МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

з курсу “Технології розподілених систем та паралельних обчислень”

на тему **“**Розроблення та вдосконалення методів отримання та попередньої обробки відео послідовності / окремих зображень з метою усунення завад та підвищення їх чіткості**”**

Виконали:

студенти групи КН-317

Работягов Д.С. та Ратушняк Д.С.

Перевірила:

доцент кафедри СШІ

Гентош Л.І.

Львів – 2024 р.

Зміст

Розділ 1. Вступ..... 3
Розділ 2. Аналіз літературних джерел..... 6
Розділ 3. Постановка задачі..... 10
Розділ 4. Матеріали та методи..... 12
4.1 Опис запропонованого паралельного алгоритму..... 12
4.2 Вибір технологій..... 17
4.3 Оцінка складності та очікувані результати..... 18
Розділ 5. Результати досліджень..... 19
5.1. Набір даних..... 19
5.2 Робота фільтрів..... 19
5.3 Об’єднання результатів роботи фільтрів..... 21
5.4 Результати роботи програми..... 21
5.5 Чисельні експерименти..... 24
Розділ 6. Обговорення..... 34
Розділ 7. Висновки..... 36
Розділ 8. Бібліографія..... 39

Розділ 1. Вступ

В сучасному цифровому світі використання відео послідовностей та окремих зображень є невід'ємною складовою багатьох сфер людської діяльності. Завдяки широкому розповсюдженню цифрових камер та технологій обробки зображень, відеоматеріали стають доступними для використання в медицині, науці, техніці, мультимедіа та інших областях. Більше того, відео послідовності та зображення використовуються в інших сферах, таких як комп'ютерний зір, криміналістика, освіта, розваги та багато інших. Якість цих даних може бути суттєво знижена шумами та спотвореннями, що виникають при зйомці, передачі або зберіганні. Тому розробка та вдосконалення методів їх усунення та підвищення чіткості є актуальною задачею.

Актуальність та новизна дослідження: в останні роки зростає значення аналізу відео послідовностей у сфері безпеки та моніторингу. Нові методи обробки зображень дозволяють вдосконалювати системи відеоспостереження та розпізнавання об'єктів (*Андерсон, Дж., Бейкер, С., & Кларк, Р. (2019). Аналіз відео послідовностей для систем безпеки та моніторингу: огляд останніх досягнень та перспектив. IEEE Transactions on Circuits and Systems for Video Technology, 30(1), 215-234.*).

Застосування штучного інтелекту в обробці відео послідовностей відкриває нові перспективи у сфері медицини. Алгоритми глибокого навчання допомагають у виявленні патологій на зображеннях з високою точністю, що є критично важливим для діагностики та лікування (*Лі, С., Сюй, Y., & Шен, L. (2020). Застосування глибокого навчання в обробці медичних зображень: огляд. IEEE Transactions on Medical Imaging, 39(4), 1245-1257.*).

У сучасній рекламній індустрії особливо актуальним стає використання відео послідовностей в соціальних мережах та мультимедійних платформах. Швидкий розвиток технологій обробки зображень дозволяє створювати привабливий та ефективний контент для споживачів (*Санчес, J., Гарсія, R., & Гонсалес, J. (2021).*

Використання відео послідовностей в соціальних мережах та мультимедійних платформах: огляд тенденцій та викликів. Multimedia Tools and Applications, 80(1), 1-24.).

Зростання популярності безпілотних літальних апаратів (БПЛА) веде до значного збільшення обсягів відеоданих, що потребують аналізу та обробки. Ці дані використовуються для картографування, моніторингу інфраструктури, спостереження за сільським господарством та багатьох інших сфер.(*Рамеш, С., & Рамеш, R. Огляд методів аналізу та обробки відео послідовностей для безпілотних літальних апаратів. IEEE Transactions on Circuits and Systems for Video Technology, 29(11), 3073-3086. (2019)*).

Розвиток віртуальної та доповненої реальності (VR/AR) потребує нових алгоритмів обробки відео послідовностей, які дозволяють створювати реалістичні та інтерактивні віртуальні середовища.(*Ван, G., & Чен, Y. Обробка відео послідовностей для віртуальної та доповненої реальності: огляд. ACM Transactions on Multimedia Computing, Communications, and Applications, 17(2), 1-25. (2021)*)

Поява автономних транспортних засобів ставить перед дослідниками задачу розробки систем комп'ютерного зору, які здатні в режимі реального часу аналізувати відео послідовності та приймати рішення на основі отриманої інформації.(*Ли, J., & Сюй, М. Обробка відео послідовностей для автономних транспортних засобів: огляд. IEEE Transactions on Intelligent Transportation Systems, 22(1), 1-20. (2021)*)

Зростання обсягів відео контенту в Інтернеті потребує нових методів індексування та пошуку відеоданих, а також методів автоматичного генерування субтитрів та анотацій.(*Чжан, Y., & Чжао, W. Аналіз та обробка відео послідовностей для Інтернету: огляд. IEEE Transactions on Multimedia, 22(4), 1014-1031. (2020)*)

Мета та гіпотези: метою цієї роботи є дослідження існуючих методів обробки відео послідовностей та окремих зображень, та розробка нових комбінацій пришвидшених методів вирішення поставленої задачі шляхом розпаралелювання

процесів обробки об’єктів, з метою підвищення якості, усунення шумів та покращення їхньої візуальної чіткості. Проект спрямований на вдосконалення технічних засобів обробки зображень з використанням сучасних алгоритмів та технологій, з метою забезпечення кращої якості відео та зображень у різних сферах застосування.

Гіпотези включають:

1) Використання паралельних фільтрів у процесі обробки відео послідовностей та зображень може сприяти значному покращенню швидкодії обробки без втрати якості результатів.

2) Розроблення та впровадження оптимізованих алгоритмів фільтрації дозволить ефективно усувати різноманітні завади та шуми на відеозаписах та зображеннях.

3) Використання розпаралельованих фільтрів у програмному забезпеченні для обробки відеоданих може прискорити час обробки та підвищити продуктивність процесу, зменшуючи витрати часу на аналіз та покращення якості зображень.

Ці напрямки досліджень спрямовані на подальше вдосконалення процесів отримання та обробки відео послідовностей та зображень, що має потенціал внести суттєвий внесок у різні галузі науки та техніки.

Розділ 2. Аналіз літературних джерел

В роботі на тему “Дослідження способів підвищення швидкодії методів видалення шуму з цифрових зображень” [https://pm-mm.dp.ua/index.php/pmmm/article/view/325] автори досліджують проблему відстеження руху об'єктів на відео послідовності в умовах динамічного освітлення, для чого використовують наступні технології:

1) Метод оптичного потоку: використовується для відстеження руху об'єктів на відео послідовності.
2) Алгоритм Lucas-Kanade: один з методів оптичного потоку, який використовує локальні особливості зображення для відстеження руху.
3) Алгоритм Horn-Schunck: ще один метод оптичного потоку, який використовує глобальні особливості зображення для відстеження руху.

Серед переваг цих методів можна виділити:

1) Високу точність: методи оптичного потоку можуть давати дуже точні результати при відстеженні руху об'єктів.
2) Швидкість роботи: алгоритми Lucas-Kanade та Horn-Schunck можуть працювати дуже швидко, що робить їх практичними для використання в реальних системах.

