Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Пояснювальна записка

до дипломної роботи
магістра
(освітньо-кваліфікаційний рівень)

на тему «Розробка і дослідження алгоритму стабілізації яскравості відеопотоку та оцінка його ефективності»

ХАІ.301.362.23О.151.14135959 ПЗ

Виконав: студент 6 курсу, групи 362

Галузь знань 15 «Автоматизація та приладобудування»

Спеціальність

151 “Автоматизація та комп’ютерно-інтегровані технології”

Освітня програма
“Інженерія мобільних додатків”

Білоус О.О.
(прізвище та ініціали студента)

Керівник Краснов Л.О.
(прізвище та ініціали)

Рецензент Ковтун І. В.
(прізвище та ініціали)

Харків – 2024

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів (№301)
Рівень вищої освіти___другий (магістерський)_____
Галузь знань ____15__Автоматизація та приладобудування _____
(шифр і назва)
Спеціальність ___151__Автоматизація та комп'ютерно-інтегровані технології _
(шифр і назва)
Освітня програма_____Інженерія мобільних додатків_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

к.т.н., доц. _____Костянтин ДЕРГАЧОВ

"____" _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ ЗДОБУВАЧУ

_____Білоуса Олександра Олександровича_____
(прізвище, ім'я, по батькові)

1. Тема роботи ____ Розробка і дослідження алгоритму стабілізації яскравості відеопотоку та оцінка його ефективності
керівник роботи _____к.т.н доцент каф. 301 Краснов Л.О._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 6.11.2023 року № 1968-уч
2. Строк подання студентом роботи: _08 січня 2024 року
3. Вихідні дані до роботи програмне забезпечення для попередньої обробки відеоданих із застосуванням бібліотеки OpenCV _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)___ Аналітичний огляд предметної області, аналіз шляхів та способів рішення завдання, опис та розробка методів попередньої обробки відеоданих, розробка програмного забезпечення, аналіз результатів обробки зображень, розрахунок собівартості виготовлення програмного забезпечення.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_10 слайдів для презентації та захисту роботи(див. додаток Е)__

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Стан проблеми	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24
Аналіз і синтез СУ	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24
Конструкторська частина	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24
Технологічна частина	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24
Експер.-практ. част.	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24
Економічна частина	Краснов Л.О., к.т.н., доцент каф 301	12.09.23	08.01.24

Нормоконтроль _____ Краснов Л.О «____» _____2024 р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання_____12.09.2023_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Початок переддипломної практики	01.09.2023	
2.	Формулювання теми роботи. Розробка технічного завдання	12.09.2023	
3.	Математичний опис системи управління. Аналіз і синтез системи управління. Проведення експериментальних досліджень	20.10.2023	Залік з переддип-ломної практики
4.	Конструкторська частина роботи. Дослідницька частина роботи. Експериментально-практична частина. Економічне обґрунтування розробки.	15.12.2023	
5.	Оформлення розрахунково-пояснювальної записки і графічного матеріалу	08.01.2024	
6.	Попередній захист роботи. Рецензування роботи	12.01.2024	
7.	Захист роботи	16.01.2024	

Здобувач _____ **Олександр БІЛОУС**
(підпис) (ім'я, прізвище,)

Керівник роботи _____**Леонід КРАСНОВ**

Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського
“Харківський авіаційний інститут”

Кафедра 301

«ЗАТВЕРДЖУЮ»
Завідуючий кафедри 301
к. т. н., с.н.с, доцент

Костянтин ДЕРГАЧОВ
« » 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на дипломне проектування
Білоуса Олександра Олександровича
(П.І.П.)

1. Тема роботи: «Розробка і дослідження алгоритму стабілізації яскравості відеопотоку та оцінка його ефективності» затверджена наказом по університету від 6 листопада 2023 р., № 1968-уч
2. Термін здачі студентом закінченої роботи « 8 » січня 2024 р.
3. Область застосування розробки: робототехніка, ЛА, повсякденне користування.
4. Початкові данні для об'єкту управління
4.1. Призначення і мета створення системи: Покращення якості зображень за допомогою методів бібліотеки OpenCV, а також оцінка їх роботи.
4.2. Загальні відомості: дослідження та розробка методів попередньої обробки яскравості зображень.
5. Технічні вимоги до системи управління
5.1. Питання, що підлягають розробці: Розробка та розрахунок нового методу попередньої обробки яскравості відеозображення.
5.2. Вимоги до структури й функціонування системи: система написана на мові програмування Python, кросплатформенність, використання малого об’єму пам’яті.
5.3. Вимоги до показників якості системи : підвищення якості очікуваних параметрів.
5.4 Вимоги до конфігурації обчислювальної техніки: 64—бітний двох ядерний процесор з тактовою частотою від 2 ГГц, оперативна пам'ять від 4 ГБ DDR3 SDRAM.
6 Умови експлуатації системи:
6.1 Вимоги до програмної та інформаційної сумісності: кросплатформенність, можливість запуску на застарілих системах.

6.2 Вимоги до зовнішніх збурень: температура середовища (—10℃…+40℃), вологість середовища 20—50%, хімічно активні компоненти відсутні.
6.3 Характер роботи системи (безперервної, циклічний, одноразового дії): безперервний.
7 Додаткові функції, реалізовані системою (сигналізація про несправності, реєстрація необхідної інформації, самоконтроль самої системи і т. ін.): самоконтроль, реєстрація та запис інформації.
8 Обсяг виконуваних розроблювачем робіт
8.1 Етапи проведення роботи: 1) вибір теми; 2) розробка ТЗ на проектування ПЗ; 3) Оцінка стану проблеми, аналіз; 4) Математичний опис методів. 5) розробка ПО;6) оформлення записки; 7) захист дипломного проекту
8.2 Обсяг розробки по кожному етапу: технічне завдання (3с); реферат (1с); зміст(2с); список умовних позначень (с); Вступ (1с); Постановка задачі та аналіз існуючих методів (9с); Опис і особливості системи (4с); Аналіз якості алгоритмів обробки (5с); Оцінка швидкодії алгоритмів (8с); Реалізація програмного продукту (), Економічна частина (9с); Заключення (1с); список використаних джерел (1л).
9 Порядок контролю й приймання системи: система повинна максимально покращувати вхідне зображення та надавати показники якості вихідного зображення.
10 Дослідницька частина: опис програми, демонстрація роботи методів для покращення вхідного зображення, оцінка розроблених методів обробки зображення
11 Експериментально—практична частина: проведення тестування системи, оптимального результату тестування.
12 Економічна частина
12.1 Розробити (розрахувати, одержати): розрахувати собівартість і ціну розробки системи обробки відеозображення;
12.2 Умови і вимоги: річна програма випуску не менше 140 одиниць;
12.3 Очікуваний результат: виробництво даної системи повинно виходити на точку беззбитковості
13. Перелік графічних матеріалів та їх формат: 10 листів презентації А3 для доповіді та захисту роботи наведені у додатку
14. Мова підготовки пояснювальної записки (захисту): _українська_____

Керівник роботи

доц к.т.н Краснов Л.О.
(П. І. П.)

« » _____ 2023 р.

Прийняв до виконання

Білоус О.О.
(П. І. П. студента)

« » _____ 2023 р.

Погоджено з питань:

конструкції

доц к.т.н Краснов Л.О.
(П. І. П.)

« » _____ 2023 р.

технології

доц к.т.н Краснов Л.О.
(П. І. П.)

« » _____ 2023 р.

економіки

доц. к.т.н Краснов Л. О.
(П. І. П.)

« » _____ 2023 р.

РЕФЕРАТ

Випускна робота містить: 94 ст. 39 рис. 7 табл. 4 додатка 18 джерел.

Тема роботи: Розробка і дослідження алгоритму стабілізації яскравості відеопотоку та оцінка його ефективності.

Мета роботи: Розробити новий метод стабілізації яскравості зображення.

Предмет роботи: система управління якістю відеопотоку, що виконує задачу стабілізації яскравості кадрів.

У роботі було розроблено систему попередньої стабілізації яскравості відеопотоку з використанням технології технічного зору. Результатом роботи є розроблена система автоматичного стабілізації яскравості з використанням технологій технічного зору.

КЛЮЧОВІ СЛОВА: ОБРОБКА ЗОБРАЖЕНЬ, ГІСТОГРАМА, СТАБІЛІЗАЦІЯ ЯСКРАВОСТІ, ЕКВАЛІЗАЦІЯ, OPENCV2, ШВИДКОДІЯ.

СПИСОК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ТЗ - технічний зір;
ШІ - штучний інтелект;
ПО - програмне забезпечення;
ЦЗ - цифрове зображення;
НЧ - низькочастотний.

ЗМІСТ

ВСТУП

Останніми роками галузь штучного інтелекту (ШІ) переживає швидке зростання, революціонізуючи різні галузі завдяки своїм передовим можливостям. Однією з найбільш розповсюджених галузей застосування ШІ є комп'ютерний зір, зокрема, розпізнавання зображень і обличь. Системи комп'ютерного зору, засновані на штучному інтелекті, продемонстрували вражаючий потенціал у точній ідентифікації та аналізі візуального контенту, що дало змогу знайти безліч практичних застосувань у різноманітних галузях - від безпеки та відеоспостереження до охорони здоров'я та автомобільної техніки.

У сфері розпізнавання зображень і обличь, якість вхідних даних має першорядне значення. Якісні зображення з оптимальною різкістю і чіткістю дають змогу алгоритмам ШІ більш ефективно виявляти і витягувати необхідні ознаки, що призводить до підвищення точності ідентифікації об'єктів і осіб. Таким чином, підвищення різкості та чіткості зображень є однією з основних умов досягнення високої продуктивності та мінімізації ризику неправильної класифікації або помилкової ідентифікації.

Таке покращення можливе за допомогою попередньої обробки зображення. Попередня обробка відеоданих позволяє підвищити якість зображень відеопослідовності для отримання на основі оригіналів максимально точних і адаптованих для автоматичного аналізу зображень. Стабільне значення яскравості є одним з основоположних елементів попередньої обробки, що дає змогу нормалізувати умови освітлення і забезпечити сталість рівня яскравості на різних зображеннях. Така нормалізація необхідна для підтримання однорідності даних і зниження впливу коливань освітленості, що сприяє підвищенню надійності та точності подальших завдань обробки зображень.

При постановці завдання на проектування було запропоновано провести дослідження методів і засобів попередньої обробки яскравості зображень, розробити алгоритми та методи програмування поставленого завдання. Тому спочатку розглянемо основні загальні особливості вирішення завдання обробки яскравості зображення. Потім проведемо огляд, класифікацію та аналіз існуючих методів і засобів, які зазвичай використовуються при вирішенні подібних завдань. Далі визначимо особливості завдання і запропонуємо конкретні технічні варіанти її вирішення.

1. ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз технічного завдання

У ТЗ наведені технічні вимоги до програмного продукту і сформовані питання, що підлягають розробці в проектувальній, експериментальній та реалізаційній частинах. Наведені вимоги до структури та функціонування системи, показників якості, конфігурації обчислювальної техніки. Також в ТЗ передбачені умови експлуатації системи. За допомогою всіх вимог наведених в ТЗ можливо визначити структуру та можливості програмного продукту, що зможе забезпечити стабільні показники якості зображення. А також оцінити ефективність методів і алгоритмів. Також система повинна мати здатність до безперервної роботи у реальному часі і мати змогу оцінювати якість оброблених відео даних, тому насамперед треба ввести критерії оцінки якості даних.

1.2 Загальна характеристика проблеми стабілізації яскравості зображення

Неточності у вхідних даних можуть суттєво вплинути на продуктивність і точність моделей ШІ, особливо тих, що базуються на завданнях комп'ютерного зору, таких як класифікація, виявлення об'єктів і сегментація зображень. Поширеними проблемами є зміни умов освітлення, шуми на зображенні, розбіжності в роздільній здатності та інші артефакти, які можуть перешкоджати навчанню та висновкам алгоритмів штучного інтелекту.

Методи попередньої обробки мають важливе значення для покращення вхідних даних. Хоча попередня обробка використовувалася для поліпшення візуальних характеристик для людського ока, зараз вона стала ще більш важливою для поліпшення якості вхідних даних для систем ШІ. Методи попередньої обробки спрямовані на стандартизацію вхідних даних, покращення співвідношення сигнал/шум і зменшення впливу неузгодженостей, тим самим забезпечуючи більш надійні та достовірні дані.

Забезпечення правильного рівня яскравості зображення є особливо важливим для ШІ, оскільки це безпосередньо впливає на інтерпретацію та виконання завдань на основі зображень. Стабільний рівень яскравості зображення допомагає моделям точно ідентифікувати та розрізняти об'єкти, шаблони та особливості на зображенні. Нерівномірна яскравість може вносити

спотворення, які вводять систему в оману, призводячи до помилкових класифікацій, поганого виявлення об'єктів або неточних результатів сегментації.

Якість вхідних даних суттєво впливає на продуктивність і надійність систем, тому впровадження ефективних методів попередньої обробки, включно зі стабілізацією яскравості, має першорядне значення. Такі високоякісні стандартизовані дані допоможуть системам досягти точного й ефективного прийняття рішень та аналізу.

Тому задачею даної роботи було обрано пошук, дослідження і проектування оптимального методу попередньої обробки яскравості кадру зображення, у якому можлива поточна обробка відеозображення без впливу на швидкодію.

1.3 Патентний пошук

Патент (від лат. Patens – відкритий, ясний) – охоронний документ, що засвідчує виключне право, авторство і пріоритет винаходу, корисної моделі або промислового зразка.

Першим було знайдено патент методу налаштування кольору [1], яскравості та тонової шкали візуалізованих цифрових зображень для оброблення для одержання поліпшеного цифрового зображення, що включає одержання нелінійної шкали тонів для візуалізації, а також одержання або обчислення функції шкали тонів настроювання та об'єднання функцій шкали тонів візуалізації та настроювання. Але такому рішенню притаманне значне погіршення візуального сприйняття, суттєва зміна колірного балансу та неприпустимі зміни середньої яскравості кадрів відео.

Інший відомий метод підвищення контрастності зображень [2] має пристрій призначений для підвищення контрастності одного або декількох кольорових зображень. Пристрій містить джерело зображення, яке забезпечує кольорове зображення з колірними значеннями для областей зображення, де кожна область зображення може конкретно відповідати пікселю. Процесор покращення контрастності може спеціально генерувати компенсовані значення яскравості відносно номінального значення яскравості для кольору пікселів. Потім виконується підсилення контрасту яскравості для компенсованих значень яскравості. Такий підхід може дозволити покращити якість зображення і, зокрема, зменшити появу неприродних ефектів затінення через варіації

кольору. Цьому методу притаманні ті самі недоліки − порушення колірного балансу і помітне погіршення візуального сприйняття кадрів відео.

Найбільш близьким по технічній суті і результату є пристрій і спосіб керування яскравістю зображення [3]. Яскравість і контрастність зображення в цьому пристрої підвищується за допомогою гістограмного вирівнювання. Але такому рішенню притаманні недоліки. За такого гістограмного вирівнювання може виникнути суттєве порушення колірного балансу та неконтрольована зміна середньої яскравості кадру. Причиною цього є нелінійність процедури еквалізації.

Список патентів представлений у списку використаних джерел

1.4 Аналіз існуючих методів попередньої обробки яскравості

Яскравість у контексті зображень означає загальну інтенсивність або яскравість пікселів зображення. Це міра того, скільки світла випромінюють або відбивають об'єкти сцени, яку представляє зображення. У цифрових зображеннях яскравість часто представлена значеннями пікселів, де вищі значення вказують на світліші ділянки, а нижчі - на темніші.

Поняття яскравості тісно пов'язане зі сприйняттям освітленості зображення. Однак важливо зазначити, що термін "яскравість" може бути неоднозначним, оскільки його часто використовують як взаємозамінний з іншими термінами, такими як "освітленість" або " насиченість".

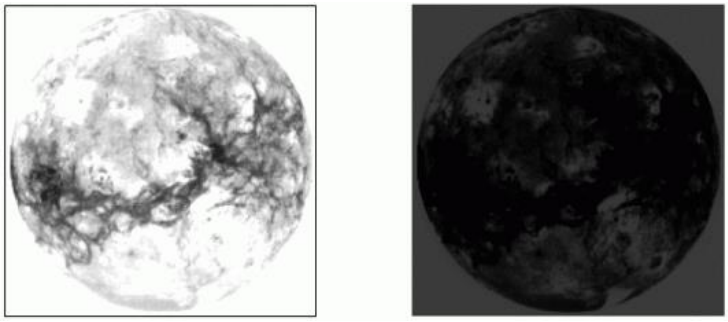
В обробці зображень регулювання яскравості зазвичай передбачає зміну значень пікселів, щоб зробити зображення яскравішим або темнішим. Це можна зробити за допомогою різних методів, таких як вирівнювання гістограми, розтягування контрасту або просто додавання чи віднімання постійного значення від усіх значень інтенсивності пікселів.

Варто зазначити, що яскравість - це лише один з аспектів загального вигляду зображення, а інші фактори, такі як контраст, колірний баланс і насиченість, також відіграють важливу роль у візуальному сприйнятті зображення. Регулювання лише яскравості не завжди може призвести до бажаного візуального покращення, тому для досягнення оптимального результату зазвичай використовують комбінацію налаштувань.

Зображення повинно мати належну яскравість і контрастність для зручного перегляду. Яскравість - це загальна світлість або темрява зображення. Контраст - це різниця в яскравості між об'єктами або регіонами.

Наприклад, білий кролик, який біжить засніженим полем, має погану контрастність, тоді як чорний собака на тому ж білому тлі має хорошу контрастність. На рис. 1.1 показано чотири можливі способи неправильного налаштування яскравості та контрастності.

Коли яскравість занадто висока, як у варіанті (a), найбіліші пікселі перенасичуються, знищуючи деталі в цих областях. Зворотна ситуація показана на (b), де яскравість встановлена занадто низько, що призводить до насичення найчорніших пікселів.

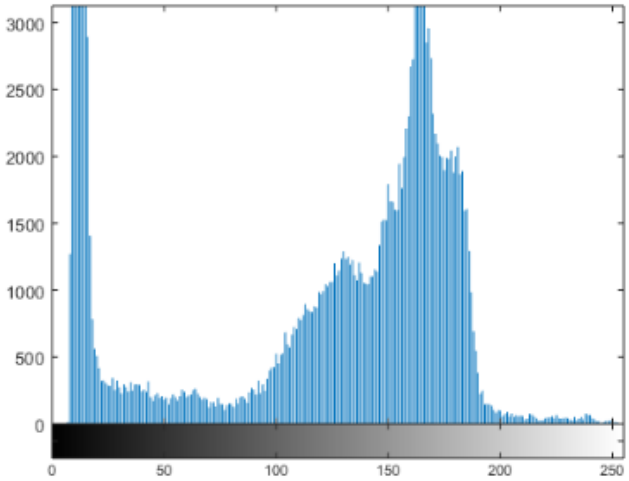


а) б)

Рисунок 1.1 – Приклади неправильного налаштування яскравості

Найкориснішим інструментом для корекції налаштувань експозиції та балансу є гістограма. Гістограма - це графік тональних значень зображення. Від чорного кольору, розташованого ліворуч, до білого, розташованого праворуч. Висота графіка в кожній точці показує відносну кількість пікселів зображення певного тону та їхній рівень яскравості. Що вищий графік, то більше пікселів певного тону присутні на зображенні. Приклад гістограми зображено на рис. 1.2.

Гістограми яскравості присвоюють значення яскравості кожному пікселю зображення чи кадру і використовують його для побудови графіка. Колір окремих пікселів ігнорується на користь відображення загальної яскравості сцени. Однак це може бути оманливим за наявності яскравих, насичених



об'єктів.

Рисунок 1.2 – Гістограма яскравості зображення

Серед різних існуючих методів покращення якості зображень, метод на основі звичайного вирівнювання гістограми є одним з найпопулярніших завдяки своїй простоті та ефективності. Основна ідея полягає в тому, щоб відтворити рівні сірого на зображенні на основі функції щільності розподілу ймовірностей. Він працює шляхом вирівнювання та розтягування динамічного діапазону гістограми, що призводить до загального покращення контрастності зображення. Результати роботи алгоритму приведені на рис. 1.3.

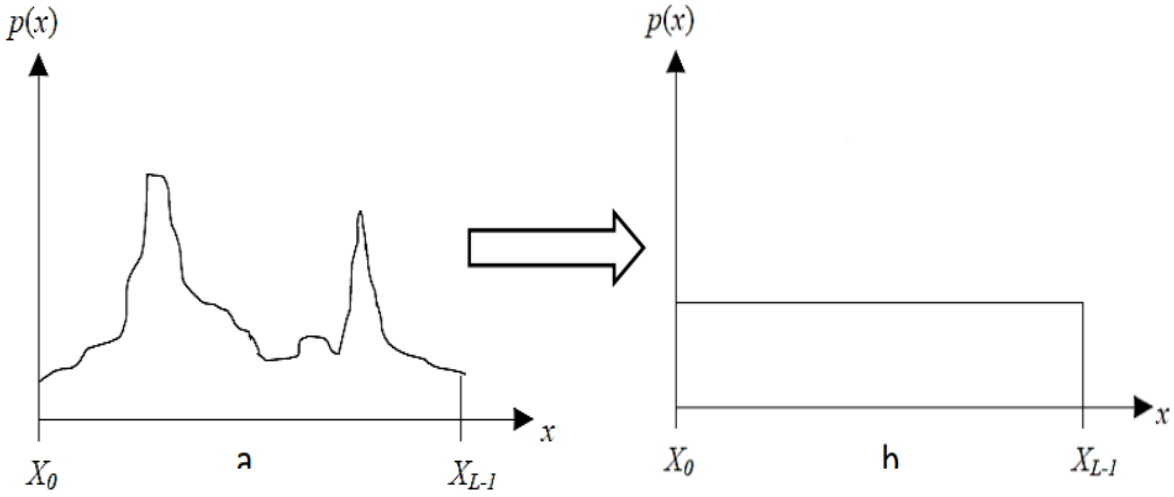


Рисунок 1.3 – Результати роботи гістограмної евалізації

Незважаючи на свою популярність, еквалізація має відомий недолік: він змінює початкову яскравість вхідного зображення. Метод завжди підкреслює області зображення з більшою кількістю входжень сірого кольору. Ці області часто бувають надмірно підсиленими. І навпаки, області, що містять відносно невелику кількість пікселів, можуть бути вилучені, що призводить до так званого розмитого вигляду. Деякі деталі на зображенні зникають зі зменшенням рівня сірого у вихідному зображенні. Розтягування контрасту також обмежується певними домінуючими областями. Надмірне злиття рівнів сірого на зображенні призводить до появи хибних контурів, які генерують небажані артефакти та неприродне підсилення на зображенні. На рис.1.4 зображено негативний вплив перетворень еквалізації.

Для подолання цих недоліків запропоновано кілька методів на основі еквалізації, які більше зосереджені на збереженні яскравості зображення, ніж на покращенні контрастності зображення[6]. Деякі методи часто генерують зображення з візуальними артефактами та неприродним виглядом, хоча яскравість зображення зберігається до певної міри.

