.

19. Wu, T. F., Chen, S. H. (2006). Vision-based gesture recognition: A review. Journal of Robotics and Autonomous Systems

20. Yang, J., Yuan, J., Liu, Z. (2018). Recent advances in vision-based hand gesture recognition. Frontiers of Computer Science

21. Breazeal, C. (2002). Designing sociable robots. MIT press.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

А.1 Програма для збору зображень

```
# Import libraries
import os
import cv2

# Images path
img_dir = './images'

# Create images directory if it does not exist
if not os.path.exists(img_dir):
os.makedirs(img_dir)

# There are 32 comninations of fingers on one
        hand
combinations_amount = 32
# Amount of images to generate for one run
images_amount = 100

cap = cv2.VideoCapture(0)

for i in range(combinations_amount):
# Create image class directory if it does
        not exist
if not os.path.exists(os.path.join(img_dir,
        str(i))):
os.makedirs(os.path.join(img_dir, str(i)))

# Get ready to start i class capturing
```

```
        print(f"Capturing images for {i+1}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣combination...")
        capture_success = False

        # Waiting to press 'c' button to start
                collecting class images
        while True:
        ret, frame = cap.read()
        cv2.imshow('Collecting images', frame)
        if cv2.waitKey(25) == ord('c'):
        break

        # Starting collected images counter from 0
        img_count = 0
        # Saving images in range of images_amount
        while img_count < images_amount:
        ret, frame = cap.read()
        cv2.imshow('Collecting images', frame)
        cv2.waitKey(25)
        cv2.imwrite(os.path.join(img_dir, str(i),
                '{}.jpg'.format(img_count)), frame)
        img_count += 1

        cap.release()
        cv2.destroyAllWindows()
```

