Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Дослідження доцільності виконання попередньої обробки зображень на системах із масовим паралелізмом
(тема)

Виконав:

студент II курсу, групи СПзм-20-1

Суліма В.М.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Торба А.А.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Сулімі Валентині Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження доцільності виконання попередньої обробки зображень на системах із масовим паралелізмом

затверджена наказом по університету від " 25 " березня 2022 р. № 33 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 р.

3. Вхідні дані до роботи
Набір вихідних зображень різного розміру, різного ступеню деталізації, різної контрастності; обчислювачі на базі ЦПУ та ГПУ.

4. Перелік питань, що потрібно опрацювати у роботі
1) огляд найуживаніших сфер застосування систем аналізу зображень;
2) вибір та обгрунтування методики та засобів дослідження;
3) програмна реалізація методів оконтурювання;
4) проведення експериментальних досліджень;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд найуживаніших сфер застосування	3.04.2022	
	систем аналізу зображень		
2	Вибір та обгрунтування методики та засобів	8.04.2022	
	проведення дослідження		
3	Програмна реалізація методів	28.04.200	
4	Проведення експериментальних досліджень	7.05.2022	
5	Оформлення матеріалів кваліфікаційної роботи	11.05.2022	
6	Подання кваліфікаційної роботи керівникові та	15.05.2022	
	опередній захист		
7	Подання кваліфікаційної роботи на рецензу-	18.05.2022	
	вання		

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Торба А.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 65 с., 35 рис., 1 дод., 26 джерел.

ЗОБРАЖЕННЯ, ПРИСКОРЕННЯ, ВИДІЛЕННЯ КОНТУРУ, ПАРАЛЕЛІЗМ, ОКОНТУРЮВАННЯ.

Метою кваліфікаційної роботи є дослідження доцільності виконання етапів обробки зображень на обчислювальних системах різних типів на прикладі задачі оконтурювання обєктів на вихідному зображенні. Для рішення поставленої мети були вирішені такі задачі: проаналізовані методи препроцесінгу у рамках аналізу проблемої області, досліджені методи контурної сегментації, запропоновано удосконалення існуючого методу оконтурювання для систем із масовим паралелізмом, виконано тестування методу та аналіз отриманих результатів.

У ході виконання кваліфікаційної роботи показано, що запропонований метод виконує більш швидку побудову зв'язного реалістичного контуру на зображеннях, в яких мало об'єктів. Збільшення кількості об'єктів на зображенні призводить до збільшення часу обчислень контуру. Для деяких зображень досягається прискорення вдвічі, в порівнянні з OpenCV реалізацією.

ABSTRACT

Master’s thesis: 65 pages, 35 figures, 1 appendices, 26 sources.

IMAGE, ACCELERATION, CONTOUR SELECTION, PARALLELISM, CONTOURING.

The purpose of the qualification work is to study the feasibility of performing stages of image processing on computer systems of different types on the example of the problem of contouring objects in the original image. To solve this goal, the following tasks were solved: analyzed preprocessing methods in the analysis of the problem area, studied methods of contour segmentation, proposed improvement of the existing contouring method for systems with mass parallelism, tested the method and analyzed the results.

In the course of qualifying work it is shown that the proposed method performs a faster construction of a coherent realistic contour on images in which there are few objects. Increasing the number of objects in the image increases the time of contour calculations. For some images, the acceleration is doubled compared to the OpenCV implementation.

ЗМІСТ

ВСТУП

Зображення краще ніж будь-яка інша форма інформації для сприйняття людиною. Зір дозволяє людям сприймати і розуміти навколишній світ. Розуміння зображень, їх аналіз та комп’ютерний зір мають на меті дублювати ефект людського зору за допомогою електронного (цифрового, у теперішньому контексті) сприйняття та розуміння зображень. Сфера обробки зображень постійно розвивається. Протягом останніх п’яти років відбулося значне зростання інтересу до морфології зображень, нейронних мереж, обробки повно-кольорових зображень, стиснення даних зображень, розпізнавання зображень та систем аналізу зображень.

Методи обробки зображень випливають з двох основних сфер застосування: покращення графічної інформації для інтерпретації людиною та обробка даних сцени для автономного машинного сприйняття. У системі цифрової обробки зображень першим кроком є отримання зображення. Після отримання цифрового зображення наступний крок стосується попередньої обробки, її функція полягає в покращенні зображення. Наступний крок стосується сегментації, це розділення вхідного зображення на складові частини або об'єкти. Функція представлення та опису займається створенням даних у формі, придатній для комп'ютерної обробки, щоб обробити зображення, це означає застосувати якусь операцію над зображенням. В основному існують засоби покращення зображення, які покращують зовнішній вигляд, та засоби відновлення зображення, для відновлення зображення в такому вигляді, яким воно є після застосування будь-якої операції, наприклад стиснення зображення для зменшення обсягу даних зображення для зменшення його розміру. Цифрова обробка зображень може здатися складною темою для багатьох людей, але насправді існує лише кілька принципів, які потрібно знати, щоб використовувати більшість графічних програм. Сучасні цифрові технології зробили можливим

маніпулювати багатовимірними сигналами за допомогою систем, які варіюються від простих цифрових схем до передових паралельних комп’ютерів.

Обробка зображень здебільшого стикається з проблемою великих накладних витрат на обробку цього зображення, що є великою перешкодою для програм реального часу. Щоб вирішити цю проблему, потрібна високопродуктивна обчислювальна платформа. Використання апаратного прискорювача зменшує час обробки. В останніх розробках графічний процесор використовується в багатьох програмах. Цей графічний процесор (GPU) відіграє важливу роль у сучасних обчислювальних машинах. За останні роки відбулося помітне зростання продуктивності та можливостей графічних процесорів. Сучасний графічний процесор – це не тільки потужний графічний движок, але й високопаралельний програмований процесор, який має максимальну арифметику та пропускну здатність пам’яті, що значно випереджає його аналог – ЦП (CPU). Швидке зростання можливостей програмного забезпечення та можливостей графічного процесора породило дослідницьку спільноту, яка успішно відобразила широкий спектр вимогливих до обчислень, складних проблем на GPU. Ці обчислення загального призначення на GPU, позиціонували GPU як переконливу альтернативу традиційним мікропроцесорам у високопродуктивних комп’ютерних системах майбутнього.

Метою даної роботи є аналіз доцільності використання обчислювальної продуктивності трьох основних обчислювальних платформах: CPU, GPU та FPGA для рішення задач попередньої обробки зображень.

1 ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд існуючих аналогів

Актуальність обраної теми пояснюється поширенням застосунків інтелектуального аналізу зображень, які використовують методи попередньої обробки зображень [1-4] на етапі підготовки даних для подачі на інтелектуальний модуль [5-7].

Серед прикладних областей, які потребують застосуванні віддаленого автоматизованого аналізу інформації, отриманої із зображень, можна виділити (рисунок 1.1) транспортні системи, mo-cap технології, системи аутентифікації та розпізнавання об’єктів. Всі представлені технології інтелектуальних систем, заснованих на обробці зображень, націлені на те, щоб зробити життя людей безпечнішим, простішим і здійснювати ті процедури, які раніше без цих технологій виконати було неможливо.

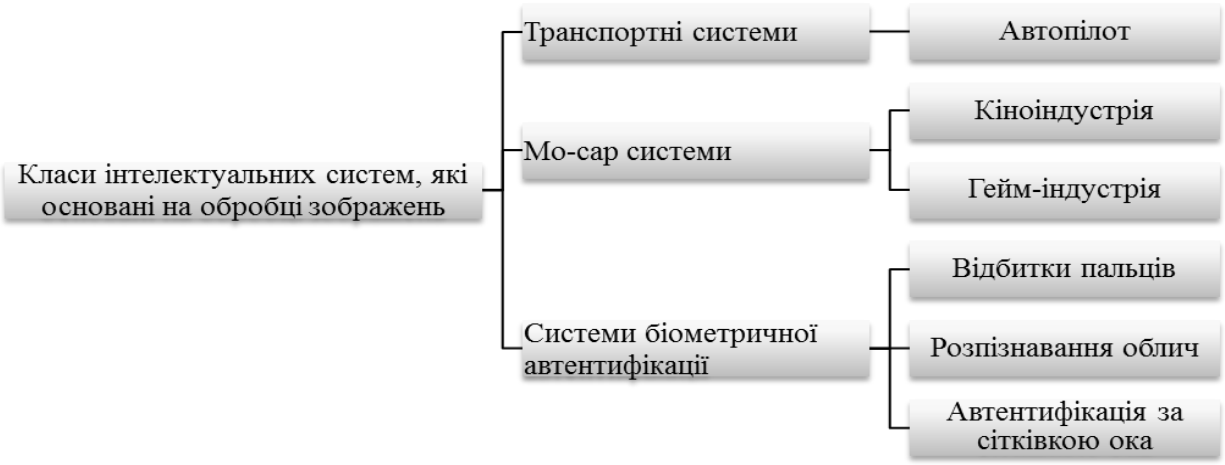


Рисунок 1.1 – Сфери застосування візуального інтелекту

1.1.1 Технологія інтелектуальної системи, що базується на обробці довкілля «Автопілот»

Технологія інтелектуальної системи, що базується на обробці довкілля

«Автопілот» може викликати велику соціальну революцію, оскільки здатні вирішити всілякі проблеми, такі як затримки руху, аварії, спричинені помилкою водія, попередження перед зіткненням, допомога в рульовому управлінні та автоматичне гальмування. Але це не зупиняється на досягнутому: автономні транспортні засоби виведуть на ринок безліч нових та цікавих додатків для різних галузей, таких як судноплавство, транспортування та екстрена допомога (рисунок 1.2).

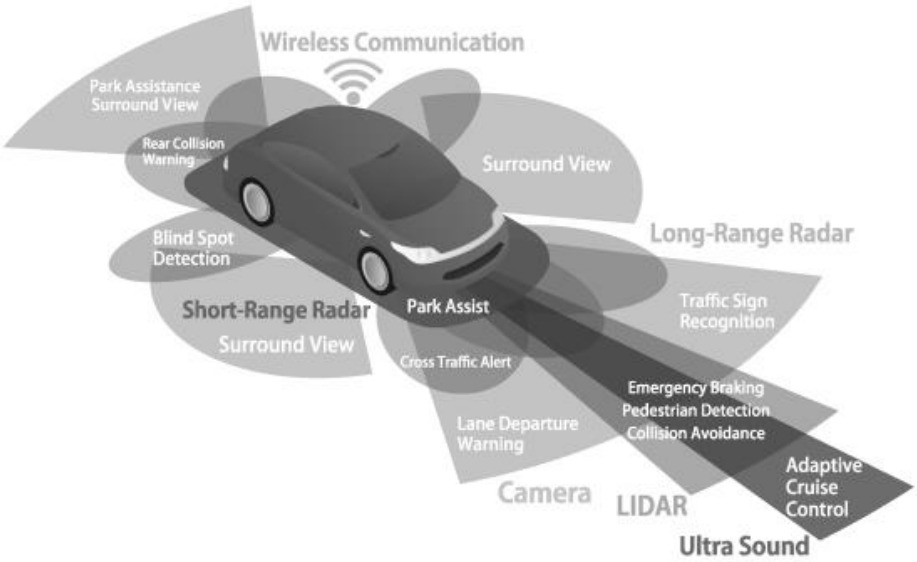


Рисунок 1.2 – Дані, що забезпечують роботу системи автоматичного керування

Вивести технології автоматичного керуванням на новий рівень, повністю усуваючи потребу у водії, допомагають радари, камери, ультразвук та радіоантени, які використовуються для фіксації інформації та подальшого аналізу.

Електромобілі Tesla виділяються серед інших, крім іншого, системою просунутої системи допомоги водієві (ADAS) Tesla Autopilot, доданої у вигляді програмного розширення.

Аппаратна частина Autopilot, крім комп'ютера FSD, що відповідає за обробку всіх даних, включає вісім камер, які створюють панорамну картинку навколо автомобіля, 12 ультразвукових датчиків, що обчислюють відстань до

перешкоди, та радар, встановлений у носовій частині.

Але на відміну від Waymo та інших розробників технологій автономного водіння, які покладаються на лідарах (лазерні 3D-сканери, що дозволяють виявляти предмети та визначати відстань до них) та HD-картографію, Tesla спирається на комп'ютерний (машинний) зір (рисунок 1.3).

Рисунок 1.3 – Програмне розширення для Tesla Autopilot

Переваги: систему легше масштабувати на великий парк автомобілів, тоді як картографічний підхід, що вимагає постійного відстеження найменших змін, пов'язаних із великими витратами і є «немасштабованим».

1.1.2 Технологія захвату руху Motion Caption

Технологія захвату руху Motion Caption або mo-cap, є технологія поєднує в собі реальне життя і анімацію, дозволяючи знімати когось наживо і переводити їх у комп'ютерну форму.

Захоплення руху - це передовий метод захоплення всієї гри актора або її частини, щоб її можна було перетворити на дії комп'ютерного 3D-персонажу на екрані. Оскільки точні рухи актора фіксуються та відображаються на цифровому персонажі, корисно думати про це як про

цифровий грим всього тіла; спільний процес, що дозволяє кінематографістам наповнювати персонажів неймовірним рівнем тонкості та натуралізму, не ставлячи при цьому жодних обмежень на творчий підхід до зовнішнього вигляду. Цей процес дуже успішно використовувався при спільній роботі над екстраординарними виступами у таких фільмах, як «Аватар», «Володар кілець» та «Війна світів Z».

Нещодавно технологія захоплення руху використовувалася у поєднанні з віртуальним виробництвом у таких фільмах, як "Книга джунглів" та "Першому гравцю приготуватися". Це нове використання технології зробило її популярнішою, ніж будь-коли.

Відстеження руху та захоплення мають широкий спектр застосування у різних галузях, у тому числі:

- кіно та ігри. Захоплення руху використовується для запису рухів акторів та перенесення їх на віртуальних чи доповнених комп'ютером персонажів;
- спортивна терапія та охорона здоров'я. Медичні працівники можуть використовувати захоплення рухів для аналізу рухів пацієнтів та діагностики проблем, наприклад, аналізу ходи;
- військові. У поєднанні з віртуальною реальністю технології захоплення руху використовуються підвищення ефективності військової підготовки.

1.1.3 Технологія біометрична автентифікація та ідентифікація

Біометрична автентифікація (БА) порівнює дані про характеристики людини з біометричним «шаблоном» цієї людини, щоб визначити схожість (фотографія обличчя, запис їх голосу або зображення відбитків пальців).