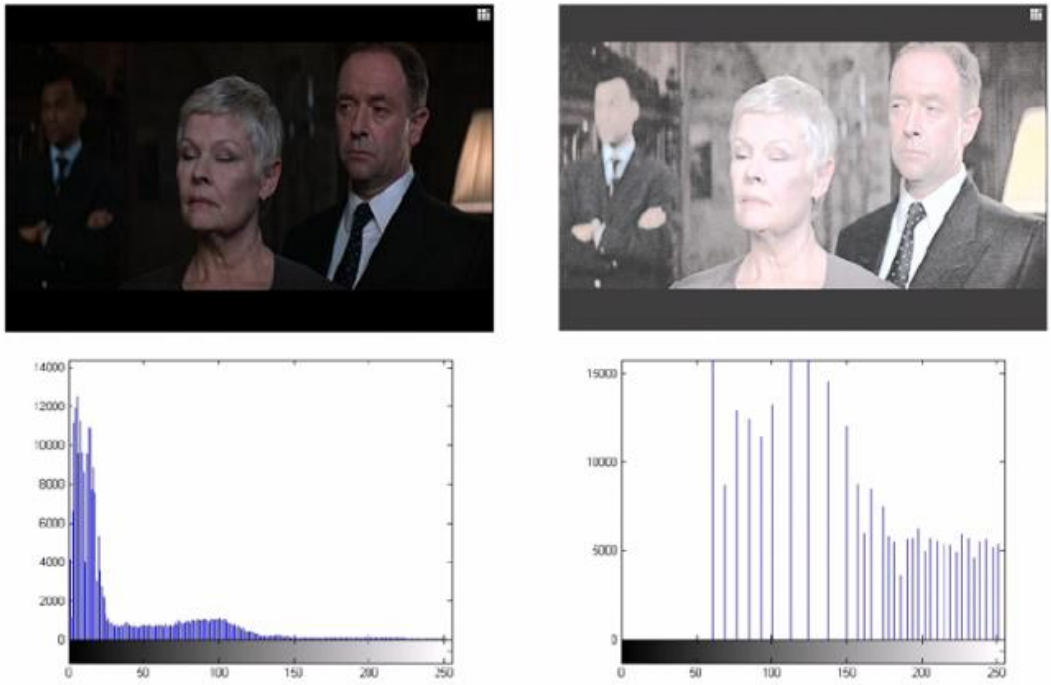


Рисунок 1.4 – Приклад перенасичення зображення після класичної гістограмної еквалізації

Метод бігістограмного вирівнювання[12], представлений на рис. 1.5, гістограма ділиться на дві підгістограми на основі різних точок поділу. Пізніше кожна субгістограма вирівнюється окремо на основі еквалізації гістограми. Ці методи можуть краще зберігати яскравість зображення порівняно з методом вирівнювання гістограми.

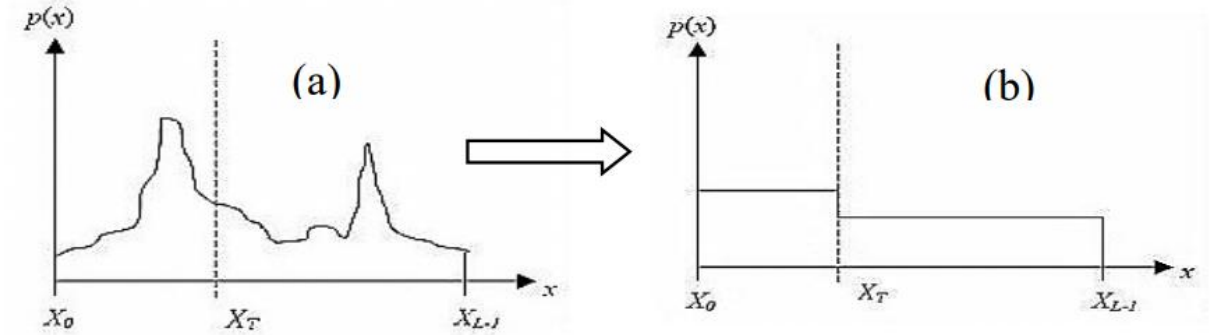
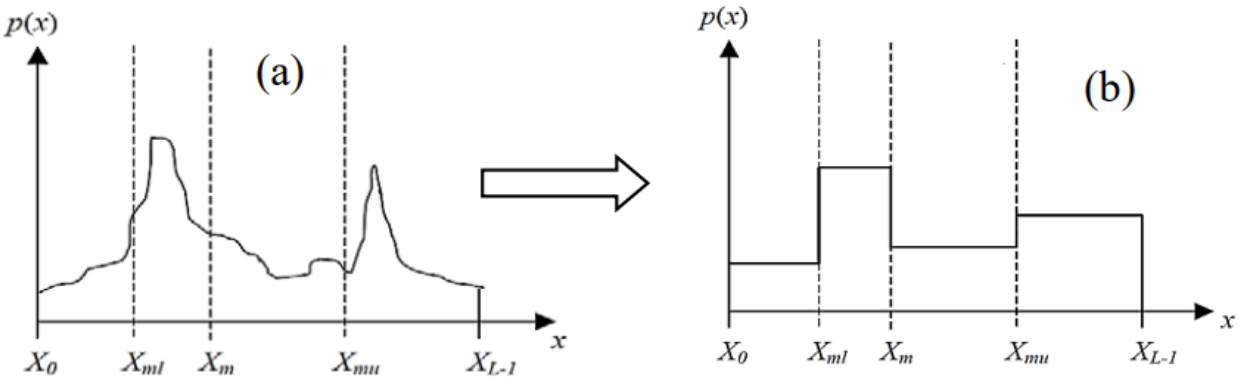


Рисунок 1.5 – Результати роботи бігістограмної евалізації

Однак отримані зображення можуть не мати природного вигляду. Щоб подолати ці недоліки, можливе розкладання зображення на декілька підзображень, таким чином, щоб підвищення контрастності зображення, яке забезпечується еквалізацією на кожному підзображенні, було менш інтенсивним. Таких алгоритм зображений на рис. 1.6[7]. Зображення, оброблене методами мультигістограмної еквалізації, зберігає яскравість

зображення і запобігає появі небажаних артефактів, але не суттєво посилює



контрастність.

Рисунок 1.6 - Результати роботи мультигістограмної евалізації

Особливість використання цих методів зумовлена нелінійністю процедури еквалізації, виконання якої в ортогональному колірному просторі RGB призводить до істотного порушення колірного балансу. Основна причина, чому колірна модель RGB вважається неоптимальною для обробки яскравості, полягає в тому, що вона не має доступу до прямого впливу на яскравість.

При маніпулюванні яскравістю в колірному просторі RGB, представлений на рис. 1.7, проста зміна інтенсивності кожного окремого компонента (R, G або B) не призводить до точних змін яскравості. Наприклад, збільшення числового значення всіх трьох компонентів однаково призведе до зміни як кольору, так і яскравості, що не є ідеальним варіантом, якщо метою є регулювання лише яскравості.

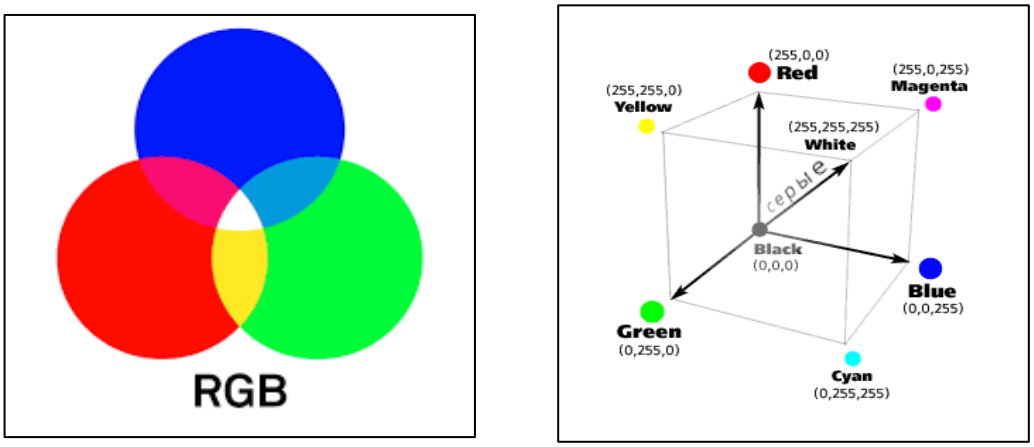


Рисунок 1.7 – а) Палітра моделі RGB , б) Відображення кольорового простору моделі

Для багатьох методів обробки інтерактивне використання цього адитивного змішування кольорів не завжди підходить. Визначаючи колірні простори як тривимірні об'єми, що представляють доступний колірний спектр,

можна отримати більш відповідні методи для вибору кольорів. Тому потрібен перехід в інший колірний простір із незалежною компонентою яскравості[9].

Такі моделі, як HSV (Hue, Saturation, Value) (рис. 1.8) або HSL (Hue, Saturation, Lightness) мають доступ до перетворень тільки окремо взятого каналу насиченості/яскравості зображення[11]. Компонент яскравості відокремлено від інформації про колір (відтінок і насиченість). Таке розділення полегшує виділення та маніпулювання інформацією про яскравість незалежно від кольору. При корегуванні яскравості це розділення може спростити обробку і зробити алгоритми більш ефективними. Це дає змогу провести еквалізацію точніше і позбавляє від одержуваних дефектів на зображенні, повного руйнування кольору. Перехід на ці моделі полегшить регулювання яскравості, покращуючи керування рівнями яскравості.

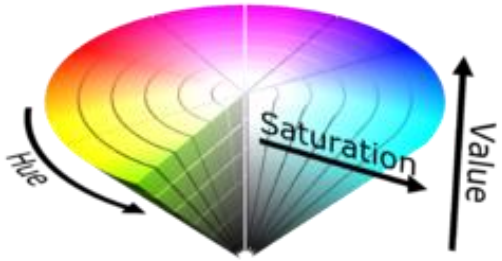


Рисунок 1.8 – Кольорова модель HSV

А колірні моделі, такі як YUV або YIQ, відокремлюють представлення яскравості від кольоровості, що полегшує виділення компонента яскравості та маніпулювання ним незалежно від інформації про колір.

Колірна модель YIQ на рис. 1.9, хоча в першу чергу призначена для телебачення, може надати певні переваги для задач комп'ютерного зору, включаючи стабілізацію рівня яскравості.

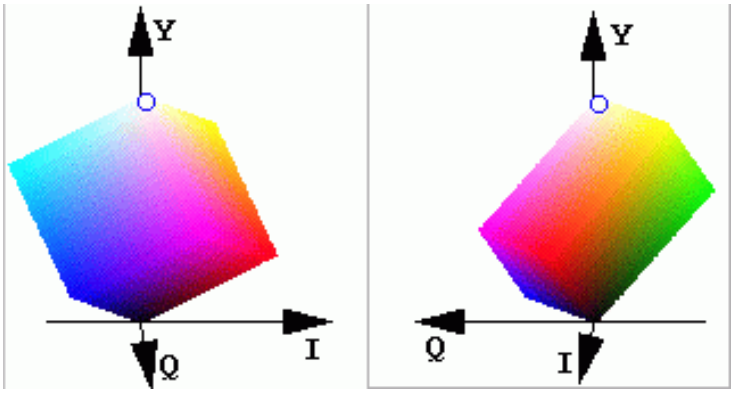


Рисунок 1.9 - Кольорова модель YIQ

YIQ розділяє компоненти яскравості (Y) і кольоровості (I і Q). Це розділення дозволяє незалежно маніпулювати інформацією про яскравість і колір. При виконанні обробки яскравості це може спростити обробку,

зосередившись на компоненті яскравості, який, по суті, є представленням у відтінках сірого. Таке відокремлення може сприяє більш точному і цілеспрямованому регулюванню рівнів яскравості, покращуючи загальну якість і узгодженість нормалізації яскравості зображення.

Гістограмна еквалізація працює з усім зображенням, і її мета - розподілити значення інтенсивності так, щоб покрити весь динамічний діапазон. В результаті цього процесу локальний контраст може бути посилений, але водночас це може призвести до втрати локальних деталей і дрібних структур.

У підсумку, оскільки цей метод є нелінійним, його використання може і буде призводити до появи дефектів і артефактів, можлива повна втрата колірного балансу зображення. Такі перетворення можуть призвести до неможливості проведення подальших обробок інформації. А оскільки процес є неконтрольованим, якось протидіяти цьому ефекту не є можливим.

1.5 Постановка задач проектування

Метою даної роботи є створення системи попередньої обробки яскравості відеозображення з метою покращення роботи систем штучного інтелекту.

Для реалізації цього проекту необхідно:

1. Розробити новий метод стабілізації яскравості зображення.
2. Створити апаратно-програмні засоби розробленого метода на мові програмування Python
3. Підготувати методичні матеріали.

2. ОПИС І ОСОБЛИВОСТІ СИСТЕМИ СТАБІЛІЗАЦІЇ ЯСКРАВОСТІ

2.1 Основний критерій оцінки яскравості кадру

Під час аналізу відеозаписів або відео в реальному масштабі часу необхідно використовувати найуніверсальніший, найстійкіший та найінформативніший критерій, що дає змогу дати об'єктивну оцінку якості відеоданих за різних умов зйомки та характеру освітленості сцени. Найбільш підходящим і чи не єдиним стійким показником якості є середня яскравість фрейму AFB (Average Frame Brightness)[16].

Для отримання об'єктивної оцінки зміни яскравості відео доцільно простежити залежність середньої яскравості фреймів AFB від часу та розрахувати статистичні характеристики цього випадкового процесу (тренди середніх значень на всьому інтервалі аналізу або окремих ділянках, гістограми розподілу параметра AFB).

Результатом є значення, яке представляє загальну яскравість всього кадру. Корекція цього показника фокусується виключно на регулюванні рівнів яскравості зображення, залишаючи при цьому інформацію про колір незмінною. Вона так само дає змогу уникнути артефактів як на еквалізації, що забезпечує більш точне представлення вихідного зображення.

Таким чином, лінійна стабілізація яскравості зображення забезпечує більший контроль, дає змогу уникнути колірних артефактів, зберігає природний колір і, як правило, найкраще підходить для застосунків, де важливо зберегти точне представлення вихідного зображення.

2.2 Вибір основної колірної моделі

Однак, окремі компоненти фрейму R, G і B так і не мають повної інформації про яскравість пікселів зображення. Для можливості виконання даного процесу ноебхідно виконати перехід у кольорову модель, що включає у себе окремий канал яскравості, наприклад моделі HSV та YIQ, які були представлені раніше.

Перехід із колірного простору RGB у простір HSV дає змогу просто оцінити середній рівень яскравості фрейма за компонентою яскравості V, який обчислюється як

$$\mathrm{AFB} = \frac{1}{\mathrm{MN}} \sum_{n=1}^{N} \sum_{m=1}^{M} V(n, m)\,, \tag{2.1}$$

де V(i,j) – двомірний масив чисел, що визначають яскравість пікселів зображення кадру розміром M×N.

Сам процес представляє просту лінійну зміну значення параметру яскравості зображення. Зміна цього показника виконується додаванням сталої до значення каналу V у кольоровому форматі HSV.

2.3 Алгоритм обробки та збереження даних

Для повноти аналізу необхідно досліджувати залежність середньої яскравості кадрів AFB від часу, і розрахувати статистичні характеристики цього випадкового процесу (середнє значення за усім інтервалом аналізу чи окремих ділянках відеопослідовності). Для цього було використано універсальний алгоритм просторово-часової обробки відеоданих, що формує з вихідного відео послідовність значень середньої яскравості кадрів AFB[15].

Структуру цього алгоритму наведено на рис. 2.1.

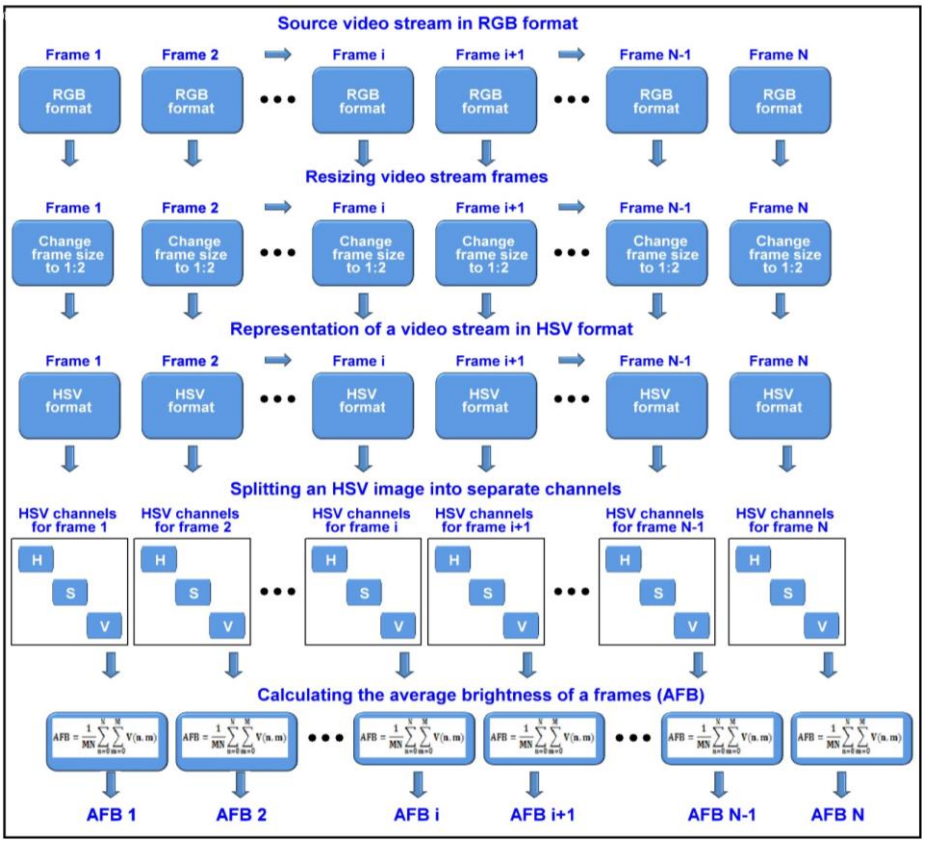


Рисунок 2.1 – Алгоритм формування тренда змін середньої яскравості відеозображення

2.4 Алгоритм фільтрації

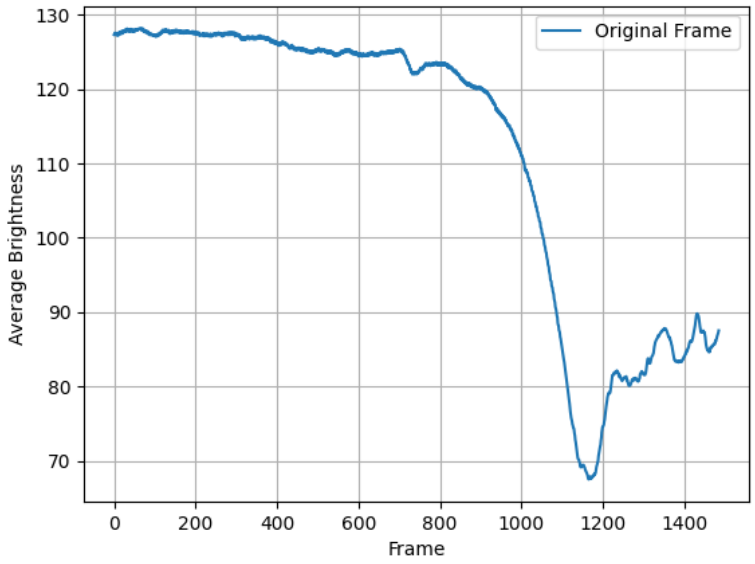


Рисунок 2.2 – Приклад графіку тренду середньої яскравості відеозображення

На рис. 2.2 можна побачити вплив високочастотної компоненти шумоподібного характеру, що свідчить про швидку і спонтанну зміну яскравості кадрів відео. Високочастотну компоненту слід розглядати як перешкоду, для усунення якої доречно використовувати усереднюючий фільтр зі ковзним вікном.

Такий алгоритм працює за принципом буфера, де зберігаються останні дані для усереднення. Це числа, що відповідають середній яскравості кадрів AFB, кількість яких визначена розмірами вікна фільтра та швидкістю зміни кадрів (fps). На кожному кроці фільтрації процедура усереднення даних визначається формулою

$$\mathrm{AFB_{filtr}} = \frac{1}{\mathrm{W}} \sum_{\mathrm{w}=1}^{\mathrm{W}} \mathrm{AFB_w}, \tag{2.2}$$

де $\mathrm{AFB_w}$ – одновимірний масив чисел, що становлять послідовність значень середньої яскравості кадрів AFB не більше вікна W фільтра. На кожному новому кроці буфер зсувається, до нього додається нове значення $\mathrm{AFB_{w+1}}$ і забирається найстаріше $\mathrm{AFB_1}$, після чого процедура усереднення повторюється.

Повна блок-схема пропонованого алгоритму представлена на рис. 2.3.

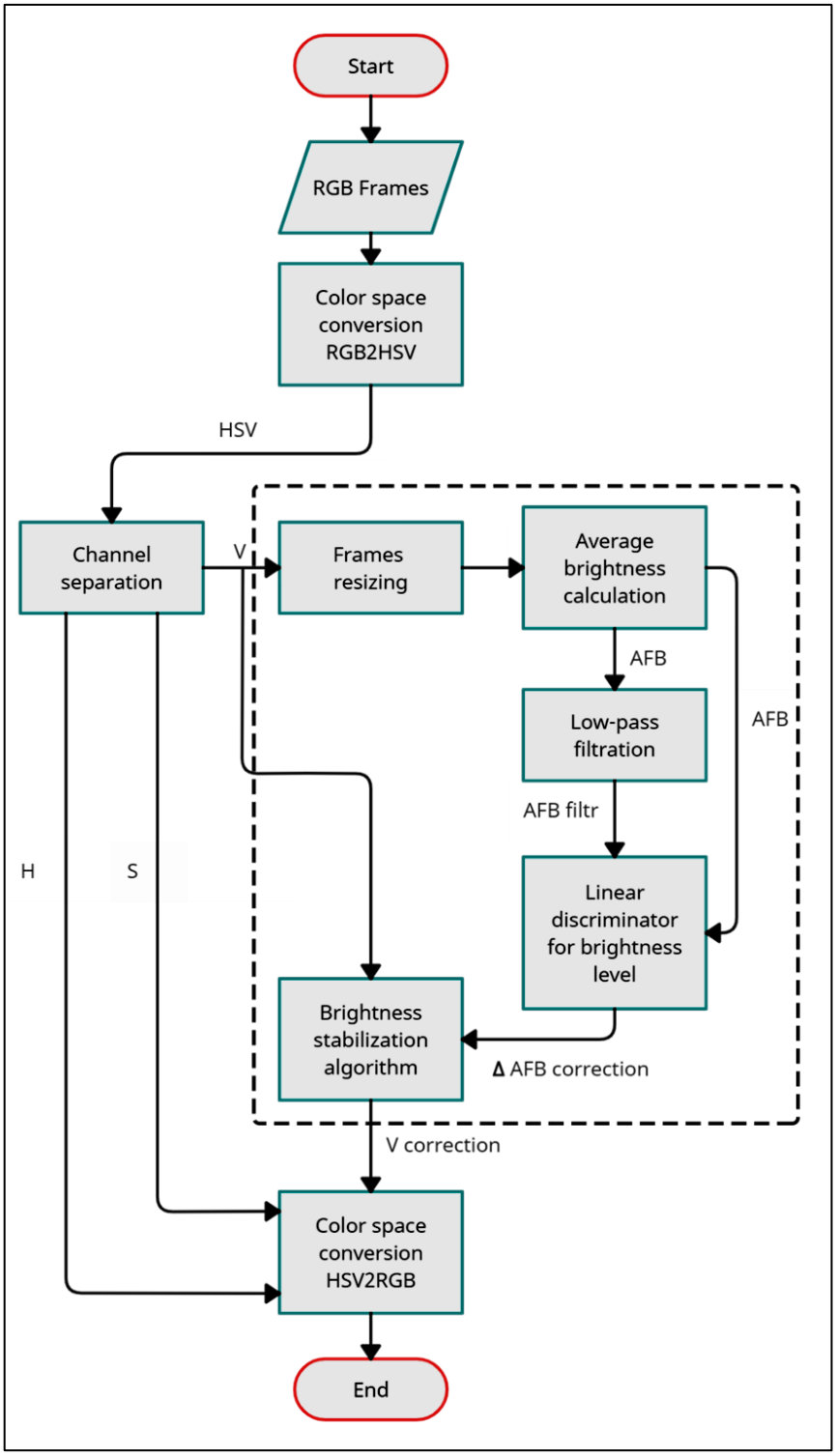


Рисунок 2.3 – Блок-схема алгоритму стабілізації яскравості

Основна проблема такого методу полягає у перевищені межових значень показника яскравості пікселя. Щоб уникнути цих проблем, дуже важливо виконувати регулювання яскравості в межах відповідного діапазону і застосовувати відповідні методи для збереження деталей зображення та

візуальної точності. Для цього необхідно проводити перевірку на перенасичення меж яскравості.

$$\Delta\mathbf{AFB}_{\boldsymbol{i}} = \mathbf{AFB}_{\boldsymbol{i}} - \mathbf{AFB}_{\mathbf{filtr\ i}};$$
$$\text{if } \Delta\mathbf{AFB}_{\boldsymbol{i}} \geq \mathbf{0} \text{ then } \mathbf{AFB}_{\mathbf{i\ correction}} = \mathbf{AFB}_{\boldsymbol{i}} - \Delta\mathbf{AFB}_{\boldsymbol{i}}; \quad (2.3)$$
$$\text{if } \Delta\mathbf{AFB}_{\boldsymbol{i}} < \mathbf{0}, \text{ then } \mathbf{AFB}_{\mathbf{i\ correction}} = \mathbf{AFB}_{\boldsymbol{i}} + \Delta\mathbf{AFB}_{\boldsymbol{i}};$$

Уникнення обмеження межових значень може призвести до втрати інформації та деталей на яскравих ділянках зображення, призвести до відсікання яскравих ділянок, внаслідок чого вони виглядатимуть рівномірно білими без будь-якої помітної текстури або деталей, або перейдуть но нижніх рівнів яскравості, ставши повністю чорними[14].