A.2 Програма реалізації методів виявлення контурів

```
        # Import libraries
        import numpy as np
```

```
import cv2
import matplotlib.pyplot as plt
import time
import os

# Load an image
image_path = "./images/12/100.jpg"
image = cv2.imread(image_path)

def imgCompare(images, titles, fig_shape=(1, 2),
        fig_size=(10, 5)):
fig = plt.figure(figsize=fig_size)
for i, img in enumerate(images):
ax = fig.add_subplot(*fig_shape, i + 1)
ax.imshow(img, cmap='gray')
ax.grid(False)
ax.set_xticks([])
ax.set_yticks([])
plt.title(titles[i])
plt.tight_layout()
plt.show()

# Sobel's Operator
def sobel_operator(image, threshold):
# Convert the image to grayscale
gray_image = cv2.cvtColor(image,
        cv2.COLOR_BGR2GRAY)

# Apply Sobel operator
sobel_x = cv2.Sobel(gray_image,
        cv2.CV_64F, 1, 0, ksize=3)
```

```
sobel_y = cv2.Sobel(gray_image,
        cv2.CV_64F, 0, 1, ksize=3)

# Calculate the magnitude of gradients
gradient_magnitude = np.sqrt(sobel_x**2
        + sobel_y**2)

# Normalize the gradient magnitude to
        the range [0, 255]
gradient_magnitude = cv2.normalize(
        gradient_magnitude, None, 0, 255,
        cv2.NORM_MINMAX, dtype=cv2.CV_8U)

# Apply a threshold to obtain the binary edge map
_, edges = cv2.threshold(gradient_magnitude,
        threshold, 255, cv2.THRESH_BINARY)

return edges

# Prewitt's Operator
def prewitt_operator(image, threshold):
# Convert the image to grayscale
gray_image = cv2.cvtColor(image,
        cv2.COLOR_BGR2GRAY)

# Define the Prewitt kernels
prewitt_kernel_x = np.array([[-1, 0, 1],
[-1, 0, 1],
[-1, 0, 1]])
prewitt_kernel_y = np.array([[-1, -1, -1],
[0, 0, 0],
```

```
[1, 1, 1]])

# Apply the Prewitt kernels to compute
        gradients in the x and y directions
gradient_x = cv2.filter2D(gray_image,
        -1, prewitt_kernel_x)
gradient_y = cv2.filter2D(gray_image,
        -1, prewitt_kernel_y)

# Compute the gradient magnitude
gradient_magnitude = np.sqrt(
        np.square(gradient_x) +
        np.square(gradient_y))

# Normalize the gradient magnitude to [0, 255]
gradient_magnitude = np.uint8(
        gradient_magnitude * 255 /
        np.max(gradient_magnitude))

# Apply a threshold to obtain the binary edge map
_, edges = cv2.threshold(gradient_magnitude,
        threshold, 255, cv2.THRESH_BINARY)

return edges

# Robert's Operator
def roberts_operator(image, threshold):
# Convert the image to grayscale
gray_image = cv2.cvtColor(image,
        cv2.COLOR_BGR2GRAY)
```

```
# Define the Roberts kernels
roberts_kernel_x = np.array([[1, 0],
[0, -1]])
roberts_kernel_y = np.array([[0, 1],
[-1, 0]])

# Apply the Roberts kernels to compute
        gradients in the x and y directions
gradient_x = cv2.filter2D(gray_image,
        -1, roberts_kernel_x)
gradient_y = cv2.filter2D(gray_image,
        -1, roberts_kernel_y)

# Compute the gradient magnitude
gradient_magnitude = np.sqrt(np.square(
        gradient_x) + np.square(gradient_y))

# Normalize the gradient magnitude to [0, 255]
gradient_magnitude = np.uint8(gradient_magnitude
        * 255 / np.max(gradient_magnitude))

# Apply a threshold to obtain the binary edge map
_, edges = cv2.threshold(gradient_magnitude,
        threshold, 255, cv2.THRESH_BINARY)

return edges

# Canny Method
def canny_edge_detection(image,
        threshold1, threshold2):
# Convert the image to grayscale
```

```
    gray_image = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)

    # Apply Gaussian Blur
    blurred_image = cv2.GaussianBlur(
            gray_image, (5, 5), 0)

    # Apply Canny edge detection
    edges = cv2.Canny(gray_image,
            threshold1, threshold2)

    return edges

    # Laplacian
    def laplacian_edge_detection(image):
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)

    # Apply Laplacian edge detection
    edges = cv2.Laplacian(gray_image, cv2.CV_8U)

    return edges

    # Dilation & Erosion
    def dilation_erosion_contour_detection(
            image, threshold1, threshold2):
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)
```

```
    # Apply binary thresholding to obtain
            a binary image
    _, binary_image = cv2.threshold(gray_image,
            threshold1, threshold2,
                    cv2.THRESH_BINARY)

    # Define the structuring element (kernel)
            for morphological operations
    kernel = np.ones((3, 3), np.uint8)

    # Apply dilation to connect and enhance
            the contours
    dilated_image = cv2.dilate(binary_image,
            kernel, iterations=1)

    # Apply erosion to further refine the contours
    eroded_image = cv2.erode(dilated_image,
            kernel, iterations=1)

    return eroded_image

    # Opening & Closing
    def opening_closing_contour_detection(image):
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)

    # Apply binary thresholding to obtain
            a binary image
    _, binary_image = cv2.threshold(gray_image,
            160, 200, cv2.THRESH_BINARY)
```

```
    # Define the structuring element (kernel)
        for morphological operations
    kernel = np.ones((3, 3), np.uint8)

    # Apply opening to remove noise and separate
        connected objects
    opened_image = cv2.morphologyEx(binary_image,
        cv2.MORPH_OPEN, kernel)

    # Apply closing to close gaps between contours
        and connect objects
    closed_image = cv2.morphologyEx(opened_image,
        cv2.MORPH_CLOSE, kernel)

    # Find contours in the closed image
    contours, _ = cv2.findContours(closed_image,
        cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)

    # Draw contours on the original image
    result_image = cv2.drawContours(gray_image,
        contours, -1, (0, 255, 0), 2)

    return result_image

    # Marr-Hildreth
    def marr_hildreth_edge_detection(
        image, sigma, threshold):
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image,
```

```
            cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to the grayscale image
    blurred_image = cv2.GaussianBlur(
            gray_image, (0, 0), sigma)

    # Calculate the Laplacian of the blurred image
    laplacian = cv2.Laplacian(blurred_image,
            cv2.CV_64F)

    # Find zero crossings in the Laplacian image
    zero_crossings = np.zeros(laplacian.shape,
            dtype=np.uint8)
    zero_crossings[laplacian > 0] = 255

    # Apply thresholding to obtain binary edges
    edges = np.zeros(zero_crossings.shape,
            dtype=np.uint8)
    edges[zero_crossings > threshold] = 255

    return edges

    # Shen-Castan
    def shen_castan_edge_detection(image,
            radius, threshold):
    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image,
            cv2.COLOR_BGR2GRAY)

    # Normalize the image to [0, 1] range
    normalized_image = gray_image.astype(
```

```
        np.float32) / 255.0

    # Apply the Shen-Castan filter in multiple
        directions
    directions = [(1, 0), (0, 1), (1, 1), (1, -1)]
        # Horizontal, vertical, diagonal
        (top-left to bottom-right), diagonal
        (top-right to bottom-left)
    filtered_images = []
    for direction in directions:
    filtered_image = cv2.sepFilter2D(
        normalized_image,
        -1, np.array([1, -2, 1]),
        np.array(direction),
                borderType=cv2.BORDER_REFLECT)
    filtered_images.append(filtered_image)

    # Combine the filtered images
    edge_image = np.zeros_like(gray_image,
        dtype=np.uint8)
    for filtered_image in filtered_images:
    edge_image += np.abs(filtered_image) > threshold

    # Apply non-maximum suppression
    edges = cv2.morphologyEx(edge_image,
        cv2.MORPH_DILATE, np.ones((3, 3)))

    return edges
```