Зіткнувшись із підробкою документів та крадіжкою особистих даних, тероризмом та кіберзлочинністю, змінами в міжнародному законодавстві, впроваджуються нові рішення для біометричної безпеки.

Біометрію можна визначити як найпрактичніший засіб ідентифікації та аутентифікації людей надійним та швидким способом за допомогою унікальних біологічних характеристик.

Звичайно, ширше суспільне визнання, значний приріст точності, багата пропозиція та зниження цін на датчики, IP-камери та програмне забезпечення спрощують встановлення біометричних систем та підвищують їх актуальність.

Біометричні ідентифікатори ділять на морфологічні, поведінков і біологічні:

- морфологічні ідентифікатори в основному складаються з відбитків пальців, форми руки, пальця, малюнка вен, ока (райдужної оболонки та сітківки) та форми обличчя;

- для біологічних аналізів медичні бригади та поліцейські криміналісти можуть використовувати ДНК, кров, слину;

- поведінкові ідентифікатори основані на розпізнаванні голосу, динаміки підпису (швидкість руху пера, прискорення, тиск, нахил), динаміки натискання клавіш, як ми використовуємо предмети, ходи, звуку кроків, жестів та ін.

Різні методи, що використовуються, є предметом постійних досліджень і розробок і постійно вдосконалюються (рисунок 1.4).

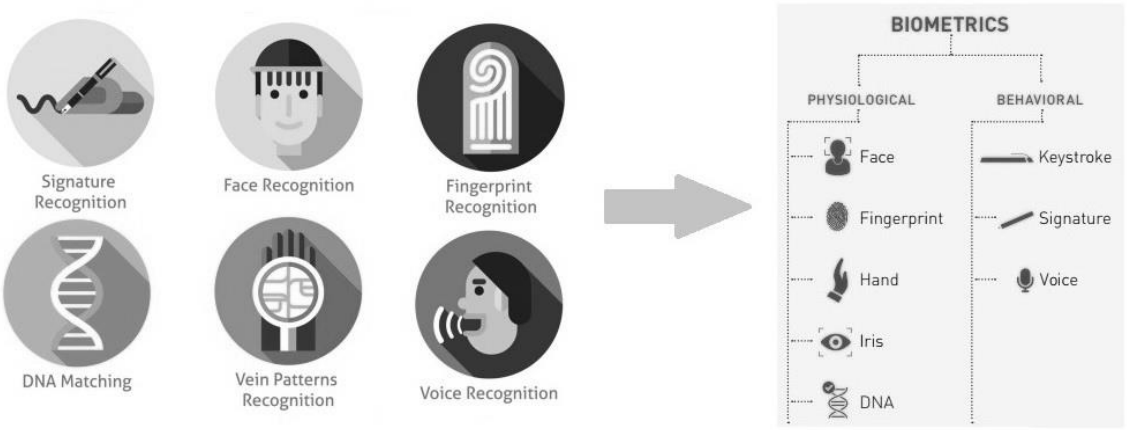


Рисунок 1.4 – Біометричні ідентифікатор. Класифікація

Фізіологічні виміри зазвичай дають перевагу в тому, що вони

залишаються стабільнішими протягом усього життя людини.

Найзначнішою перевагою біометричної автентифікації є рівень безпеки та точності, який гарантує цей підхід. На відміну від паролів, бейджів чи документів, біометричні дані неможливо забути, обміняти, вкрасти чи підробити.

Серед секторів, які демонструють реальний інтерес до переваг біометрії , можна виділити:

- правоохоронні органи та громадська безпека (ідентифікація злочинців/підозрюваних);
- військові (ідентифікація ворога/союзника);
- прикордонний, виїзний та міграційний контроль (ідентифікація мандрівника/мігранта/пасажира);
- громадянська ідентифікація (ідентифікація громадянина/резидента/виборця);
- охорона здоров'я та субсидії (ідентифікація пацієнта/бенефіціара/медичного працівника);
- фізичний та логічний доступ (ідентифікація власника/користувача/співробітника/підрядника/партнера);
- комерційні програми (ідентифікація споживача/клієнта).

Технічні проблеми автоматичного розпізнавання людей на основі їх біологічних та поведінкових характеристик пов'язані з перетворенням аналогової (зображення особи, відбиток пальця, зразок голосу) у цифрову інформацію (образи, дрібні деталі), яку потім можна обробляти, порівнювати та зіставляти за допомогою ефективних алгоритмів.

1.2 Обґрунтування доцільності аналізу платформ для прискореного рішення задач попередньої обробки зображень

Серед обговорюваних тут обчислювальних платформ, центральні процесори, з більш ніж півстолітньою історією, є найбільш відомими та

розповсюджуваними [8-9]. Архітектуру ЦП іноді називають скалярною архітектурою, оскільки вона розроблена для ефективної обробки послідовних інструкцій. За допомогою багатьох доступних методів процесори оптимізовані для збільшення паралелізму на рівні інструкцій, щоб послідовні програми могли виконуватися якнайшвидше.

Скалярне конвеєрне ядро ЦП може виконувати інструкції, поділені на етапи, зі швидкістю до однієї інструкції за такт, якщо немає залежностей (рисунок 1.5, 1.6). Щоб підвищити продуктивність, ядра сучасних процесорів є багатопоточними суперскалярними процесорами зі складними механізмами, які використовуються для пошуку паралельності на рівні інструкцій і виконання кількох інструкцій, що не в порядку, за такт. Вони отримують багато інструкцій одночасно, знаходять графік залежності цих інструкцій, використовують складні механізми передбачення розгалужень і виконують ці інструкції паралельно [10-12].

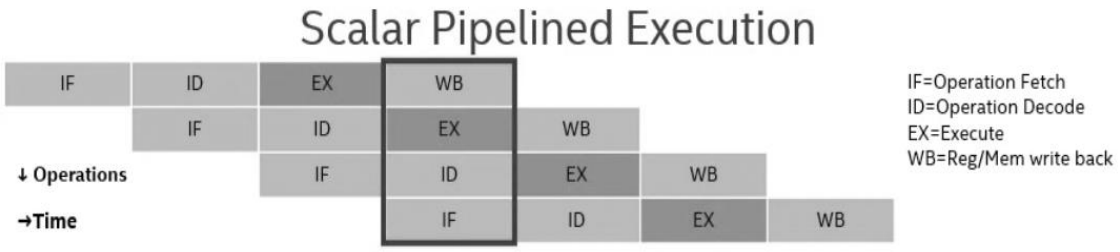


Рисунок 1.5 - Скалярне конвеєрне виконання

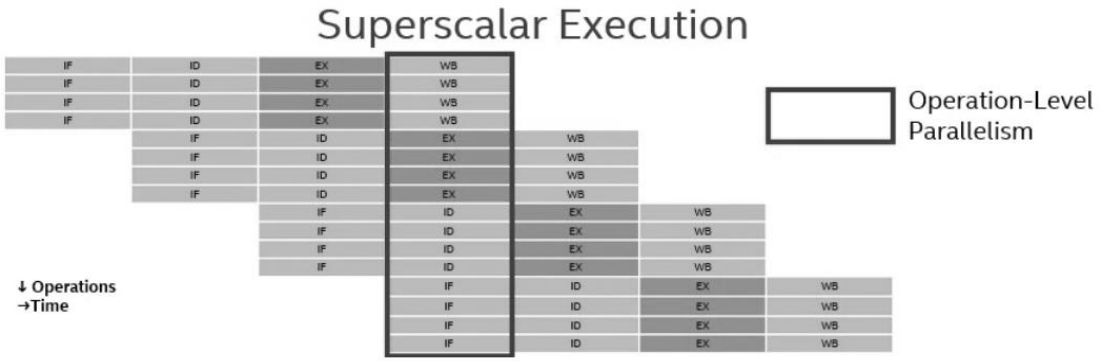


Рисунок 1.6 - Суперскалярне виконання

Використання ЦП для обчислень має багато переваг у порівнянні з розвантаженням на співпроцесор, такий як графічний процесор (GPU) або FPGA. По-перше, оскільки дані не потрібно вивантажувати, затримка зменшується з мінімальними витратами на передачу даних. Оскільки високочастотні ЦП налаштовані для оптимізації скалярного виконання, а більшість програмних алгоритмів мають послідовний характер, високої продуктивності можна легко досягти на сучасних ЦП. Для частин алгоритму, які можуть бути векторно-паралелізовані, сучасні ЦП підтримують інструкції з однією командою, кількома даними (SIMD), як-от Intel® Advanced Vector Extensions 512. В результаті ЦП підходять для широкого спектру робочих навантажень. Навіть для дуже паралельних робочих навантажень ЦП можуть перевершити прискорювачі для алгоритмів з великою розбіжністю гілок або великою паралельністю на рівні команд, особливо там, де велике відношення розміру даних і обчислень.

Переваги центрального процесора:

- автоматичний паралелізм у послідовному коді;
- велика кількість підтримуваних інструкцій;
- нижча затримка в порівнянні з прискоренням за рахунок GPU;
- послідовне виконання коду спрощує розробку;
- невпорядковане супер скалярне виконання;
- точне прогнозування гілок;
- удосконалене управління для отримання неймовірного рівня паралелізму інструкцій.

Графічні процесори — це процесори, що складаються з масивно паралельних, менших і більш спеціалізованих ядер, ніж ті, які зазвичай зустрічаються у високопродуктивних ЦП [13,15]. Архітектура графічного процесора оптимізована для сукупної пропускної здатності всіх ядер, не підкреслюючи затримки та продуктивність окремих потоків. Архітектура графічного процесора ефективно обробляє векторні дані (масив чисел) і її часто називають векторною архітектурою. Графічні процесори виділяють

більше пам’яті для обчислень та менше для кешування та управління. В результаті апаратне забезпечення графічного процесора досліджує менший паралелізм на рівні інструкцій і покладається на паралелізм, заданий програмним забезпеченням, для досягнення продуктивності та ефективності.

Графічні процесори — це стандартні процесори і не підтримують складне передбачення розгалужень. Натомість у них є безліч арифметичних логічних блоків (АЛУ) і глибоких конвеєрів. Продуктивність досягається за рахунок багатопотокового виконання великих і незалежних даних, що амортизує витрати на простіший контроль і меншу кількість кеш-пам'яті [14,16]. Графічні процесори використовують модель виконання однієї інструкції з кількома потоками (single instruction, multiple threads - SIMT), де багатопоточність і SIMD використовуються разом. У моделі SIMT кілька потоків (робочі елементи або послідовність операцій на лінії SIMD) обробляються послідовно в одному потоці інструкцій SIMD. Кілька потоків інструкцій SIMD відображаються на один виконавчий блок (EU), і GPU EU може перемикати контекст між цими потоками команд SIMD, коли один потік зупиняється.

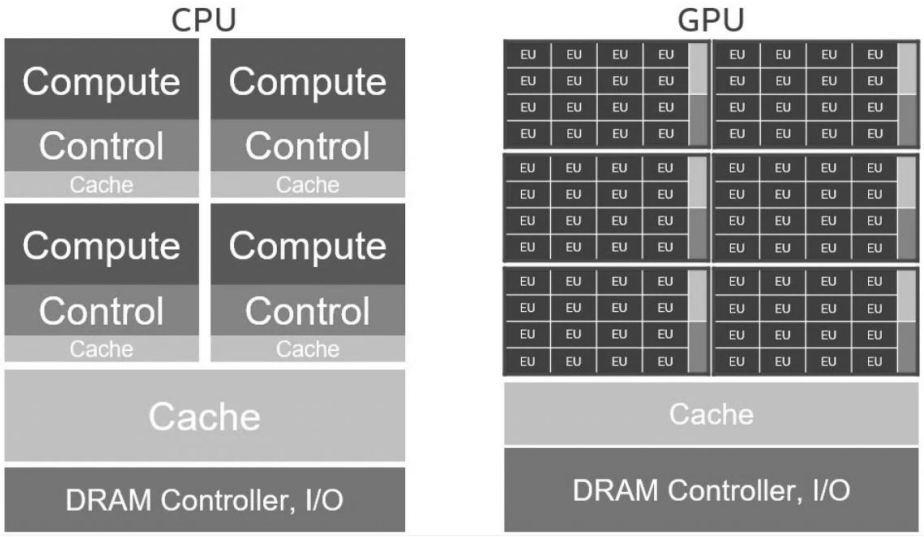


Рисунок 1.7 – Архітектурні відмінності CPU та GPU

На діаграмі вище показано різницю між CPU та GPU (рисунок 1.7). EUs

є основною одиницею обробки на GPU. Кожен EU може обробляти кілька потоків інструкцій SIMD. У тому ж просторі пам’яті графічні процесори мають більше ядер/EU, ніж ЦП. Графічні процесори організовані ієрархічно. Кілька EU об’єднуються, щоб утворити обчислювальний блок із спільною локальною пам’яттю та механізмами синхронізації (він же субчастковий або потоковий мультипроцесор, виділений фіолетовим кольором). Обчислювальні одиниці об’єднуються, утворюючи графічний процесор.

Переваги графічного процесора:

- величезна паралельність, до тисяч малих і ефективних ядер SIMD/EU;
- ефективне виконання паралельного з даними коду;
- висока пропускна здатність динамічної пам’яті з довільним доступом (DRAM).

На відміну від ЦП і графічних процесорів, які є програмно-програмованими фіксованими архітектурами, FPGA можна переналаштувати, а їх обчислювальні механізми визначаються користувачем. Під час написання програмного забезпечення, націленого на FPGA, скомпільовані інструкції стають компонентами апаратного забезпечення, які розміщуються на матриці FPGA в просторі, і всі ці компоненти можуть виконуватися паралельно. Через це архітектуру FPGA іноді називають просторовою архітектурою.

FPGA — це величезний набір невеликих процесорів, що складається з мільйонів програмованих 1-бітових адаптивних логічних модулів (кожен може функціонувати як однорозрядний ALU), до десятків тисяч конфігурованих блоків пам'яті та десятків тисяч математичних механізмів, відомих як блоки цифрової обробки сигналів (DSP), які підтримують операції з плаваючою та фіксованою комою змінної точності (рисунок 1.8). Усі ці ресурси з’єднані сіткою програмованих провідників, які можна активувати за потребою.