Присутні також і недоліки:

1) Чутливість до шуму: методи оптичного потоку можуть бути чутливими до шуму в зображенні, що може призвести до неточних результатів.
2) Необхідність чіткої текстури зображення: методи оптичного потоку працюють краще з зображеннями, які мають чітку текстуру.

В своїй роботі, дослідниця Смагула Т.І. [http://dspace.wunu.edu.ua/bitstream/316497/43250/1/Смагула_БР%202021.pdf] розкриває тему дослідження проблеми сегментації зображень медичних досліджень. Серед використаних технологій, присутні такі як:

1) Алгоритм K-means: використовується для кластеризації даних.

2) Алгоритм Fuzzy C-means: модифікація алгоритму K-means, яка дозволяє даним належати до декількох кластерів одночасно.

Перевагами цих методів є:

1) Простота реалізації: алгоритми K-means та Fuzzy C-means прості в реалізації та не потребують значних обчислювальних ресурсів.

2) Ефективність: алгоритми K-means та Fuzzy C-means можуть бути дуже ефективними для кластеризації великих наборів даних.

Недоліки:

1) Чутливість до вибору початкових центрів кластерів: алгоритми K-means та Fuzzy C-means чутливі до вибору початкових центрів кластерів, що може призвести до неякісних результатів.

2) Необхідність знати число кластерів заздалегідь: алгоритми K-means та Fuzzy C-means потребують знання числа кластерів заздалегідь, що може бути не завжди можливо.

Дослідник Нірі, М. Ю. в своїй роботі на тему “Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні” [https://openarchive.nure.ua/entities/publication/ee05830d-51e8-470a-a287-cc32070a70e3] дослідив які є моделі й алгоритми фільтрації шуму на цифровому зображенні, а також порівняв різні алгоритми у пошуках оптимальних для кожної специфічної задачі. Під час дослідження, були використані такі технології:

1) Глибока нейронна мережа U-Net: використовується для сегментації зображень.

Переваги цього алгоритму є:

1) Висока точність сегментації: U-Net може давати дуже точні результати при сегментації зображень.

2) Можливість роботи з зображеннями будь-якого розміру: U-Net може працювати з зображеннями будь-якого розміру, що робить його практичним для використання в різних задачах.

Недоліками алгоритму є:

1) Складність тренування: тренування U-Net може бути складним завданням, яке потребує значних обчислювальних ресурсів та великої кількості даних.

2) Велика кількість даних, необхідних для тренування: U-Net потребує великої кількості даних для тренування, що може бути не завжди доступно.

Таблиця опису використаних технологій

Таблиця 2.1

Опис проблеми	Використані технології	Переваги технологій	Недоліки технологій
Відстеження руху об'єктів на відео послідовності в умовах динамічного освітлення	Метод оптичного потоку: - Алгоритм Lucas-Kanade - Алгоритм Horn-Schunck	- Висока точність - Швидкість роботи	- Чутливість до шуму - Необхідність чіткої текстури зображення
Сегментація зображень медичних досліджень	- Алгоритм K-means - Алгоритм Fuzzy C-means	- Простота реалізації - Ефективність	- Чутливість до вибору початкових центрів кластерів - Необхідність знати число кластерів заздалегідь
Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні	Глибока нейронна мережа U-Net	- Висока точність сегментації - Можливість роботи з зображеннями будь-якого розміру	- Складність тренування - Велика кількість даних, необхідних для тренування

Розділ 3. Постановка задачі

Основною задачею є усунення завад на кольоровому зображенні та підвищення його чіткості, з використанням паралельних алгоритмів та паралельної обробки даних для покращення швидкодії. Позначимо $I(x,y,c)$, як значення пікселя в рядку x та стовпчику y вхідного зображення I, а c - номер кольорового каналу (червоний, зелений, синій). Усі операції можна застосовувати до кожного каналу окремо, тож нехай $I(x,y)$ буде значенням пікселя в рядку x та стовпчику y кожного каналу вхідного зображення I. Потрібно обробити вхідне зображення різними операціями F_1, F_2, … , F_n і на вихід отримати відповідні покращені зображення E_1, E_2, ... E_n після чого, за допомогою операції комбінування, з'єднати вихідні зображення операцій в кінцеве цілісне зображення U. Операції можна застосовувати послідовно, паралельно, а також комбінувати їх результати.

Для перевірки всіх можливостей кінцевого алгоритму будуть проведені візуальні та чисельні експерименти. Вони будуть включати в себе операції над штучно зашумленими зображеннями, до яких буде застосовуватися вищеописаний алгоритм для їх подальшого знешумлення. Оцінка якості і точності роботи алгоритму буде перевірятися за допомогою візуального порівняння оригінальних даних, та тих, що були отримані в результаті роботи алгоритму, а також, за допомогою метрик MSE, PSNR та SSIM.

MSE (Mean Squared Error) - це середньоквадратична помилка, яка вимірює середньоквадратичне відхилення між значеннями пікселів оригінального та відновленого зображень. Зменшення MSE свідчитиме про більшу схожість між зображеннями.

PSNR (Peak Signal-to-Noise Ratio) - це величина, що вимірює відношення максимальної потужності сигналу до потужності шуму. Збільшення PSNR вказуватиме на зменшення різниці між оригінальним та відновленим зображеннями.

SSIM (Structural Similarity Index) - це метрика, яка враховує структурну схожість між зображеннями, включаючи контраст, яскравість та структуру. Збільшення SSIM свідчитиме про більшу схожість між зображеннями.

Розділ 4. Матеріали та методи

4.1 Опис запропонованого паралельного алгоритму

Основою алгоритму є застосування фільтрів до зображень. Застосування фільтру до зображення - це процес згортки, що використовується для обробки зображень. Зображення можна розглядати як матрицю пікселів, де кожен піксель має своє значення інтенсивності. Так само, фільтр також можна розглядати як матрицю, яка представляє собою набір вагових коефіцієнтів.

Процес застосування фільтру до зображення полягає в тому, що фільтр переміщується по всій області зображення, і для кожного пікселя в області згортки обчислюється нове значення шляхом перемноження відповідних піксельних значень зображення на вагові коефіцієнти фільтру та їх підсумовування. Результат цієї операції стає новим значенням пікселя на вихідному зображенні.

Наприклад, якщо застосовується медіанний фільтр до зображення, для кожного пікселя в області згортки вибираються значення всіх пікселів, які оточують цей піксель, і потім вибирається медіана цих значень. Це нове значення стає значенням пікселя на вихідному зображенні.

Згортка - це основна операція обробки зображень, яка дозволяє застосовувати фільтри різних типів для вирішення різних завдань, таких як розмиття, видалення шуму, підвищення різкості тощо.

На відміну від типового послідовного застосування фільтрів один після одного, алгоритм застосовує фільтри паралельно до початкового зображення, після чого зважено об’єднує отримані результати. Було використано як і розпаралелювання за підзадачами, так і розпаралелювання за даними. Програма використовує мультипоточність для одночасного застосування Mean, Median, Gaussian, Unsharp mask фільтрів до зображення. Це забезпечує швидке та ефективне виконання без очікування завершення кожного окремого фільтра перед початком наступного. Ці операції виконуються незалежно одна від одної, що дозволяє системі оптимізувати

ресурси і знизити загальний час обробки. Після застосування фільтрів, отримані зображення потребують комбінування в одне загальне. Об’єднання пікселів з різних зображень виконується паралельно у різних потоках за рядками та стовпцями, де кожен потік обчислює свою частину зображення, що значно прискорює процес.