Рисунок 2.4 – Приклад перенасичення кадру

Як показано на рис. 2.4, перенасичення параметру яскравості призводить до чорний плям на зображенні.

2.5 Висновки

В даному розділі було представлено лінійний алгоритм стабілізації яскравості відеопослідовності. Було розроблено структурну схему і основні необхідні параметри. Представлений метод використовує низькочастотну фіотрацію по рівню середньої яскравості кадру та тренду серендії показників.

3. АНАЛІЗ ЯКОСТІ АЛГОРИТМІВ ОБРОБКИ ВІДЕОДАНИХ

3.1 Створення тестової вибірки відеозображень

Для об'єктивного оцінювання якості існуючих та запропонованого алгоритмів, необхідно провести тестування на наборі відеозаписів із різними властивостями - різною роздільною здатністю кадрів і різною частотою їхнього проходження (fps).

Було сформовано набір відеофайлів для тестування, Їхні характерні фрейми показано на рис. 3.1.

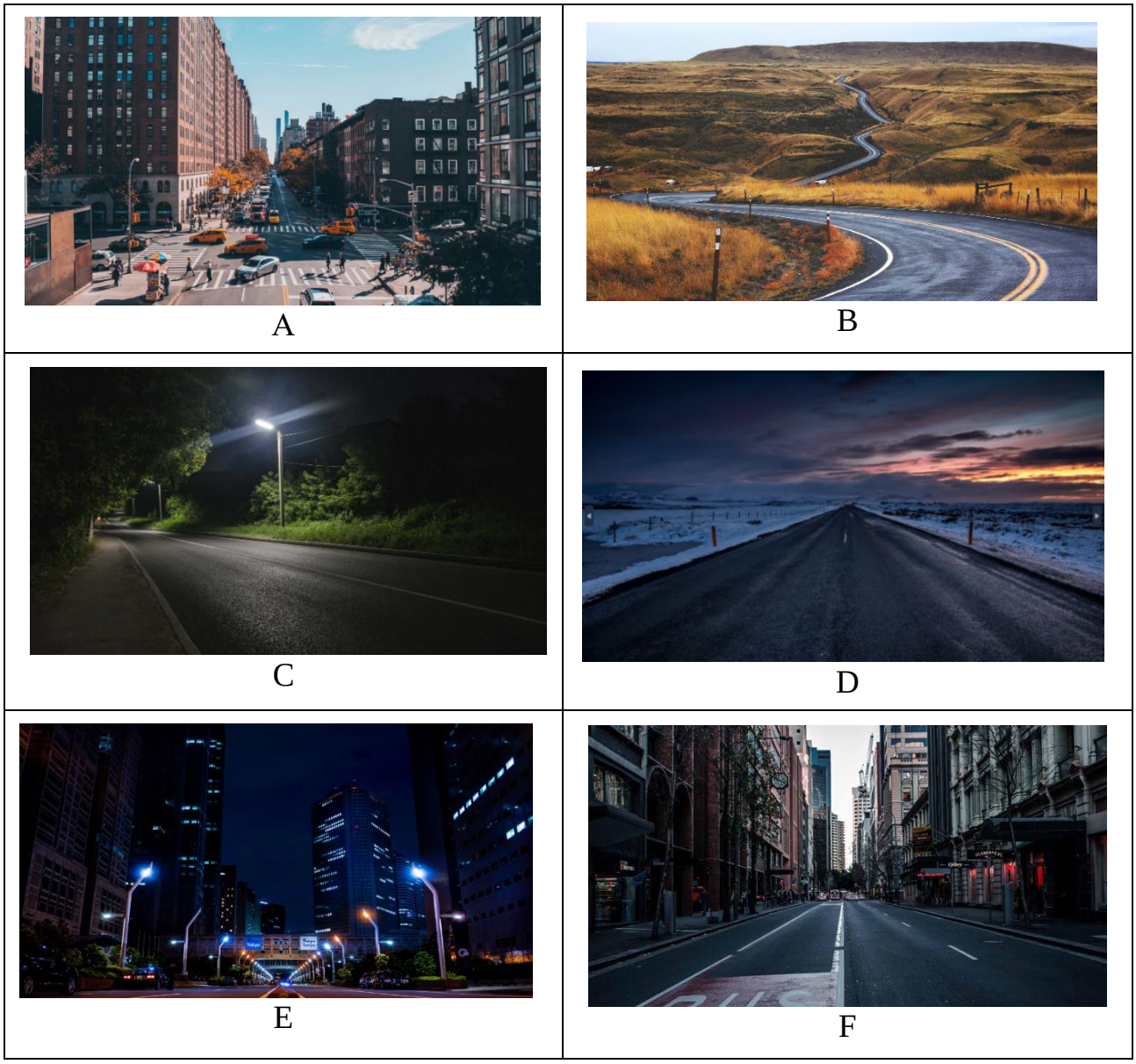
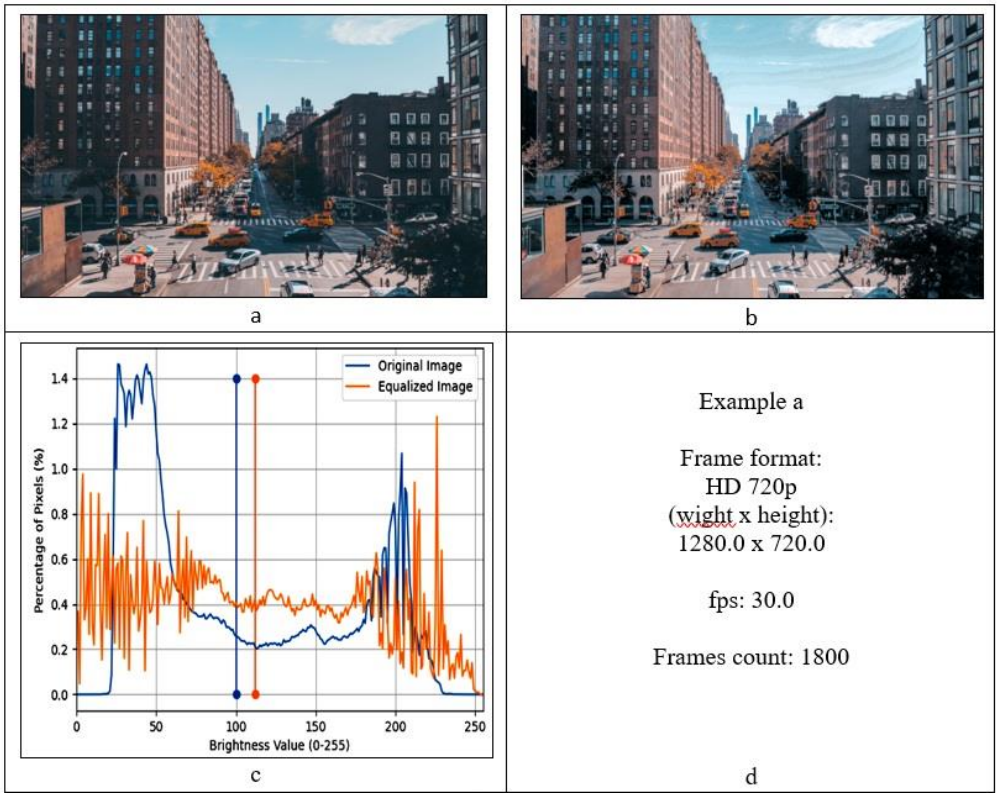


Рисунок 3.1 – Вибірка тестових відеозображень

3.2 Оцінка візуальних маркерів якості еквалізації

Наведено кілька прикладів оцінювання якості за візуальними маркерами.

Для тесту використовувся класичний алгоритм еквалізації відеоданих із переходом з колірного простору RGB у простір HSV. На рис. 3.2 a і рис. 3.2 b показано зображення фрейму до і після еквалізації, а на рис. 3.2 c в одному вікні наведено гістограми розподілу яскравостей кадру до і після перетворення. У вигляді різнокольорових стовпчиків показано розрахункові значення середньої яскравості кадру до і після еквалізації. На рис. 3.2 ,b помітно структурні зміни кадру після еквалізації – освітлені частини зображення помітно перенасичені та набули різкості. Гістограма після перетворення набуває більш коливального характеру, баланс кольору зображення сильно пошкоджено, а середня яскравість кадру зростає на 10-15 одиниць шкали



яскравості кадру.

Рисунок 3.2 – Еквалізація світлого кадру при переході у простір HSV

При використанні переходу у простір YIQ, результати мають значно гірший вигляд. Вони представлені на рис. 3.3. Візуальне сприйняття погіршується, набагато помітнішими стають структурні зміни, гістограма

розподілу яскравостей має хаотичний характер, а зміни середньої яскравості кадру досягають 25-30 одиниць шкали.

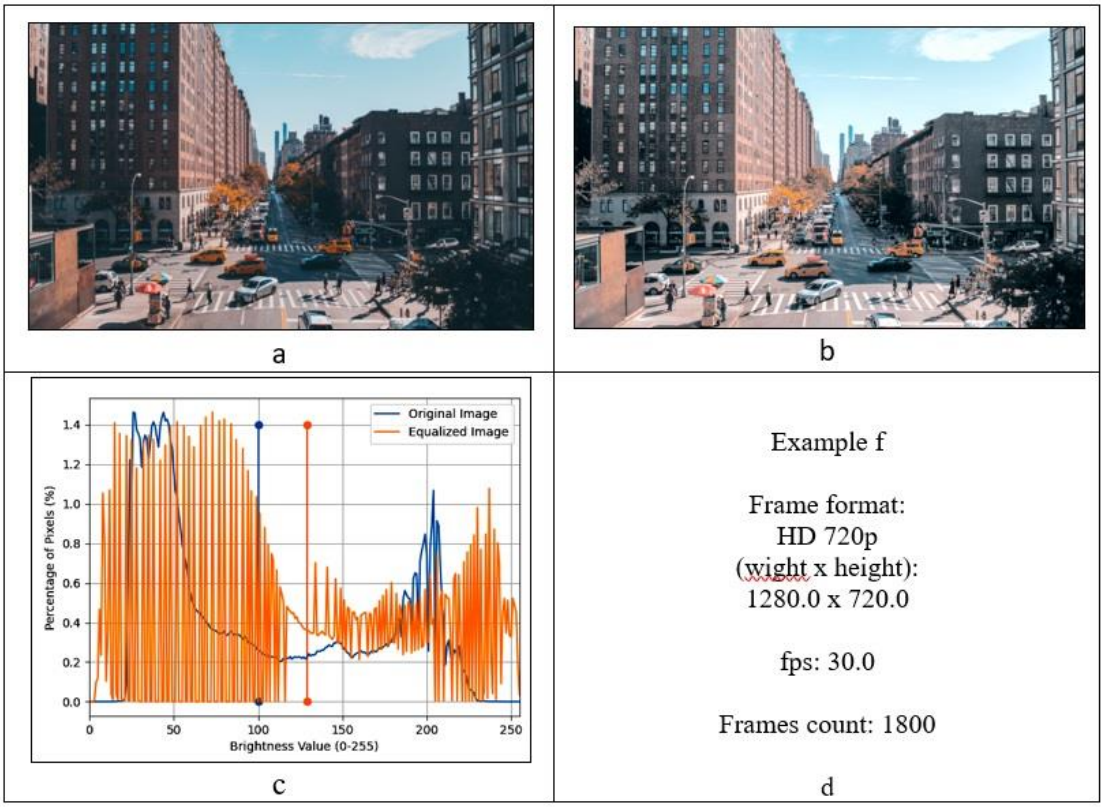


Рисунок 3.3 – Еквалізація світлого кадру при переході у простір YIQ

Слід провести дослідження впливу методу на більш затемнену сцену відеозображення. Результати обробки показано з переходом у простір HSV на рис.3.4.

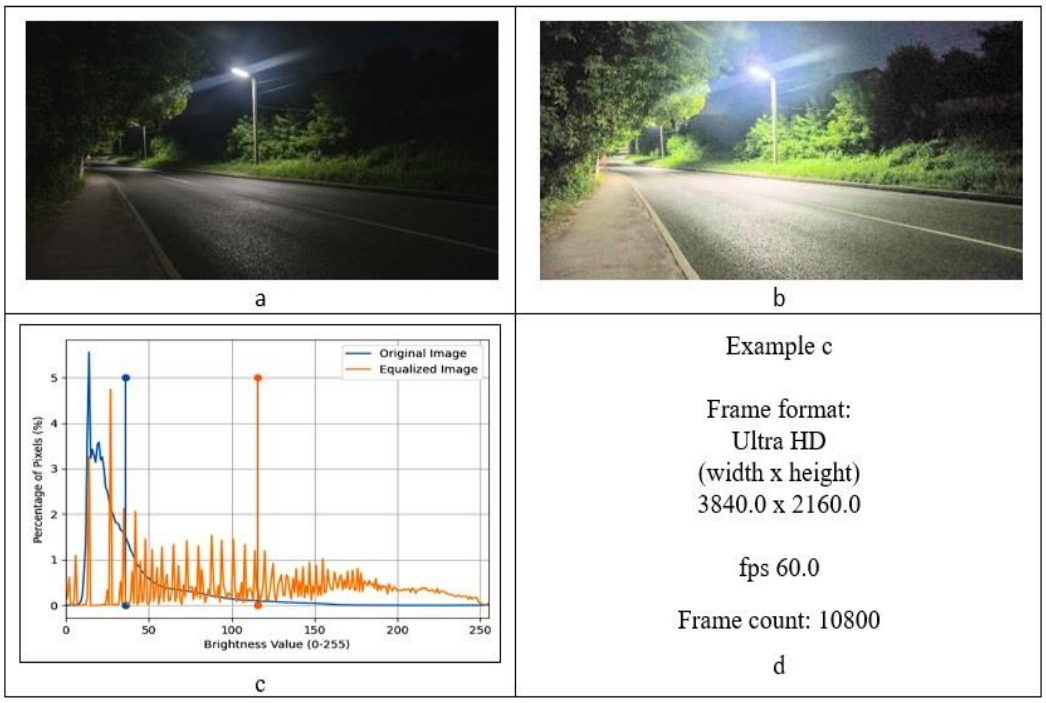


Рисунок 3.4 - Еквалізація затемненого кадру при переході у кольоровий простір HSV

На рис. 3.4 с видно, що рівень середньої яскравості кадру помітно зріс − від 40 до 120 одиниць шкали яскравості, гістограма розподілилася більш рівномірно в діапазоні всієї шкали яскравості, але почала мати коливальний характер, візуальне сприйняття кадру стало дещо комфортнішим. Але зображення отримало сильне перенасичення, що призвело до втрати детальності на засвітлених ділянках та загальній втраті читаності.

У разі використання колірного простору YIQ, негативні тенденції тільки посилилися. На рис. 3.5, середній рівень яскравості кадру зріс ще більше (від 40 до 130 одиниць шкали), а гістограма розподілу яскравості стала ще більш гребінчастою, візуальне сприйняття погіршилося до дискомфортного, візуально кадр було зруйновано.

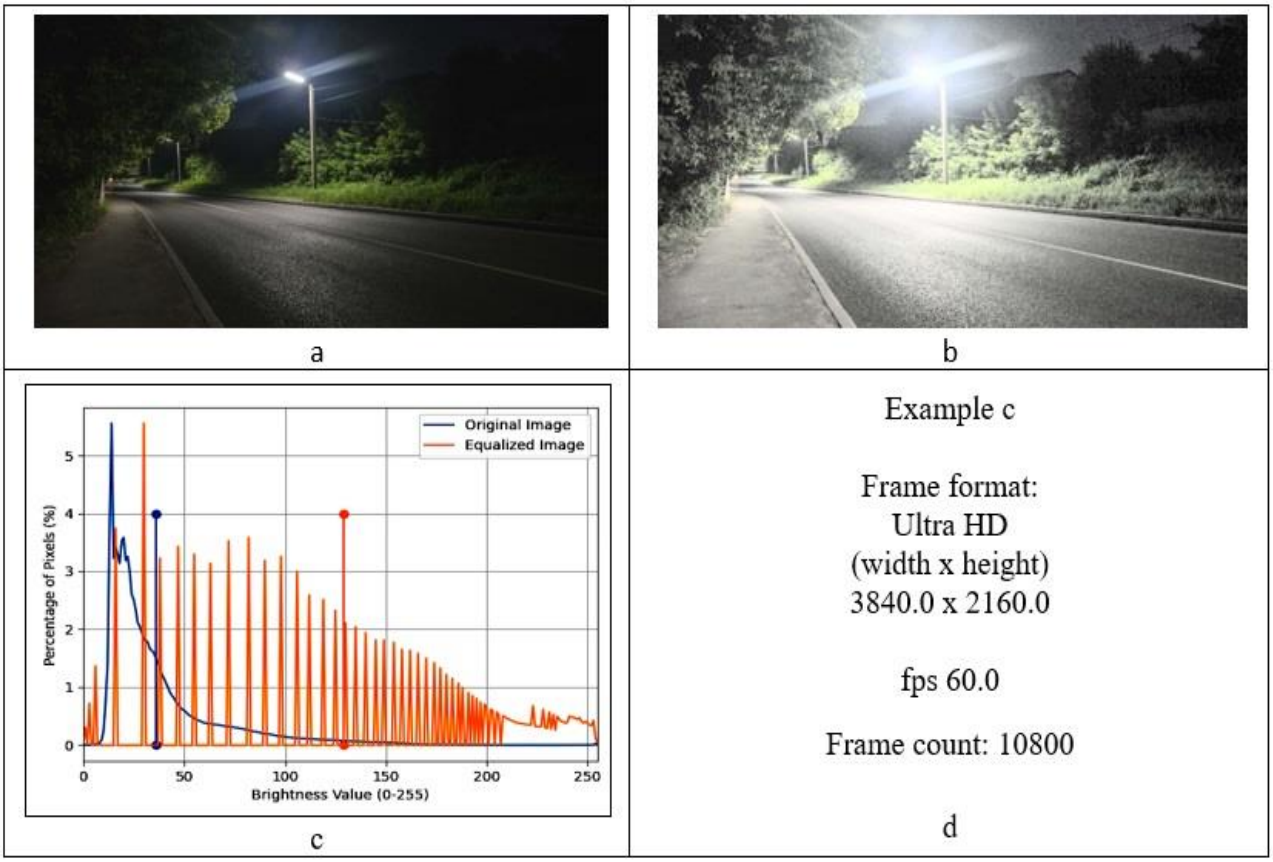


Рисунок 3.5 – Еквалізація затемненого кадру при переході у простір YIQ

Слід також представити приклад відносно справної роботи еквалізації. Результати обробки показано з переходом у простір HSV на рис.3.6.

Даний кадр став контрастнішим та чіткішим, але менш комфортним для людського ока. Через вирівнювання яскравості темних ділянок, було втрачено глибину зображення.

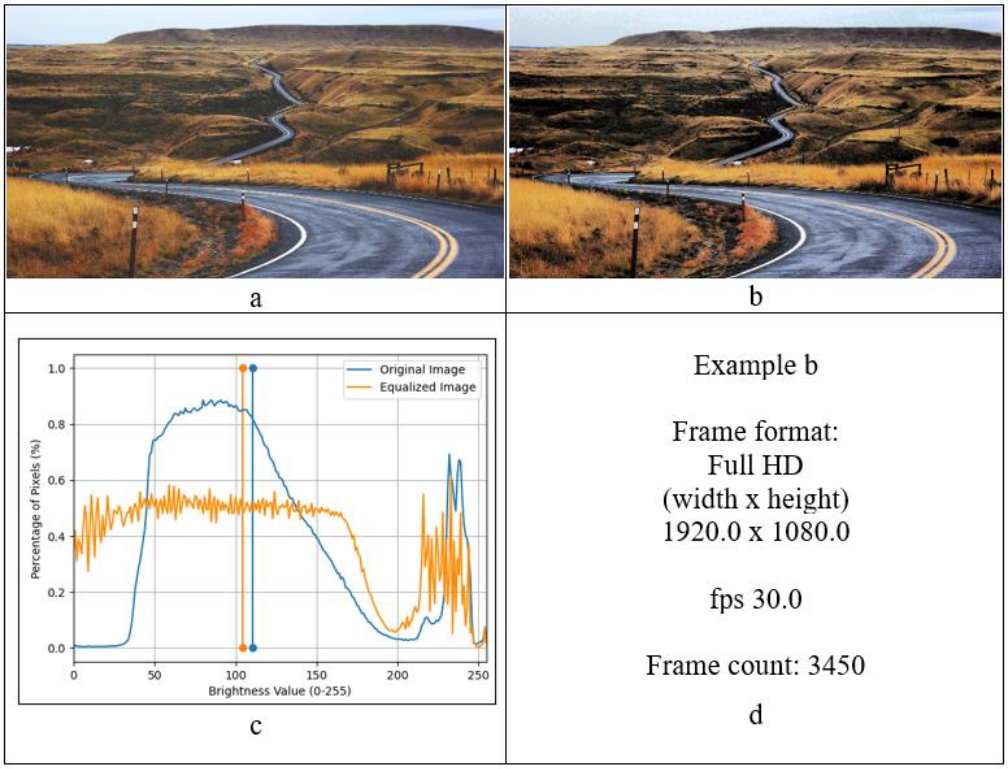


Рисунок 3.6 – Еквалізація при переході у кольоровий простір HSV

На гістограмі видно більш коливальний та хаотичний характер розподілу, було порушено баланс. Рівень середньої яскравості незначно виріс, яскравість зображення була посунута в середньому на 10-15 одиниць.