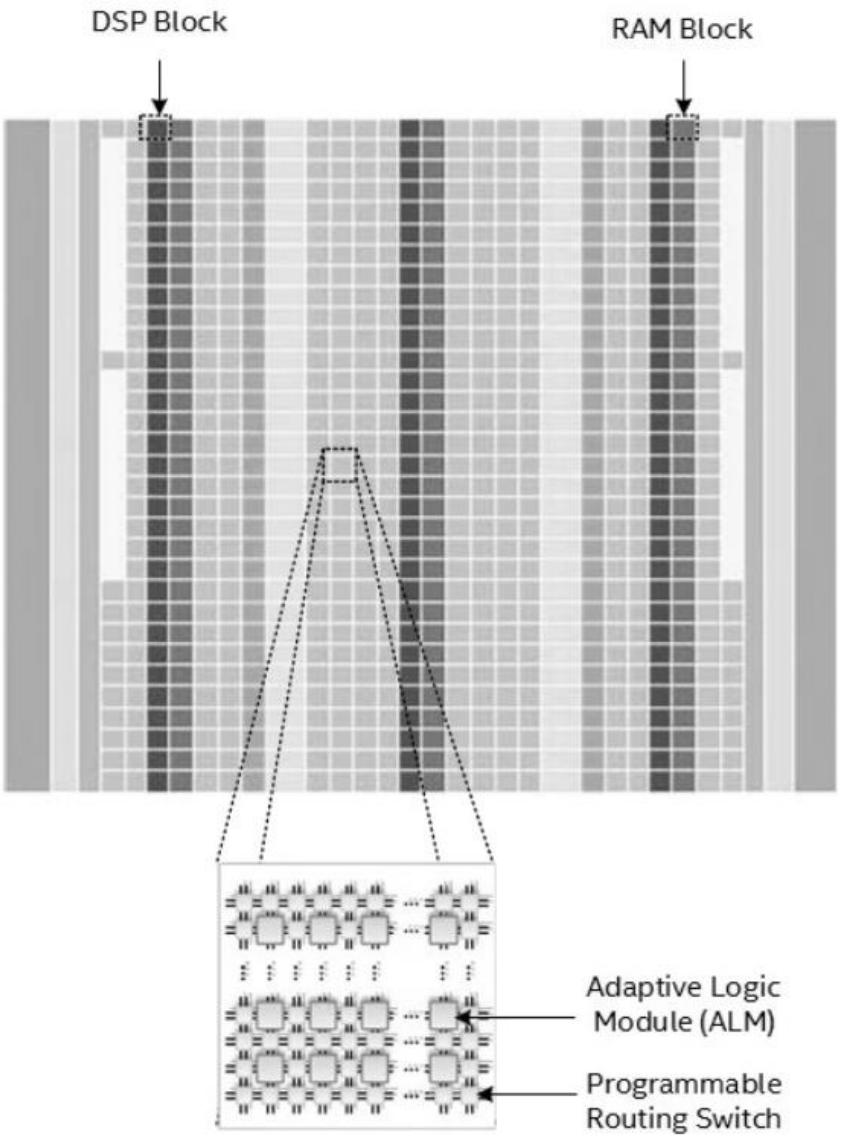


Рисунок 1.8 – Архітектурні особливості FPGA

Коли програмне забезпечення «виконується» на FPGA, воно не виконується в тому ж сенсі, що скомпільовані та зібрані інструкції виконуються на CPU та GPU. Натомість потік даних здійснюється через налаштовані глибокі конвеєри на FPGA, які відповідають операціям, вираженим у програмному забезпеченні. Оскільки апаратне забезпечення конвеєра потоку даних відповідає програмному забезпеченню, накладні витрати на керування усуваються, що призводить до покращення продуктивності та ефективності. З процесорами та графічними процесорами етапи інструкцій конвейеруються, і нова інструкція починає виконувати

кожен такт. У FPGA операції конвеєрні, тому нові потоки інструкцій, що працюють з різними даними, починають виконувати кожен такт (рисунок 1.9, 1.10).

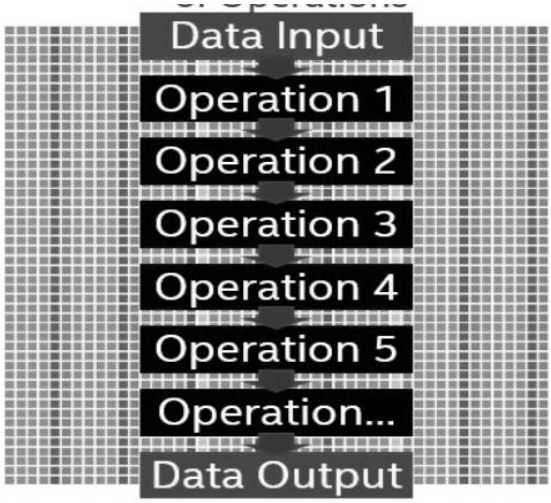


Рисунок 1.9 – Послідовність виконання операцій на FPGA

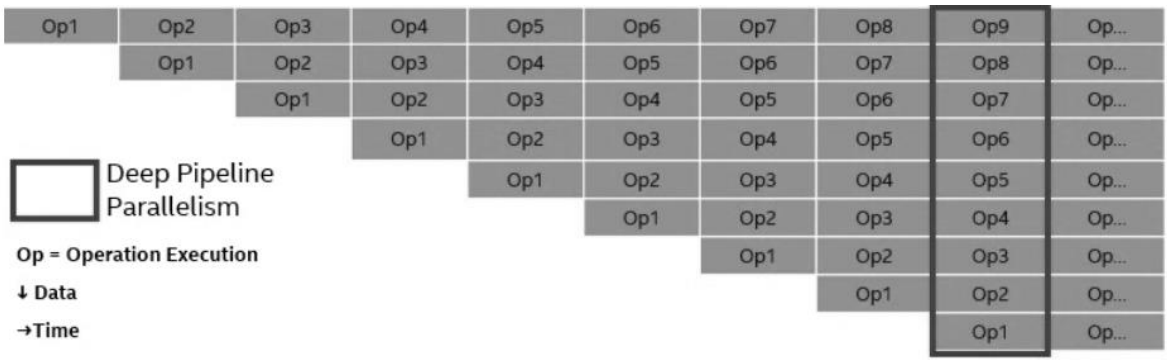


Рисунок 1.10 – Паралельність конвеєру FPGA

Хоча паралельність конвеєра є основною формою паралелізму для FPGA, його можна поєднувати з іншими типами паралельності. Наприклад, паралелізм даних (SIMD), паралельність завдань (кілька конвеєрів) і суперскалярне виконання (кілька незалежних інструкцій, що виконуються паралельно) можна використовувати з паралельним конвеєром для досягнення максимальної продуктивності.

Переваги FPGA:

- ефективність: конвеєр обробки даних точно налаштований на

потреби програмного забезпечення. Немає потреби в блоках керування, блоках вибору інструкцій, зворотному запису в регістр та інших накладних витратах на виконання;

- індивідуальні інструкції: інструкції, які не підтримуються CPU/GPU, можуть бути легко реалізовані та ефективно виконані на FPGA (наприклад, маніпуляції з бітами);
- залежності даних у паралельній роботі можна вирішити без зупинок конвеєра;
- гнучкість: FPGA можна переналаштувати для розміщення різних функцій і типів даних, включаючи нестандартні типи даних;
- спеціальна топологія вбудованої пам’яті, налаштована на алгоритм: вбудована пам’ять із великою пропускною здатністю, створена для адаптації шаблону доступу для мінімізації або усунення затримок;
- багатий ввід-вивід: ядро FPGA може безпосередньо взаємодіяти з різними мережами, пам’яттю, користувацькими інтерфейсами та протоколами, що призводить до рішень з низькою та детермінованою затримкою.

1.3 Постановка задачі

Метою кваліфікаційної роботи є дослідження доцільності виконання етапів обробки зображень на обчислювальних системах різних типів на прикладі задачі оконтурювання обєктів на вихідному зображенні.

Для рішення поставленої мети мають бути вирішені такі задачі:

- аналіз методів препроцесінгу у рамках аналізу проблемої області;
- дослідження методів контурної сегментації;
- удосконалення існуючого методу оконтурювання для систем із масовим паралелізмом;
- тестування методу та аналіз отриманих результатів.

2 АНАЛІЗ ІСНУЮЧИХ ТЕХНОЛОГІЙ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Сучасні обчислювальні системи є гетерогенними і включають центральний процесор, графічні процесори, FPGA та інші прискорювачі. Різні архітектури демонструють різноманітні характеристики, які можна підібрати до конкретних робочих навантажень для найкращої продуктивності. Наявність кількох типів обчислювальних архітектур призводить до різних потреб у програмуванні та оптимізації для кожної архітектури.

Архітектура графічного процесора є найбільш щільною для обчислень. Якщо обчислювальне ядро просте і вимагає великої кількості обчислень, воно, швидше за все, найкраще працюватиме на графічному процесорі. Архітектура FPGA є найбільш ефективною для обчислень. Хоча FPGA і створені як конвеєри обчислень, їх можна використовувати для прискорення практично будь-якого ядра, просторова природа реалізації FPGA означає, що доступні ресурси FPGA можуть бути обмеженням. Нарешті, архітектура ЦП є найгнучкішою з найширшою підтримкою інструкцій. Сучасні процесори підтримують багато типів паралелізму і можуть ефективно використовуватися, як доповнення до інших прискорювачів.

2.1 Визначення апаратної бази для експериментальної частини

Продуктивність реалізації задачі попередньої обробки зображень великих розмірів залежить від апаратної бази та алгоритму, що використовується.

Для прискореної обробки на центральному процесорі (ЦП) та для досягнення високої продуктивності, можуть бути використані декілька видів паралельності: паралелізм даних SIMD, паралелізм на рівні інструкцій та

паралельність на рівні потоків із кількома потоками, що виконуються на різних логічних ядрах. Ці типи паралелізмів можна досягти такими способами:

Паралелізм даних SIMD характеризується тим, що:

- кожен робочий елемент (одної і тієї ж робочої групи) може зіставлятися зі смугою SIMD ЦП. Робочі елементи (підгрупи) виконуються разом у SIMD;
- векторні типи даних можна використовувати для явного визначення операцій SIMD;
- компілятор може виконувати циклізацію векторизації для створення SIMD-коду. Одна ітерація циклу відображається на лінії SIMD CPU. Кілька ітерацій циклу виконуються разом у SIMD.

Паралелізм ядра ЦП (рисунок 2.1) і гіперпотоку характеризується тим, що різні робочі групи можуть виконуватися на різних логічних ядрах паралельно. Машина з 16 ядрами і 32 гіперпотоками може виконувати 32 робочі групи паралельно.

Традиційний паралелізм на рівні інструкцій досягається завдяки складному управлінню ЦП, як традиційне послідовне виконання програмного забезпечення.

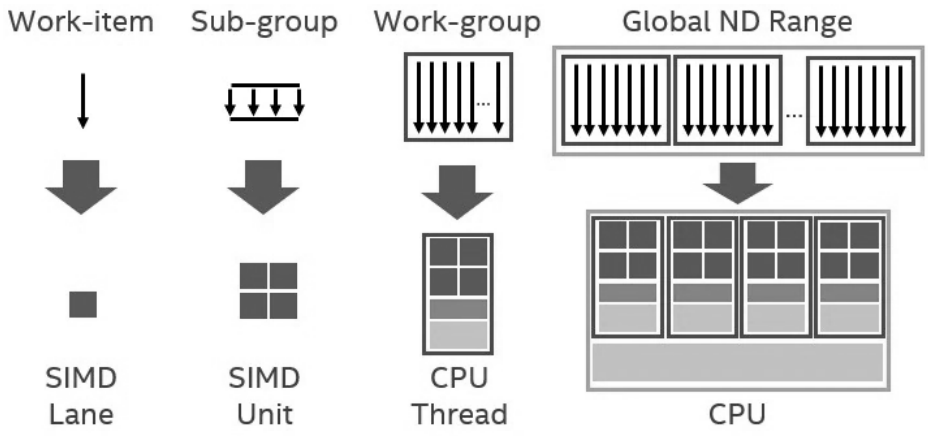


Рисунок 2.1 – Рівні паралелізму ЦП

Для прискореної обробки на GPU та для досягнення високої

продуктивності, графічні процесори покладаються на паралельні робочі навантаження великих обсягів даних. В результаті ядра з одним завданням використовуються рідко, а ядра NDRange необхідні для повного заповнення конвеєра глибокого виконання GPU (рисунок 2.2). Під час виконання ядра на графічному процесорі кожен робочий елемент відображається на SIMD. Підгрупа формується з робочих елементів, які виконуються в стилі SIMD, а підгрупи відображаються на GPU EU. Робочі групи, які включають робочі елементи, які можуть синхронізувати та обмінюватися локальними даними, призначаються для виконання на обчислювальних блоках (він же потокові мультипроцесори або суб-фрагменти). Нарешті, весь глобальний діапазон робочих елементів NDR відображається на весь графічний процесор.

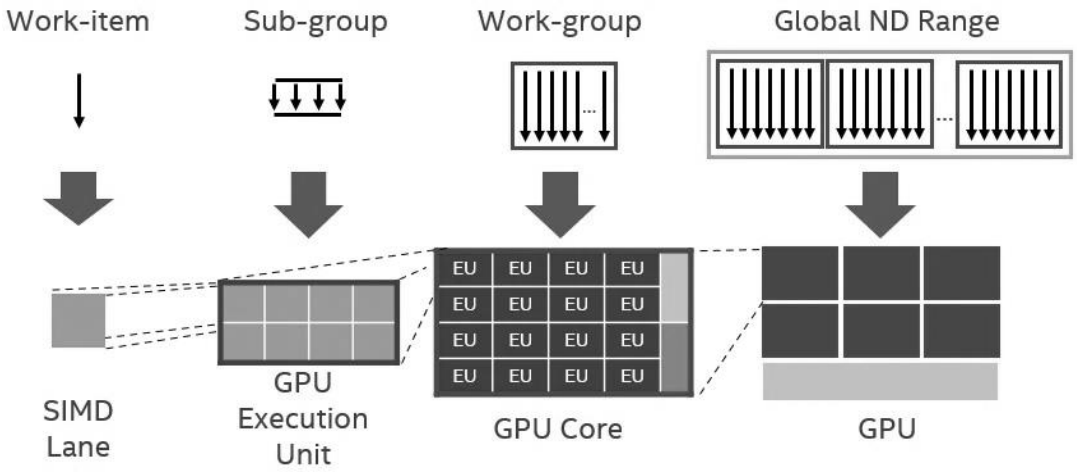


Рисунок 2.2 – Рівні паралелізму ГПУ

Коли ядра компілюються для FPGA, операції в ядрі розміщуються просторово. Залежні операції ядра глибоко конвеєрні, тоді як незалежні операції виконуються паралельно. Важливою перевагою конвеєрної реалізації є те, що залежності між сусідніми робочими елементами можна вирішити, просто перенаправивши дані з одного етапу конвеєра назад на попередній, без зупинки конвеєра. Ключ до продуктивності полягає в тому, щоб глибокий конвеєр був повністю зайнятим. Хоча продуктивність може бути отримана на FPGA або за допомогою ядер NDRange, або ядра одного завдання, ядра, які сприятливо працюють на FPGA, часто виражаються як

ядра одного завдання.