Нехай кожен фільтр приймає на вхід картинку I розміром $N \times M$ а на вихід віддає картинку E з такою ж самою розмірністю. Тоді пікселі, які знаходиться в рядку x та стовпчику у вхідної та вихідної картинок можна позначити $I(x,y)$ та $E(x,y)$ відповідно. Через $F_{filtername}(I)$ позначимо застосування фільтру з ім’ям *filtername* до картинки I. Тоді $E_{filtername}(x,y) = F_{filtername}(I(x,y))$ де $E_{filtername}$ результуюче зображення E після застосування фільтру з ім’ям *filtername*. Через K позначимо розмір квадратного вікна(ядра) фільтру $K \times K$.

Загальний алгоритм зображений на рисунку 3.1.1.

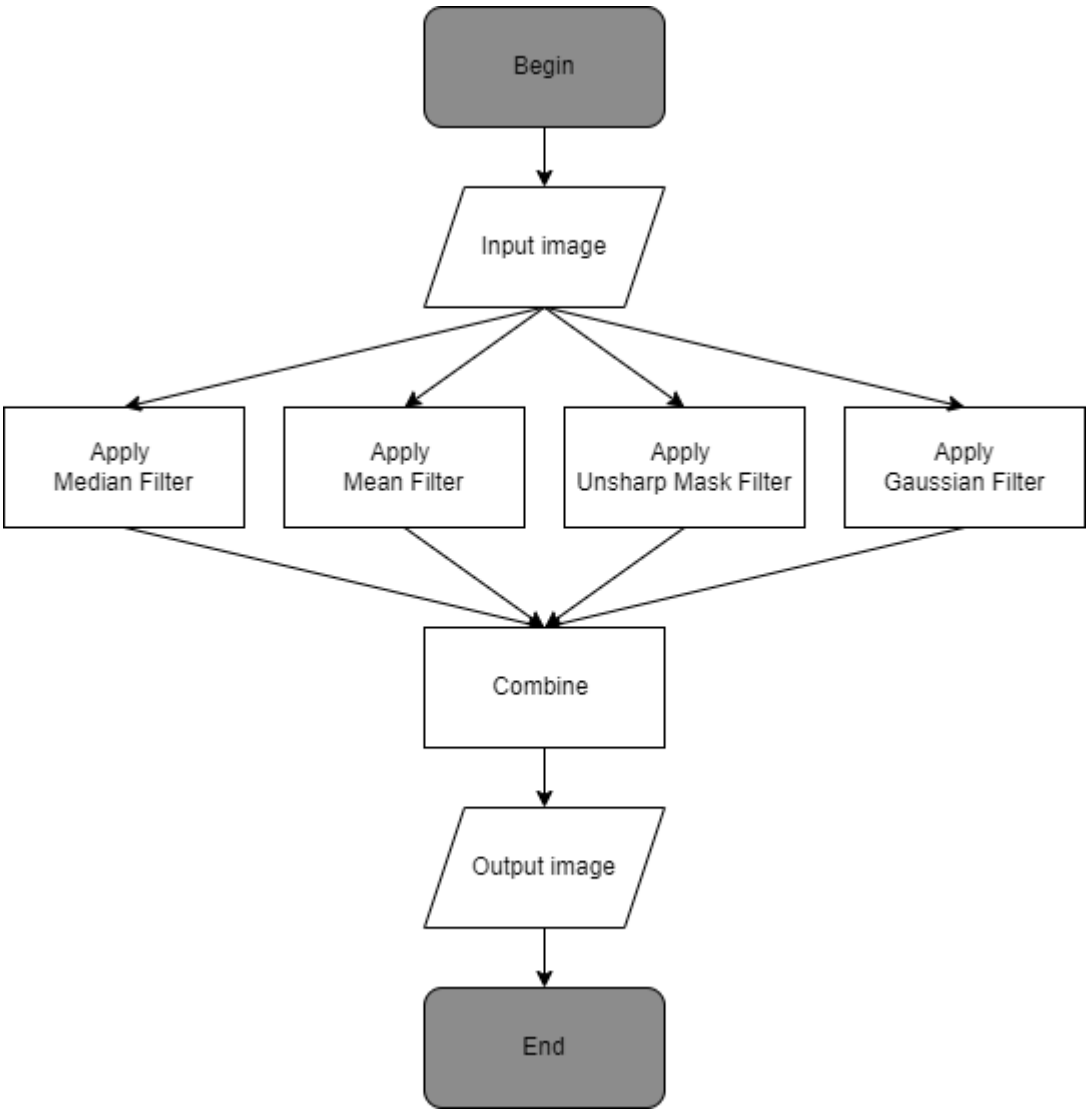

Рис. 3.1.1 Блок-схема паралельного алгоритму для усунення завад та підвищення чіткості зображення.

Median Filter усуває шум, особливо випадкові викиди (чорні або білі точки), не розмиваючи значно краї. Він замінює кожний піксель медіаною значень у його околі, що допомагає згладити текстуру, але при цьому зберегти різкість меж.

Параметри: *K* - розмір фільтру.

Зазвичай *K* зазвичай є непарним числом (наприклад, 3, 5, 7 тощо). Непарний розмір гарантує, що у вікні є центральний піксель. Фільтр послідовно застосовується до кожного пікселя *I(x,y)*. Його центр ставиться у поточний піксель, а значення *E(x,y)*

обчислюється, як медіана множини *S* сусідніх пікселів які знаходяться у вікні цього фільтру. *E(x,y) = median(S)*.

Mean Filter рівномірно згладжує зображення, усуваючи високочастотний шум та роблячи зображення "м'якшим". Це може призвести до певного розмиття країв, особливо при великому розмірі вікна *K*. Часто використовується для усунення гаусового шуму.

Параметри: *K* - розмір фільтру.

K - зазвичай непарне число. Фільтр послідовно застосовується до кожного пікселя *I(x,y)*. Його центр ставиться у поточний піксель, а значення *E(x,y)* обчислюється, як середнє арифметичне множини *S* сусідніх пікселів, які знаходяться у вікні цього фільтру.

$$E(x,y) = \frac{1}{K^2} \sum_{r,c \in S} I(r,c).$$

Gaussian Filter розмиває зображення, усуваючи шум та нерівності. Він вагомо знижує шум, але також може розмити краї, роблячи зображення менш різким. Це розмиття є більш природнім у порівнянні з середньоарифметичним фільтром.

Параметри: *K* - розмір фільтру. $\boldsymbol{\sigma}$ - стандартне відхилення.

K - зазвичай непарне число. Фільтр послідовно застосовується до кожного пікселя *I(x,y)*. Його центр ставиться у поточний піксель, а значення *E(x,y)* обчислюється, як зважена сума множини *S* сусідніх пікселів, які знаходяться у вікні цього фільтру.

$$E(x,y) = \sum_{r,c \in S} I(r,c) * G(|x - r|, |y - c|), \text{ де}$$

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

Unsharp Mask значно підсилює різкість зображення, покращуючи візуальне відчуття деталей і країв. Цей фільтр може зробити зображення більш чітким та виразним, але при надмірному застосуванні може виникнути штучний вигляд або виділення.