А.3 Програма реалізації навчання моделей

```
import os
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pickle

# Step 1: Prepare your dataset
train_data_dir = './images'
valid_data_dir = './test_images'
num_classes = 32
# Number of classes in your dataset

# Step 2: Load and preprocess the data
image_size = (224, 224)  # Size of input images
batch_size = 32

train_data_generator =
        keras.preprocessing.image.ImageDataGenerator(
rescale=1.0 / 255,  # Normalize pixel values
rotation_range=10,  # Randomly rotate images
width_shift_range=0.1,
# Randomly shift images horizontally
height_shift_range=0.1,
# Randomly shift images vertically
horizontal_flip=True
# Randomly flip images horizontally
)
```

```
valid_data_generator =
        keras.preprocessing.image.ImageDataGenerator(
rescale=1.0 / 255  # Normalize pixel values
)

train_generator =
        train_data_generator.flow_from_directory(
train_data_dir,
target_size=image_size,
batch_size=batch_size,
class_mode='categorical'
)

valid_generator =
        valid_data_generator.flow_from_directory(
valid_data_dir,
target_size=image_size,
batch_size=batch_size,
class_mode='categorical'
)

# Step 3: Design model architecture
model = keras.Sequential([
keras.layers.Conv2D(32, (3, 3),
        activation='relu', input_shape=(224, 224, 3)),
keras.layers.MaxPooling2D((2, 2)),
keras.layers.Conv2D(64, (3, 3),
        activation='relu'),
keras.layers.MaxPooling2D((2, 2)),
keras.layers.Conv2D(128, (3, 3),
        activation='relu'),
```

```
keras.layers.MaxPooling2D((2, 2)),
keras.layers.Flatten(),
keras.layers.Dense(128, activation='relu'),
keras.layers.Dense(num_classes,
        activation='softmax')
])

# Step 4: Define loss function and optimizer
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])

# Step 5: Train the model
epochs = 20
model.fit(
train_generator,
steps_per_epoch=len(train_generator),
epochs=epochs,
validation_data=valid_generator,
validation_steps=len(valid_generator)
)

# Step 6: Evaluate the model
test_data_dir = './test_images'
test_data_generator =
        valid_data_generator.flow_from_directory(
test_data_dir,
target_size=image_size,
batch_size=batch_size,
class_mode='categorical',
shuffle=False
```

```
)

_, test_accuracy = model.evaluate(
        test_data_generator, steps=len(
                test_data_generator))
print('Test accuracy:', test_accuracy)
```