З однозадачними ядрами FPGA намагається виконати конвеєрний цикл. Кожен такт, послідовні ітерації циклу входять у перший етап конвеєра. Залежності між ітераціями циклу, виражені в програмному забезпеченні, можуть бути вирішені в апаратному забезпеченні шляхом перенаправлення виводу одного етапу на вхід більш раннього залежного етапу.

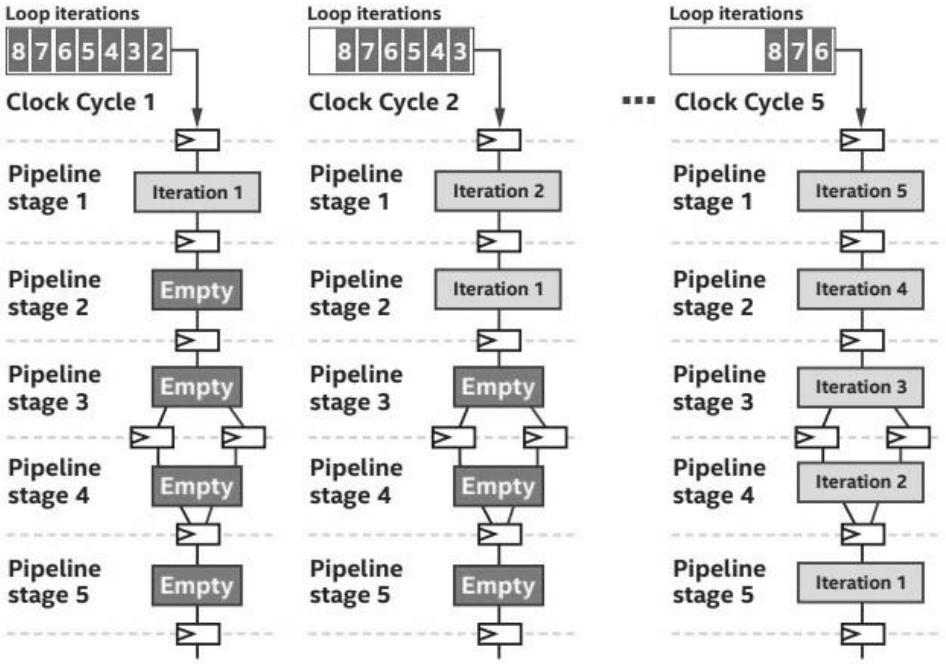


Рисунок 2.3 - Конвеєрний цикл на основі FPGA ядер

При виконанні ядра NDRange на кожному такті інший робочий елемент входить на перший етап користувацького конвеєра обчислень.

На відміну від інших архітектур із користувальницькими обчислювальними блоками конвеєра FPGA (рисунок 2.3), розробники мають додаткову опцію компромісу ресурсів FPGA за пропускну здатність шляхом збільшення ширини конвеєра SIMD для паралельної обробки більшої кількості робочих елементів. Наприклад, ітерації циклу можна розгорнути для досягнення паралельності даних (рисунок 2.4).

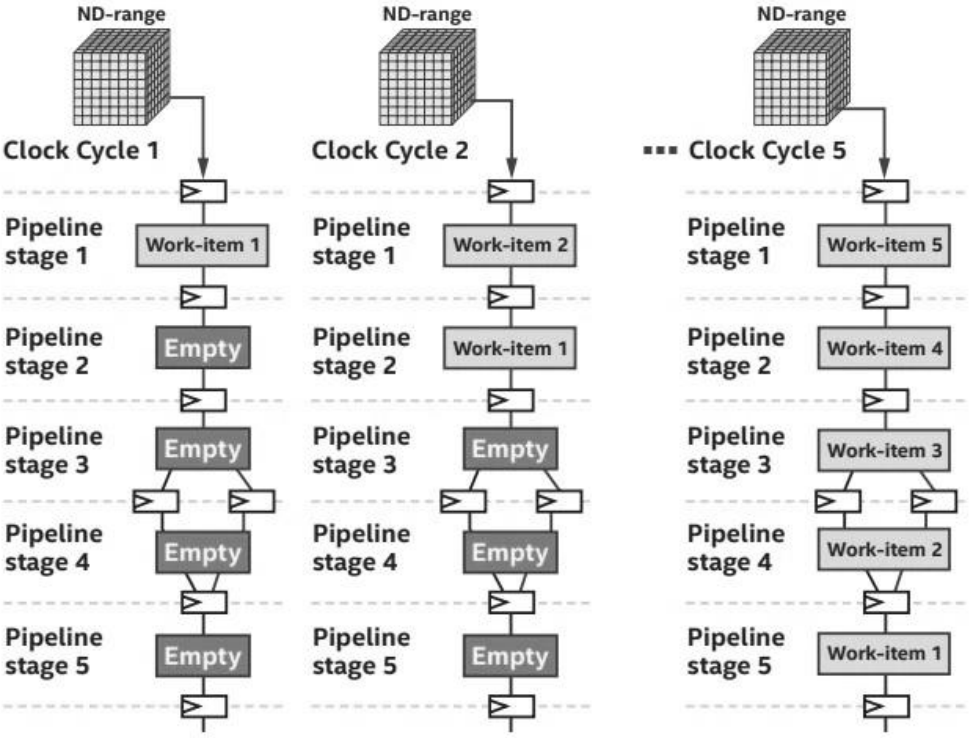


Рисунок 2.4 – SIMD конвеєр для паралельної обробки на основі FPGA [1]

2.2 Аналіз технологій для вирішення поставленої задачі

З розвитком потужностей, комп'ютерний зір та обробка зображень набули широкого застосування. Апаратна реалізація дозволяє використовувати більш якісні та складні алгоритми. Доступна велика кількість бібліотек примітивів, що дозволяють використовувати можливості відеокарти.

Будь-яка мова, яка дозволяє коду, що виконується на CPU, опитувати шейдер GPU для повернення значень, може створити фреймворк GPGPU. Попередня обробка зображень є обчислювально-вимогливим прикладом GPGPU. Проте, окрім мануального розпаралелювання, розробники пропонують використання спеціалізованих бібліотек для прискореної обробки зображень, перевагою яких є простота використання та великий набір реалізованих алгоритмів. Стандарти програмування для паралельних обчислень загального призначення на GPU включають OpenCL (незалежний

від виробника), OpenACC , OpenMP і OpenHMPP. Проте, ці бібліотеки, не охоплюють алгоритмів обробки зображень. Серед спеціалізованих бібліотек прискореної обробки зображень на ЦПУ та ГПУ можна виділити:

- OpenCV (Open Source Computer Vision Library) у поєднанні із GpuCV;
- Kornia (Python библиотека для обработки изображений в задачах CV);
- Nvidia Perfomance Primitives;
- GPU4VISION;
- OpenVIDIA;
- Scikit-Image;
- підмодуль scipy.ndimage бібліотеки Scipy.

Бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV (Open Source Computer Vision Library) – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення. Реалізована мовою C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов.

Функції OpenCV:

- інтерпретація зображень;
- калібрування камери за зразком;
- усунення оптичних спотворень;
- визначення подібності;
- аналіз переміщення об'єкта;
- визначення форми об'єкта та стеження за об'єктом;
- 3D-реконструкція;
- сегментація об'єкта;
- розпізнавання жестів.

Задачею бібліотеки GpuCV є реалізувати примітиви OpenCV на GPU, а також надати CPU, OpenGL, CUDA реалізації примітивів.

До складу функцій GpuCV входять:

- перетворення колірних форматів;
- обчислення країв об'єктів, морфологія (рисунок 2.5);
- дискретні перетворення (FFT, DCT);
- геометричні перетворення;
- гістограми.

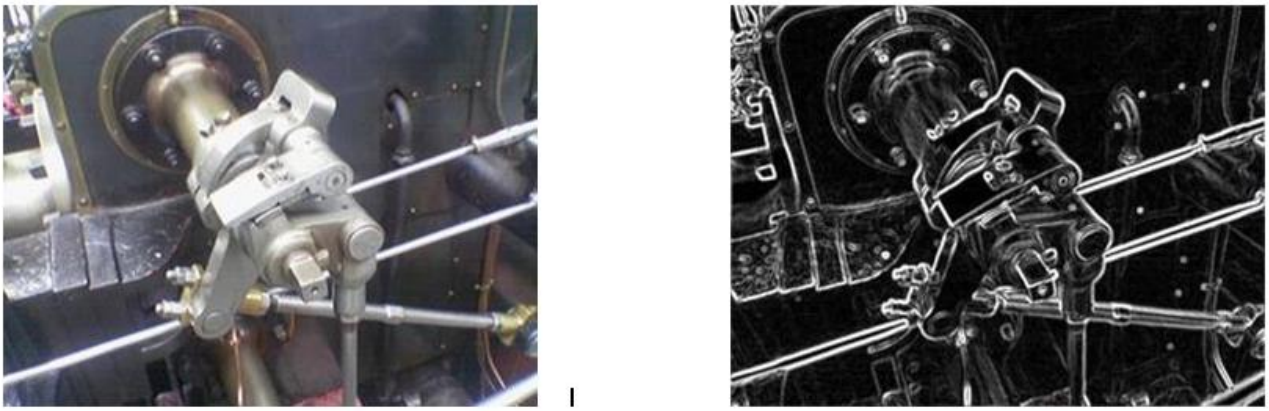


Рисунок 2.5 – Результат реалізації оператора Собеля

Бібліотека Kornia написана з допомогою pytorch, у її основі лежать готові рішення, такі як torchvision, PIL, skimage, tf.image, OpenCV. У Kornia реалізована можливість виконання обчислень не тільки з використанням CPU, а й GPU, що робить фреймворк досить продуктивним.

Рисунок 2.6 – Результат роботи методу adjust_contrast

Бібліотека містить у своєму складі ряд компонентів, серед основних є:

1) kornia.enhance – надає набір функцій для покращення зображень: adjust_brightness – керує яскравістю зображення, adjust_contrast - управління контрастністю (рисунок 2.6), adjust_saturation – насиченість зображення, adjust_gamma – гамма-контроль, adjust_hue – управління відтінками;

2) kornia.color - функції для роботи з кольором:bgr_to_rgb – наводить формат кольору BGR до RGB. rgb_to_grayscale – RGB для напівтонового (рисунок 2.7), rgb_to_hsv – привести зображення RGB до колірної моделі HSV;

Рисунок 2.7 – Результат роботи методу rgb_to_grayscale

3) kornia.filters – набір методів для фільтрації зображень, а також функції визначення меж на зображеннях: box_blur, blur_pool2d, gaussian_blur2d, max_blur_pool2d, median_blur, motion_blur (різні види розмиття, прямокутне, за Гаусом, рухом тощо), spatial_gradient, sobel, laplacian, canny (функції визначення меж на зображеннях) на рисуноку 2.8;

4) kornia.morphology - набір інструментів для математичної морфології - це функції ерозії, нарощування, розмикання тощо: morphology.dilation, morphology.erosion, morphology.opening, morphology.closing, morphology.gradient, morphology.bottom_hat, morphology.top_hat;

Рисунок 2.8 – Результат роботи методу kornia.filters.canny()

5) kornia.geometry - методи для роботи з геометрією, у тому числі зі спотвореннями перспективи: get_perspective_transform - функція обчислення трансформації перспективи, warp_perspective – застосування трансформації зображення (рисунок 2.9), rotate – поворот зображення, get_rotation_matrix2d - розрахунок матриці множинного повороту, warp_affine – застосувати обертання із матриці, rescale – зміна розміру (є antialiasing);

Рисунок 2.9 – Результат роботи методу get_perspective_transform() и warp_perspective()

6) kornia.augmentation – інструменти аугментації: ColorJitter, RandomAffine, RandomPerspective.

Методи та функціонал бібліотеки допомагають заощадити час на написання коду для розробників програмного забезпечення в області computer vision. Наприклад, в kornia реалізований алгоритм Connected-component labeling (CCL) для сегментації дрібних об'єктів на зображенні, на базі цього фреймворку легко реалізувати навчання моделі Torch з автоматичною аугментацією датасету та поділом даних на батчі, при цьому є можливість вибору між CPU або GPU.

Функціонал, що реалізовано у Nvidia erfomance Primitives є наступним:

- арифметичні операції (підсумовування, множення);
- перетворення колірних форматів;
- обчислення стандартних різниць зображень (SAD, SSD, MAD);
- Ппорівняння та трешхолдинг зображень;
- 1D фільтри (лінійний фільтр, Window Sum);

2D фільтри (морфологія, медіанний фільтр, лінійні фільтри, що настроюються).

Недоліками наведеної бібліотеки є те, що реалізовано невелику кількість примітивів (рисунок 2.10), багато часу витрачається на копіювання даних.

Особливістю бібліотеки GPU4VISION є те, розробники викладають частини своїх проектів (FlowLib (Anisotropic Huber-L1 OpticalFlow), VMFLib (Image Denoising & Segmentation)), алгоритми заточені під realtime, реалізовані інтерфейси до камер та сенсорів.

Open-source проект OpenVIDIA реалізує новітні алгоритми комп'ютерного зору, містить ефективно реалізовані та відпрофільовані алгоритми, на сайті проекту збирається інформація про алгоритми комп'ютерного зору.

OpenVIDIA включає в себе:

- Optical Flow;

- Feature Detection & Tracking;
- згортку невеликих радіусів із налаштуванням;
- детектор країв та медіанний фільтр;
- побудова гістограм;
- геометричні та афінні перетворення.