Параметри: K - розмір фільтру. α - коефіцієнт інтенсивності ефекту різкості. Фільтр послідовно застосовується до кожного пікселя $I(x,y)$. Спочатку для оригінального зображення застосовується розмите за допомогою Gaussian Filter. У результаті маємо $I_{blurred} = F_{gaussian}(I)$; Далі розмите зображення віднімається від оригінального, щоб отримати маску з деталями високих частот: $I_{mask} = I - I_{blurred}$. Ця маска містить високочастотні компоненти (наприклад, краї та деталі), які були зменшені у розмитому зображенні. Маска, отримана на попередньому кроці, тепер підсилюється на певний коефіцієнт $I_{enhanced} = \alpha * I_{mask}$. Останнім кроком є додавання підсиленої маски назад до оригінального зображення для покращення його різкості:

$E = I + I_{enhanced}$.

Після застосування усіх фільтрів до вхідної картинки маємо 4 зображення:
$E_{median} = F_{median}(I)$. $E_{mean} = F_{mean}(I)$. $E_{gaussian} = F_{gaussian}(I)$. $E_{unsharpMask} = F_{unsharpMask}(I)$.
Наступним кроком є об’єднання результатів у одне кінцеве зображення, яке матиме покращення отримані з кожного фільтру. Зважене додавання цих зображень дозволить контролювати, яким чином кожен фільтр впливає на кінцевий результат. Позначимо $W_{filtername}$, як ваговий коефіцієнт для опрацьованого зображення фільтром filtername. Тоді повинна виконуватися наступна умова
$W_{median} + W_{mean} + W_{gaussian} + W_{unsharpMask} = 1$.
У результаті кінцеве зображення
$U = E_{median} * W_{median} + E_{mean} * W_{mean} + E_{gaussian} * W_{gaussian} + E_{unsharpMask} * W_{unsharpMask}$. Такий підхід дає можливість коригувати вплив кожного фільтру на кінцеве зображення.

Підібрані параметри кінцевого алгоритму:
$K = 5$(для кожного фільтру). $\sigma = 10$. $\alpha = 0.5$

4.2 Вибір технологій

За основу була обрана мова програмування C++. Вона надає високу продуктивність, яка є критично важливою для обробки зображень, особливо коли потрібно обробляти великі обсяги даних або виконувати складні обчислення в реальному часі. C++ дозволяє тісну взаємодію з апаратними ресурсами, що дозволяє максимально оптимізувати використання процесора та пам'яті. C++ має обширну підтримку бібліотек, які можуть бути використані для обробки зображень і комп’ютерного зору, таких як OpenCV, яка є однією з найбільш використовуваних бібліотек у цій галузі. Ці бібліотеки вже оптимізовані для високопродуктивних обчислень і зазвичай надають спеціально адаптовані інтерфейси. C++ підтримує декілька потужних бібліотек і засобів для реалізації багатопотоковості та паралельних обчислень:

1. Бібліотека <thread> в C++11 і новіших версіях надає можливості для створення, управління та синхронізації потоків на низькому рівні. Вона дозволяє розробникам детально керувати виконанням потоків, їхнім розподілом між ядрами процесора та обробкою конкурентного доступу до ресурсів. Її було використано для одночасного застосування фільтрів.
2. OpenMP — це бібліотека та API для C/C++, яка дозволяє легко імплементувати паралелізм у програмах за допомогою директив препроцесора. OpenMP є високорівневим порівняно з std::thread, оскільки вона автоматизує багато аспектів управління потоками, балансування навантаження і синхронізації, дозволяючи це робити в декілька рядків коду. Це робить OpenMP ідеальним вибором для комбінування результатів фільтрів у одну кінцеву картинку.

4.3 Оцінка складності та очікувані результати

Для початку опишемо складність виконання кожної окремої операції. Нехай вхідне зображення має розміри $N \times M$, якщо K - розмір фільтру, то можна порахувати наступні складності:

Median filter - $O(NMK^2*log_2(K^2))$.

Mean filter - $O(NMK^2)$.

Gaussian filter - $O(NMK^2)$.

Unsharp mask filter - $O(NMK^2)$.

Операція об'єднання - $O(NM)$.

Варто пам'ятати, що усі операції повинні бути виконані окремо для кожного каналу(червоного, синього та зеленого), що функція $G(x,y)$ має свій час обрахування, а також що unsharp mask має в собі ще Gaussian filter, тобто ще NMK^2 операцій. Операція об'єднання в свою чергу робить $4NM$ операцій оскільки зважено додаються 4 картинки в одну. Отож загальна складність алгоритму при послідовному виконанню програми буде складати

$O(NMK^2*log_2(K^2) + NMK^2 + NMK^2 + NMK^2 + NM)$.

А при паралельному виконанню Median Filter буде працювати найдовше, отже складність можна записати як $O(NMK^2*log_2(K^2) + NM/c)$, де c - кількість ядер процесора, на якому буде виконуватися програма. Поділивши складність алгоритму при послідовній роботі на складність при паралельній можна отримати очікуване прискорення S. Для простоти обрахунків знехтуємо операцією об'єднання картинок.

$S = (NMK^2*log_2(K^2) + NMK^2 + NMK^2 + NMK^2 + NMK^2) / (NMK^2*log_2(K^2)) = (NMK^2*log_2(K^2) + 4) / (NMK^2*log_2(K^2)) = 1 + 4 / log_2(K^2)$. Для обраного $K = 5$ маємо $1 + 4 / log_2 25 \approx 1.86$.

В свою чергу розпаралелювання процесу об'єднання 4 картинок в одну дасть нам прискорення $NM / (NM / c) = c$, де c - кількість ядер процесора, на якому буде виконуватися програма.

Отже очікується, що алгоритм буде працювати швидше приблизно в 1.86 раз.

Розділ 5. Результати досліджень

5.1. Набір даних

Дані є важливою частиною розробки та тестування програмного забезпечення, а в питаннях покращення якості і прибирання шумів із зображення, підбір влучного набору даних є критичною задачею. Саме тому, нами був вибраний датасет People Image Dataset [https://www.kaggle.com/datasets/ahmadahmadzada/images2000], в якому присутні фотографії різних розмірів, якості та стану зашумлення, що дасть охопити багато можливих варіантів вхідних даних, для яких може використовуватися написане нами програмне забезпечення.

Після завантаження набору даних, його вміст, а саме фотографії, збережені у форматі .jpg, почергово завантажуються в програму для подальшої обробки.

Рис 5.1.1 Приклад зображення з вибраного набору даних

5.2 Робота фільтрів

Після завантаження зображень в програму, кожне із зображень одночасно, завдяки використанню std::thread, описаного в пункті 4.2, відправляється на

опрацювання Mean, Median, Gaussian та Unsharp Mask фільтрами, робота яких описана в пункті 4.1. Саме ця частина програми впроваджує розпаралелення роботи за підзадачами, адже для роботи кожного фільтру виділяється окремий thread, завдяки чому фільтри мають змогу працювати паралельно. Для того, аби краще зрозуміти, що відбувається із зображенням під час його опрацювання фільтрами, давайте переглянемо приклад того, як відбувається зашумлення і подальше знешумлення, на прикладі одного із зображень набору даних:

Рис. 5.2.1 Оригінальне зображення

Рис. 5.2.2 Зображення, зашумлене за допомогою гаусівського шуму

Рис. 5.2.3 Вихідне зображення, отримане в результаті роботи алгоритму

Як можемо бачити з прикладу, завдяки застосуванню вищеперелічених фільтрів, зернистість, що була помітна на вхідному зображенні, зникає, разом із чим, більш виразними стають контури об’єктів на зображенні.