ДОДАТОК Б ВЕЛИКІ РИСУНКИ ТА ТАБЛИЦІ

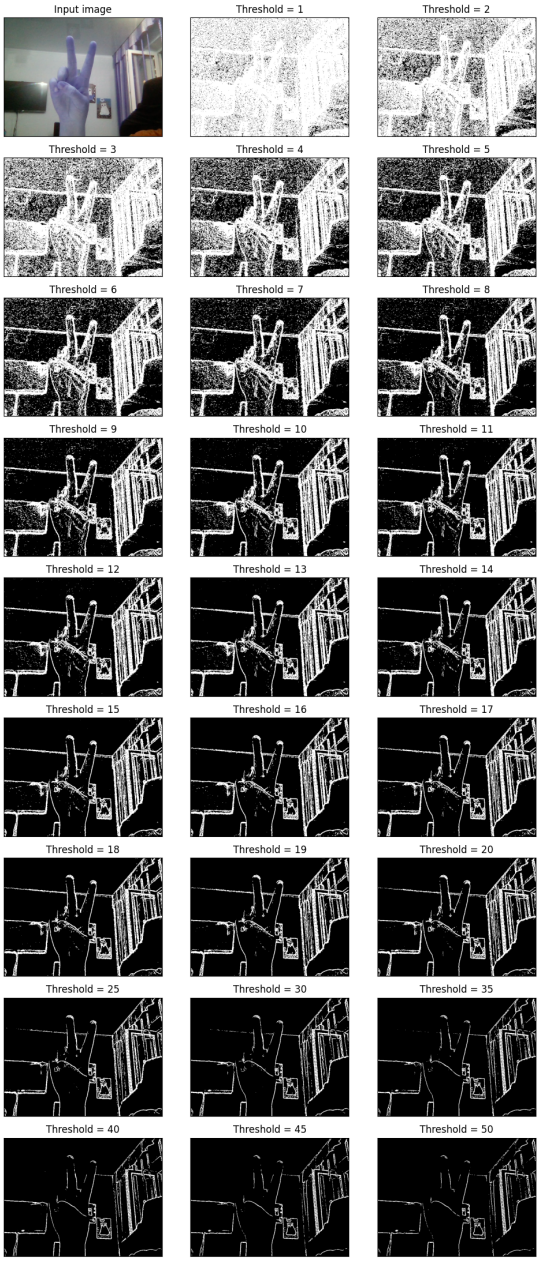


Рисунок Б.1 – Тестування оператору Собеля