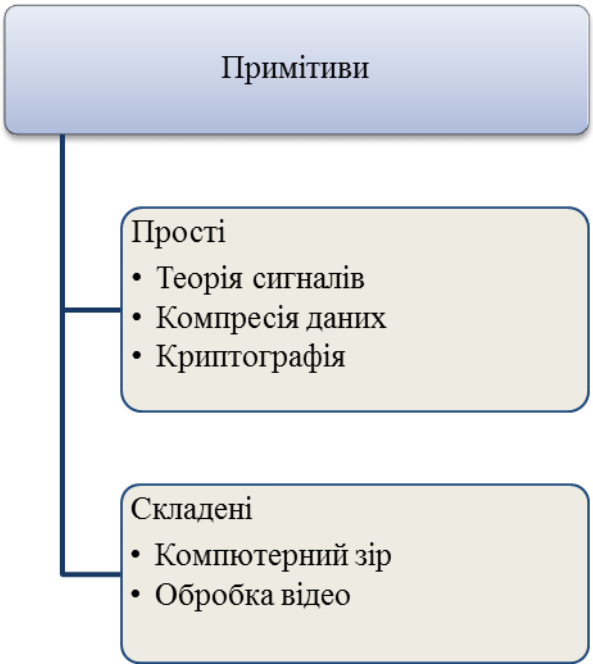


Рисунок 2.10 – Робота з примітивами у бібліотеці NPP

Scikit-image — це бібліотека обробки зображень на основі Python, яка має деякі частини, написані на Cython, що забезпечеує продуктивність та подібні до мови програмування C. Включає в себе алгоритми для: сегментації, геометричних перетвореннь, маніпулювання кольоровим простором, аналізу, фільтрації.

Scipy використовується для математичних і наукових обчислень, але також може виконувати багатовимірну обробку зображень за допомогою підмодуля scipy.ndimage. Він надає функції для роботи з n-вимірними масивами Numpy. Scipy пропонує найбільш часто використовувані операції обробки зображень, такі як: читання зображень, сегментація зображення, згортка, розпізнавання облич, вилучення функцій тощо.

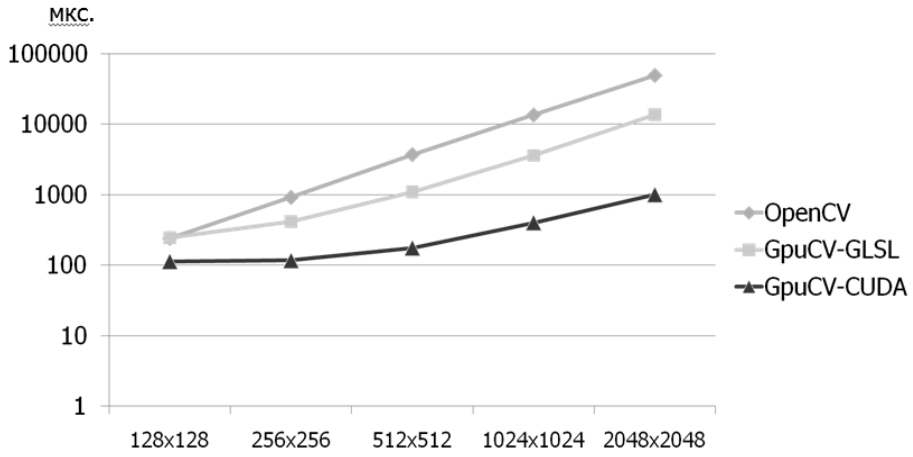


Рисунок 2.11 – Результати аналізу продуктивності реалізації оператору Собеля на різних програмних та апаратних платформах

Аналіз підтверджує ефективність використання технології CUDA для реалізації задач (рисунок 2.11), де є потреба у масовому паралелізмі, що забезпечується особливістю архітектури GPU. Задачі виділення контуру об’єктів зображення вимагають великих обчислювальних ресурсів, тому використання паралельних підходів та апаратного прискорення завдяки обчислювальному ресурсу графічного процесора є обґрунтованим.

2.3 Аналіз методологічного підґрунтя для рішення поставленої задачі

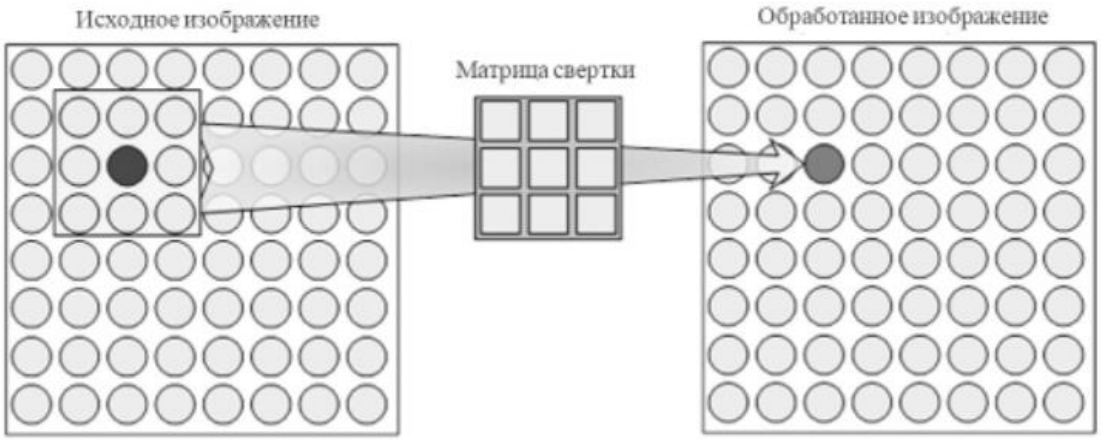
Методи попередньої обробки зображень залежать від завдань досліджень і можуть включати такі види робіт [17-19]:

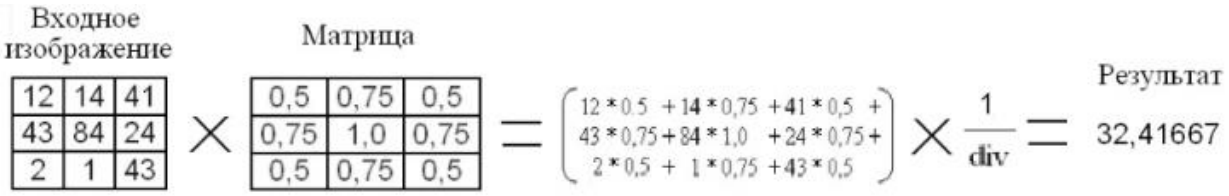
- фільтрцію зашумлених зображень;
- корекція яскравості та контрасту.

Під фільтрцією зашумлених зображень як правило розуміють видалння **ц**ифрового шуму зображення — дефектів зображення, який вносить фотосенсори та електроніка пристроїв, які їх (рисунок 2.12) Для його придушення використовують такі методи:

- лінійне усереднення точок сусідів. Розмиття по Гаусу;

- медіанна фільтрація;
- морфологічні перетворення.





Рсунок 2.12 – Приклад накладання матриці згортки при шумопригніченні

Лінійне усереднення точок сусідів — найпростіший вид алгоритмів видалення шуму. Основна ідея їх у тому, щоб брати середнє арифметичне значення точок в деякій околиці як нове значення точки.

Фізично така фільтрація реалізується за допомогою обходу пікселів зображення матрицею згортки, що має вигляд, наведений на рисунку ….

Гаусове розмиття (також відомий як Gaussian згладжування) є результатом нечіткості зображення від Gaussian функції. Це широко використовуваний ефект у графічному програмному забезпеченні, зазвичай зменшення шумів зображення. Візуальним ефектом цієї техніки розмиття є плавне розмиття, що нагадує ефект перегляду образу через напівпрозорий екран, помітно відрізняється від боке ефекту, створюваний не у фокусі об'єктивом або тінню об'єкта при звичайному освітленні. Гаусове згладжування (рисунок 2.13) також використовується як стадія попередньої

обробки в комп'ютерному зорі алгоритми для поліпшення структури зображення в різних масштабах.

Рисунок 2.13 – Результат роботи фільтру Гауса

Різновид лінійного згортання реалізується за допомогою обходу пікселів зображення матрицею згортки, що має вигляд, наведений на рисунку

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Рисунок 2.14 – Обхід пік селів матрицею згортки при роботі фільтру Гауса

Матриця 5×5 заповнюється за нормальним (гаусовим законом). Від розміру матриці залежить сила розмиття.

Медіанний фільтр являє собою віконний фільтр, що послідовно сканує зображення, і повертає на кожному кроці один з елементів, що потрапили у вікно фільтра.

Пікселі, які «попадають» у вікно, сортуються в порядку зростання і вибирається значення, яке знаходиться посередині відсортованого списку.

Медіанний фільтр зазвичай використовується для зменшення шуму або згладжування зображення.

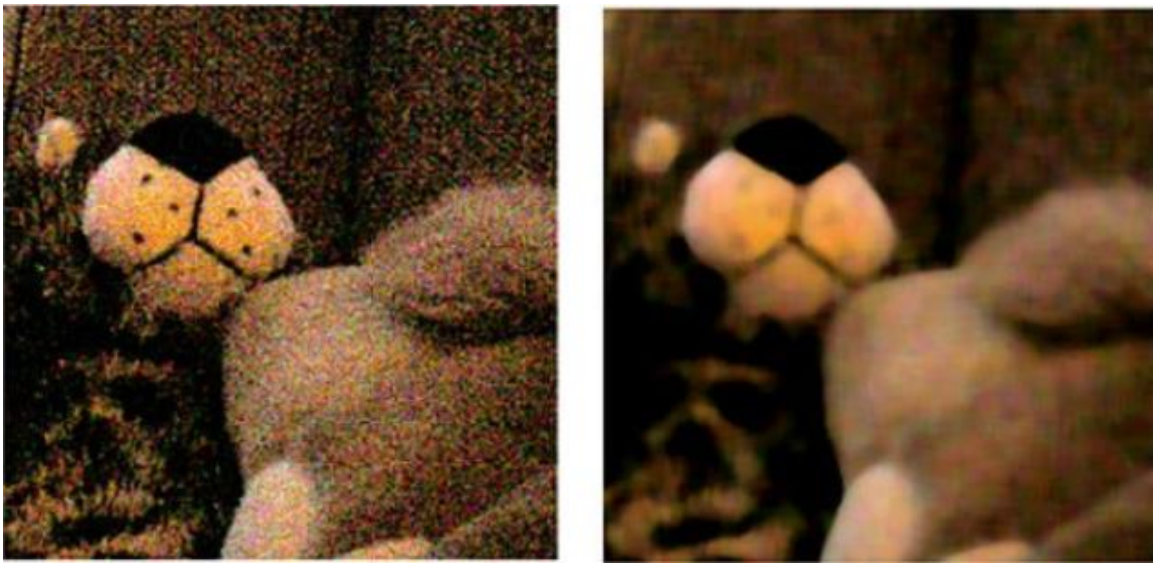


Рисунок 2.15 – Результат роботи медіанного фільтру

Для покращення чіткості зображення використовується наступний фільтр (div=1) .

Дилатація (морфологічне розширення) – згортка зображення або виділення зображення деяким шаблоном. Шаблон може мати довільну форму та розмір. При цьому в ньому виділяється єдина провідна позиція (anchor), яка поєднується з поточним пікселем при обчисленні згортки.

Застосування дилатації зводиться до проходу шаблоном по всьому зображенню та застосування оператора пошуку локального максимуму інтенсивності пікселів зображення, які накриваються шаблоном. Якщо максимум дорівнює 1 то точка, в якій знаходиться анкор шаблону буде біла. Така операція викликає зростання світлих областей на зображенні.

Ерозія(морфологічне звуження) - операція, обернена дилатації. Дія ерозії подібна до дилатації, різниця лише в тому, що використовується оператор пошуку локального мінімуму. Якщо мінімум дорівнює 0, то точка, де знаходиться анкор шаблону, буде чорна. На малюнку праворуч сірим кольором відмічені пікселі, які стануть чорними внаслідок ерозії.

Операція "Дилатація" - аналог логічного "або", операція "Ерозія" - аналог логічного "і".

Результат морфологічних операцій багато в чому визначається

шаблоном (структурним елементом). Вибираючи різний структурний елемент, можна вирішувати різні завдання обробки зображень:

- шумопридушення;
- виділення меж об'єкта;
- нарощування точок області;
- виділення скелета об'єкта.

Яскравість є характеристикою, що визначає те, наскільки сильно кольори пікселів відрізняються від чорного кольору. Наприклад, якщо оцифрована фотографія зроблена в сонячну погоду, її яскравість буде значною. З іншого боку, якщо фотографія зроблена ввечері або вночі, її яскравість буде невелика.

Контраст є характеристикою того, наскільки великий розкид мають кольори пікселів зображення. Чим більший розкид мають значення кольорів пікселів, тим більший контраст має зображення.

За аналогією з термінами математичної статистики можна відзначити, що яскравість - середнє значення; контраст - середньоквадратичне відхилення від середнього значення.

Серед важливих методів попередньої обробки зображень, які зустрічаються при детектуванні об’єктів на зображення та подальшому їх аналізі, є методи оконтурювання зображень [20-22], так звані, детектори кордонів. Методи оконтурювання можна поділити на [23,24]:

- - оконтурювання Canny;
- - оконтурювання Sobel;
- - оконтурювання LoG.

Достоїнством детектора Кенні є те, що для зашумлених зображень цей метод забезпечує найкраще виявлення меж порівняно з іншими способами цієї групи. Детектор має високу ймовірність детектування, високою точністю локалізації, єдиністю відгуку на один контрастний перепад. Недоліком цього алгоритму є те, що він вимагає значно більшого часу, що відіграє важливу роль, особливо в системах, що працюють у режимі реального часу. Також

недоліками цього є складність реалізації і дуже велика ресурсоємність. Детектор Кенні вимагає ручної установки двох порогів та розміру ядра, значення яких суттєво змінюють малюнок одержуваних контурів [25-26]. Не дозволяє отримати автоматичні алгоритми виділення кордонів.

Подальше дослідження буде зосереджено на даному методі.

Алгоритм детектора країв Canny названий на честь його винахідника Джона Ф. Кенні, який винайшов алгоритм у 1986 році. Детектор країв Canny зазвичай приймає зображення в градаціях сірого як вхідні дані і створює зображення, що показує розташування розривів інтенсивності як вихідні дані.

3 РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Критерії сегментації

У комп'ютерному зорі, сегментація – це процес поділу цифрового зображення на кілька сегментів (безліч пікселів, також званих суперпікселями). Мета сегментації полягає у спрощенні та/або зміні уявлення зображення, щоб його було простіше та легше аналізувати. Сегментація зображень зазвичай використовується для того, щоб виділити об'єкти та межі (лінії, криві, тощо) на зображеннях. Точніше, сегментація зображень – це процес присвоєння таких міток кожному пікселю зображення, що пікселі з однаковими мітками мають загальні візуальні характеристики.

Результатом сегментації зображення є безліч сегментів, які разом покривають все зображення або безліч контурів, виділених із зображення. Всі пікселі в сегменті схожі за деякою характеристикою або властивістю, наприклад, за кольором, яскравістю або текстурою. Сусідні сегменти значно відрізняються за цією характеристикою.