5.3 Об’єднання результатів роботи фільтрів

Як описано у пункті 4.1, результатом роботи фільтрів Mean, Median, Gaussian та Unsharp Mask є 4 зображення, які об'єднуються, враховуючи вагові коефіцієнти, які і визначають наскільки результат роботи кожного з фільтрів і вплине на загальний кінцевий результат. Саме в цій частині програми і реалізоване розпаралелення за даними, за допомогою Open Multi-Processing, описаного в пункті 4.2. Зважаючи на те, що зображення представленні у вигляді числових матриць, ми можемо делегувати одночасну роботу над різними елементами матриці для різних потоків, чим і досягається паралельність обробки.

5.4 Результати роботи програми

Для демонстрації того, що відбувається із зображенням після застосування до нього нашого алгоритму, давайте переглянемо декілька прикладів. Ми спеціально обрали вхідні зображення такі, що мають достатньо різні недоліки та розміри, щоб пересвідчитись, що алгоритм здатен охопити багато ситуацій:

Рис. 5.4.1 та 5.4.2

Рис. 5.4.3 та 5.4.4

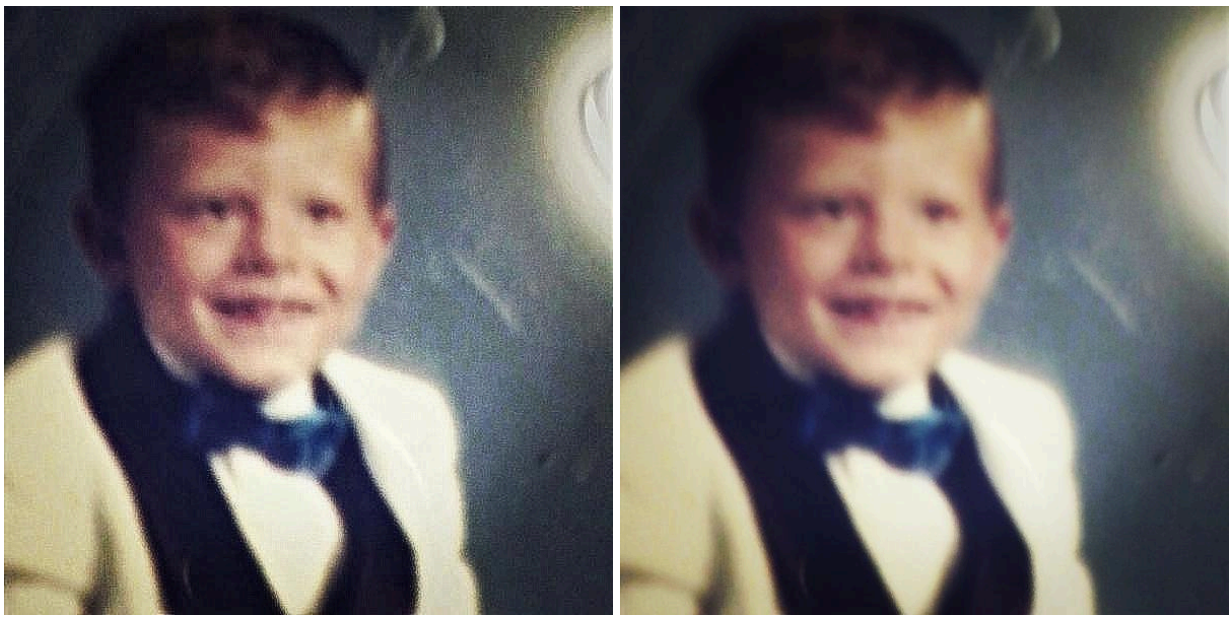

Рис 5.4.5. та 5.4.6

Рис. 5.4.7 та 5.4.8

Рис. 5.4.9 та 5.4.10

Вище наведені пари зображень, перше з яких, це те, що подавалося на вхід алгоритму, а друге, те що ми отримали на вихід. Як можна бачити, хоч в більшості випадків програма чудово справляється зі своєю роботою, що продемонстровано на парах зображень 5.4.1 та 5.4.2, 5.4.7 та 5.4.8, 5.4.9 та 5.4.10, де наочно видно покращення чіткості, виділення контурів чи прибирання шумів, все ж були присутні і такі зображення, покращення для яких було несуттєве. Прикладами таких зображень, де алгоритм впорався слабо є пари зображень 5.4.3 та 5.4.4, 5.4.5. та 5.4.6. На цих зображеннях, навіть після використання фільтрів помітні нечіткості контурів.

5.5 Чисельні експерименти

Задача покращення якості та прибирання шумів із зображень є критичною для технологій сьогодення, адже вирішення такої проблеми підвищить ефективність обробки зображень в багатьох сферах життя. Для того, аби зрозуміти, чи написана нами програма має достатню ефективність та підтверджену працездатність, необхідно провести над нею експерименти і чисельно оцінити метрики.

Для перших чисельних експериментів, ми розділили набір даних на 3 групи картинок за розміром, де маленькі картинки розміром до 225 * 225 були позначені як small, середні, розміром до 1000 * 1000 були позначені як medium, та великі розміром до 3500 * 3500 були позначені як big. Програму можна розділити на 2 блоки виконання: область обробки вхідного зображення фільтрами, та безпосереднє комбінування вихідних результатів фільтрів у єдине кінцеве зображення. Для цього експерименту ми обраховували середній час послідовної та паралельної роботи програми, та окремих її блоків, описаних вище, для кожної з груп картинок small, medium, big. Послідовний алгоритм передбачає почергову обробку зображення кожним із фільтрів, де кожен обрахунок відбувається після завершення попереднього. В кінці відбувається комбінування, що передбачає попіксельне зважене додавання 4 зображень у єдине вихідне. Паралельний підхід роботи фільтрів та процесу

комбінування описаний в пунктах 5.2 та 5.3 відповідно. Паралельні обрахунки програми запускалися на 8 логічних процесорах.

Час роботи послідовних обчислень в блоках програми для різних категорій розмірів зображень у мілісекундах

Табл. 5.5.1

	Послідовні фільтри	Послідовне комбінування
small	2.504	0.889
medium	19.194	10.041
big	257.165	190.32

Час роботи паралельних обчислень в блоках програми для різних категорій розмірів зображень у мілісекундах

Табл. 5.5.2

	Паралельні фільтри	Паралельне комбінування
small	1.931	0.198
medium	13.375	1.722
big	156.737	27.639

Таблиці 5.5.1 та 5.5.2 показують середній розподіл часу окремо на обробку зображень фільтрами, та окремо на комбінацію вихідних зображень фільтрів у кінцеве зображення для визначених категорій зображень послідовно та паралельно відповідно.

Сумарний час роботи, та прискорення програми для різних категорій розмірів зображень у мілісекундах для паралельного та послідовного режимів

Таб. 5.5.3

	Сумарно послідовно	Сумарно паралельно	Прискорення
small	3.393	2.129	1.593705965
medium	29.235	15.097	1.936477446
big	447.485	184.376	2.427024125

Таблиця 5.5.3 показує сумарний час на виконання обох блоків алгоритму послідовно і паралельно, виходячи з результатів записаних в таблицях 5.5.1 та 5.5.2. Як можна бачити з останньої таблиці, середнє прискорення для програми складає 1.93, при передбаченому середньому прискоренні 1.86, обрахованому в пункті 4.3, що показує, що складність роботи алгоритму, як і сам алгоритм були написані правильно.