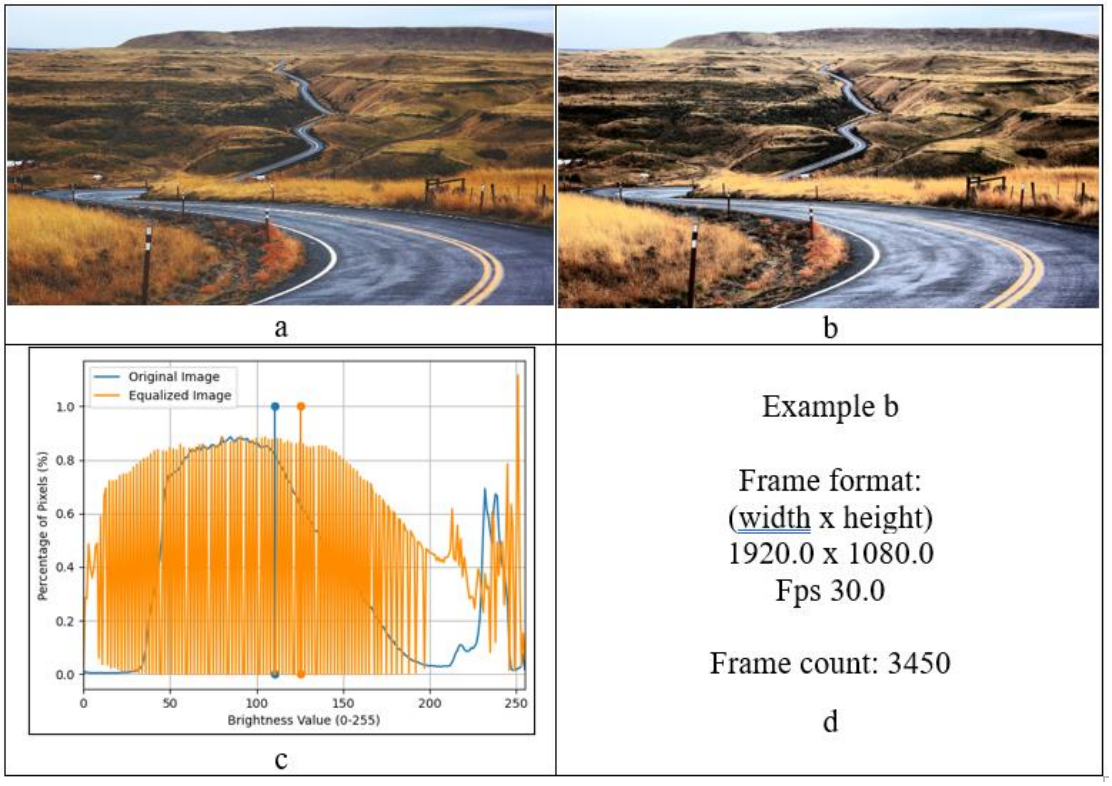


Рисунок 3.7 - Еквалізація при переході у кольоровий простір YIQ

При використанні переходу до колірного простору YIQ, усі позитивні сторони використання еквалізації відсутні. Зображення стало занадто чітким та перенасиченим, колір перейшов до більш сірих та блідих відтінків. Хоча на рис. 3.7 середній рівень яскравості виріс незначно більше ніж при використанні простору HSV (від 10 до 20), гістограма розподілу має повністю хаотичний характер (кожна друга одиниця яскравості відсутня на розподілі).

На підставі аналізу цих двох прикладів можна зробити висновок про те, що в розглянутих алгоритмах підвищення контрастності доцільно використовувати тільки перехід із колірного простору RGB у простір HSV. Еквалізація за каналом Y у форматі YIQ дає великі спотворення, зображення втрачає в контрастності та кольорі, руйнується колірний баланс. Тому що канал V в HSV − це яскравість пікселя, а канал Y в YIQ − це освітленість. Це означає, що еквалізація, застосована до каналу V в HSV, з більшою ймовірністю дасть візуально приємніші результати з підвищення контрастності, оскільки вона безпосередньо націлена на рівні яскравості зображення.

3.3 Розрахунок кількісних показників якості еквалізації

Крім візуальних маркерів для повноцінного аналізу розраховувалися і кількісні статистичні показники для всіх тестових відео, показаних на рис. 6.

У практиці оброблення цифрових зображень заведено використовувати кілька популярних кількісних статистичних показників для оцінювання якості перетворень. У нашому випадку доцільно використовувати такий набір показників[8]:

Середньоквадратична помилка (MSE) - це викривлення зображення, яке обчислюють шляхом усереднення квадрата різниць інтенсивностей зображення обкладинки й отриманих у результаті перетворення пікселів зображення. Нижче значення MSE вказує на кращу якість зображення.

$$\mathrm{MSE} = \frac{1}{\mathrm{M} \times \mathrm{N}} \sum_{\mathrm{i}=0}^{\mathrm{M}-1} \sum_{\mathrm{j}=0}^{\mathrm{N}-1} \big(\mathrm{I}(\mathrm{m},\mathrm{n}) - \mathrm{C}(\mathrm{m},\mathrm{n})\big)^2. \tag{3.1}$$

PSNR вимірює якість зображення з точки зору відношення сигнал/шум. Більш високі значення PSNR вказують на кращу якість зображення і широко використовуються під час стиснення зображень і відео. Однак, як і MSE, воно не завжди добре корелює з людським сприйняттям.

$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSE}}, \tag{3.2}$$

Нормалізована крос-кореляція (NCC) вимірює схожість між двома зображеннями. Значення NCC лежить у діапазоні від 0 до 1. Якщо значення NCC дорівнюють 1, це означає, що ці зображення повністю ідентичні. NCC представляється формулою

$$\mathrm{NCC} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(S(i,j)\times C(i,j)) \,/\, \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\big(S(i,j)\big)^2 . \tag{3.3}$$

Індекс SSIM оцінює помилки, що сприймаються, що означає, що спотворення зображення розглядається як зміна структурної інформації, що сприймається. Він заснований на оцінці того, коли пікселі мають взаємозалежності, особливо коли ці пікселі просторово близькі. Пропонується як поліпшення PSNR.

$$\mathrm{SSIM}(C,S) = \frac{(2\mu_C\mu_S + c_1)\times(2\sigma_{CS} + c_2)}{(\mu_C^2 + \mu_S^2 + c_1)\times(\sigma_C^2\sigma_S^2 + c_2)}, \tag{3.4}$$

де μ_C- середнє значення C кадру I_{output}, μ_S - середнє значення S пікселів кадру I_{input}, σ_C^2 - дисперсія C, σ_S^2 - дисперсія S, σ_{CS} - коваріація C і S, $c_1 = (K_1L)^2$ та $c_2 = (K_2L)^2$ - дві змінні, що стабілізують розподіл зі слабким знаменником, L - динамічний діапазон значень пікселів, $K_1 = 0{,}01$ и $K_2 = 0{,}03$ за замовчуванням.

Абсолютна помилка середньої яскравості (AMBE) визначається як абсолютна різниця між середнім значенням вхідного і вихідного зображень і пропонується для оцінки ефективності збереження вихідної яскравості.

$$\mathrm{AMBE} = \left|\mathrm{AFB}_{\mathrm{Ioutput}} - \mathrm{AFB}_{\mathrm{Iinput}}\right| \tag{3.5}$$

Крім візуальних маркерів для повноцінного аналізу необхідно створити базу кількісних статистичних показників для всіх тестових відео, для алгоритму підвищення контрастності з переходом у простір HSV, і алгоритму з

використанням колірного простору YIQ. Результати розрахунків зведено в табл. 3.1 і табл. 3.2 відповідно.

Таблиця 3.1

Показники якості алгоритму покращення контрастності у просторі HSV							
	A	B	C	D	E	F	Average
MSE	86.58	88.27	106.134	93.85	89.56	106.49	95.147
PSNR	28.76	28.67	27.87	28.41	28.61	27.86	28.36
NCC	0.9705	0.9766	0.8352	0.93	0.83	0.90	0.906
SSIM	0.87	0.88	0.35	0.77	0.41	0.58	0.643
AMBE	82.47	88.27	84.93	77.18	47.41	55.15	72.61

Таблиця 3.2

Показники якості алгоритму покращення контрастності у просторі YIQ							
	A	B	C	D	E	F	Average
MSE	116.55	101.99	111.09	85.31	105.18	85	100.85
PSNR	27.47	28.04	27.67	28.82	17.91	28.84	26.458
NCC	0.96	0.9375	0.8	0.88	0.6755	0.8603	0.8522
SSIM	0.81	0.82	0.3	0.5	0.1	0.41	0.49
AMBE	116.55	93.26	99.09	75.11	100.88	75.55	93.4

Аналізуючи кількісні показник помітно що зображення в обох випадках еквілізації втрачає схожість з оригіналом. Залежно від екземпляра, результат практично повністю втрачає колірний і яскравий зв'язок, але використання алгоритму підвищення контрасту кадрів з використанням колірного простору HSV безумовно краще.

3.4 Оцінка візуальних маркерів якості лінійного алгоритму стабілізації яскравості

Результати алгоритму стабілізації яскравості приведені на рис. 3.8. В результаті роботи алгоритму перетворене зображення кадру не викликає зорового дискомфорту, а середня яскравість кадру трохи зменшилася (приблизно на 5 одиниць шкали яскравості), але водночас істотно згладилася гістограма яскравості та зрушилася в бік зменшення. Це дуже важливо в різних технічних додатках, і свідчить про ефективну фільтрацію високочастотних коливань яскравості.

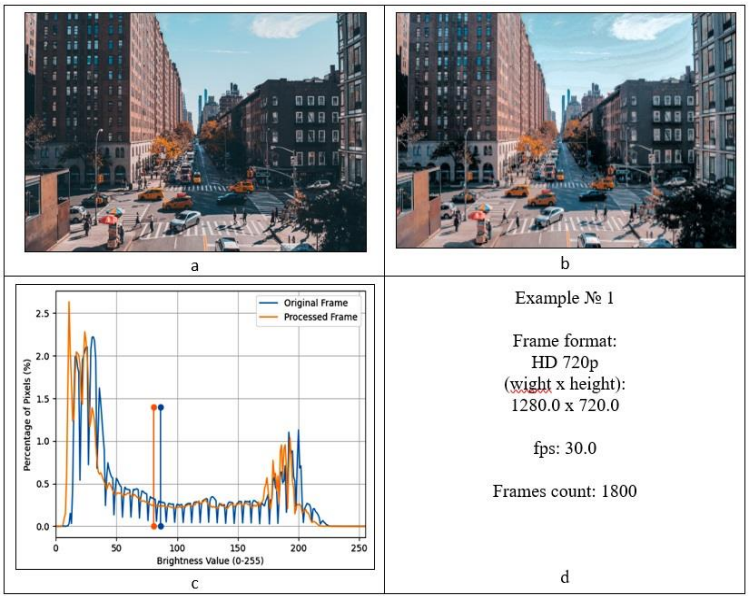


Рисунок 3.8 – Виконання алгоритму лінійної стабілізації

Представлено ще одним приклад, але із заниженим рівнем освітлення сцени (тестове відео C). Результати розрахунків з переходом у простір HSV наведено на рис. 3.9

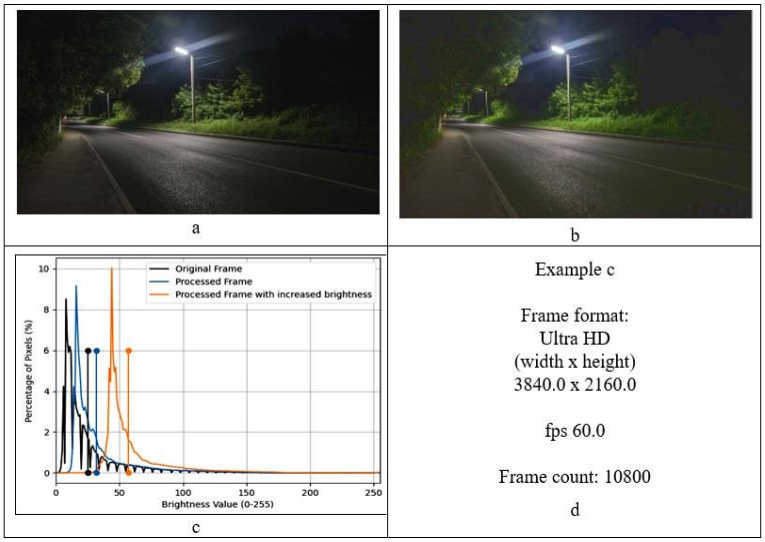


Рисунок 3.9 – Виконання алгоритму лінійної стабілізації при низькому рівні освітленя

Як і в попередньому прикладі, гістограма розподілу яскравості успішно згладилася, але форма її залишилася незмінною. Для фільтрації тренда середньої яскравості було використано НЧ-фільтр із постійною часу 5 сек. Відбулося невелике збільшення яскравості, яке складно помітити візуально. Але в цих випадках яскравість може бути збільшена додаванням до кожного пікселя декількох градацій яскравості. У цьому прикладі додано 30 одиниць. Таке додавання значень повинно бути виконано з захистом від переповнення шкали яскравості.

На прикладі, представленому на рис. 3.10 видно прямий вплив алгоритму на розподіл яскравості. Після обробки кадр став темнішим, а на гістограмі видно зсув графіку на 5-10 одиниць, при цьому повністю відсутні зміни у балансу яскравості,

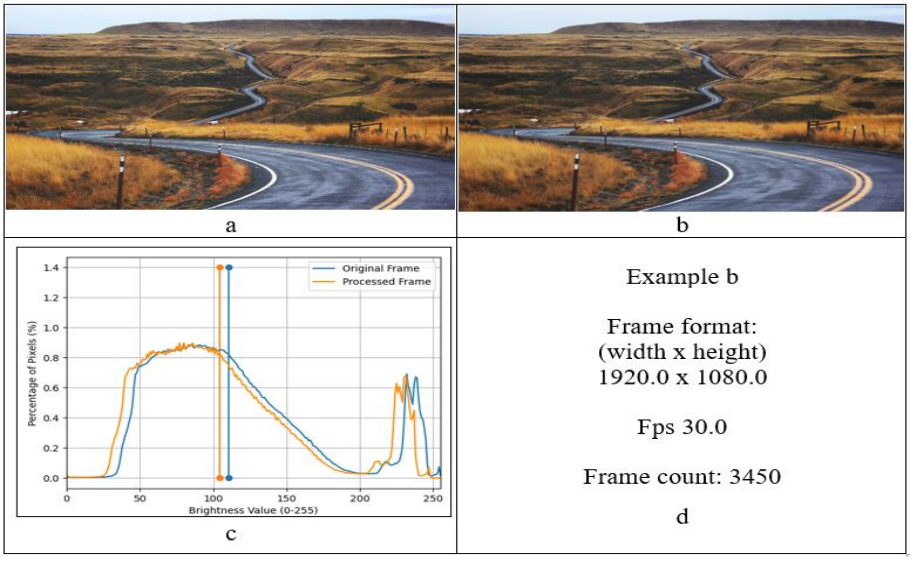


Рисунок 3.10 - Виконання алгоритму лінійної стабілізації

Кількісні оцінки якості оброблення алгоритмом наведено в табл.3.3

Таблиця 3.3

Показники якості алгоритму лінійної стабілізації яскравості у просторі HSV							
	A	B	C	D	E	F	Average
MSE	16.5710	14.40	18.49	12.85	11.10	15.40	14.80
PSNR	35.94	36.55	35.46	37.04	37.68	36.25	36.48
NCC	0.999	0.997	0.996	0.99	0.997	0.997	0.997
SSIM	1.0	1.1	0.99	1.0	0.95	0.98	0.97

AMBE	4.06	4.24	4.21	3.79	3.85	3.78	3.98

3.5 Висновки

В даному розділі було проведено оцінку якості перетворень існуючих та запропонованого методів стабілізації яскравості. Виходячи з отриманих візуальних та статистичних показників, можна зробити висновок що запропонований алгоритм являється більш ефективним. Статичні показники показують, що лінійні перетворення не змінюють кольоровий баланс та не перенасичують зображення.

4. ОЦІНКА ШВИДКОДІЇ АЛГОРИТМІВ

Паралельне опрацювання і використання спеціалізованих обчислювальних пристроїв допомагають впоратися з цією проблемою. Важливо враховувати і стійкість алгоритмів до змін параметрів відео. Час обробки має бути мінімізовано.

4.1 Дослідження швидкодії алгоритму еквалізації

Для проведення тестування швидкодії запропонованого методу було створено додаток, основні функції якого були описані у розділі 5. Цей додаток дозволив провести дослідження роботи лінійного алгоритму стабілізації яскравості відеопотоку з метою вивчення швидкодії в порівнянні з алгоритмом еквалізації, зробити вибір оптимальних розмірів вихідних кадрів, провести аналіз впливу розмірного коефіцієнта на швидкість роботи. Як вихідні дані для досліджень використовувалися відеофайли з різною роздільною здатністю.

Спочатку було проведено дослідження алгоритму еквалізації та вплив роздільної здатності відеозображення на загальну швидкодію методу. Для перевірки було обрано відеофайл розмірності 1920x1080p.

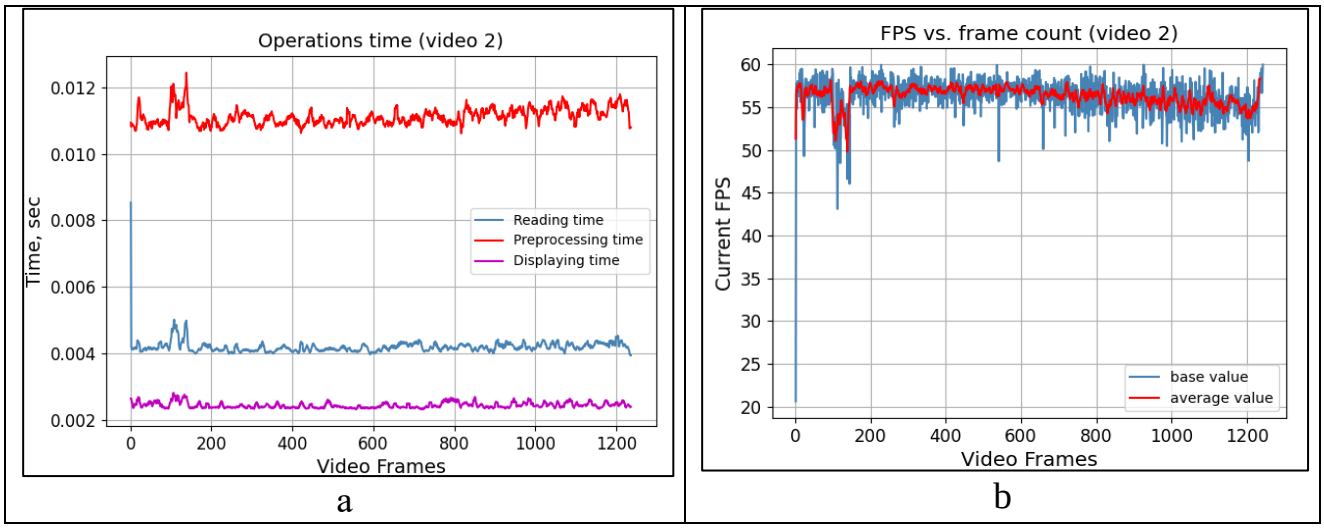


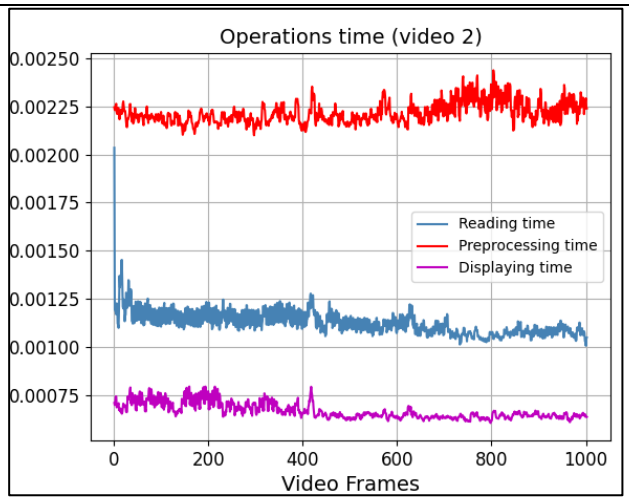
Рисунок 4.1 – Показники швидкодії алгоритму еквалізації при розмірності кадру 1920x1080p

На рис. 4.1 а та 4.1 в зображено витрати часу на повне виконання алгоритму з урахуванням зчитування окремих кадрів (Reading time), обробки алгоритмом еквалізації (Preprocessing time) і відображення результату у вікні програми (Displaying time). За отриманими показниками видно за метод повністю покриває необхідну швидкодію для використання при обробці

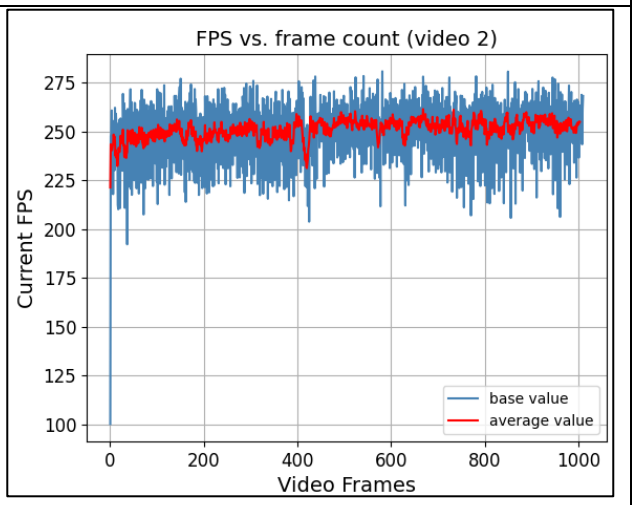
відеофайлів з частотою оновлення у 30 кадрів на секунду, що є стандартом для систем відеоспостереження та загальної систем комп'ютерного зору.

Було проведено дослідження роботи алгоритму еквалізації при більшому та меншому розмірі відеофайлу по роздільній здатності. Результати приведені на рис. 4.2.

Файл розмірності 854x480

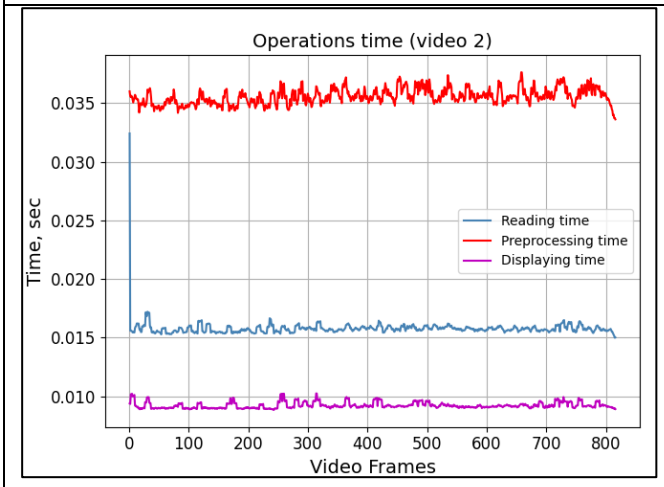


a

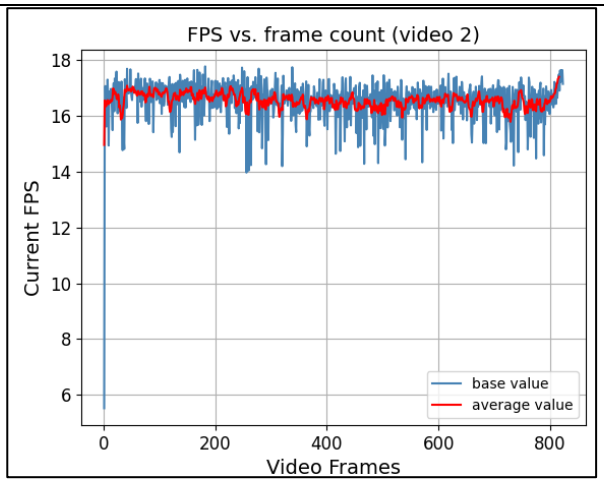


b

Файл розмірності 3840x2160



c



d

Рисунок 4.2 - Показники швидкодії алгоритму еквалізації при різній розмірності кадру

На рисунках 4.2 a та 4.2 в показано витрати часу на повне виконання алгоритму еквалізації при вхідному розмірі зображення 854x480. На риснуках 4.2 c та 4.2d зображено часові показники методу при розмірності відеозображення 3840x2160. Як видно з показників, розмір зображення майже впливає як на затрати часу на зчитування та відображення кадрів, так і критично впливає на швидкість обробки кадру методом. Тому великоформатні

файли та відеопослідовності (Ultra Hd, 4k) недоступні для обробки у реальному часі таким алгоритмом через низьку частоту оновлення кадрів вихідного зображення.