Алгоритм сегментації повинен чітко визначати де закінчується один сегмент і починається інший. Як правило, межі являють собою характерні перепади яскравості та/або відтінків кольору, що зустрічаються на ділянках об'єкт-фон, площина-тінь, папужка-море, напис на паркані… І якщо «перепад» більший за деякий «поріг» (threshold), то з цього має бути, що це різні сегменти. Але є невелика проблемка: перепади можуть сильно змінюватись для різних об'єктів, і сегменти складно «відокремити» однозначно заданою (const) граничною величиною (threshold). Для поверхні столу на тлі стіни: перепади сусідніх пікселів вздовж поверхні столу (стіни) будуть відносно невеликими, а ось на межі стіл-стіна буде стрибок, який розділить сегменти. Це зрозуміло. А якщо у нас папужка на тлі моря? Він же сам дуже «строкатий», всередині нього великі перепади інтенсивності

(зелений-жовтий-...), щоб його «відокремити» від моря, потрібен інший «поріг» (threshold). І поступово приходимо до висновку, що поріг, який вирішує, що сусіднім сегментам «не судилося бути разом», має спиратися не лише на локальні показники: перепад інтенсивностей вздовж одного ребра (що поєднує сусідні пікселі), а й на те, наскільки ці сегменти самі по собі гладкі (у плані кольору) або «строкаті».

Сегментація є проміжним етапом між виявленням руху та етапами супроводу та розпізнавання об'єктів. На цьому етапі відбувається групування розрізнених ділянок або фрагментів зображення в область, що належить одному об'єкту, або поділ якоїсь ділянки зображення на області, що належать різним об'єктам. При цьому групування здійснюється за різними ознаками, такими як яскравість, колір, текстура, рух в одному напрямку, з однаковою швидкістю тощо. Застосування способу групування, за тією чи іншою ознакою, залежить від завдання, яке необхідно розв'язати. Оскільки етап сегментації передує етапам вищого рівня обробки зображення (аналіз, розпізнавання об'єктів), то методам сегментації застосовуються певні вимоги:

- максимальна відповідність сегментованої області реальному об'єкту;
- робота у режимі реального часу;
- низька ймовірність помилок;
- стала робота у складних умовах.

Аналіз методів сегментації показує, що вони не позбавлені недоліків. Тому вибір того чи іншого методу залежить від конкретного завдання. Видно, більшість методів орієнтовано під певні ознаки сегментації (яскравість, колір, текстура, швидкість тощо.). Так само при виборі методу сегментації необхідно враховувати і те, що властивості об'єктів під впливом зовнішніх факторів можуть змінюватися. Наприклад, при зміні освітленості (інтенсивності сонячного світла) змінюються яскраві характеристики об'єктів, або об'єкти мають неоднорідну яскравість або слабко виражену текстуру. У разі знаходження в кадрі безлічі об'єктів зі схожими ознаками, що рухаються довільними траєкторіями, що перекривають один одного,

провести точну сегментацію практично неможливо.

Сегментація методом вододілу. У сегментації методом вододілу розглядається абсолютна величина градієнта зображення як топографічної поверхні. Пікселі, що мають найбільшу абсолютну величину градієнта яскравості, відповідають лініям вододілу, які становлять межі областей. Вода, поміщена будь-який піксель всередині загальної лінії вододілу, тече вниз до загального локального мінімуму яскравості. Пікселі, від яких вода стікається до загального мінімуму, утворюють водозбір, який є сегментом.

Сегментація за допомогою моделі. Основне припущення цього підходу – те, що структури, що цікавлять, або органи мають повторювані геометричні форми. Отже, можна знайти ймовірну модель для пояснення змін форми органу і потім, сегментуючи зображення, накладати обмеження, використовуючи цю модель як апріорну. Таке завдання включає приведення тренувальних прикладів до загальної пози, імовірнісне представлення змін наведених зразків та статистичний висновок для моделі та зображення. Сучасні методи в літературі для сегментації, заснованої на знанні, містять активні моделі форми та зовнішності, активні контури, деформовані шаблони та методи встановлення рівня.

Багатомасштабна сегментація. Сегментація зображень виконується в різних масштабах у масштабному просторі та іноді поширюється від дрібних масштабів до великих.

Критерій сегментації може бути складним і може брати до уваги як локальні, і глобальні критерії. Загальна вимога – те, що кожна область має бути пов'язана у певному сенсі

Методи розростання областей. Першим був метод розростання областей із насіння. Як вхідні дані цей метод приймає зображення та набір насіння. Насіння відзначає об'єкти, які потрібно виділити. Області поступово розростаються, порівнюючи всі незайняті сусідні пікселі з областю. Різниця delta між яскравістю пікселя та середньою яскравістю області використовується як міра схожості. Піксель з найменшою такою різницею

додається до відповідної області. Процес триває, доки всі пікселі не будуть додані до одного з регіонів.

Метод розростання областей із насіння вимагає додаткового введення. Результат сегментації залежить від вибору насіння. Шум на зображенні може спричинити те, що насіння погано розміщене. Метод розростання областей без використання насіння – це змінений алгоритм, який не вимагає явного насіння. Він починає з однієї області A1 - піксель, вибраний тут, незначно впливає на кінцеву сегментацію. На кожній ітерації він розглядає сусідні пікселі так само як метод розростання областей з використанням насіння. Але він відрізняється тим, що якщо мінімальна delta менше, ніж заданий поріг T, він додається у відповідну область Aj. Інакше піксель вважається дуже різним від усіх поточних областей Ai і створюється нова область An+1, що містить цей піксель.

Один із варіантів цього методу, запропонований Хараліком та Шапіро (1985), заснований на використанні яскравості пікселів. Середня та дисперсія області та яскравість пікселя-кандидата використовується для побудови тестової статистики. Якщо тестова статистика досить мала, то піксель додається до області, і середнє та дисперсія області перераховуються. Інакше піксель ігнорується та використовується для створення нової області.

3.2 Застосування оконтурювання на практиці

Процес сегментації, заснований виключно на використанні низькорівневих локальних атрибутів, таких як яскравість і колір, загрожує суттєвими помилками. Щоб надійно виявляти межі, пов'язані з об'єктами, необхідно також використовувати високорівневі знання про те, які об'єкти можуть ймовірно зустрітися в даній сцені. При розпізнаванні мови така можливість виникла завдяки використанню формальних засобів прихованої марківської моделі; у контексті обробки зображень пошук такої універсальної інфраструктури залишається темою інтенсивних досліджень.

Так чи інакше, уявлення високорівневих знань про об'єкти є темою даного та наступного розділу.

Межі об'єктів на зображенні значно зменшують кількість даних, які необхідно обробити, і водночас зберігає важливу інформацію про об'єкти на зображенні, їх форму, розмір, кількість. Головною особливістю техніки виявлення кордонів є можливість отримати точну лінію з гарною орієнтацією. У літературі описано багато алгоритмів, які дозволяють виявляти межі об'єктів, але ніде немає опису того, як оцінювати результати обробки. Результати оцінюються суто індивідуально та залежать від сфери їх застосування.

Виявлення меж - фундаментальний інструмент для сегментації зображення. Такі алгоритми перетворюють вхідне зображення на зображення з контурами об'єктів, переважно в сірих тонах. У обробці зображень, особливо у системах комп'ютерного зору, з допомогою виділення контуру розглядають важливі зміни рівня яскравості на зображенні, фізичні та геометричні параметри об'єкта на сцені. Це фундаментальний процес, який описує в загальних рисах об'єкти, отримуючи цим деякі знання про зображення. Виявлення кордонів є найпопулярнішим підходом виявлення значних неоднорідностей.

Контур є місцевою зміною яскравості зображення. Вони зазвичай проходять по краю між двома областями. За допомогою меж можна отримати базові знання про зображення. Функції їх отримання використовуються передовими алгоритмами комп'ютерного зору та таких галузях, як медична обробка зображень, біометрія тощо. Виявлення меж - активна область досліджень, оскільки він полегшує високорівневий аналіз зображень. На напівтонових зображеннях існує три види розривів: точка, лінія та межа. Для виявлення всіх трьох видів неоднорідностей можна використовувати просторові маски. У технічній літературі наведено та описано велику кількість алгоритмів виділення контурів та кордонів.

Таким чином, можна виділити такі етапи оконтурювання:

- перетворення зображення на двомірний масив чисел, де кожне число це RGB уявлення кожного кольору на картинці;
- знаходження середнього значення контрастності зображення;
- перетворення кольорового зображення на відтінки сірого;
- сегментація;
- оконтурювання.

На рисунку 3.1.представлено алгоритм роботи модулю попередньої обробки зображень більшості інтелектуальних систем аналізу зображень.

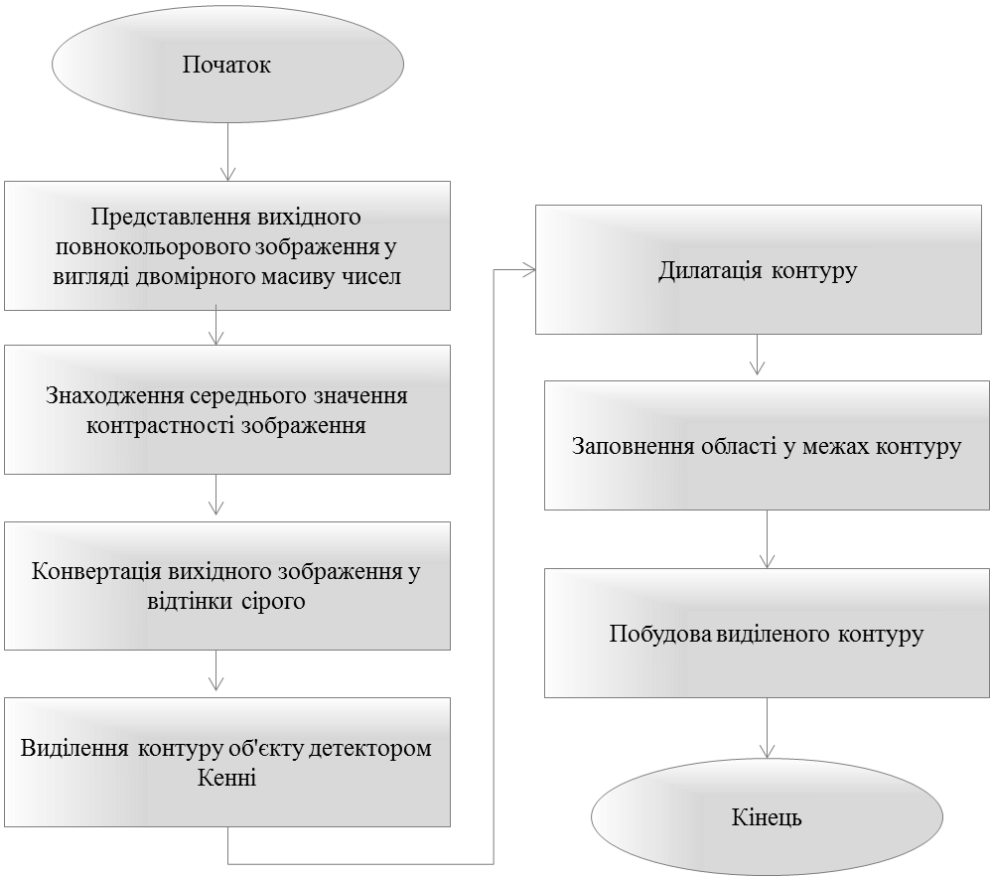


Рисунок 3.1 – Традиційна (послідовна) схема алгоритму контурної сегментації комбінованим методом

3.3 Програмна реалізація

3.3.1 Робота програми знаходження контурів за допомогою OpenCV

Для вирішення цього завдання було розроблено дві програми. У першій

програмі завдання вирішувалося за допомогою бібліотеки Open Source Computer Vision, а у другій програмі вирішувалося завдання реалізації розробленого методу оконтурювання.

Для першої програми, яка написана за допомогою бібліотеки OpenCV, було створено консольну програму. Після цього, для підключення бібліотеки до проекту, знадобилося у властивостях проекту підключити «Lib» та «Include» файли. Потім, у властивостях лінковника прописуємо необхідні файли для реалізації способу. Реалізація програми представлена у прикладі 3.1.

Лістинг 3.1 – OpenCV реалізація

```
// вікно для відображення картинки
cvNamedWindow("binary", CV_WINDOW_AUTOSIZE);
double start = clock();
// Перетворимо на градації сірого
cvCvtColor(image, gray, CV_RGB2GRAY);
CvMemStorage* storage = cvCreateMemStorage(0);
CvSeq * contours = 0;
// Перетворимо на двійкове
cvInRangeS(gray, cvScalar(40), cvScalar(150), bin); //
atoi(argv[2])
// знаходимо контури
int contoursCont = cvFindContours(bin, storage, &contours,
sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE,
cvPoint(0, 0));
std::cout <<"Time work: " <<clock() - start << endl;
// намалюємо контури
for (CvSeq* seq0 = contours; seq0 != 0; seq0 = seq0->h_next) {
cvDrawContours(dst, seq0, CV_RGB(255, 216, 0), CV_RGB(0, 0,
250), 0, 1, 8); // малюємо контур
}
// показуємо картинки
cvShowImage("binary", bin);
// чекаємо натискання клавіші
cvWaitKey(0);
// звільняємо ресурси
cvReleaseImage(&image);
cvReleaseImage(&gray);
cvReleaseImage(&bin);
cvReleaseImage(&dst);
// видаляємо вікна
cvDestroyAllWindows();
```

3.3.2 Реалізація знаходження контурів на зображенні методом сегментації

Для реалізації другої програми знадобилося створити win32 application тому, що в консолі виведення зображення стандартними засобами скрутне. Для того, щоб перетворити колір у RGB була використана функція GetPixel із бібліотеки Windows.h. Вона повертає цілісне уявлення кольору. Завантажити зображення в контекст пристрою можливо функцією LoadImage (дозволяє завантажити зображення в структуру HBITMAP) із бібліотеки Windows.h. Функція працює лише з форматом bmp. І тому програма працює тільки з цим форматом. У прикладі 2.2 представлений фрагмент реалізації.

Лістинг 3.2 –Реалізація перетворення картинки на масив чисел

```
//завантаження зображення
HBITMAP hBitmap = (HBITMAP)LoadImage(hInst, (LPCSTR)_imName,
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
     GetObject(hBitmap, sizeof(BITMAP), &bm);
     //завантаження у контекст пристрою
     SelectObject(hMemDC, hBitmap);
     s.cy = bm.bmHeight;
     s.cx = bm.bmWidth;
     //по піксельному захопленню кожного кольору
     for (size_t i = 0; i < x; ++i)
          for (size_t j = 0; j < y; ++j)
               out[i][j] = GetPixel(hMemDC, i, j);
```

Далі знаходимо середнє значення контрастності зображення за формулою 3.1.

$$r*0.299+g*0.587+b*0.114 \quad (3.1)$$

Після знаходження в масиві середнє значення контрастності, перетворюємо вихідний масив на масив,де ціле чисельне уявлення кольору замінюється на значення контрастності на зображенні. Після цього виконуємо сегментацію. Кількість сегментів залежатиме якраз від середнього значення контрастності. У прикладі 3.3 описано фрагменти реалізації

вищезазначеного.