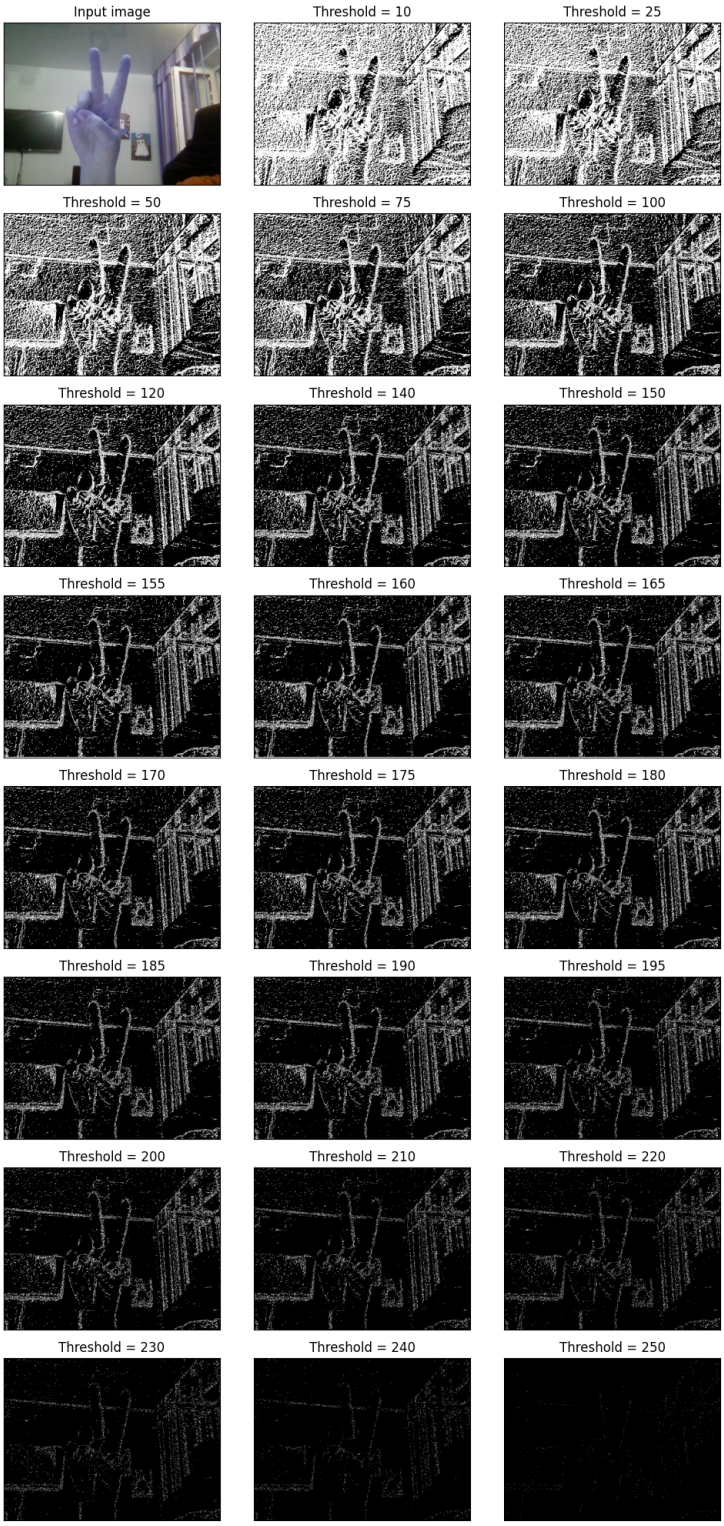


Рисунок Б.2 – Тестування оператору Прюїтта

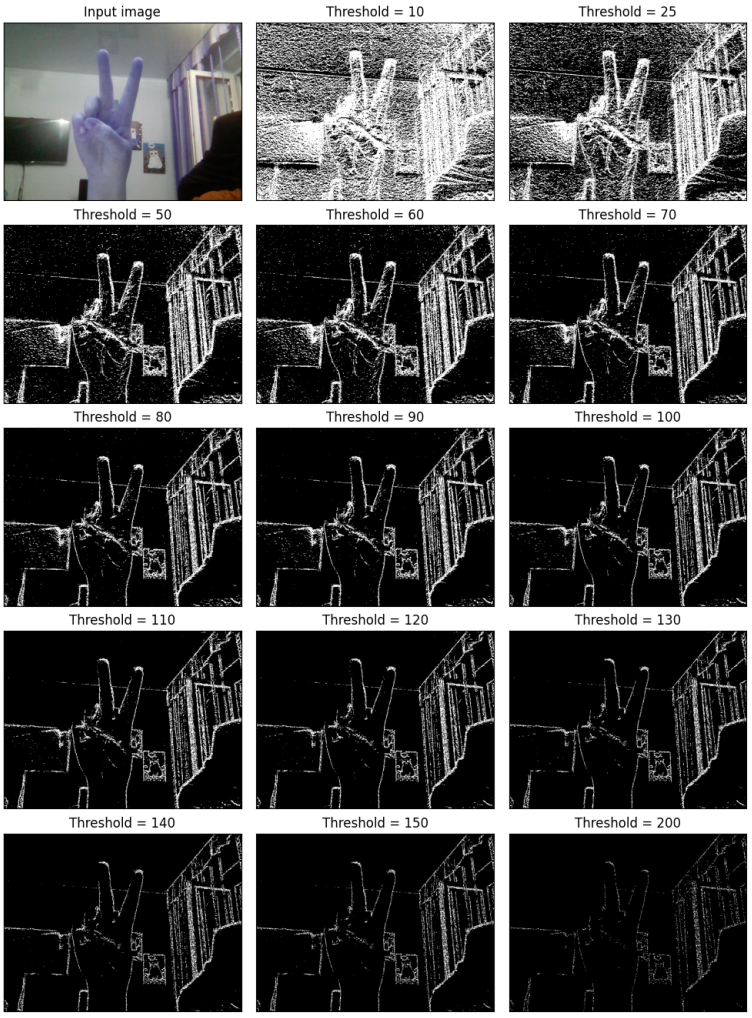


Рисунок Б.3 – Тестування оператору Робертса