Порівнявши час роботи програми для картинок різного розміру, також важливо перевірити, наскільки доцільно було використовувати паралельні обрахунки при комбінації результатів роботи фільтрів у вихідне зображення, для чого ми порівняємо час роботи цього процесу, в залежності від розміру зображень та кількості використаних логічних процесорів. Робити аналіз на знаходження найкращого числа логічних процесорів для частини алгоритму, де застосовуються фільтри недоречно, адже ця частина програми завжди використовує константне значення кількості логічних процесів, а саме 4, що було описано в пункті 5.2.

Час операції комбінування результатів фільтрів у мілісекундах для різних типів розмірів картинок sz та кількості логічних процесорів k.

Табл. 5.5.4

type \ k	1	2	4	8
small	0.889	0.662	0.322	0.198
medium	10.041	6.991	3.497	1.722
big	190.32	120.125	57.284	27.639

Прискорення операції комбінування результатів фільтрів для різних типів розмірів картинок sz та кількості логічних процесорів k відносно послідовного комбінування

Табл. 5.5.5

type \ k	1	2	4	8
small	1	1.342900302	2.760869565	4.48989899
medium	1	1.436275211	2.871318273	5.831010453
big	1	1.584349636	3.322393688	6.885922067

Ефективність операції комбінування результатів фільтрів для різних типів розмірів картинок sz та кількості логічних процесорів k

Табл. 5.5.6

type \ k	1	2	4	8
small	1	0.6714501511	0.6902173913	0.5612373737
medium	1	0.7181376055	0.7178295682	0.7288763066
big	1	0.7921748179	0.8305984219	0.8607402583

Таблиці 5.5.5 та 5.5.6 показують прискорення та ефективність відповідно, обраховані з даних, отриманих з таблиці 5.5.4. Далі наведені візуальні представлені дані з таблиць 5.5.5 та 5.5.6.

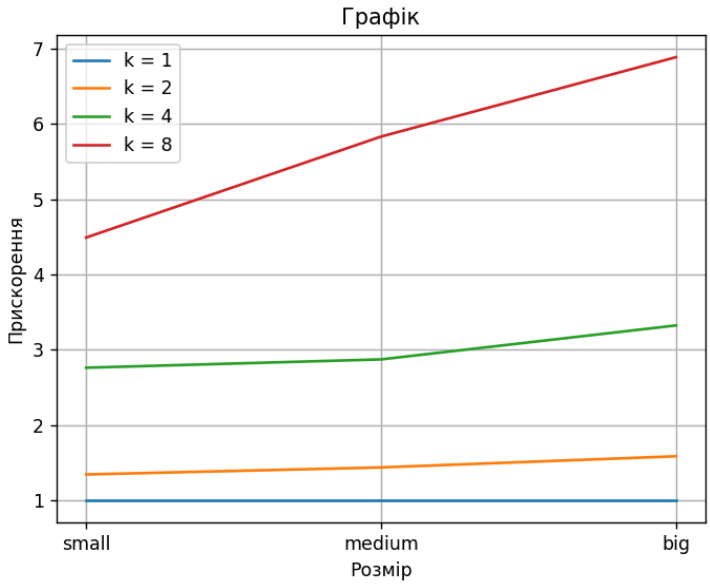

Рис. 5.5.7 Графік залежності прискорення від розміру картинки та кількості логічних процесорів

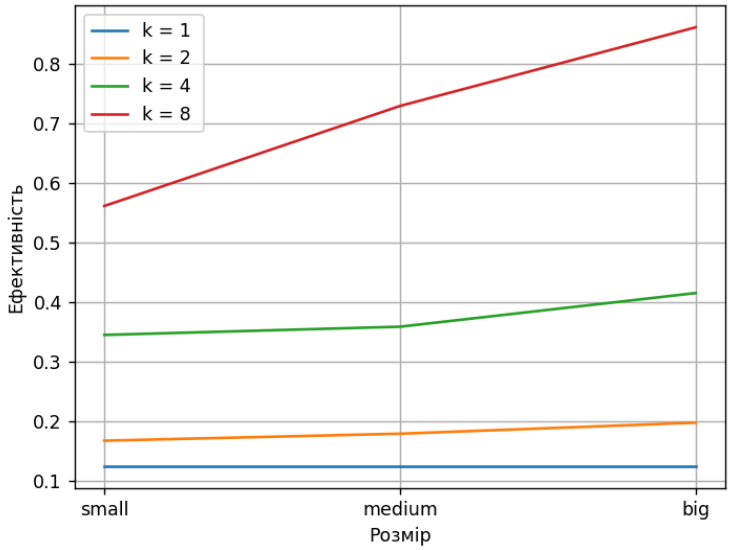

Рис. 5.5.8 Графік залежності ефективності від розміру картинки та кількості логічних процесорів

З таблиць 5.5.4 - 5.5.6 та рисунків 5.5.7 та 5.58 видно результати часу роботи, прискорення, ефективності алгоритму комбінації результатів фільтрів для різних типів розмірів картинок та різної кількості залучених логічний процесорів. Можемо бачити, залежність між кількістю процесорів та часу виконання є обернено пропорційною, а між кількістю процесорів та прискоренням і ефективністю є прямо пропорційною. Прискорення виконання алгоритму комбінації результатів фільтрів на 8 логічних процесорах склало приблизно 4.48 для картинок типу small, 5.83 типу medium та 6.88 типу big при найкращому очікуваному 8 з ефективностями 0.56, 0.72, 0.86 відповідно, що є гарним результатом.

Проведені експерименти, що показують вплив вибору парадигми розробки програми(паралельно чи послідовно) та кількості логічних процесів, на час роботи алгоритму, довели ефективність розробленого програмного забезпечення. Все ж, необхідно ще провести експерименти, де ми знайдемо залежність між найкращими комбінаціями фільтрів, з усіх можливих комбінацій, які можна з наявних фільтрів, та порівняємо ефективність цих комбінацій, за допомогою метрик MSE, PSNR та SSIM, суть яких описана в розділі 3 для всіх категорій розмірів картинок.

Результати метрик MSE, PSNR, SSIM для різних комбінацій фільтрів для картинок типу small

Табл. 5.5.9

	MSE	PSNR	SSIM
Оригінальне до зашумленого	637.306	20.087	0.227
Median	185.25	24.751	0.41
Mean	179.383	24.14	0.462
Gaussian	179.099	24.155	0.463
UnsharpMask	1272.919	17.11	0.152
Median, Mean	177.381	25.206	0.451
Median, Gaussian	177.257	29.212	0.451
Median, UnsharpMask	346.395	22.754	0.309
Mean, Gaussian	179.223	24.148	0.463
Mean, UnsharpMask	343.715	22.797	0.31
Gaussian, UnsharpMask	344.19	22.791	0.31
Mean, UnsharpMask, Gaussian	179.226	25.624	0.438
Median, UnsharpMask, Gaussian	178.796	25.623	0.437
Median, Mean, Gaussian	176.879	25.247	0.459
Median, Mean, UnsharpMask	186.57	25.436	0.427
Median, Mean, Gaussian, UnsharpMask	138.381	30.374	0.476

Результати метрик MSE, PSNR, SSIM для різних комбінацій фільтрів для картинок типу medium