Слід зазначити, що при проведенні дослідження швидкодії використовувався базовий метод еквалізації. Подальше вдосконалення методу у бі-гістограмну чи мульти-гістограмну еквалізації збільшить час на виконання у геометричній прогресії.

4.2 Вибір найефективнішого методу реалізації лінійного алгоритму стабілізації

В ході проектування запропонованого лінійного алгоритму стабілізації було розроблено декілька методів реалізації, тому необхідно експериментальним методом визначити найефективніший варіант.

Першим варіантом реалізації алгоритму лінійної корекції є вбудована у бібліотеку NumPy мови Pyhon функція clip(). Вона використовується для обмеження значень масиву в межах заданого діапазону[5]. Вона допомагає гарантувати, що всі елементи масиву потрапляють у певні мінімальні та максимальні значення. Так як це заготовлена функція, вона може містити в собі сторонні залишкові елементи, які можуть сповільнити швидкодію. Приклад лістингу коду представлений на рис. 4.3.

```
48      if value >= 0:
49          # Increase the brightness of V channel by beta, avoiding border values
50          v = np.clip(v + value, 0, 255)
51      else:
52          v = np.clip(v - value, 0, 255)
```

Рисунок 4.3 – Лістинг коду лінійної корекції за допомогою функції clip()

Іншим варіантом є використання базових операцій над масивом за допомогою бібліотеки NumPy. Така операція буде забезпечувати попереднє утримання порогу даних для всіх елементів масиву, потім виконуючи просту функцію додавання чи віднімання. Для використання таких маніпуляцій необхідно використовувати перехід у інший тип даних, що може призвести до незначної втрати часу. Приклад лістингу коду представлений на рис. 4.4.

```
59      v = v.astype('int32')
60      v[v < (0 + value)] = 0 + value
61      v -= value
62      v = v.astype('uint8')
```

Рисунок 4.4 - Лістинг коду лінійної корекції за допомогою NumPy

Останнім методом лінійної корекції є доповнення попереднього алгоритму. У цьому випадку масив даних зображення не піддається додатковим перевіркам та перетворенням, лише при необхідності (переповнення ліміту). Таке вдосконалення позволить не навантажувати алгоритм постійними перевірками перенасичення, що прискорить сам алгоритм. Лістинг коду цього методу представлений на рис. 4.5.

```
38      if value >= 0:
39          lim = 255 - value
40          v[v > lim] = 255
41          v[v <= lim] += value
42      else:
43          v[v >= -value] -= -value
44          v[v < -value] = 0
```

Рисунок 4.5 - Лістинг коду лінійної корекції за допомогою лімітів

Слід зазначити, що різниця даних методів залежить тільки від підходу до обробки та зміні даних кадру, тому результати цих методів будуть ідентичні. Єдиною відзнакою буде лише швидкодія процесу.

Було проведено порівняльних аналіз алгоритмів. Тестування проводилося на відеозображенні с роздільною здатністю 1280x720 при значені розмірного коефіцієнта Size = 1. Результати приведені на рис. 4.6.

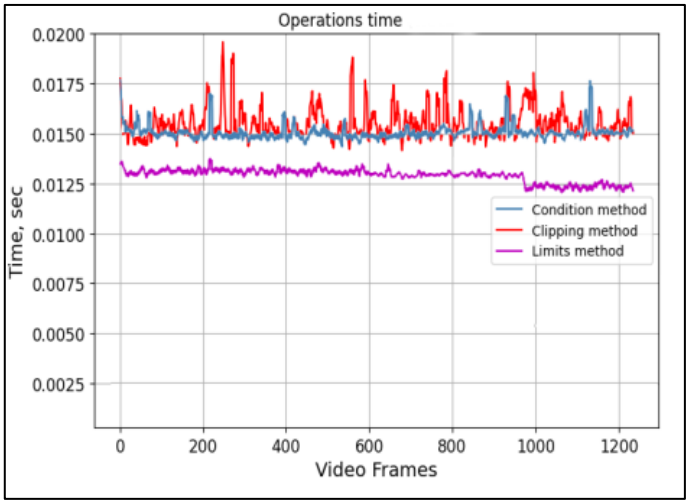


Рисунок 4.6 – Графік швидкодії запропонованих алгоритмів

Як видно з отриманих показників, найшвидшим варіантом є використання лімітів, через відсутніть додаткових навантажень на систему. Різниця у 0.0025 секунди на кадр позволить збільшити показник кадрів на секунду приблизно на 5-10 одиниць.

4.3 Вплив розмірного коефіцієнту на швидкодію лінійного алгоритму стабілізації

Було проведено дослідження залежності швидкодії алгоритму стабілізації яскравості від значення розмірного коефіцієнта. Так, відповідно до рис. 2.1, кожен кадр відеопотоку перетворюється у формат HSV і зменшується в розмірах для подальшої обробки. Робота алгоритму стійка до змін розмірності кадру, що дає змогу зменшити час обробки без втрати загальної якості. Отже, стоїть завдання оцінити рівень впливу розмірного коефіцієнта на швидкодію алгоритму. Отримані результати показано на рис. 4.7, вони являють собою графіки параметрів часу обробки та кількості кадрів за секунду від часу, вираженого в порядкових номерах кадрів.

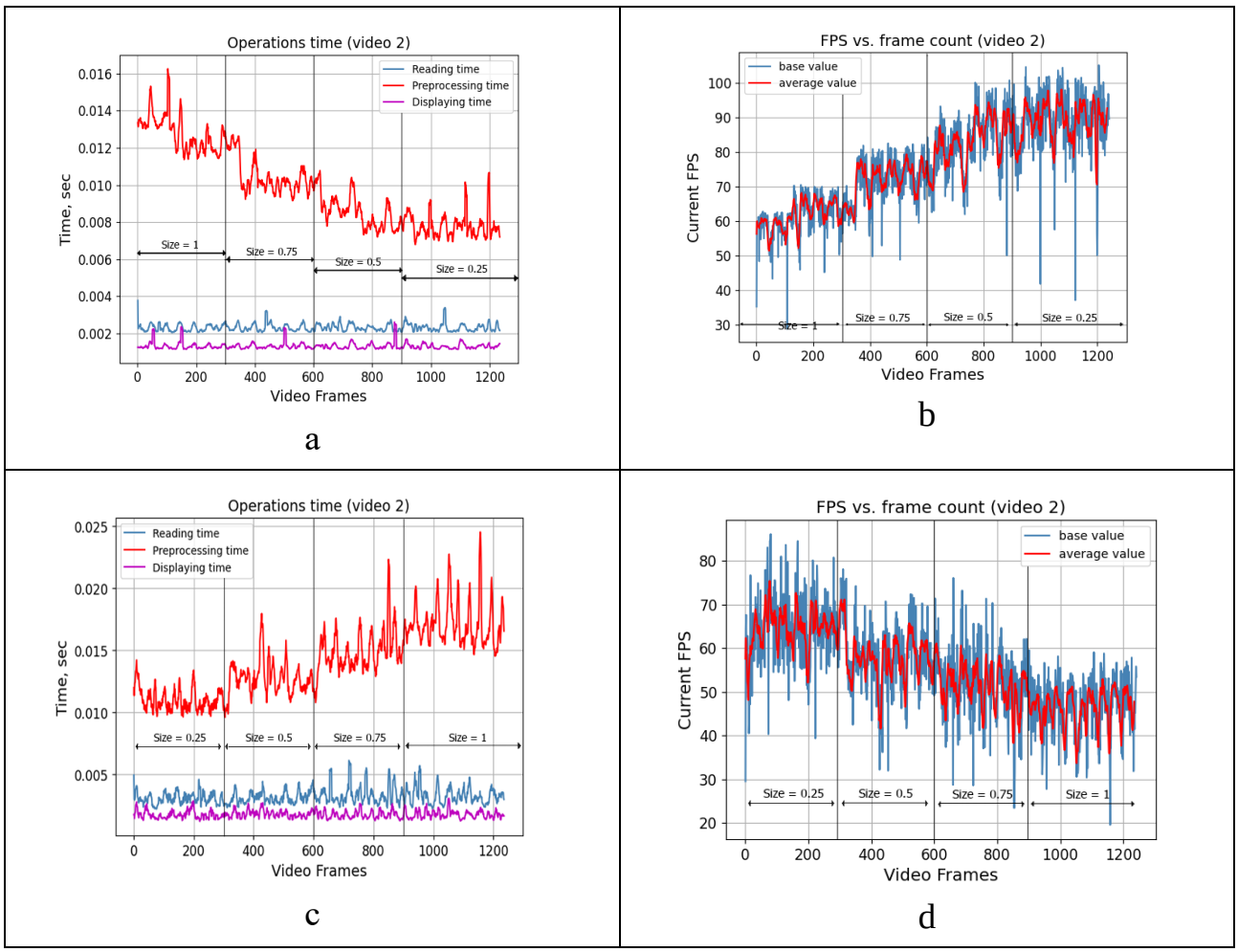


Рисунок 4.7 – Вплив розмірного коефіцієнту алгоритму на витрати часу

На рис. 4.7a та 4.7 c показано витрати часу на повне виконання алгоритму з урахуванням зчитування окремих кадрів (Reading time), обробки алгоритмом стабілізації яскравості (Preprocessing time) і відображення результату у вікні програми (Displaying time). При цьому в обох прикладах виділено часові

області, що належать до різних значень розмірного коефіцієнта (Size). На рис. 4.7 a виконується зменшення значення коефіцієнта з 1.0 до 0.25, а на рис. 4.7 c - збільшення з 0.25 до 1. Проаналізувавши отримані графіки, можна помітити, що час читання і відображення кадрів не залежать від розмірного коефіцієнта Size. Це повністю відповідає загальній ідеї, оскільки зміну розмірності виконують тільки в процесі алгоритму стабілізації яскравості для прискорення обчислювального етапу показника AFB, а читання і відображення виконують із кадром вихідного розміру.

Зменшення розмірного коефіцієнта призводить до зменшення часу опрацювання, однак при цьому виразно спостерігається ефект насичення. Так, для першого прикладу під час зменшення розмірного коефіцієнта зі значення 1.0 до 0.75 час обробки зменшився на 17 %, а зі значення 0.5 до 0.25 - лише на 6 %. Відповідно, надмірне зменшення розміру (Size < 0.25) уже не принесе відчутного (>5 %) прискорення роботи алгоритму.

Для оптимального значення розмірного коефіцієнта Size = 0.25, було розраховано процентне значення прискорення роботи алгоритму. У результаті отримано що для першого прикладу прискорення склало близько 43 %, для другого прикладу − 44 %. Зазначені тенденції можна побачити на рис. 4.7 b і рис. 4.7 d, оскільки значення Current FPS є зворотним значенням загального часу виконання. Отже, зменшення розмірного коефіцієнта призводить до збільшення FPS.

4.4 Вплив роздільної здатності вхідного кадру на швидкодію

Крім значення розмірного коефіцієнта, для оптимальної роботи алгоритму необхідно проаналізувати швидкість його роботи для різних розмірностей вихідного відеопотоку. Для цього було проведено дослідження, під час якого для постійного значення використовувалися відеодані роздільних здатностей 1280x720, 1920x1080, 2560x1440, 3840x2160. Отримані результати дослідження впливу розмірності вихідних кадрів на швидкодію представлені на рис. 4.8.

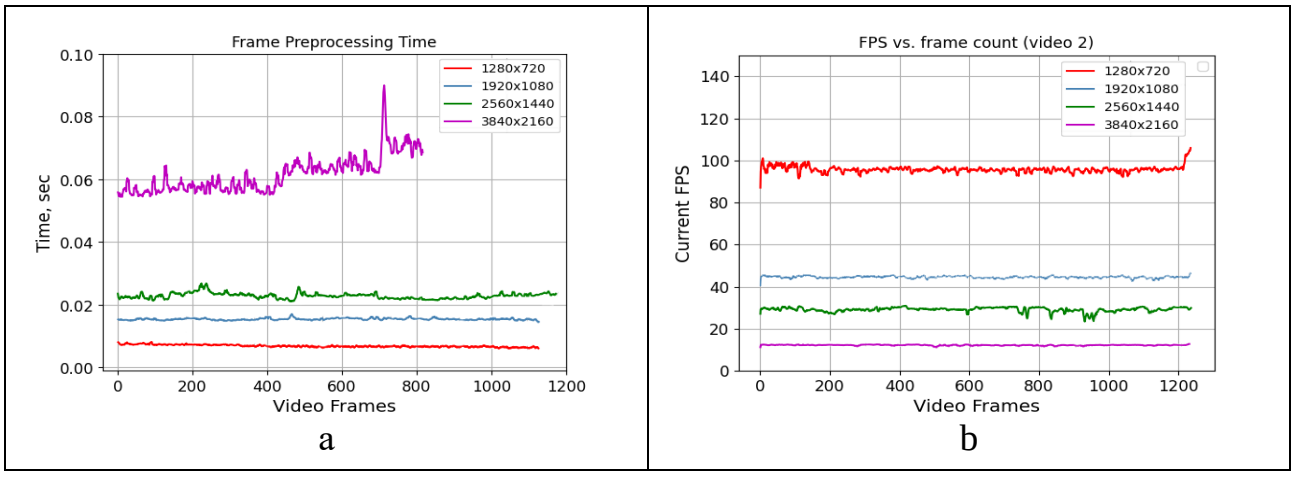


Рисунок 4.8 – Швидкодія алгоритму при різних розмірностях вхідного кадру

Можна помітити, що для відеоданих розмірності до 3840x2160 вдалося одержати достатній рівень швидкодії, що відповідає значенню FPS більше 30. Що стосується розміру 3840x2160 - швидкість роботи алгоритму абсолютно не дає змоги одержувати прийнятну якість вихідного відеопотоку, оскільки кількість кадрів за секунду перебуває на рівні 15.

Зазначимо, що дослідження проводилися на ноутбуці з невисокими характеристиками (2-х ядерний процесор Intel(R) Core(TM) i5-6200U @ 2.3 ГГц, об'єм ОЗП - 16 ГБ, дискретна відеокарта GeForce MX110) тому всі тимчасові показники алгоритму можуть змінитися при виборі більш потужного і спеціалізованого обладнання.

Останнє дослідження являє собою порівняльний аналіз роботи лінійного алгоритму стабілізації яскравості та алгоритму еквалізації каналу яскравості в режимі реального часу. Для перевірки було обрано відеофайл розмірності 1280x720 і розмірний коефіцієнт 0.25. Отримані результати впливу розмірності вихідних даних на швидкодію представлені на рис. 4.9. Тут video1 − еквалізація, а video2 – стабілізація яскравості.

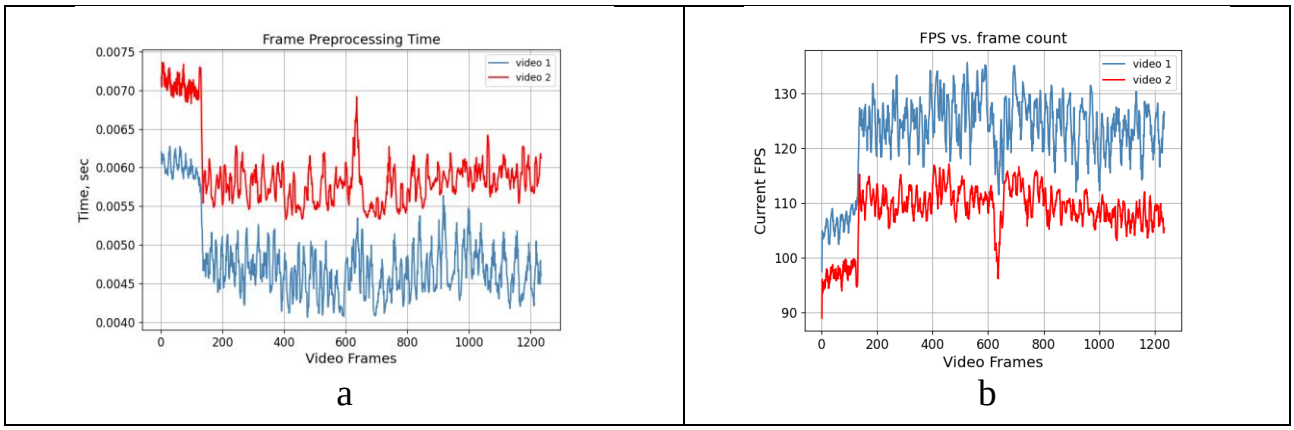


Рисунок 4.9 – Порівняльний аналіз швидкодії еквалізації та лінійної стабілізації

Проаналізувавши рис. 4.9,a і рис. 4.9,b можна зробити висновок, що алгоритм стабілізації яскравості незначно поступається алгоритму еквалізації за показником швидкодії. Так, різниця в FPS становить у середньому 15 кадрів на секунду. Однак при цьому важливо відзначити, що порівняльна повільність роботи є ціною за описані в попередніх розділах переваги. Також, під час роботи з відеоданими розмірності до 4К, алгоритм стабілізації яскравості володіє достатньою швидкодією (FPS >=30) для оптимальної роботи, тому дана різниця у швидкості роботи з еквалізацією не має жодного ефекту на виконання головного завдання, а саме підвищення якості вихідних даних за рахунок стабілізації яскравості.

4.5 Висновки

В даному розділі було проведено оцінку можливості реалізації алгоритму для роботи у режимі реального часу. Виходячи з отриманих показників швидкодії алгоритму, система може працювати у режимі 30 кадрів на секунду при роздільній здатності відео до 4К (3840x2160).

5. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

5.1 Програма досліджень «Video Stream Brightness Preprocessing»

Для оцінки ефективності пропонованих методів підвищення якості відеоданих були проведені експериментальні дослідження за допомогою програми «Video Stream Brightness Preprocessing». Вона написана на мові Python із залученням відповідних ресурсів бібліотеки OpenCV. Ця програма заснована на використанні алгоритмів введення відеоданих, прийнятих за допомогою web-камери або записаних відеофайлів, попередньої обробки та запису в файл отриманих даних.

Програма «Video Stream Brightness Preprocessing» дозволяє провести попередню обробку зареєстрованих даних. У програмі передбачена можливість запису інформації в відеофайли потрібного формату з різними кодеками стиснення і можливістю подальшого їх зберігання.

5.2 Вибір бібліотек для реалізації методів системи

Для реалізації методів підвищення яскравості були вибрані певні бібліотеки, які надають зручні та ефективні інструменти для роботи з зображеннями. Однією з ключових бібліотек є OpenCV (Open Source Computer Vision Library), яка надає багато функцій для обробки зображень, включаючи реалізацію гістограмної еквалізації та алгоритм лінійної стабілізації. Крім того, інші допоміжні бібліотеки, такі як NumPy та Matplotlib, використовуються для роботи з масивами даних та візуалізації зображень. Відображення вікон та інструментарію було виконано за допомогою бібліотеки PyQt5.

5.3 Реалізація основних алгоритмів роботи з відеофайлами

Для оптимізації та компонування програми, елементи систему були розділені на окремі файли. Процесс вводу та виводу відеозображення базується на базовому інструментарії бібліотеки OpenCV для мови програмування Python. При зчитуванні відеопослідовність розкладається на окремі кадри, що дозволяє безпосередньо впливати на відеоряд, рахуючи оброблену кількість кадрів. А додавання у програму функції `cv2.VideoCapture()` дозволяє використовувати відеозображення з камери у програмі.

Реалізація методів корекції відеозображень теж базується на бібліотеці OpenCV та мові програмування Python.

Метод еквалізації впроваджено у вигляді функції `equalize_value_channel (frame)`, яка приймає вхідний кадр `frame` та повертає еквалізований кадр зі зміненою яскравістю. У реалізації використовується базова функція `cv2.equalizeHist(v)`, яка є частиною бібліотеки OpenCV. Ця функція застосовує еквалізацію по обраному каналу зображення. Алгоритм еквалізації потребує переходу у кольоровий простір, що містить у собі окремий канал яскравості, тому перед виконанням функції проводиться перехід у колірний простір HSV за допомогою функції `brightness_hsv()`. Даний код виконується за допомогою стандартних операцій бібліотеки OpenCV.

Після застосування еквалізації, функція повертає відфільтрований кадр.

Описана реалізація алгоритму еквалізації дозволяє ефективно обробляти зображення, не використовуючи додаткові ускладнення, що можуть призвести до уповільнення усієї системи.

```
66  def equalize_value_channel(image_bgr):
67      """Еквалізація Value каналу HSV зображення"""
68      # Переводимо зображення до формату HSV
69      image_hsv = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2HSV)
70      # Відокремлюємо канали
71      h, s, v = cv2.split(image_hsv)
72      # Еквалізуємо канал яскравості Value (V)
73      equalized_v = cv2.equalizeHist(v)
74      # Відновлюємо HSV зображення з еквалізованним каналом яскравості
75      equalized_hsv = cv2.merge((h, s, equalized_v))
76      # Переводимо до формату RGB
77      equalized_rgb = cv2.cvtColor(equalized_hsv, cv2.COLOR_HSV2BGR)
78
79      return equalized_rgb
```

Рисунок 5.1 - Код, використаний для реалізації еквалізації

Лінійну стабілізацію впроваджено у вигляді функції `brightness_change(image_hsv, value)`, яка приймає вхідний кадр `frame` та різницю поточної яскравості з середнім показником за обраний част. Реалізація даного алгоритму було показано у розділі 4. Повертає функція кадр зі зміненою яскравістю. Вона теж потребує переходу у кольоровий простір HSV за домогою функції `equalize_value_channel (frame)`. Для можливості поточної зміни змінних алгоритму лінійної стабілізації використовується функція `resize(input_frame, size_multiplier)`, що змінює роздільну здатність кадру базуючись на перемикачах на інтерфейсі. Зміна розміру зображення виконується операндами бібліотеки OpenCV.

Лістинг коду класи опрацювання відеопослідовностей та обробки та корекції кадрів приведені у додатку Б.

5.4 Збереження та відображення показників якості

Для збереження великих масивів даних було впроваджено відведений для цього дата клас AdditionalInfo, код якого приведений у додатку Д. Впроваджені методи у класі дозволяють передавати дані без додаткового навантаження циклу роботи програми. Усі необхідні дані кожного кадру (середня яскравість, витрачений час) використані для побудови графіків та гістограм за допомогою функції `histogram(input_images)`.Їх відображення проводиться за допомогою бібліотеки NumPy та функцій `multi_plot()` та `plt.figure()`.

Код функцій побудови графіків приведений у додатку Г.

5.5 Інтерфейс програми

На рис. 5.1 показаний вигляд стартового вікна програми. Після натискання кнопки «Start» програма переходить у головне вікно програми для обробки вхідних даних.