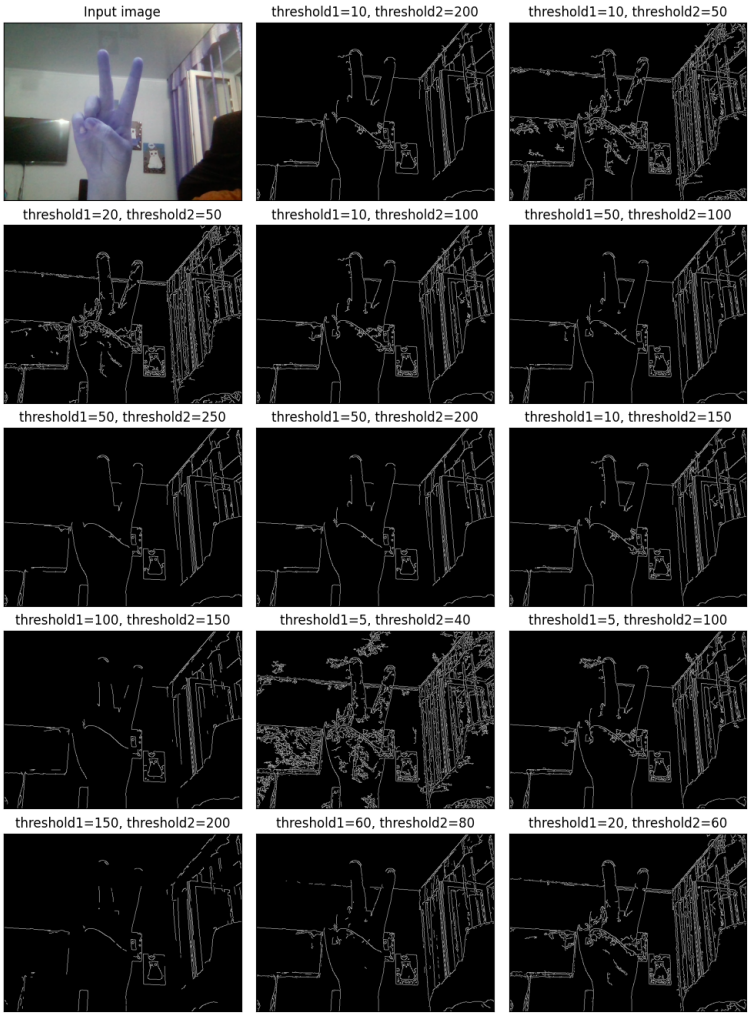


Рисунок Б.4 – Тестування методу Кенні

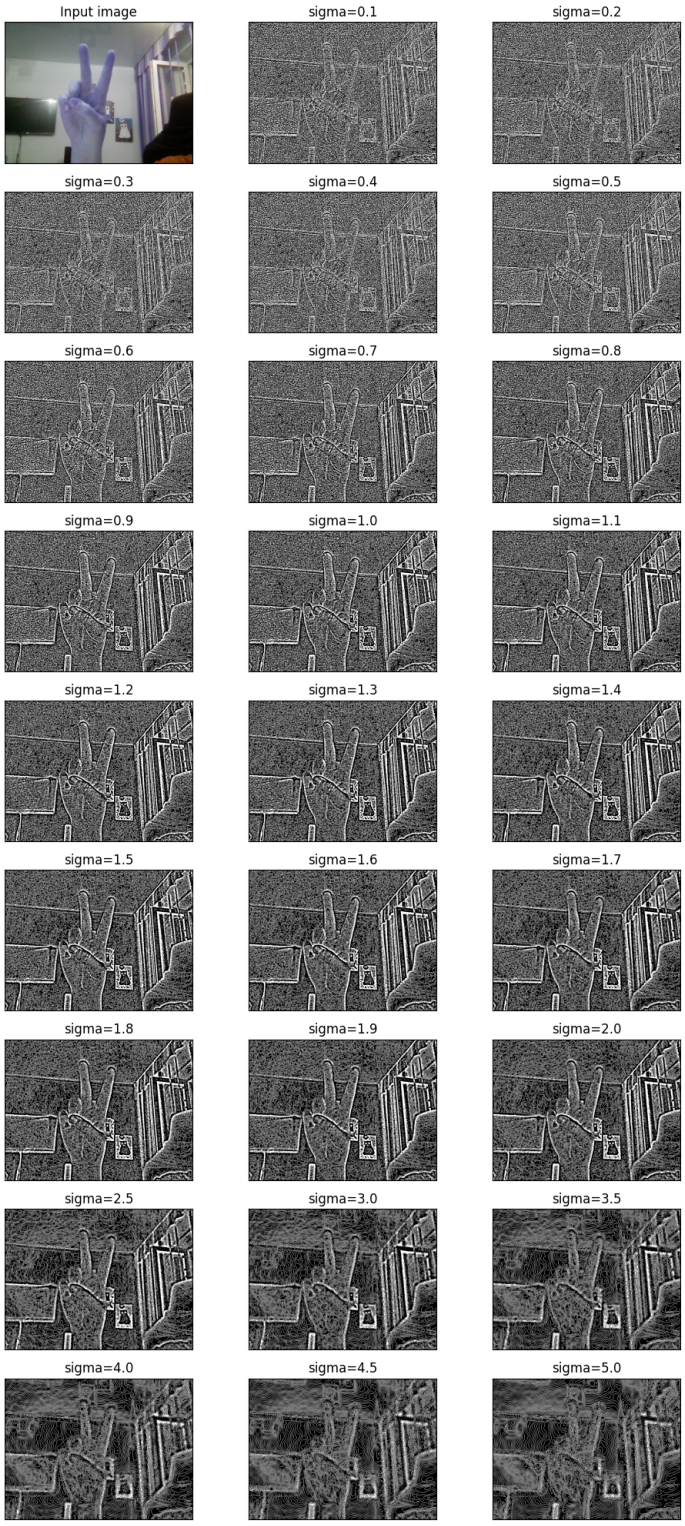