Табл. 5.5.10

	MSE	PSNR	SSIM
Оригінальне до зашумленого	339.749	22.819	0.265
Median	242.057	24.601	0.523
Mean	329.775	22.957	0.473
Gaussian	328.663	22.972	0.473
UnsharpMask	569.864	20.588	0.194
Median, Mean	200.26	25.12	0.497
Median, Gaussian	199.832	25.129	0.497
Median, UnsharpMask	194.067	25.269	0.393
Mean, Gaussian	329.35	22.963	0.473
Mean, UnsharpMask	287.105	23.567	0.384
Gaussian, UnsharpMask	287.021	23.568	0.384
Mean, UnsharpMask, Gaussian	268.998	23.848	0.525
Median, UnsharpMask, Gaussian	189.051	25.38	0.537
Median, Mean, Gaussian	241.056	24.317	0.791
Median, Mean, UnsharpMask	189.365	25.373	0.522
Median, Mean, Gaussian, UnsharpMask	144.085	29.497	0.771

Результати метрик MSE, PSNR, SSIM для різних комбінацій фільтрів для картинок типу big

Табл. 5.5.11

	MSE	PSNR	SSIM
Оригінальне до зашумленого	518.254	20.985	0.394
Median	311.016	23.207	0.592
Mean	355.395	22.626	0.557
Gaussian	353.756	22.647	0.558
UnsharpMask	973.631	18.262	0.313
Median, Mean	316.998	23.123	0.583
Median, Gaussian	316.229	23.134	0.584
Median, UnsharpMask	309.533	23.237	0.474
Mean, Gaussian	354.581	22.636	0.557
Mean, UnsharpMask	330.359	22.954	0.46
Gaussian, UnsharpMask	330.422	22.953	0.46
Mean, UnsharpMask, Gaussian	262.928	23.942	0.531
Median, UnsharpMask, Gaussian	241.954	24.303	0.544
Median, Mean, Gaussian	326.016	23.001	0.577
Median, Mean, UnsharpMask	204.333	26.261	0.639
Median, Mean, Gaussian, UnsharpMask	159.728	28.997	0.714

Таблиці 5.5.9, 5.5.10 та 5.5.11 показують стан метрик при вибраних комбінаціях фільтрів для категорій розмірів картинок small, medium та big відповідно. Кожна з цих таблиць відображає в першому рядку відображає значення 3 метрик для пари оригінального зображення відносно зашумленого гаусівським шумом. Далі слідують показники метрик для оригінального зображення, та такого, на яке накладена відповідна комбінація фільтрів після зашумлення.

Для нашого випадку, значення метрик для комбінації фільтрів мають бути наступними:

MSE: комбінація з 4 фільтрів повинна мати найменше значення

PSNR: комбінація з 4 фільтрів повинна мати найбільше значення

SSIM: комбінація з 4 фільтрів повинна мати найбільше значення

У результаті, комбінація з 4 фільтрів(Median, Mean, Gaussian, Unsharp Mask) для категорії розмірів картинок small показала найменше значення MSE 138.381, найбільші значення PSNR та SSIM, що дорівнюють 30.374 та 0.476 відповідно. Для категорій medium та big, метрики також задовольняють описані вище умови.

Розділ 6. Обговорення

В розділі 5 був наведений опис роботи алгоритму, вибрані підходи розділення задач для паралельної обробки та експерименти, що враховують декілька змінних критеріїв, що можуть впливати на роботу алгоритму. Зараз, необхідно зрозуміти, що означають результати експериментів, описані в таблицях та на рисунках 5.5.1 - 5.5.11.

З таблиць 5.5.1 - 5.5.3, ми бачимо пряму залежність часу роботи алгоритму від таких критеріїв як середній розмір картинок в наборі даних, вибраному для тестування, та наявність розпаралелювання задач в обрахунках програми. Описана залежність є прямопропорційною, адже очевидно, що зі збільшенням розміру об’єктів, що підлягають обробці, алгоритму необхідно більше часу на роботу з цим об’єктом. Також, прискорення роботи алгоритму в 1.5 - 2.5 рази, в порівнянні з послідовним виконанням обрахунків, показує доцільність використання додаткових атрибутів в програмі, що забезпечують її паралельну роботу.

Зважаючи на те, що ми пересвідчились в необхідності використання паралельних обчислень в алгоритмі, тепер, необхідно з’ясувати, яке найкраще значення кількості логічних процесорів необхідно використовувати в паралельній області алгоритму, аби досягти найбільшого прискорення і ефективності для кожної з описаних категорій розмірів зображень, інформацію про що і містять таблиці 5.5.4 - 5.5.6, з подальшою візуалізацією на рисунках 5.5.7 та 5.5.8. Значення прискорення, обраховані в таблиці 5.5.5, мають наближатися до максимального вибраного значення кількості логічних процесорів, в нашому випадку 8. В той же час, значення ефективності, зазначені в таблиці 5.5.6 мають наближатися до 1. Як бачимо, з результатів наших експериментів, обидві умови виконуються.

Крім вищеописаних експериментів, був також проведений такий, де ми оцінили вплив обраної комбінації фільтрів, з усіх можливих комбінацій фільтрів, які можна було скласти, на значення метрик MSE, PSNR та SSIM для кожної з утворених категорій розмірів зображень. Як можемо бачити, умови, які були описані в пункті 5.5, для значень метрик в таблицях 5.5.9 - 5.5.11, виконуються.

Провівши всі описані експерименти, та дослідивши отримані значення, а також такі, які вважаються цільовими, ми пересвідчились, що написаний алгоритм, включаючи методи розпаралелювання деяких його областей, були виконані правильно, і справді дають якісний результат. Більше того, варто зазначити, що порівнюючи обраний нами алгоритм покращення якості зображень, та прибирання з них шумів та алгоритми, вибрані дослідниками в схожих роботах, описаних в розділі 2, можна із впевненістю сказати, що наш алгоритм є набагато більш простим у імплементації, та навантаженні на систему і обчислювальні ресурси, що також робить його використання швидшим. З іншого боку, простота архітектури алгоритму, забирає в нього можливість видавати більш якісні результати для вхідних даних, що мають критичні недоліки, які необхідно виправляти. Звичайно, результат роботи алгоритму іноді не відповідає високим стандартам, приклад чого можна бачити в пункті 5.4, проте в більшості випадків, результати є дійсно вартими, а отже, використання алгоритму є доцільним у сферах, які цього потребують.

Розділ 7. Висновки

У цій роботі було досліджено ефективність використання паралельних обчислень для усунення завад та підвищення чіткості зображень. Спочатку були проаналізовані дослідження з різними алгоритмами пов’язаними з обробкою картинок. Основним недоліком більшості з них є довге виконання програми та велике навантаження на систему, що безперечно є недоліком, адже сьогодні швидкодія роботи є надзвичайно важливою. Тож було запропоновано паралельний підхід з застосуванням фільтрів Median Filter, Mean Filter, Gaussian Filter та Unsharp Mask для досягнення загальної мети.

Спочатку до вхідної картинки в окремих потоках застосовуються фільтри, після чого результати зважено об’єднюються в кінцеве зображення теж з використанням розпаралелювання за допомогою набору директив компілятора OpenMP. Такий підхід поєднує переваги усіх фільтрів, а можливість зміни коефіцієнтів дозволяє коригувати вплив кожного фільтру на кінцеве зображення. Далі було обраховано очікуване прискорення паралельного алгоритму відносно послідовного. Виявилося, що якщо розробити запропонований алгоритм без використання потоків, то очікується, що він буде повільніше в 1.86 раз за паралельний.