Лістинг 3.3 –Реалізація сегментації

```
          getAvaragValue();

          int countSegment = getAvaragValue();/ kof;
          int *colors;
          colors = new int[countSegment];
/#pragma omp parallel for
          for (int i = 0; i < countSegment; i++)
               colors[i] = RGB(i * sensentive, i * sensentive, i
* sensentive);
          #pragma omp parallel for
          for (int i = 0; i < size.cx; i++)
               for (size_t j = 0; j < size.cy; j++)
                    image[i][j] =
colors[getContrast(image[i][j]) / countSegment];
          delete[]colors;
```

Для того що виділити контур, був реалізований алгоритм ковзаючого вікна, основна суть якого полягає в порівнюванні пікселів, що поруч стоять, на відмінності. Відмінність обчислюється формулою 2.2.

$$Sqrt((color1.b - color2.b)^2 + (color1.g - color2.g)^2 + (color1.r - color2.r)^2) \quad (3.2)$$

Після того як знайдено піксель, який відрізняється від сусіднього, аналізуємо інші пікселі на подібність, щоб прибрати шум з картинки. Після цих перетворень отримуємо зображення, на якому виділено контури всіх наявних об'єктів.

Для того, щоб прискорити роботу програми, були використані директиви компілятора OpenMP. Для роботи з OpenMP необхідно увімкнути його у властивостях проекту та підключити бібліотеку «omp.h». Прописуючи директиву #pragma omp for, програма автоматично розпаралелюється при компіляції. Лістинг 3.4 демонструє описане вище.

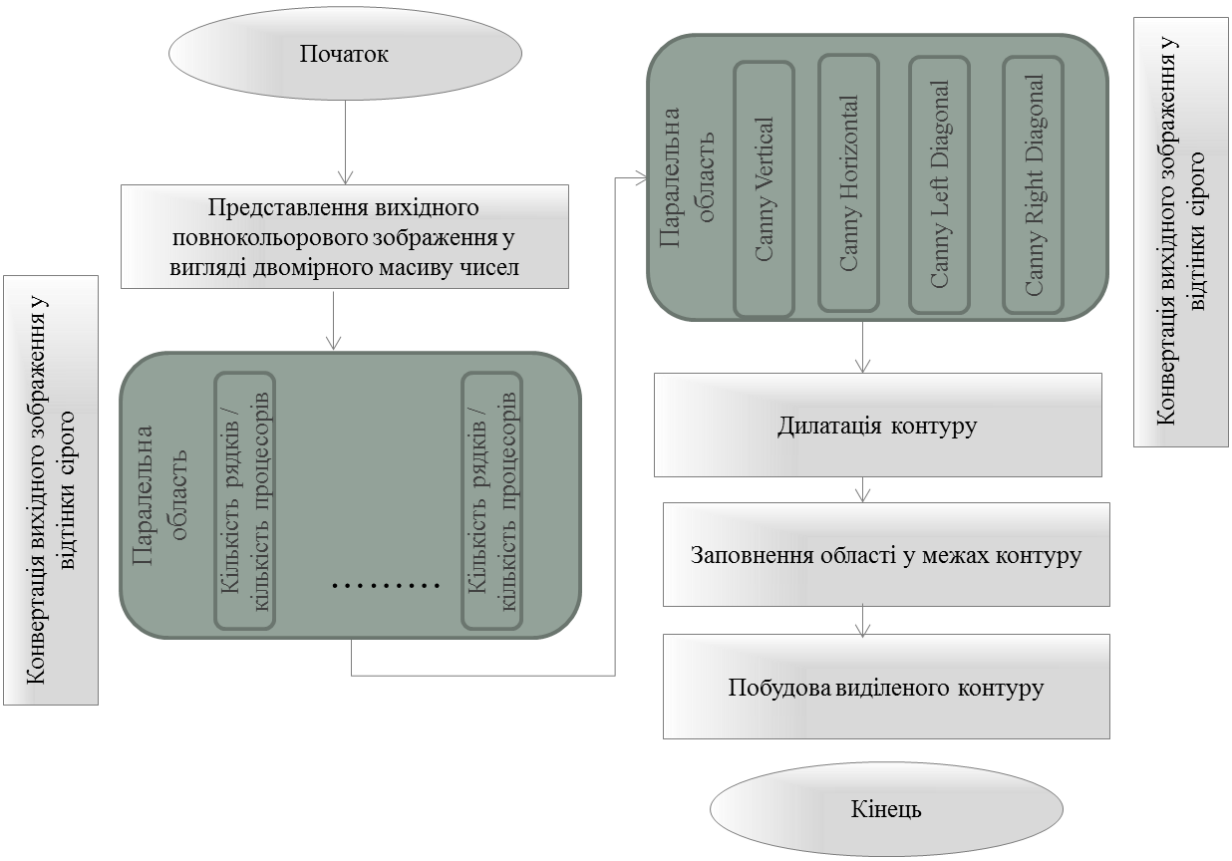


Рисунок 3.2 – Модифікована схема виділення контуру об’єктів вихідного зображення

Лістинг 3.4 - Розпаралелена програма

```
#pragma omp parallel for
          for (int i = 0; i < size.cx; i++)
               for (size_t j = 0; j < size.cy; j++)
                    image[i][j] =
colors[getContrast(image[i][j]) / countSegment];
```

Для зручності роботи з програмою була використана функція SaveFileName із стандартної бібліотеки commdlg.h. Ця функція дозволяє користувачеві вибрати зображення зі свого каталогу картинок.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У ході роботи було досліджено продуктивність операцій оконтурювання на обчислювальних системах різних типів – із масовим паралелізмом та із загальною памяттю. Перша – OpenCV реалізація методів виділення контурів із використанням функцій GpuCV, а також без наведених функцій, а друга – мануальная реалізація запропонованого способу виділення контурів на зображенні із використанням обчислювального ресурсу CPU. Після кількох випробувань було виявлено, що запропонований метод працює швидше, ніж послідовна OpenCV реалізація. У цій роботі порівняно час роботи програм на різних розмірах зображення із різною кількістю контрастних зон. Також було вивчено вплив розміру зображення на швидкість роботи застосунку, і показана залежність часу роботи застосунку від інших факторів.

Перший фактор оцінки роботи програми – зв'язність та реалістичність отриманого контуру. Наскільки добре запропонований та існуючий методи оконтурювання впоралися із завданням наведено на малюнку 4.1.

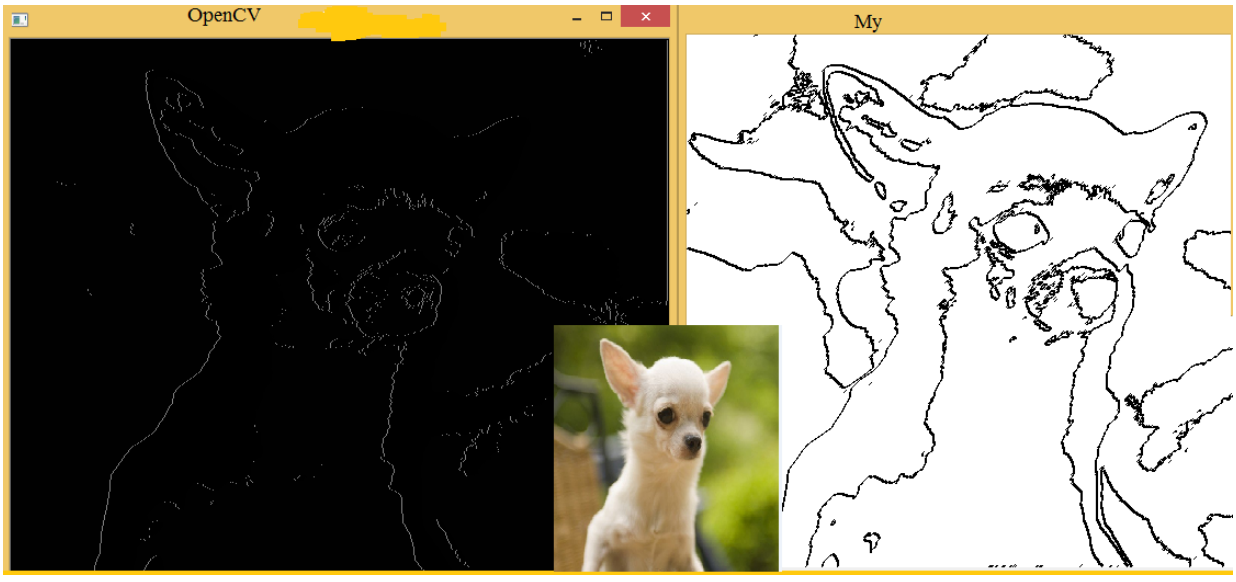


Рисунок 4.1 – Результат виділення контуру

На рисунку 4.2 представлений графік, який показує час роботи тестових програм на тих самих зображеннях. Можна помітити, що запропонований метод дозволяє отримати результат за більш короткий час, ніж послідовна реалізацію OpenCV. У аналізі брали участь зображення 1920*1080 із різним ступенем деталізації. На деяких зображеннях прискорення досягається вдвічі і більше.

На графі видно, що зображення номер 4 і 5 швидше були оброблені OpenCV реалізацією, оскільки саме ці зображення мають багато об'єктів та структурний елемент (ковзаюче вікно) витрачає багато часу для обробки зображення через велику кількість точок.

Також, варто відзначити, що в OpenCV час обробки зображення практично не залежить від розмірів картинки та обробляє близько 650мс. Запропонований метод дуже чутливий до кількості об'єктів на зображенні.

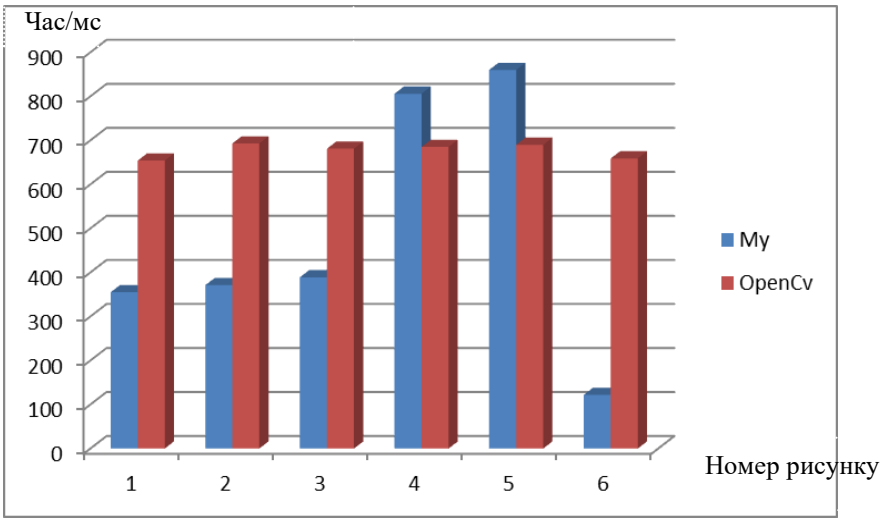


Рисунок 4.2 – Час роботи тестових програм на кольорових зображеннях

Тобто, чим більше деталізоване вихідне зображення, тим більшу продуктивність у порівнянні із запропонованим методом показує метод оконтурювання, запропонований в OpenCV.

Також було перевірено, як розмір зображення впливає на час обробки. Для цього було вираховано кількість пікселів на зображенні. На малюнку 4.3

видно, що розмір зображення практично не впливає на час знаходження контурів на зображення. Тому, можна зробити висновок, що на час вирішення поставленої задачі має вплив кількість об'єктів на зображенні, а не розмір вихідного зображення.

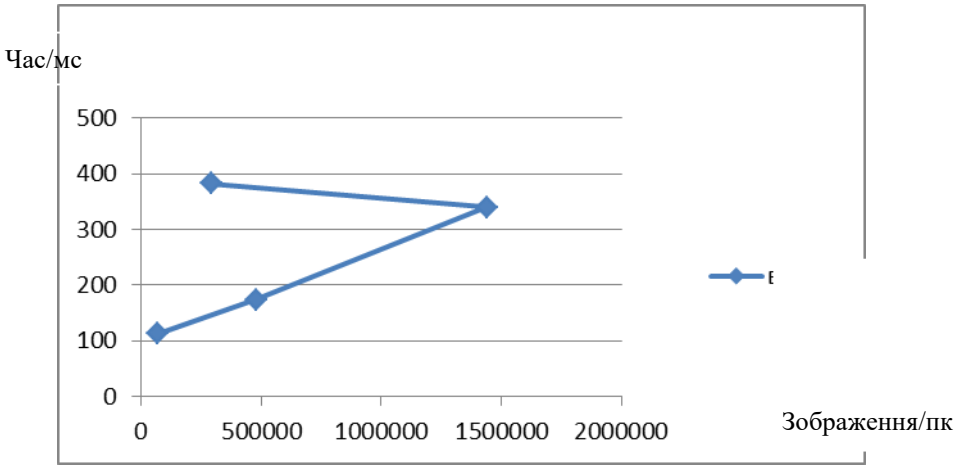


Рисунок 4.3 – Графік залежності часу та розміру зображення

Далі досліджувалися параметри алгоритму та його вплив на час виконання програми:

- розмір ковзаючого вікна (структурного елементу);
- кількість сегментів на зображенні.

На рисунку 4.4 явно показано, що розмір ковзаючого вікна сильно впливає на час роботи програми. Лінійна залежність. Також спостерігається зменшення зв’язності отриманого контуру із збільшенням розміру ковзаючого вікна.