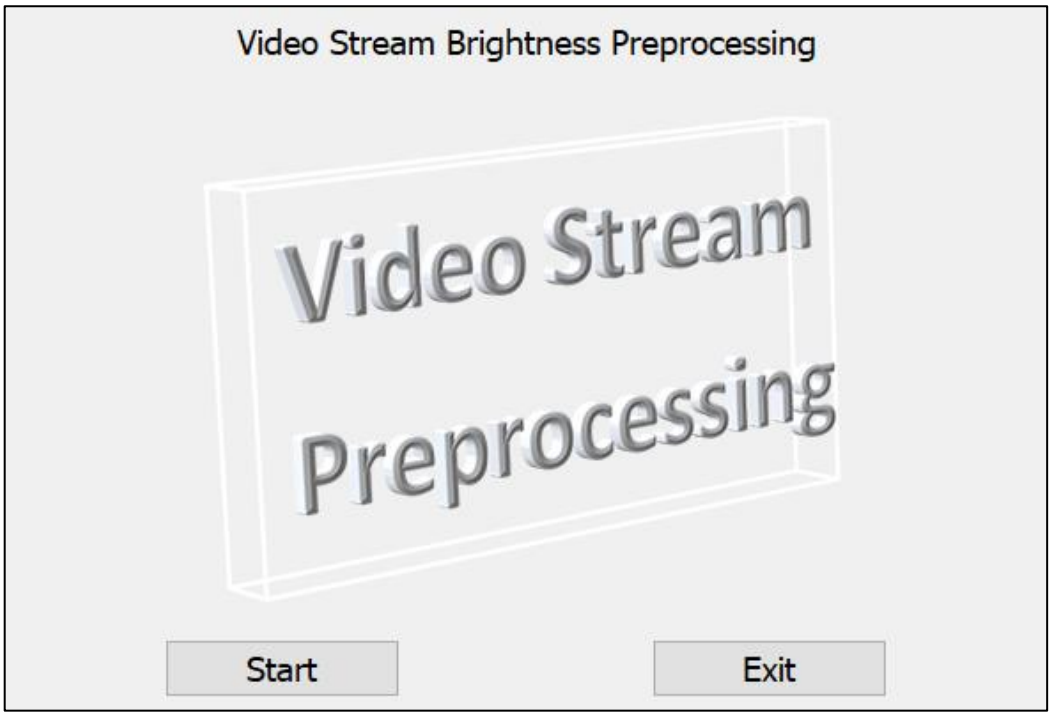


Рисунок 5.1 - Стартове вікно програми

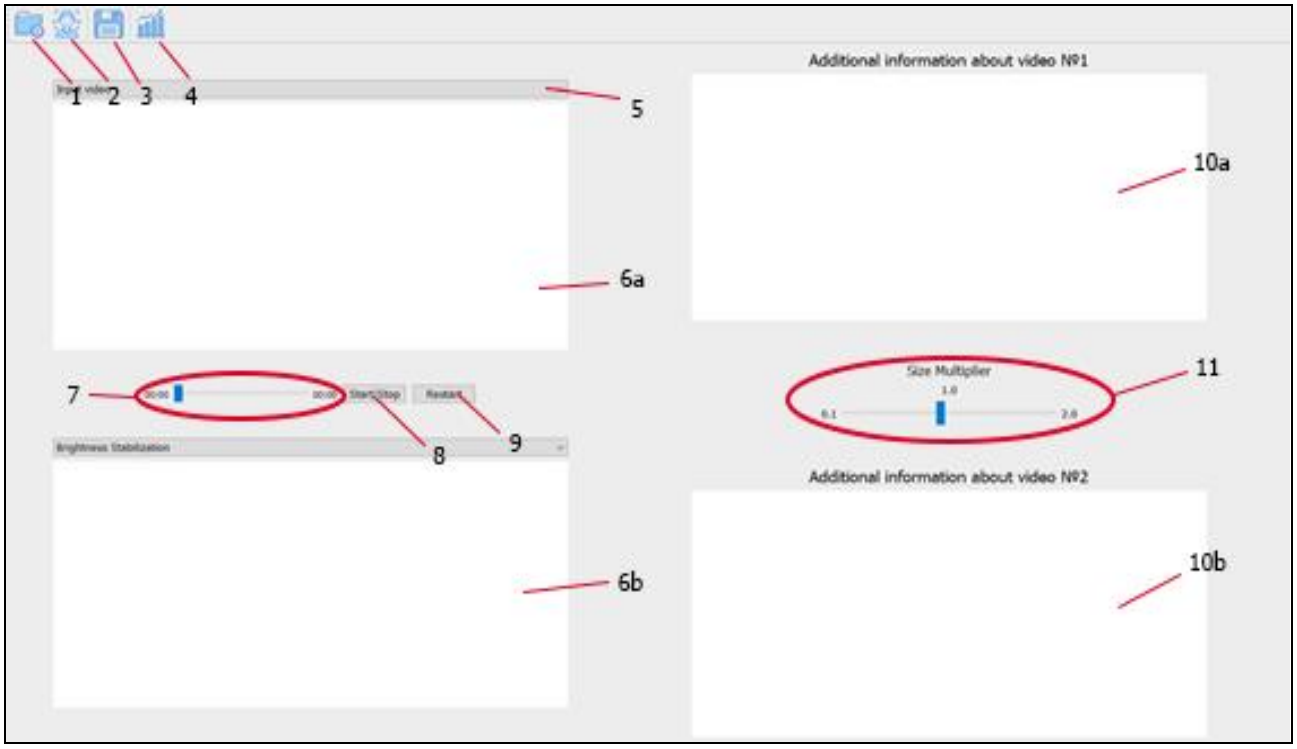


Рисунок 5.2 - Зовнішній вигляд головного екрану програми

Позначення на рис. 5.2 такі: 1 − іконка для вибору поточного відеофайлу для роботи; 2 − іконка для початку роботи з відеоданими, отриманими з web-камери;

3 − іконка для запису відеоданих у файл; 4 − іконка для побудови графіків залежностей параметрів відеопотоку; 5 − поле списку, що випадає, для вибору варіанта відображення кадру; 6a, 6b − вікна відображення потокових кадрів відеопотоку; 7 − часова шкала для відеопотоку; 8 − кнопка для запуску/зупинки відеопотоку; 9 − кнопка для перезапуску відеопотоку; 10a, 10b − інформаційні вікна для параметрів кожного з вікон відображення; 11 − повзунок адаптивного налаштування розмірного коефіцієнта алгоритму.

Інтерфейс побудований за допомогою бібліотеки PyQt5, лістинг коди приведені у додатку А.

Зв’язок інтерфейсу та основної частини програми виконується за домогою класу-контролеру, лістинг код якого приведений у додатку Б.

5.5.1 Панель інструментів

На верхній панелі зображено декілька розділів з кнопок. Перший розділ містить тип вхідних даних. Клік на іконку папки відкриє провідник для вибору відеофайлу для обробки, а іконка портрету запустить запис даних з відеокамери.

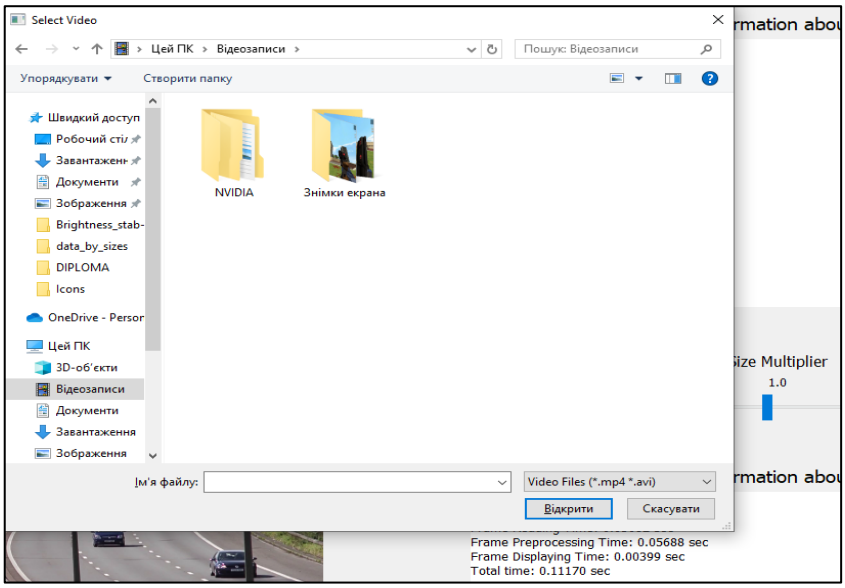


Рисунок 5.4 - Вікно вибору відеофайлу для відображення

Для повноцінного контролю якості відеоданих, одержуваних за допомогою пропонованих алгоритмів і програмних засобів, необхідно не тільки забезпечувати можливість введення даних в реальному часі з високим FPS, але і записувати отримані дані для повторного аналізу. Для цього в інтерфейсі програми передбачено кнопку з зображенням дискети «Save file», що знаходиться у другому розділі панелі інструментів основного вікна перегляду відеоданих, представлений на рис. 5.5. Після її натискання починається програвання відео з початку (при використанні файлових відео) та запис відеоінформації в файл. Після повторного натискання кнопки реєстрація закінчується.

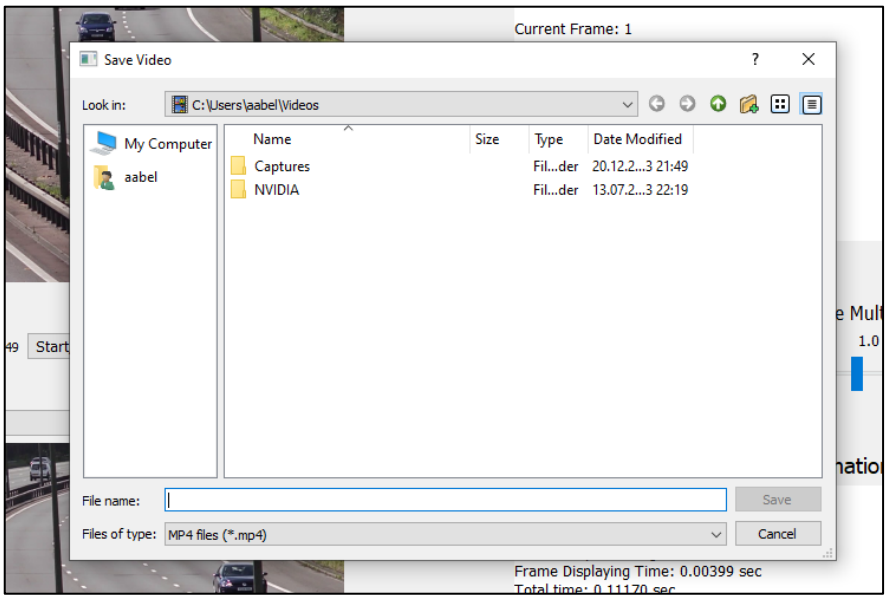


Рисунок 5.5 - Збереження і розміщення відеофайлів в архіві

Кнопка на панелі «Data Plotter» відповідає за розрахунок якісних показників та побудову графіків. Натискання почне запис даних з обох вікон додатку. При повторній активації запис закінчиться і будуть побудовані графіки, що використовувалися у минулих розділах.

5.5.2 Вікно відображення відео

Додаток має два вікна відображення відеопослідовностей та візуального порівняння та аналізу (рис. 5.6). Над вікнами присутні списки, що випадають, де можна обрати тип відображуваного зображення (оригінальне, оброблене методом гістограмної еквалізації, оброблене методом лінійної стабілізації).

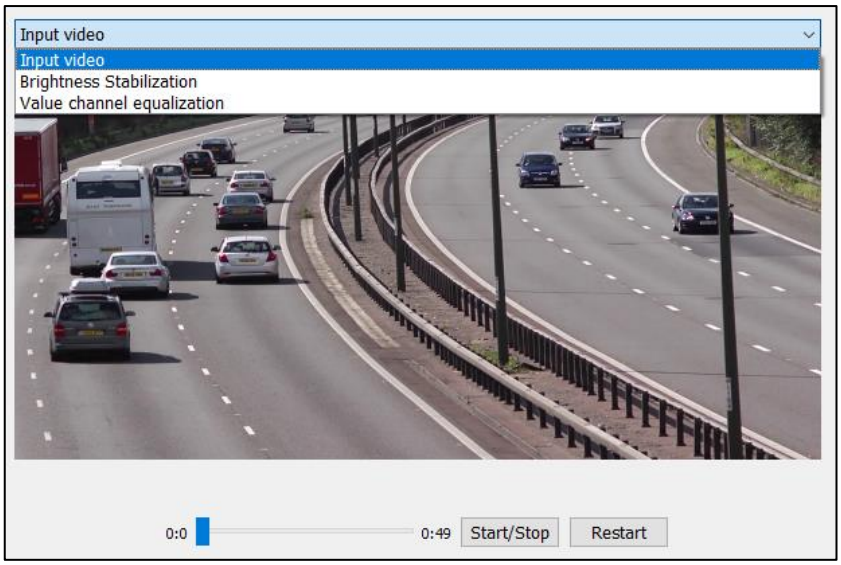


Рисунок 5.6 - Вікно відображення відеозображення

Поміж вікон розташовано плеєр-маніпулятор, що дозволяє перемотувати відео та ставити на паузу. Збоку вікон розташовано додаткові поля з інформацію про роздільну здатність зображення та швидкодію обраного методу, що представлені на рис. 5.7

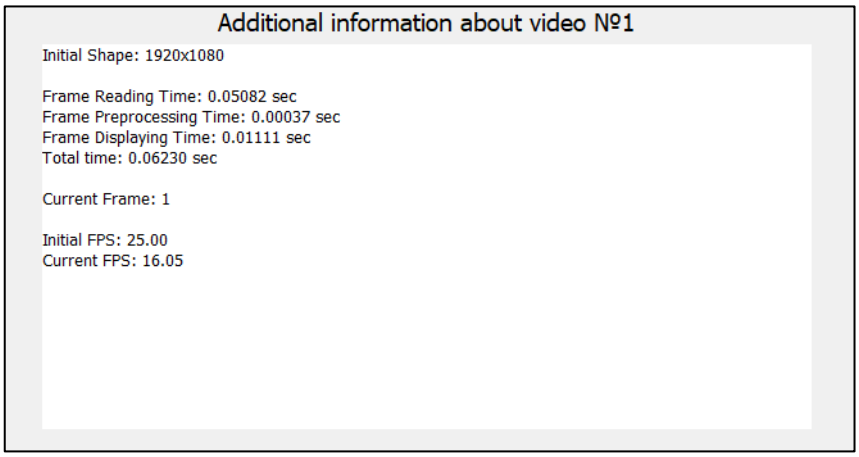


Рисунок 5.7 - Панель додаткової інформації про відеопослідовність

5.6 Тестування розробленого продукту

У процесі розробки програми проводилося поетапне тестування з метою виявлення програмних помилок і невідповідностей ТЗ (технічному завданню). Для цього були використано декілька персональних систем з комплектуючими різних потужностей.

Программний продукт, що тестується, послідовно запускався на цих системах, його поведінку було аналізовано, і, за необхідності, за результатами аналізу вносилися зміни до коду. Для тестування окремих модулів роботи з дата-класов у текст програми було внесено спеціальні функції, що дають змогу аналізувати базу даних і, в разі підозри на помилку, виводять повідомлення в системний журнал. Вони також відомі як юніт-тести. У разі змін у базі даних проводилася перевірка цілісності (перевірку на відповідність ключів - індексів).

Було проведено наведені нижче тести:

1) кожен клас та функцію було піддано юніт-тестуванню з метою виявлення помилок, викликаних невідповідністю очікуваних і отриманих параметрів. У разі аномальної поведінки активності або її збою, поведінка аналізувалася та помилка виправлялася;

2) кожен клас та функцію було піддано юніт-тестуванню з метою виявлення помилок, викликаних невідповідністю очікуваних і отриманих параметрів. У разі аномальної поведінки активності або її збою, поведінка аналізувалася та помилка виправлялася;

3) у програму навмисно вносилися неприпустимі дані у постійні та змінні, які могли бути невірно інтерпретовані програмою. Аналізувалася поведінка системи під час обробки неприпустимих даних;

4) застосунок було запущено на пристроях, що працюють під управлінням різних версій Windows з метою виявлення особливостей роботи застосунку, запущеного в різних операційних системах;

5) після завершення циклу розробки, програмний продукт тестувався на реальних пристроях. За результатами тестування було що продукт задовільно працює навіть на застарілих системах.

5.6 Висновки

Було розроблено систему для дослідження якості перетворень яскравості кадру та оцінки швидкодії системи. Для реалізації програмного забезпечення було використано мову Python та її бібліотеку OpenCV для обробки відеоданих. Інтерфейс буд побудован за допомогою вібліотеки PyQt5. Було проведено тестування програмного продукту на відповідність технічному завданню та виявлення технічних помилок.

6. ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Розвиток науково-технічного прогресу на сучасному етапі значною мірою залежить від впровадження обчислювальної техніки в усі сфери людської діяльності, в тому числі і в науково-дослідні проекти. Створювані на базі комп'ютерів інформаційні системи дозволяють обробляти велику кількість інформації і своєчасно приймати управлінські рішення в різних сферах, де потрібна обробка великої кількості даних. А це в свою чергу вимагає розробки засобів програмного забезпечення. За оцінками фахівців витрати на розробку програмного забезпечення ростуть швидше, ніж витрати на створення обчислювальної техніки, тому розробники програм повинні враховувати не тільки технічну, а й економічну доцільність розробки. Економічна доцільність розробки і впровадження програмного забезпечення визначається економічним ефектом, який буде отриманий виробниками при їх реалізації і споживачами при їх використанні. За величиною очікуваного економічного ефекту приймається рішення про доцільність інвестицій в розробку того чи іншого програмного продукту. За характером об'єкта вкладень інвестиції в розробку програмного забезпечення відносять до інтелектуальних інвестицій.

При створенні програмного продукту дуже важливо оцінити його вартість. Це тим більш важливо, якщо програмний продукт є затребуваним і має всі шанси конкурувати вийти на ринок програмного забезпечення.

Існують способи і рекомендації до розрахунку собівартості розробки і написання програмного забезпечення .

Обсяг вихідних текстів програми, перш за все, відображає трудомісткість і тривалість розробки програмного забезпечення і дозволяє оцінювати відносні показники продуктивності праці фахівців-розробників. Обсяг програм в сучасних публікаціях наводиться в різних одиницях, які можна розділити на дві групи:

1) група, яка характеризує обсяг початкового програмного коду, які розробляються й аналізуються програмістом (це символи в початковому тексті програми на будь-яких мовах програмування; лексеми, які об'єднують групи символів, що мають загальне смисловий зміст в тексті програми; оператори мови програмування рівня асемблера; рядки тексту програми на мові програмування високого рівня);

2) група, яка відображає обсяг програми, що розміщується в реалізує ЕОМ (це байти, зайняті текстом програми в пам'яті ЕОМ; команди або операції реалізує , складові текст програми в об'єктних коді; слова пам'яті, обумовлені структурою даної реалізує , які використовуються для зберігання виконуваної програми і / або бази даних при функціонуванні програмних засобів).

Обсяг програми, що розміщується на , впливає на характеристики і вартість машин, яка залежить від необхідної пам'яті і продуктивності. З огляду на, що при розробці програми " Video Stream Brightness Preprocessing»" вибір не проводився і конкретні вимоги до реалізує машині не пред'являлися, будемо користуватися одиницями першої групи.

6.1 Основні статті витрат при розробці програми

Основна праця фахівця, який розробляє програмне забезпечення, вкладається в розробку тексту програми і розробку алгоритмів, за якими текст написаний. Бажано, щоб обрана одиниця виміру була б найбільшою мірою адекватна трудомісткості розробки. Крім того, одиниця виміру обсягу повинна бути наочною і просто вимірюванню. З цих позицій застосування числа лексем для характеристики обсягу програми поки скрутна, тим більше що відсутній досвід використання цього показника. Таким чином, базовим показником для визначення складових витрат праці є умовне число операторів в програмі.

Різні джерела радять вважати за число операторів в програмі наступні величини:

- число команд на мові асемблера;
- число логічних операторів в програмі, операторів переходу, арифметичних операторів і інших операторів у вихідному коді програми;
- число рядків в програмі (для мов високого рівня).

Програма "Image Brightness Preprocessing" розроблялася на мові високого рівня Python.

При його розробці були враховані такі сучасні рекомендації до структурному програмуванню як відсутність умовних і безумовних переходів, запис операторів в один рядок (за несуттєвими винятками), лінійний підхід до програмування. Отже, за число операторів в програмі можна взяти число рядків в програмі. Слід зазначити, що в це число не входять коментарі, вказівки і

заголовки, так як ці конструкції не використовуються при нормальному функціонуванні програми.

6.1.1 Розрахунок трудомісткості розробки програмного забезпечення

Базовий показник для визначення складових витрат праці обчислюється за формулою:

$$Q = q * c\ (1 + p), \tag{6.1}$$

(де q - число операторів (вихідних команд) в програмному продукті, так само 1086; с - коефіцієнт складності програми; р - коефіцієнт корекції програми в ході її розробки, залежить від точності і коректності поставленого завдання - приймаємо рівним 0.06.

Коефіцієнт складності програми визначається з табл. 6.1 на перетині "групи складності" і "ступеня новизни". При цьому новизна визначається за принципом: А - розробка принципово нових завдань, Б - розробка оригінальних програм, В - розробка програм з використанням типових рішень, Г - разова звичайне завдання. А складність визначається виходячи з типу вирішуваних завдань: 1 - алгоритми оптимізації і моделювання систем, 2 - завдання обліку, звітності та статистики, 3 - стандартні алгоритми. Крім того, в таблиці вказано коефіцієнт недостатності опису програми, який буде потрібно трохи пізніше.

Таблиця 6.1 - Коефіцієнти розрахунку трудомісткості

Мова програмування	Група складності	Ступінь новизни				коефіцієнт В
		А	Б	В	Г	
високого рівня	1	1,38	1,26	1,15	0,69	1,2
	2	1,30	1,19	1,08	0,65	1,35
	3	1,20	1,10	1,00	0,60	1,5
низького рівня	1	1,58	1,45	1,32	0,79	1,2
	2	1,49	1,37	1,24	0,74	1,35
	3	1,38	1,26	1,15	0,69	1,5

Програма "Image Brightness Preprocessing" написана на мові високого рівня, відноситься до моделювання систем і є принципово новою розробкою; тобто коефіцієнт складності програми в даному випадку: з = 1,38. Таким чином, знаходимо базовий показник: Q = тисяча п'ятсот вісімдесят дев'ять.

Далі, розрахуємо складові витрати праці, серед яких виділяють: витрати праці на підготовку і опис алгоритму, витрати праці на дослідження алгоритму, витрати праці на розробку алгоритму, витрати праці на програмування і налагодження та витрати праці на підготовку документації. Майже всі ці параметри будуть залежати від базового показника, що розраховується за формулою (6.1). Витрати праці на підготовку і опис завдання може визначатися емпірично або за формулою (6.2):

$$t_{\text{оп}} = \frac{T_{min} + 4 \cdot T_{\text{нв}} + T_{max}}{6} = \frac{26 + 4 \cdot 52 + 78}{6} = 52 \text{ [люд. год]} \qquad (6.2)$$

де T_{max} - трудомісткість операції в найбільш несприятливих умовах (песимістична оцінка); Tmin - трудомісткість операції при сприятливих умовах (оптимістична оцінка); Тнв - трудомісткість операції при нормальних умовах (найбільш ймовірна оцінка). Орієнтовні величини оцінки трудомісткості операції підготовки опису завдання в залежності від числа операторів q наводяться в табл. 6.2.

Таблиця 6.2 - Оцінка часу підготовки опису завдання

q	Tmin	$T_{\text{нв}}$	Tmax
100	10	15	20
500	20	35	50
1 000	25	50	75
1500	30	60	90
2000	40	70	100
2500	50	80	110
5000	70	110	150
10000	100	150	200

Витрати праці на дослідження алгоритму розв'язання задачі визначаються формулою (6.3):

$$t_{\text{ис}} = \frac{Q \cdot B}{(75 \div 85)k} \text{ [люд. год]} \qquad (6.3)$$

де Q - базовий коефіцієнт, що розраховується за формулою (6.1), B - коефіцієнт недостатності опису завдання, який береться з таблиці 3 і дорівнює 1,2; k - коефіцієнт кваліфікації програміста, залежить від стажу працівника і визначається з табл. 6.3.

Таблиця 6.3 - Коефіцієнти кваліфікації програміста

Досвід роботи	коефіцієнт кваліфікації
До двох років	0.8
2-3 роки	1
3-5 років	1.1 - 1.2
5-7 років	1.3 - 1.4
більше 7 років	1.5 - 1.6

По таблиці визначаємо коефіцієнт k = 0.8.

Таким чином, знаходимо витрати праці на дослідження алгоритму розв'язання задачі: tic = 30 [люд.год].