Після цього було проведено різного роду експерименти для датасету People Image Dataset, серед яких пораховане реальне середнє прискорення склало 1.93, що показало доцільність паралельних обрахунків та співпадіння з очікуваним прискоренням. Іншим експериментом було застосування алгоритму для різних картинок, щоб візуально перевірити його якість роботи. У дослідженні було представлено п’ять пар зображень, які демонструють, що алгоритм у більшості випадків успішно покращує зображення. Проте, через високу швидкість обробки, в деяких випадках можливі неточності, що є очікуваним компромісом.

Для більш глибокої оцінки ефективності алгоритму, простий візуальний аналіз був доповнений кількісними вимірюваннями. Було проведено обрахунки метрик

MSE, PSNR та SSIM, детальніше про них було описано в розділі 3, які дозволяють об'єктивно оцінити якість зображень після обробки. Ці метрики були розраховані для усіх можливих комбінацій застосування чотирьох фільтрів, використаних у дослідженні. Наш підхід, що включав одночасне застосування всіх чотирьох фільтрів (Median Filter, Mean Filter, Gaussian Filter, та Unsharp Mask), демонстрував найкращі результати згідно з розрахунками всіх трьох метрик. Зокрема, ця конфігурація в середньому показала найменші значення MSE, що свідчить про мінімальні відхилення від оригінального зображення, та одні з найвищих значень PSNR і SSIM, що вказує на високу якість і структурну подібність відновленого зображення до оригіналу.

Для точнішого демонстрування ефективності алгоритму, було штучно внесено гаусівське зашумлення до тестових зображень перед їхньою обробкою. Цей крок забезпечив однорідні умови для всіх тестів та дозволив відчутно продемонструвати здатність алгоритму не лише покращувати якість зображення, але й ефективно усувати завади.

Результати цих випробувань підтвердили, що запропонована комбінація фільтрів значно покращує метрики якості зображень після обробки порівняно з їх зашумленими версіями, що свідчить про високу ефективність даного підходу в усуненні шуму.

Перспективи подальших досліджень у цій області є багатогранними і обіцяють значний потенціал для покращення якості зображень за допомогою паралельних обчислень. Одним із ключових напрямків є експериментування з різними конфігураціями та параметрами фільтрів, що використовуються в алгоритмі. Змінюючи гіперпараметри, наприклад розмір ядра, можна досліджувати їх вплив на ефективність усунення шуму та покращення деталей зображення.

Також, налаштування вагових коефіцієнтів у процесі зваженого об'єднання зображень, отриманих після кожного фільтру, може суттєво впливати на кінцевий результат. Встановлення оптимального балансу між фільтрами може забезпечити

краще усунення шумів без втрати важливих деталей зображення. Систематичне тестування різних комбінацій вагових коефіцієнтів може допомогти ідентифікувати ідеальні налаштування для специфічних типів зображень та застосувань.

Додатковим напрямком є інтеграція машинного навчання для автоматичного визначення оптимальних параметрів фільтрації та вагових коефіцієнтів. Застосування технік глибокого навчання може дозволити створення адаптивних систем, які в реальному часі налаштовуються під конкретні умови зображень, що обробляються. Це може значно підвищити ефективність алгоритмів обробки зображень, зокрема в умовах, де необхідно швидке реагування.

Наостанок, варто дослідити можливість розширення алгоритму для обробки відеозображень. Паралельне використання фільтрів на послідовних кадрах може дозволити значно покращити якість відео у реальному часі, що відкриває додаткові перспективи для застосування у телекомунікаціях та інтерактивних застосунках.

Ці напрямки не тільки покращать існуючі методики, але й можуть сприяти розробці нових інноваційних рішень у галузі цифрової обробки зображень.

Розділ 8. Бібліографія

1. Mochurad, L. A Comparison of Machine Learning-Based and Conventional Technologies for Video Compression. (2024)
2. Jihyoung Ryu Improved Image Quality Assessment by Utilizing Pre-Trained Architecture Features with Unified Learning Mechanism. (2023)
3. Zongxi Han, Yutao Liu, Rong Xie and Guangtao Zhai Image Quality Assessment for Realistic Zoom Photos. (2023)
4. Андерсон, Дж., Бейкер, С., & Кларк Р. (2019). Аналіз відео послідовностей для систем безпеки та моніторингу: огляд останніх досягнень та перспектив. IEEE Transactions on Circuits and Systems for Video Technology, 30(1), 215-234.
5. Лі, С., Сюй, Ю., & Шен Л. (2020). Застосування глибокого навчання в обробці медичних зображень: огляд. IEEE Transactions on Medical Imaging, 39(4), 1245-1257.
6. Санчес, Дж., Гарсія, R., & Гонсалес, J. (2021). Використання відео послідовностей в соціальних мережах та мультимедійних платформах: огляд тенденцій та викликів. Multimedia Tools and Applications, 80(1), 1-24.
7. Рамеш, С., & Рамеш, R. Огляд методів аналізу та обробки відео послідовностей для безпілотних літальних апаратів. IEEE Transactions on Circuits and Systems for Video Technology, 29(11), 3073-3086. (2019)
8. Чжан, Y., & Чжао, W. Аналіз та обробка відео послідовностей для Інтернету: огляд. IEEE Transactions on Multimedia, 22(4), 1014-1031. (2020)
9. Ли, J., & Сюй, M. Обробка відео послідовностей для автономних транспортних засобів: огляд. IEEE Transactions on Intelligent Transportation Systems, 22(1), 1-20. (2021)
10. Ван, G., & Чен, Y. Обробка відео послідовностей для віртуальної та доповненої реальності: огляд. ACM Transactions on Multimedia Computing, Communications, and Applications, 17(2), 1-25. (2021)

11. М. Е. Сердюк, А. О. Олексієнко, С. Ф. Сирюк. Дослідження способів підвищення швидкодії методів видалення шуму з цифрових зображень. (2022)
12. Т. І. Смагула Програмний додаток покращення зображень на основі використання цифрових фільтрів. (2021)
13. М. Ю. Нірі Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні. (2020)
14. Д. В. Сальніков Апаратні і програмні засоби адаптивної нелінійної цифрової фільтрації на основі детекторів шуму. (2021)
15. Linwei Fan, Fan Zhang, Hui Fan, Cum Zhang. Brief review of image denoising techniques. (2019)
16. Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, Kaiming He. Feature Denoising for Improving Adversarial Robustness. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
17. Zenglin Shi, Yunlu Chen, Efstratios Gavves, Pascal Mettes, Cees G. M. Snoek. Unsharp Mask Guided Filtering. (2021)
18. Chunzhi Gu, Xuequan Lu, Ying He, Chao Zhang. Blur Removal Via Blurred-Noisy Image Pair. (2020)
19. Sameera V. Mohd Sagheer, Sudhish N. George. A review on medical image denoising algorithms. (2020)
20. Bhawna Goyal, Ayush Dogra, Sunil Agrawal, B.S. Sohi, Apoorav Sharma. Image denoising review: From classical to state-of-the-art approaches. (2020)
21. Yaqiao Cheng, Zhenhong Jia, Huicheng Lai, Jie Yang, Nikola K. Kasabov. A Fast Sand-Dust Image Enhancement Algorithm by Blue Channel Compensation and Guided Image Filtering. (2020)
22. Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, Gregory Slabaugh; DeepLPF: Deep Local Parametric Filters for Image Enhancement. Proceedings of

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2020)

23. Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, Jie Zhou. Global Filter Networks for Image Classification. Part of Advances in Neural Information Processing Systems 34. (2021)