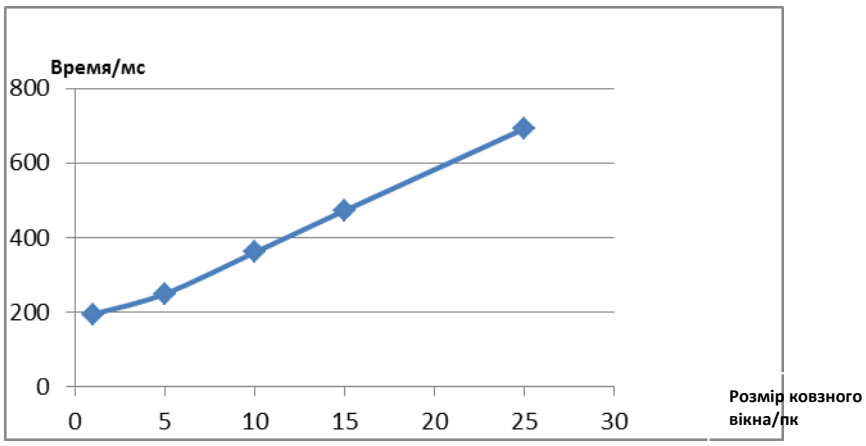
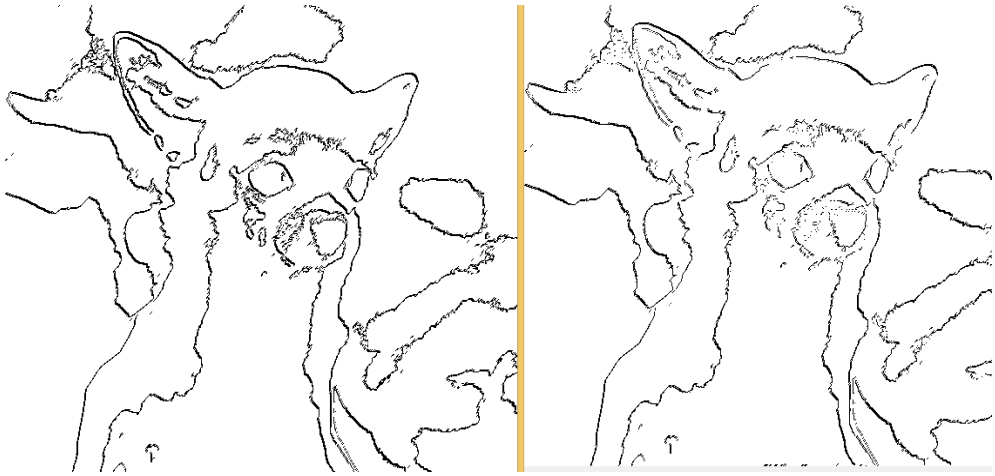


Рисунок 4.4 – Графік залежності часу та розмір ковзаючого вікна

На рисунку 4.5 можна помітити різниця при різних значеннях розмірів ковзаючого вікна.



а) б)

Рисунок 4.5 – Робота програми при різних значеннях ковзного вікна: а) розмір структурного елементу 5*5; б) розмір структурного елементу 15*15

Наступний параметр, який впливає на час виділення контуру - кількість сегментів на зображенні. Тобто від нього залежить, на скільки частин буде розбита палітра кольору зображення. Виходячи з графіка, який зображений на малюнку 4.6, можна зробити висновок, що цей параметр один з найважливіших параметрів, оскільки, неправильно визначивши його, можна

нічого не побачити на зображенні., тобто контури не будуть відображені

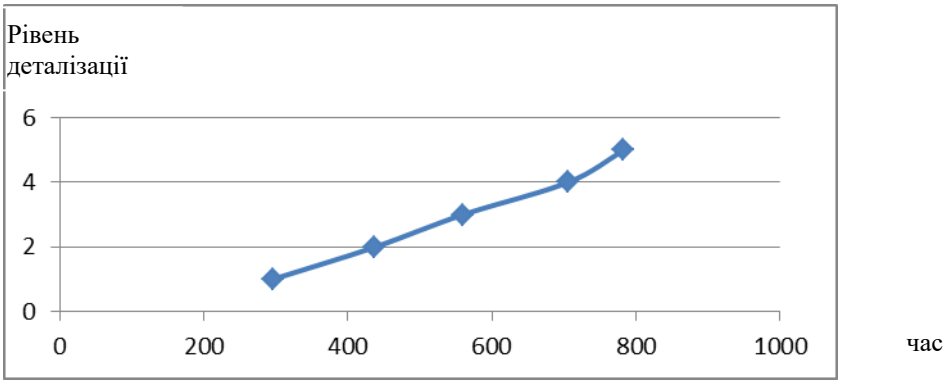


Рисунок 4.6 – Графік залежності часу та кількості сегментів

На рисунку 4.7 показано зображення за різних значень сегментів. Значення зліва направо, починаючи з двох.

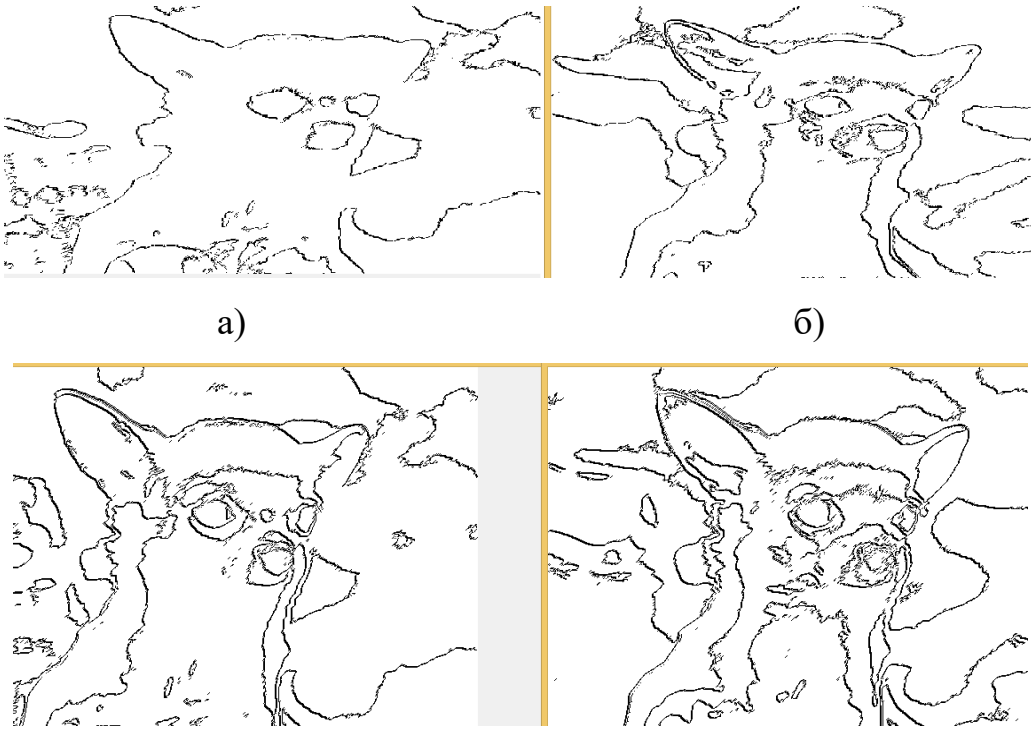


Рисунок 4.7 – Робота програми при різних значеннях сегментів: а) 2 сегменти, б) 3 сегменти, в) 4 сегменти, г) 4 сегменти

Таким чином, збільшення кількості сегментів призводить до збільшення детермінованих деталей на зображенні, але, в одночас, призводить до збільшення часу отримання результату у 2,5 рази для кількості сегментів від 2 до 4.

В результаті аналізу наведених у цьому розділі графіків видно, що запропонований метод виконує швидке зв'язне реалістичне знаходження контуру на зображеннях, в яких мало об'єктів. Збільшення кількості об'єктів на зображенні призводить до збільшення часу обчислень контуру. Для деяких зображень досягається прискорення вдвічі, в порівнянні з OpenCV реалізацією.

У роботі повністю дотримано вимог, що ставились у завданні – розглянуто існуючі методів оконтурювання та обґрунтовано необхідність та актуальність дослідження методів виділення контуру, через їх поширене використання в інтелектуальних системах аналізу вихідних зображень, запропоновано власний методу виділення контуру на вихідному зображенні шляхом удосконалення існуючого завдяки розпаралелюванню етапу конвертації до відтінків сірого та роботи ковзаючи вікон, а також порівняно реалізацій розроблених та існуючих методів на центральному та графічному процесорах.

Серед недоліків, які є об’єктом подальших досліджень, слід зазначити, що методи сегментації досліджувалися на вихідних зашумлених зображеннях, не використовуючи такі етапи попередньої обробки зображень, як шумовидалення та нормалізація яскравості.

ВИСНОВКИ

В результаті аналізу отриманих результатів видно, що запропонований метод виконує швидке зв'язне реалістичне знаходження контуру на зображеннях, в яких мало об'єктів. Збільшення кількості об'єктів на зображенні призводить до збільшення часу обчислень контуру. Для деяких зображень досягається прискорення вдвічі, в порівнянні з OpenCV реалізацією.

У роботі повністю дотримано вимог, що ставились у завданні – розглянуто існуючі методів оконтурювання та обґрунтовано необхідність та актуальність дослідження методів виділення контуру, через їх поширене використання в інтелектуальних системах аналізу вихідних зображень, запропоновано власний методу виділення контуру на вихідному зображенні шляхом удосконалення існуючого завдяки завдяки розпаралелюванню етапу конвертації до відтінків сірого та роботи ковзаючи вікон, а також порівняно реалізацій розроблених та існуючих методів на центральному та графічному процесорах.

Серед недоліків, які є об’єктом подальших досліджень, слід зазначити, що методи сегментації досліджувалися на вихідних зашумлених зображеннях, не використовуючи такі етапи попередньої обробки зображень, як шумовидалення та нормалізація яскравості.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Senthilkumaran, N. Edge Detection Techniques for Image Segmentation – A Survey / N. Senthilkumaran, R. Rajesh // Proceedings of the International Conference on Managing Next Generation Software Applications. – 2008. – P.749-760.

2. Vincent L. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms //IEEE transactions on image processing. – 1993. – Т. 2. – №. 2. – С. 176-201.

3. Сойфер, В.А. Методы компьютерной обработки изображений / В.А. Сойфер // М.: ФИЗМАТЛИТ. – 2003. – 192 с.

4. Огнев, И. В. Обработка изображений методами математической морфологии в ассоциативной осцилляторной среде / И.В. Огнев, Н.А. Сидорова // Технические науки. Информатика и вычислительная техника. – №4. – 2007.

5. Haralick R. M., Sternberg S. R., Zhuang X. Image analysis using mathematical morphology //IEEE transactions on pattern analysis and machine intelligence. – 1987. – №. 4. – С. 532-550.

6. Erhardt R. et al. FAZYTAN: a system for fast automated cell segmentation, cell image analysis and feature extraction based on TV-image pickup and parallel processing //Analytical and quantitative cytology. – 1980. – Т. 2. – №. 1. – С. 25-40.

7. Onoe M. (ed.). Real-Time Parallel Computing: Image Analysis. – Springer Science & Business Media, 2012.

8. Henri E Bal Jennifer G Steine and Andrew S Tanenbaum Programming Languages for Distributed Com puting Systems ACM Computing Surveys September 1989.

9. Robert D Blumofe Christopher F Joer Bradley C Kuszmaul Charles E Leiserson Keith H Randal and Yuli Zhou Cilk an encient mult

threaded ru time system In Proc th ACM SIGPLAN Symp Principles and Practice of Parallel Programming pages 207-216, 1995.

10. Mérigot A., Zavidovique B. Image analysis on massively parallel computers: an architectural point of view //Parallel Image Processing. – 1992. – С. 177-183.

11. Ducourthial B., Merigot A. Parallel asynchronous computations for image analysis //Proceedings of the IEEE. – 2002. – Т. 90. – №. 7. – С. 1218-1229.

12. R Bisseling Parallel Scient c Computation A Structured Approach using BSP and MPI Oxford University Press, 2004

13. NVIDIA. 2007. CUDA Programming Guide 1.1; http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf.

14. Stratton, J.A., Stone, S. S., Hwu, W. W. 2008. M-CUDA:An efficient implementation of CUDA kernels on multicores.IMPACT Technical Report 08-01, University of Illinois at Urbana-Champaign, (February).

15. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P. Brook for GPUs: Stream computing on graphics hardware. 2004.Proceedings of SIGGRAPH (August): 777-786; http://doi.acm.org/10.1145/1186562.1015800.

16. Stone, S.S., Yi, H., Hwu, W.W., Haldar, J.P., Sutton,B.P., Liang, Z.-P. 2007. How GPUs can improve the quality of magnetic resonance imaging. The FirstWorkshop on General-Purpose Processing on Graphics Processing Units (October).

17. Фисенко, В.Т. Компьютерная обработка и распознавание изображений. Учебное пособие / В.Т. Фисенко, Т.Ю. Фисенко // Электрон. дан. – СПб.: НИУ ИТМО. – 2008. – 192 с.

18. Levialdi S. Integrated Technology for Parallel Image Processing. – 1985.

19. Haralick R. M., Sternberg S. R., Zhuang X. Image analysis using mathematical morphology //IEEE transactions on pattern analysis and machine intelligence. – 1987. – №. 4. – С. 532-550.

20. Безуглов Д.А., Кузин А.П. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ВЫДЕЛЕНИЯ КОНТУРОВ ИЗОБРАЖЕНИЙ НА ФОНЕ ШУМА // Современные проблемы науки и образования. – 2015. – № 2-2.

21. Фурман, Я.А. Введение в контурный анализ. Приложения к обработке изображений и сигналов / Я.А. Фурман, А.В. Кревецкий, А.К. Передреев, А.А. Роженцов, Р.Г. Хафизов, И.Л. Егошина, А.Н. Леухин // 2-е изд., испр. – М.: Физматлит. – 2003. – 592 с.

22. Papari, G. Edge and line oriented contour detection: State of the art. Image and vision computing / G. Papari, N. Petkov // 2011. – P. 79-103.

23. Gong Xin-Yi. An Overview of Contour Detection Approaches / Xin-Yi Gong, Hu Su, De Xu, Zheng-Tao Zhang, Fei Shen, Hua-Bin Yang // International Journal of Automation and Computing. – 2018. – Vol. 15, No 6. – P. 656-672.

24. Баранник, В.В. Анализ методов обнаружения границ объектов на изображениях и их классификация / В.В. Баранник, А.В. Яковенко, А.В. Власов // Сучасна спеціальна техніка. – 2012. – № 3. – С. 20-27.

25. Ding, Lijun. On the Canny edge detector. Pattern Recognition / Lijun Ding, Ardeshir Goshtasby // Vol. 34, Issue 3. – 2001. – P. 721-725.

26. Liu, Ruiyuan. Research on Improved Canny Edge Detection Algorithm / Ruiyuan Liu, Jian Mao // MATEC Web of Conferences. – Vol. 232, No 14. – 2018. – P. 1- 4.