Витрати праці на розробку блок-схем алгоритмів (представлених на малюнках 2.7 і 2.8) обчислюються за формулою (6.4):

$$t_{\text{ал}} = \frac{Q}{(20 \div 25)k} = 88[\text{люд. год}] \quad (6.4)$$

Витрати праці на програмування алгоритму по блок-схемі і налагодження програми обчислюються за формулами (6.5, 6.6):

$$t_{\text{пр}} = \frac{Q}{(20 \div 25)k} = 88[\text{люд. год}] \quad (6.5)$$

$$t_{\text{отл}} = \frac{Q}{(20 \div 25)k} = 88[\text{люд. год}] \quad (6.6)$$

Витрати праці на підготовку документів по завданню складаються з витрат праці на підготовку рукописів і часу на оформлення документів і обчислюються за формулою (6.7):

$$t_{\partial} = t_{\text{рук}} + t_{\text{оф}} = \frac{Q}{(20 \div 25)k} + 0.75 \cdot t_{\text{рук}} = 113 + 88 = 198[\text{люд. год}] \quad (6.7)$$

Сумарні витрати праці розраховуються як сума складових витрат праці за формулою (6.8):

$$t_c = t_{\text{оп}} + t_{\text{ис}} + t_{\text{ал}} + t_{\text{пр}} + t_{\text{отл}} + t_{\partial} = 52 + 30 + 88 + 88 + 397 + 198 = 853[\text{люд. год}] \quad (6.8)$$

7.1.2 Розрахунок витрат на розробку програмного забезпечення

Заробітна плата складається з двох складових: основної заробітної плати і додаткової.

Основна заробітна плата розраховується за формулою (6.9):

$$З_{\text{осн}} = \frac{t_{\Sigma}}{t_{\text{ср}} \cdot 8} \cdot \text{ТС}[\text{грн}] \quad (6.9)$$

де t_{Σ} - сумарні витрати праці, які обчислюють за формулою (6.2); $t_{\text{ср}}$ - середня кількість днів у місяці, так само 21-го дня, множиться на кількість годин в робочому дні - 8; ТС - тарифна ставка.

Тарифна ставка являє собою МРОТ (мінімальний розмір оплати праці, станом на грудень 2023 року дорівнює 6700 гривень [Закон України 12 січня 2023 року № 357/96-ВР "Про мінімальний розмір оплати праці"]), збільшений в залежності від тарифного коефіцієнта, яке відповідає даному виду робіт. Для 12-го розряду робіт [від 1.10.2022 пост. КМУ № 29 від 20.01.2021], який відповідає роботі програміста, тарифний коефіцієнт дорівнює 2.44. Таким чином, основна заробітна плата буде становити:

$$З_{\text{осн}} = \frac{853}{21 \cdot 8} \cdot 6700 \cdot 2.44 = 83005[\text{грн}]$$

Додаткова заробітна плата становить 20% від основної заробітної плати, розраховується за формулою (6.10):

$$З_{\text{дод}} = 0.2 \cdot З_{\text{осн}} = 16601[\text{грн}] \quad (6.10)$$

Сумарна заробітна плата (або фонд заробітної плати, ФЗП) обчислюється як сума основної і додаткової заробітних плат за формулою (6.11):

$$\text{ФЗП} = З_{\text{дод}} + З_{\text{осн}} = 99606[\text{грн}] \quad (6.11)$$

6.2 Додаткові статті витрат

Серед додаткових статей витрат на розробку програмного забезпечення виділяють: витрати на матеріали і комплектуючі (вартість самого обладнання, тобто комп'ютера, до уваги не береться), відрахування на соціальне

страхування, накладні витрати, амортизаційні відрахування, витрати на технічне обслуговування обладнання і вартість витраченої електроенергії при роботі на комп'ютері. Вартість обладнання хоч і не включається в собівартість розробки програмного забезпечення, але все ж використовується при розрахунку деяких інших додаткових статей витрат. При написанні програми на в якості обладнання передбачається персональний комп'ютер, вартість якого становить: $C_{\text{обор}}$ = 26400[грн].

Витрати на матеріали і комплектуючі, які використовуються в процесі написання програмного продукту (Смик), а також витрати на технічне обслуговування і ремонт (СТО) складають, відповідно, 1.5% і 2.5% від вартості обладнання - формули (6.12 - 6.13):

$$\mathrm{C_{ММК}} = 0.0153 \cdot \mathrm{C_{обор}} = 396[\text{грн}] \quad (6.12)$$

$$\mathrm{C_{ТО}} = 0.025 \cdot \mathrm{C_{обор}} = 660[\text{грн}] \quad (6.13)$$

Амортизаційні відрахування, процес поступового перенесення вартості засобів праці у міру їх фізичного та морального зносу на вартість вироблених з їх допомогою продукції з метою акумуляції коштів для подальшого повного відновлення. Амортизаційні відрахування провадяться за встановленими нормами амортизації, виражаються, в процентах до балансової вартості обладнання і розраховуються за формулою (6.14):

$$\mathrm{A_{год}} = \mathrm{C_{обор}} \cdot \frac{H_{\mathrm{A}}}{100} \quad (6.14)$$

де $\mathrm{C_{обор}}$ - вартість комп'ютера; H_{A} - норма амортизації, яка розраховується за формулою (6.15):

$$H_{\mathrm{A}} = \frac{\mathrm{C_{обор}} - \mathrm{C_{лікв}}}{T_{\text{норм}} \cdot \mathrm{C_{обор}}} \cdot 100\% \quad (6.15)$$

де $\mathrm{C_{лікв}}$ - ліквідаційна вартість, становить 5% від вартості обладнання: $\mathrm{C_{лікв}}$ = 0.05 · $C_{\text{обор}}$ = 1320[грн]; $T_{\text{норм}}$ - нормативний термін служби (для персонального комп'ютера приймемо $T_{\text{норм}}$ = 6 [років]). Таким чином, отримуємо: $\mathrm{A_{год}}$ = 4 180 [грн].

Варто також враховувати і витрати електроенергії при написанні програмного забезпечення. Вартість електроенергії обчислюється за формулою (6.16):

$$\mathrm{C_{ЕЕ}} = M \cdot k_{\text{З}} \cdot F_{\text{еф}} \cdot C_{\text{кВт.год}} 100\% \quad (6.16)$$

де М - потужність (450 Вт); $k_З$- коефіцієнт завантаження (0.8); $C_{\text{кВт.год}}$- вартість 1 кВт-год електроенергії (2 грн. 31 коп. На момент написання цієї роботи); $F_{\text{еф}}$- ефективний фонд робочого часу, розраховується за формулою (6.17):

$$F_{\text{еф}} = Д_{\text{ном}} \cdot d \cdot (1 - \frac{f}{100}) \quad (6.17)$$

де $Д_{\text{ном}}$ = 258 - номінальне число робочих днів у році; d = 8 - тривалість робочого дня [год]; f = 2% - планований відсоток часу на ремонт.

При даних значеннях параметрів і коефіцієнтів вартість електроенергії складе C_{EE} = 1 682 [грн].

Однак, отримані значення амортизаційних відрахувань і витрат на електроенергію - значення річних витрат, необхідно їх скорегувати відповідно до тимчасового коефіцієнтом, який визначається виходячи з сумарних річних експлуатаційних витрат, які розраховуються за формулою (6.18):

$$E_з = t_\Sigma \cdot \frac{C_E}{F_{\text{еф}}} \quad (6.18)$$

де $C_E = C_{EE} + C_{TO} + A_{\text{год}}$ - сумарна річна вартість експлуатаційних витрат, $F_{\text{еф}}$- ефективний фонд робочого часу, обчислений за формулою (6.17), t_Σ- загальний час використання для вирішення завдання, яке обчислюється аналогічно формулі (7.8), враховуючи лише час роботи на комп'ютері:

$$t = t_{\text{пр}} + t_{\text{отл}} + t_{\text{д}} = 88 + 397 + 198 = 683[\text{год}].$$

Отже, сумарні витрати на експлуатацію становитимуть: $E_з$= 2 203 [грн], а сам тимчасової коефіцієнт обчислюється за формулою (6.19):

$$E_з = \frac{E_з}{C_E} \quad (6.19)$$

Таким чином, з огляду на тимчасової коефіцієнт, з сумарних експлуатаційних витрат скорегуємо:

витрати на електроенергію $C_{EE} = C_{EE} \cdot w$ = 568 [грн];

амортизаційні відрахування $A_{\text{год}} = A_{\text{год}} \cdot w$ = 1411 [грн].

Відрахування на соціальне страхування становлять 22% від усієї заробітної плати [Закон від 08.07.2010 № 2464-VI «Про збір та облік єдиного внеску на загальнообовязкове державне соціальне страхування»], обчислюються за формулою (6.20):

$$СС = 0.22 \cdot ФЗП = 21913[грн]. \quad (6.20)$$

Накладні витрати, пов'язані з управлінням і обслуговуванням, утриманням та експлуатацією устаткування Ві іншими додатковими витратами на забезпечення процесів виробництва і обігу, становлять 50% від фонду заробітної плати, визначаються за формулою (6.21):

$$С_{накл} = 0.5 \cdot ФЗП = 49803[грн]. \quad (6.21)$$

6.3 Результуюча таблиця собівартості

Сумарні витрати на розробку програмного забезпечення вважаються як сума фонду заробітної плати, експлуатаційних витрат, витрат на соціальне страхування, накладних витрат і витрат на матеріали і комплектуючі.

Підсумкова вартість розробки програмного забезпечення представлена в табл. 6.4.

Таблиця 6.4 - Результуюча таблиця собівартості

Стаття витрат		Сума, грн.	У відсотках від загальної суми
ФЗП	$З_{осн}$	83 005	47.60
	$З_{дод}$	16 601	9.52
Накладні витрати, $С_{накл}$		49 803	28.56
Соціальне страхування, СС		21 913	12.56
Експлуатаційні витрати	$С_{ЕЕ}$	568	0.32
	$С_{ТО}$	660	0.37
	$А_{год}$	1 411	0.8
Матеріали і комплектуючі, $С_{мик}$		396	0.22
Разом:		174 357	

Кількість замовлених екземплярів має бути не менше ніж 200 штук. Виробничу собівартість одного екземпляра програмного продукту (ВС) Визначається за формулою:

$$ВС_0 = ВС/кпп, \quad (6.22)$$

де ВС — виробничі витрати;

КПП — кількість замовлених екземплярів.

$$ВС_0 = \frac{174\,357}{200} = 871{,}78 \quad (6.23)$$

Повна собівартість одного екземпляру програмного продукту СП складається з суми виробничої собівартості ВСП, адміністративних витрат АВ і витрат на збут ВЗ, які приходяться на один екземпляр програмного продукту:

$$СП_0 = ВС_0 + АВ_0 + Вз_0 \quad (6.24)$$

Адміністративні витрати $АВ_0$, які приходяться на один екземпляр програмного продукту, визначається за формулою:

$$АВ_0 = \frac{АВ}{КПП} = \frac{20751}{200} = 103{,}75 \text{ грн.} \quad (6.25)$$

Витрати на збут $ВЗ_0$, які приходяться на один екземпляр програмного продукту, визначаємо за формулою:

$$ВЗ_0 = \frac{ВЗ}{КПП} = 21{,}79 \text{ грн.} \quad (6.26)$$

Таким чином, $СП_0$= 871,78+103,75+21,79= 215,4 грн.

Таблиця 6.5 — Розрахунок собівартості і ціни виробу за статтями

№	Статті	Сума, грн
10	Витрати на збут	4358,92
11	Собівартість власних робіт	199 466
12	Прибуток (П)	39 893
13	Ціна без НДС	239 359
14	НДС	47 871
15	Ціна з НДС	287 231

Рентабельність продукції (норма продукту) — це відношення загальної суми прибутку до витрат виробництва і реалізації продукції (відносна величина прибутку, що припадає на 1 грн поточних витрат):

$$Р_п = \frac{Ц - ВС}{ВС} * 100\% = \frac{287231 - 174357}{174357} * 100\% = 64\% \quad (6.27)$$

Отже рентабельність 45%. Розрахуємо величину оптової ціни одного виробу ЦПП (без врахування НДС).

$$ЦПП = СП * (1 + Р_п /100) = 997*(1+64/100) = 1635 \quad (6.28)$$

Де $Р_п$ — коефіцієнт рентабельності.

Розрахуємо точку беззбитковості. Дохід від реалізації програмних продуктів знаходимо множенням ціни одного ПП на кількість замовлених примірників ПП:

$$ДР = ЦПП*КПП = 1635*200 = 327\ 000 \text{ грн.} \quad (6.29)$$

Аналітичний розмір критичної програми (РКП) розраховують діленням постійних витрат $РП_{0СТ}В$ на різницю між ціною одного програмного продукту ЦПП і змінними витратами, які приходяться на один екземпляр програмного продукту ($З_М В_0$), тобто

$$РКП = \frac{РП0СТВ}{(ЦПП - З_м В_0)} = \frac{76893}{1635 - 106,42} = 50 \text{ один.} \quad (6.30)$$

Річні постійні витрати $РП_{0СТ}В$ складаються із суми наступних витрат:

$$РП_{ост}В = ВОУ + А_м + ДВ + АВ + ВЗ = 3460 + 1980,27 + 49803 + 20751 + 4358 = 76893 \text{ грн.} \quad (6.31)$$

Річні зміни витрати $РЗ_м В$ склаються із суми наступних витрат:

$$РЗ_м В = ВМ + ФОП + ЄСФОП = 1621 + (13496 + 2728) + 3460,5 = 21305,5 \text{ грн.} \quad (6.32)$$

Змінні витрати, які приходяться на один екземпляр програмного проукту, визначаємо діленням річних змінних витрат на річну програму випуску продукту:

$$З_м В_0 = \frac{РЗ_м В}{КПП} = \frac{21305.5}{200} = 106,42 \text{ грн.} \quad (6.33)$$

Графік критичної програми випуску продукту представлений на рис. 6.1

Рисунок 6.1 – Критична програма випуску продукту

6.4 Висновки

В результаті економічних розрахунків було проведено економічне обґрунтування розробки програмного продукту. Розрахунок вартості проведений з урахуванням всіх трудовитрат, ПДВ, відрахувань в пенсійний фонд, у фонд зайнятості і відрахувань на соціальне страхування, накладних витрат. Розрахована собівартість розробки веб-додатку та впровадження до ринку дорівнює 174 357 грн.

ЗАКЛЮЧЕННЯ

В даному дипломному проекті було розроблено систему попередньої обробки яскравості відеозображення з метою покращення роботи систем штучного інтелекту. Розробка системи проводилася у три етапи.

Перший етап полягав у дослідженні існуючих методів та визначені основних критеріїв оцінки роботи системи . Визначена схема та алгоритм системи.

Другий етап полягав у визначенні оптимальних операндів алгоритму. Було проведено аналіз показників якості існуючих варіантів вирішення завдання та запропонованого алгоритму. На основі експериментальних даних було визначено найефективніші коефіцієнти системи стабілізації.

На третьому етапі було розроблено додаток для реалізації та демонстрації запропонованої системи попередньої стабілізації яскравості відеозображення.

Для реалізації програмного забезпечення було використано мову Python та її бібліотеку OpenCV для виконання лінійних операцій над відеозображенням. Також було використано бібліотеку PyQt5 для розробки користувацького інтерфейсу додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. European Patent № EP 1 333 660 A3, МПК H 04 N 1/60, опубл. 09.03.2005 р., Бюл. № 10.

2. World Intellectual Property Organization, патент № WO 2010/043996 A3, МПК H 04 N 1/60, опубл. 22.04. 2010 р.

3. European Patent № EP 1 326 433 B1, МПК H 04 N 9/77, опубл. 11.04.2012 р., Бюл. № 15

4. Y. T. Kim, “Contrast Enhancement Using Brightness Preserving Bi-Histogram Equation”, *IEEE Transactions on Consumer Electronics*, vol. 43, no. 1, 1997 February, pp. 1-8.

5. C. H. Ooi and N. A. Mat Isa, “Adaptive Contrast Enhancement Methods with Brightness Preserving”, *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, 2010, pp. 2543-2551.

6. T. K. Kim, J. K. Paik and B. S. Kang, “Contrast enhancement system using spatially adaptive histogram equalization with temporal filtering”, *IEEE Transaction on Consumer Electronics*, vol. 44, no. 1, (1998), pp. 82-86.

7. N. Sengee and H. K. Choi, “Brightness preserving weight clustering histogram equalization”, *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, 2008, pp. 1329-1337.

8. Q. Wang and R. K. Ward, “Fast image/video contrast enhancement based on weighted threshold histogram equalization”, *IEEE transactions on Consumer Electronics*, vol. 53, no. 2, 2007, pp. 757-764.

9. S.-D. Chen and A. R. Ramli, “Minimum Mean Brightness Error Bi-Histogram Equalization in Contrast Enhancement”, *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, 2003 November, pp. 1310-1319.

10. S.-D. Chen and A. R. Ramli, “Preserving brightness in histogram equalization based contrast enhancement techniques”, *Digital Signal Processing*, vol. 14, 2004, pp. 413-428.

11. S.-D. Chen, “A new image quality measure for assessment of histogram eualization-based contrast enhancement technique”, *Digital Signal Processing*, vol. 22, pp. 640-647, 2012.

12. K. Liang, Y. Ma, Y. Xie, B. Zhou and R. Wang, “A new adaptive contrast enhancement algorithm for infrared images based on double plateaus histogram equalization”, *Infrared Physics & Technology*, vol. 55, 2012, pp. 309-315.

13. A. Raju, G.S. Dwarakish and D. Venkat Reddy “A Comparative Analysis of Histogram Equalization based Techniques for Contrast Enhancement and Brightness Preserving”, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, Vol. 6, No.5 (2013), pp.353-366, http://dx.doi.org/10.14257/ijsip.2013.6.5.31

14. Y. Su, M. Wu, Y. Yan, “Image Enhancement and Brightness Equalization Algorithms in Low Illumination Environment Based on Multiple Frame Sequences”, IEEE Access, 2023, Vol. 11, pp. 61535-61545.

15. Bilozerskyi, V., Dergachov, K. & Krasnov, L. Analiz i poperednya obrobka videodanykh dlya pidvyshchennya yakosti roboty system tekhnichnoho zoru [Analysis and pre-processing of video data to improve the quality of computer vision systems]. *Problemy keruvannya ta informatyky – Problems of control and informatics*, 2023, vol. 68, no. 2, pp. 50–66. DOI: 10.34229/1028-0979-2023-2-4. (In Ukrainian).

16. Bilozerskyi, V., Dergachov, K., Krasnov, L. “New method for video stream brightntss stabilization: algorithms and performance evaluation” *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2023, no.3, pp. 125-135. doi: 10.32620/reks.2023.3.10.

17. Egiazarian, K., Ponomarenko, M., Lukin, V. & Ieremeiev, O. Statistical Evaluation of Visual Quality Metrics for Image Denoising. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 6752-6756. DOI: 10.1109/ICASSP.2018.8462294.

18. Lukin, V. V., Zriakhov, M. S., Ponomarenko, N. N., Krivenko, S. S. & Zhenjiang, M. Lossy compression of images without visible distortions and its application. *IEEE 10th international conference on signal processing proceedings*, Beijing, China, 2010, pp. 698-701. DOI: 10.1109/ICOSP.2010.5655751.

ДОДАТОК А

Код опису інтерфейсу програми (PyQt5)