Рисунок Б.5 – Тестування методу Марра-Хілдрета

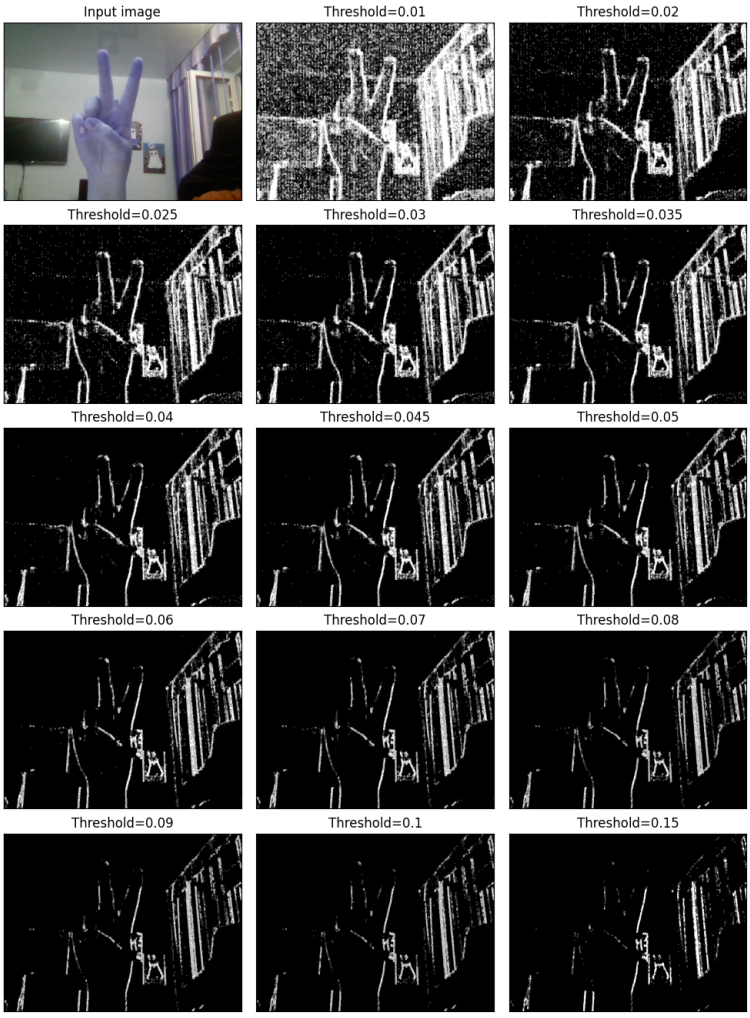


Рисунок Б.6 – Тестування методу Шен-Кастана