MainWindow.py

```
import sys
import cv2
import os
import numpy as np
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtCore import Qt, QTimer
from PyQt5.QtGui import QImage, QPixmap
from PyQt5.QtMultimedia import QMediaPlayer
from PyQt5.QtMultimediaWidgets import QVideoWidget
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton,
QFileDialog, QSlider, QSizePolicy, QVBoxLayout, QPushButton, QWidget,
QVBoxLayout, QHBoxLayout
from Interface.MainWindow import Ui_MainWindow
from timeit import default_timer as timer
from Scripts.controller import Controller


class LaunchWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Launch Window")
        self.setGeometry(100, 100, 600, 400)

        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        self.layout = QVBoxLayout(self.central_widget)

        # Image Brightness Stabilization label
        brightness_label = QLabel("Video Stream Brightness Preprocessing", self)
        font = brightness_label.font()
        font.setPointSize(14)
```

```
        brightness_label.setFont(font)  # Apply the font to the label
        brightness_label.setAlignment(Qt.AlignTop | Qt.AlignHCenter)
        self.layout.addWidget(brightness_label)

        # Placeholder image
        placeholder_image = QLabel(self)
        pixmap = QPixmap("Icons/preview.png")  # Replace with your image path
        placeholder_image.setPixmap(pixmap)
        placeholder_image.setAlignment(Qt.AlignCenter)
        self.layout.addWidget(placeholder_image)

        # Button layout
        button_layout = QHBoxLayout()
        # Open Main Window button
        self.open_main_button = QPushButton("Start", self)
        self.open_main_button.setFont(font)  # Apply the font to the button
        self.open_main_button.setFixedSize(133, 33)  # One-third smaller size
        self.open_main_button.clicked.connect(self.open_main_window)
        button_layout.addWidget(self.open_main_button)

        # Add spacing
        button_layout.addSpacing(50)  # Adjust the spacing as needed
        # Exit button
        self.exit_button = QPushButton("Exit", self)
        self.exit_button.setFont(font)  # Apply the font to the button
        self.exit_button.setFixedSize(133, 33)  # One-third smaller size
        self.exit_button.clicked.connect(self.close)
        button_layout.addWidget(self.exit_button)
        self.layout.addLayout(button_layout)

    def open_main_window(self):
        self.hide()  # Hide the launch window
        main_window = MainWindow()
        main_window.show()

# Основне вікно
```

```
class MainWindow(QMainWindow, Ui_MainWindow):
    """Клас основного вікна програми"""

    comboBoxStyleSheet = 'font: 10pt "MS Shell Dlg 2";selection-background-
color: rgb(230, 230, 230);' \
                         'background-color: rgb(245, 245, 245);color: rgb(0, 0,
255);selection-color: rgb(0, 0, 255);'
    state_values_dict = {"Wait": 0, "Start": 1, "Stop": 2, "Restart": 3}
    combobox_dict = {0: "Input video", 1: "Brightness Stabilization", 2: "Value
channel equalization"}

    def __init__(self):
        # ініціалізація вікна, створеного в QtDesigner
        # атрибути
        self.methods_for_external_use = {
            "update slider": self.update_slider_position,
            "display frame": self.display_frame_on_label,
            "print info": self.print_additional_info
        }  # словник з методами для зовнішнього користування іншими класами
        self.controller = Controller(self.methods_for_external_use,self)
        self.video_selected = False  # True - був обраний існуючий відео-файл
для роботи
        self.playing = False  # True - активне відображення відео
        self.paused = False  # True - відео стоїть на паузі
        self.isWeb = False  # True - увімкнено режим роботи з веб-камерою
        self.video1_mode = 0  # режим обробки відео у 1 вікні
        self.video2_mode = 1  # режим обробки відео у 2 вікні
        self.size_multiplier = 1  # змінна для збереження інформації про
розмірний коефіцієнт
        self.index_to_method_dict = {0: self.bgr_to_rgb, 1:
self.controller.get_afb_image, 2: self.get_equalized_image}
        super().__init__()
        self.setupUi(self)
        # Додавання елементів інтерфейсу
        self.video_widget = QVideoWidget(self)
        self.media_player = QMediaPlayer(None, QMediaPlayer.VideoSurface)
        self.media_player.setVideoOutput(self.video_widget)
        self.min_video_width = 500
```

```
self.min_video_height = 250
self.max_video_width = 800
self.max_video_height = 400
self.video1_label.setScaledContents(True)
self.video2_label.setScaledContents(True)
self.base_video_width = 600
self.base_video_height = 300
self.base_window_width = self.width()
self.base_window_height = self.height()

self.save_video_path = None
self.saving_file = False

self.video1_combobox.addItems(self.combobox_dict.values())
self.video2_combobox.addItems(self.combobox_dict.values())
self.video1_combobox.setCurrentIndex(0)
self.video2_combobox.setCurrentIndex(1)
# збільшення розміру шрифту для випадаючих меню
font_1 = self.video1_combobox.font()
font_1.setPointSize(10)
self.video1_combobox.setFont(font_1)
font_2 = self.video2_combobox.font()
font_2.setPointSize(10)
self.video2_combobox.setFont(font_2)
self.video_slider.sliderReleased.connect(self.set_frame)
self.video_slider.sliderPressed.connect(self.pause_video)
# Налаштування слайдеру масштабного коефіцієнта
self.size_slider.valueChanged.connect(self.on_size_slider_change)
self.minValue_label.setText("0.1")
self.curValue_label.setText("1.0")
self.maxValue_label.setText("2.0")
# додавання функціоналу
self.play_button.clicked.connect(self.play_video)
self.restart_button.clicked.connect(self.restart_video)
self.video1_combobox.activated.connect(self.on_video1_combobox_changed)
self.video2_combobox.activated.connect(self.on_video2_combobox_changed)
```

```
        toolbar = QtWidgets.QToolBar("Instruments")

        toolbar.setIconSize(QtCore.QSize(36, 36))

        self.addToolBar(toolbar)

        # icons_catalog_path = os.path.split(os.getcwd())[0] + "\\Icons\\"  #
шлях до каталогу з іконками

        icons_catalog_path = "Icons\\"  # шлях до каталогу з іконками

        browse_action = QtWidgets.QAction(QtGui.QIcon(icons_catalog_path +
"update_folder.png"), "Browse", self)

        browse_action.setToolTip("Change current video file")

        browse_action.triggered.connect(self.browse_btn_clicked)

        toolbar.addAction(browse_action)

        camera_action = QtWidgets.QAction(QtGui.QIcon(icons_catalog_path +
"camera.png"), "Camera", self)

        camera_action.setToolTip("Web-camera")

        camera_action.triggered.connect(self.camera_btn_clicked)

        toolbar.addAction(camera_action)

        toolbar.addSeparator()

        save_action = QtWidgets.QAction(QtGui.QIcon(icons_catalog_path +
"Save.png"), "Save_to", self)

        save_action.setToolTip("Save file")

        save_action.triggered.connect(self.save_btn_clicked)

        save_action.setCheckable(True)

        save_action.setChecked(False)

        toolbar.addAction(save_action)

        toolbar.addSeparator()

        plotter_action = QtWidgets.QAction(QtGui.QIcon(icons_catalog_path +
"Graph.png"), "Plotter", self)

        plotter_action.setToolTip("Data plotter")

        plotter_action.setCheckable(True)

        plotter_action.setChecked(False)

        plotter_action.triggered.connect(self.data_plotter_clicked)

        toolbar.addAction(plotter_action)


    def browse_btn_clicked(self):

        """Метод для обробки натискання кнопки вибору відеофайлу на пристрої"""

        # Отримуємо шлях до відеофайлу

        video_path, _ = QFileDialog.getOpenFileName(self, "Select Video", "Input
Data\\", "Video Files (*.mp4 *.avi)")
```

```
        # Через клас контролера виконуємо необхідні налаштування та отримуємо параметри відео
        video_length, current_frame, total_frames = self.controller.video_selected_action(video_path)
        if video_length is not None:
            # відображаємо параметри на слайдері
            self.slider_setup(video_length, current_frame, total_frames, False)
            self.restart_button.setEnabled(True)

    def camera_btn_clicked(self):
        """Метод для обробки натискання кнопки початку роботи з веб-камерою"""
        # Через клас контролера виконуємо необхідні налаштування та отримуємо параметри відео
        video_length, current_frame, total_frames = self.controller.web_selected_action()
        if video_length is not None:
            # відображаємо параметри на слайдері
            self.slider_setup(video_length, current_frame, total_frames, True)
            self.restart_button.setEnabled(False)

    def save_btn_clicked(self):
        if self.saving_file is False:
            self.save_video_path = self.choose_save_path()
            self.saving_file = True
        else:
            self.saving_file = False
        if self.save_video_path is not None:
            self.controller.saving_video_action(self.save_video_path)

    def data_plotter_clicked(self):
        """Метод для обробки натискання на кнопку побудови графіків даних"""
        self.controller.collecting_data_action()

    def slider_setup(self, video_length, current_frame, total_frames, is_web):
        """Метод для налаштування слайдеру"""
        # Якщо працюємо з відеофайлом
        if not is_web:
```

```
            self.display_time(video_length, self.full_time_label)
            self.video_slider.setRange(0, int(total_frames))
            self.video_slider.setValue(current_frame)
            self.video_slider.setEnabled(True)
        # Якщо працюємо з веб-камерою
        else:
            self.display_time(0, self.full_time_label)
            self.video_slider.setValue(0)
            self.video_slider.setEnabled(False)

    def play_video(self):
        """Натискання кнопки play/pause"""
        self.controller.video_play_action()

    def restart_video(self):
        """Натискання кнопки restart"""
        self.controller.video_restart_action()

    def pause_video(self):
        """Зупинка відео"""
        self.controller.video_pause_action()

    @staticmethod
    def bgr_to_rgb(frame):
        """Метод для перетворення зображення з формату BGR до QImage"""
        return frame
        # cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    def get_afb_image(self, frame):
        """Метод для обробки режиму стабілізації яскравості відео"""
        return self.controller.get_afb_image(frame)

    def get_equalized_image(self, frame):
        """Метод для обробки режиму стабілізації яскравості відео"""
        return self.controller.get_equalized_image(frame)
```

```
    @staticmethod
    def rgb_to_qimage(frame_rgb):
        """Метод для перетворення зображення з формату RGB до QImage"""
        return QImage(frame_rgb, frame_rgb.shape[1], frame_rgb.shape[0],
QImage.Format_BGR888)


    def display_frame_on_label(self, frame):
        """Метод для відображення поточного кадру"""
        # Аналіз дій перед відображенням
        point1 = timer()
        rgb1 =
self.rgb_to_qimage(self.index_to_method_dict.get(self.video1_mode)(frame))
        point2 = timer()
        rgb2 =
self.rgb_to_qimage(self.index_to_method_dict.get(self.video2_mode)(frame))
        point3 = timer()


        # Відображення кадрів
        self.video1_label.setPixmap(QPixmap.fromImage(rgb1))
        point4 = timer()
        self.video2_label.setPixmap(QPixmap.fromImage(rgb2))
        point5 = timer()


        if self.saving_file is True:
            frame_to_save =
self.index_to_method_dict.get(self.video2_mode)(frame)
            self.controller.save_single_frame(frame_to_save)
        return [point2-point1, point3-point2, point4-point3, point5-point4]


    def update_slider_position(self, current_full_time_seconds, current_frame):
        """Метод оновлення слайдера з інформацією про поточний час"""
        self.display_time(current_full_time_seconds, self.current_time_label)
        self.video_slider.setValue(current_frame)


    def print_additional_info(self, info_text, video_num):
        """Метод для відображення текстових параметрів"""
        if video_num == 1:
```

```
            self.video1_info_label.setText(info_text)
        elif video_num == 2:
            self.video2_info_label.setText(info_text)


    def set_frame(self):
        """Метод для встановлення кадру відповідно до виставленної користувачем позиції слайдера"""
        self.controller.video_frame_position_changed(self.video_slider.value())
    def on_video1_combobox_changed(self, value):
        """Метод для відстеження зміни режиму відображення першого вікна"""
        self.video1_mode = value
    def on_video2_combobox_changed(self, value):
        """Метод для відстеження зміни режиму відображення другого вікна"""
        self.video2_mode = value
    @staticmethod
    def display_time(full_time_seconds, label):
        """Метод для запису часу у хвилинах та секундах на відповідному елементі label"""
        video_minutes = int(full_time_seconds / 60)
        video_seconds = int(full_time_seconds - video_minutes * 60)
        label.setText("{}:{}".format(video_minutes, video_seconds)
    def on_size_slider_change(self, value):
        """Метод для обробки зміни значення слайдеру масштабного коефіцієнта"""
        correct_value = self.controller.size_slider_changed_action(value)
        self.curValue_label.setText(f'{correct_value:.2f}')


    def choose_save_path(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        file_path, _ = QFileDialog.getSaveFileName(None, "Save Video", "", "MP4 files (*.mp4);;All Files (*)", options=options)
        # Ensure the file path has the correct .mp4 extension
        if file_path:
            if not file_path.endswith(".mp4"):
                file_path += ".mp4"
        return file_path
    def resizeEvent(self, event):
```

```
        """Метод для масштабування вікон відображення кадрів відео при зміні
загальних розмірів вікна"""
        # Зміна масштабних коефіцієнтів
        x_shift = self.width() - self.base_window_width
        self.base_window_width = self.width()
        y_shift = (self.height() - self.base_window_height) / 2
        self.base_window_height = self.height()
        updated_width, updated_height = int(self.base_video_width + x_shift),
int(self.base_video_height + y_shift)
        # Перевірка мінімальних розмірностей вікон відображення
        self.base_video_width = updated_width
        self.base_video_height = updated_height
        if updated_width <= self.min_video_width:
            self.base_video_width = self.min_video_width
        if updated_height <= self.min_video_height:
            self.base_video_height = self.min_video_height
        if updated_width >= self.max_video_width:
            self.base_video_width = self.max_video_width
        if updated_height >= self.max_video_height:
            self.base_video_height = self.max_video_height
        self.video1_label.setFixedSize(self.base_video_width,
self.base_video_height)
        self.video2_label.setFixedSize(self.base_video_width,
self.base_video_height)
        super().resizeEvent(event)


# if __name__ == "__main__":
#     app = QApplication(sys.argv)
#     player = MainWindow()
#     player.show()
#     sys.exit(app.exec_())
if __name__ == '__main__':
    app = QApplication(sys.argv)
    launch_window = LaunchWindow()
    launch_window.show()
    sys.exit(app.exec_())
```

ДОДАТОК Б

Скрипт контролеру процесів

Controller.py

```
from Scripts.video_processing import VideoProcessing
from Scripts.AFB_correction_class import AverageFrameBrightnessClass
from Scripts.additional_info import AdditionalInfo
from PyQt5.QtCore import QTimer
from Scripts.preprocessing import resize
from timeit import default_timer as timer
import Scripts.graph_plotter as plotter
from Scripts.preprocessing import equalize_value_channel
import cv2



class Controller:
    """Фасадний клас для керування процесами"""


    def __init__(self, main_window_methods,main_window):
        self.main_window_methods = main_window_methods
        self.main_window = main_window
        self.video_processing = VideoProcessing()  # екземпляр класу для роботи
з відео
        self.afb_correction_class = AverageFrameBrightnessClass()
        self.video1_info = AdditionalInfo()
        self.video2_info = AdditionalInfo()
        self.fps = None  # кількість кадрів в секунду початкового відео
        self.timer = QTimer()  # таймер для обробки у окремому потоці
        # Приєднуємо до таймера функції відображення наступного фрейму та
передачі поточних даних
        self.timer.timeout.connect(self.video_processing_step)
        self.timer_delay = 10  # затримка для таймера
        self.collecting_data = False
        self.saving_data = False
        self.video_writer = None
        self.frame_width = None
```

```
        self.frame_height = None
        self.save_video_path = None



    def video_selected_action(self, video_path):
        """Метод для передачі шляху відеофайлу класу VideoProcessing для початку роботи"""
        # Зупинка відео при зміні відеоданих
        self.video_pause_action()
        # Відкриваємо файл за шляхом
        video_length, current_frame, total_frames, fps = self.video_processing.video_selected_action(video_path)
        if fps is not None:
            self.fps = fps
            self.afb_correction_class.set_current_video(self.fps)
            # Відображення першого кадру
            self.video_processing_step()
            return video_length, current_frame, total_frames
        else:
            return None, None, None


    def web_selected_action(self):
        """Метод для обробки натискання кнопки початку роботи з веб-камерою"""
        # Зупинка відео при зміні відеоданих
        self.video_pause_action()
        # Починаємо роботу з камерою
        video_length, current_frame, total_frames, self.fps = self.video_processing.web_selected_action()
        if self.fps is not None:
            self.afb_correction_class.set_current_video(self.fps)
            # Відображення першого кадру
            self.video_processing_step()
            return video_length, current_frame, total_frames
        else:
            return None, None, None


    def video_play_action(self):
```

```
        """Метод обробки натискання кнопки Play/Stop у головному вікні"""
        self.video_processing.play_video()
        # Інтервал роботи таймера
        self.timer_control()

    def initialize_video_writer(self):
        if self.save_video_path:
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # or 'H264'
            self.video_writer = cv2.VideoWriter(self.save_video_path, fourcc,
self.fps, (self.frame_width, self.frame_height))

    def video_processing_step(self):
        """Метод для ітераційної обробки відео та відображення даних"""
        p1 = timer()
        frame, current_frame = self.video_processing.next_frame()
        p2 = timer()
        # Передаємо кадр головному вікну для відображення
        if frame is not None:
            initial_size = frame.shape
            # Відображення кадру в інтерфейсі
            time_list = self.main_window_methods.get("display frame")(frame)
            # Розрахунок параметрів
            reading_time = p2 - p1
            total_time_1 = reading_time + time_list[0] + time_list[2]
            total_time_2 = reading_time + time_list[1] + time_list[3]
            current_fps_1 = 1.0 / total_time_1
            current_fps_2 = 1.0 / total_time_2
            info_values_1 = [self.fps, initial_size, reading_time, time_list[0],
time_list[2], total_time_1,
                             current_fps_1, current_frame]
            info_values_2 = [self.fps, initial_size, reading_time, time_list[1],
time_list[3], total_time_2,
                             current_fps_2, current_frame]
            tmp_dict_1 = dict()
            tmp_dict_2 = dict()
            for i, key in enumerate(self.video1_info.get_keys()):
                tmp_dict_1.update({key: info_values_1[i]})
```

```
                tmp_dict_2.update({key: info_values_2[i]})
            # Формування інформаційного тексту
            self.video1_info.set_current_data_iteration(tmp_dict_1)
            self.video2_info.set_current_data_iteration(tmp_dict_2)
            # Оновлення повзунка відео головного вікна
            video_selected, is_web = self.video_processing.get_state()
            if not is_web and video_selected:
                self.main_window_methods.get("update slider")(current_frame /
self.fps, current_frame)
            # Відображення текстової інформації
            if not self.collecting_data:
                text_to_display_1 = self.video1_info.display_data()
                text_to_display_2 = self.video2_info.display_data()
                self.main_window_methods.get("print info")(text_to_display_1, 1)
                self.main_window_methods.get("print info")(text_to_display_2, 2)

            self.frame_height, self.frame_width, _ = frame.shape

    def timer_control(self):
        """Метод для перевірки поточного стану відео і корекція роботи
таймера"""
        playing, paused = self.video_processing.get_video_state()
        if paused and self.timer.isActive():
            self.timer.stop()
        if playing and not self.timer.isActive():
            self.timer.start(self.timer_delay)

    def video_pause_action(self):
        """Метод для зупинки відео"""
        if self.video_processing.pause_video():
            self.timer_control()

    def video_restart_action(self):
        """Метод обробки натискання кнопки Restart у головному вікні"""
        if self.video_processing.restart_video():
            self.afb_correction_class.refresh_window()
```

```
            self.video_processing_step()
            self.timer_control()

    def video_frame_position_changed(self, new_position):
        """Метод обробки зміни поточної позиції кадру у відеофайлі"""
        self.video_processing.set_current_position(new_position)
        self.video_processing_step()

    def size_slider_changed_action(self, value):
        """Метод для передачі екземпляру класу VideoProcessing поточого значення
size_multiplier"""
        return self.afb_correction_class.set_size_multiplier(value / 100)

    def get_afb_image(self, frame):
        """Метод для отримання поточного кадру в умовах стабілізації яскравості
за AFB"""
        return self.afb_correction_class.get_image_for_brightness_stab(frame)

    def get_equalized_image(self, frame):
        """Метод для отримання поточного кадру в умовах стабілізації яскравості
за AFB"""
        return equalize_value_channel(frame)

    def collecting_data_action(self):
        """Метод для переходу у режим збору параметів"""
        video_selected, is_web = self.video_processing.get_state()
        if not video_selected and not is_web:
            return
        # Якщо був звичайний режим відображення
        if not self.collecting_data:
            # Починаємо збирати дані
            self.collecting_data = True
            # Починаємо відео з початку
            self.video_restart_action()
        # При повторному натисканні
        else:
            # будуємо графіки
```

```
            data_dict_1 = self.video1_info.get_data()
            data_dict_2 = self.video2_info.get_data()
            plotter.full_plot(data_dict_1, data_dict_2)
            # Перестаємо збирати дані
            self.collecting_data = False
            # Ставимо відео на паузу
            self.video_pause_action()
            # Видаляємо накопичені дані
            self.video1_info.clear_data()
            self.video2_info.clear_data()

    def saving_video_action(self,save_video_path):
        video_selected, is_web = self.video_processing.get_state()

        if not video_selected and not is_web:
            return

        if not self.saving_data:

            # Починаємо збирати дані
            self.saving_data = True
            self.initialize_video_writer()
            self.save_video_path = save_video_path
            self.video_restart_action()

        # При повторному натисканні
        else:
            # будуємо графіки

            self.saving_data = False
            # Ставимо відео на паузу
            self.video_pause_action()
            # Видаляємо накопичені дані
            self.video1_info.clear_data()
            self.video2_info.clear_data()
            if self.video_writer is not None:
```

```
                self.video_writer.release()
                self.video_writer = None


    def save_single_frame(self,fullframe):


        if self.video_writer is None:
            self.initialize_video_writer()


        # Write the frame to the video
        if self.video_writer is not None:
            self.saveframe = fullframe


            # rgb2 =
self.main_window_script.rgb_to_qimage(self.index_to_method_dict.get(self.video2_
mode)(frame))
            self.video_writer.write(self.saveframe)


if __name__ == "__main__":
    controller = Controller()
```

ДОДАТОК В

video_processing.py

Клас обробки відеокадрів

```
import cv2

class VideoProcessing:
    """Клас для роботи з відео-даними"""

    def __init__(self):
        """Конструктор класу"""
        self.video_path = None  # шлях до відео-файлу
        self.current_frame = 0  # лічильник поточного кадру
        self.video_selected = False  # True - був обраний існуючий відео-
файл для роботи
        self.playing = False  # True - активне відображення відео
        self.paused = False  # True - відео стоїть на паузі
        self.cap = None  # змінна для підключення відео-даних
        self.isWeb = False  # True - увімкнено режим роботи з веб-камерою
        self.total_frames = None  # загальна кількість кадрів поточного
відео
        self.fps = None  # параметр кількості кадрів в секунду
        self.video_full_time_seconds = 0  # загальна довжина відео в
секундах

    def video_selected_action(self, video_path):
        """Метод для початку роботи з відеоданими.
        Вхідні дані: videopath - шлях до відеофайлу"""
        if self.playing:  # Якщо відбувалося відображення іншого відео,
ставимо його на паузу
            self.pause_video()
        # Відкриваємо файл за шляхом
        cap = cv2.VideoCapture(video_path)
        # Перевірка коректності шляху до відео
        if cap.isOpened():
            self.video_path = video_path
            self.cap = cap
            # Зчитуємо необхідні параметри відео
            self.total_frames = self.cap.get(cv2.CAP_PROP_FRAME_COUNT)
            self.fps = self.cap.get(cv2.CAP_PROP_FPS)
            self.video_full_time_seconds = self.total_frames / self.fps
            self.current_frame = 0
            # Встановлюємо значення логічних змінних на роботу з
відеофайлом
            self.video_selected = True
```

```
                self.isWeb = False
                return     self.video_full_time_seconds,     self.current_frame,
self.total_frames, self.fps
            else:
                return None, None, None, None

        def web_selected_action(self):
            """Метод для початку роботи з веб-камерою"""
            if self.playing:   # Якщо відбувалося відображення іншого відео,
ставимо його на паузу
                self.pause_video()

            cap = cv2.VideoCapture(0)
            if cap.isOpened():  # якщо вдалося підключити камеру
                self.video_path = "web 0"
                self.cap = cap
                self.total_frames = 0
                self.fps = self.cap.get(cv2.CAP_PROP_FPS)
                self.current_frame = 0
                # Встановлюємо значення логічних змінних на роботу з веб-
камерою
                self.video_selected = False
                self.isWeb = True
                return 0, self.current_frame, 0, self.fps
            else:
                return None, None, None, None

        def play_video(self):
            """Метод для запуску або зупинки відео"""
            if not self.video_selected and not self.isWeb:   # Якщо не було
обрано відео для відображення
                return
            if not self.playing and not self.paused:   # Для першого запуску
відео
                self.playing = True
                self.paused = False
            # Якщо відео було на паузі, повертаємо відображення
            elif not self.playing and self.paused:
                self.resume_video()
            # Якщо відео відображалось, ставимо його а паузу
            elif self.playing and not self.paused:
                self.pause_video()

        def next_frame(self):
            """Перехід до наступного фрейму у відео"""
            ret, frame = self.cap.read()
```

```
        if ret:  # Якщо не має помилки
            # Збільшуємо лічильник поточного кадру
            self.current_frame += 1
            return frame, self.current_frame
        else:
            return None, None

    def restart_video(self):
        if not self.video_selected and not self.isWeb:
            return False
        """Метод для перезапуску поточного відео"""
        # Онуляємо поточний кадр
        self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        self.current_frame = 0
        self.playing = True
        self.paused = False
        return True

    def pause_video(self):
        """Тимчасова зупинка відображення відео"""
        if not self.video_selected and not self.isWeb:
            return False
        self.playing = False
        self.paused = True
        return True

    def resume_video(self):
        """Відновлення відображення відео після паузи"""
        if not self.video_selected and not self.isWeb:
            return False
        self.playing = True
        self.paused = False
        return True

    def stop_video(self):
        if not self.video_selected and not self.isWeb:
            return False
        """Повна зупинка роботи з відео"""
        self.cap.release()
        self.playing = False
        self.paused = False
        self.video_selected = False
        self.isWeb = False
        return True
```

```
        def set_current_position(self, frame_position):
            """Метод для оновлення поточної позиції відео"""
            if not self.video_selected and not self.isWeb:
                return
            self.cap.set(cv2.CAP_PROP_POS_FRAMES, frame_position)   # Установка
позиции в видео
            self.current_frame = frame_position

        def get_video_state(self):
            """Метод для повернення поточного стану роботи відео"""
            return self.playing, self.paused

        def get_state(self):
            """Метод для повернення поточного стану"""
            return self.video_selected, self.isWeb
```

Preprocessing.py

Клас обробки відеокадрів різними методами

```
import cv2
import numpy as np


def resize(input_frame, size_multiplier):
    """Збільшення/зменшення розмірів зображення відповідно до розмірного
коефіцієнту.
        Вхідні дані: input_frame - вихідне зображення;
                     size_multiplier - розмірний коефіцієнт
        Вихідні дані: зображення з оновленою розмірністю"""
        new_width = int(input_frame.shape[1] * size_multiplier)
        new_height = int(input_frame.shape[0] * size_multiplier)
        dim = (new_width, new_height)
        return cv2.resize(input_frame, dim, cv2.INTER_AREA)


def brightness_hsv(image_hsv):
    """Розрахунок середньої яскравості зображення у HSV форматі.
        Вхідні дані: image_hsv - зображення у HSV форматі
        Вихідні дані: середня яскравість за Value каналом вихідного
зображення"""
        # Виділяємо окремі канали зображення
        h, s, v = cv2.split(image_hsv)
        # Розрахунок середньої яскравості за Value каналом
        return np.mean(v)
```

```
    def brightness_change(image_hsv, value):
        """Лінійна корекція Value каналу вихідного зображення у HSV форматі
        Вхідні дані: image_hsv - вихідне зображення у HSV форматі;
                     value - значення для корекції яскравості (0-255)
        Вихідні дані: зображення у HSV форматі зі скорегованим Value каналом"""
        # Виділяємо окремі канали
        h, s, v = cv2.split(image_hsv)
        # Корекція яскравості з урахуванням крайніх значень (0, 255)


        #-----------------------------Limit method-------------------------

        if value >= 0:
            lim = 255 - value
            v[v > lim] = 255
            v[v <= lim] += value
        else:
            v[v >= -value] -= -value
            v[v < -value] = 0

        #-----------------Clip method-----------------------------------

        # if value >= 0:
        #     v = np.clip(v + value, 0, 255)
        # else:
        #     v = np.clip(v - value, 0, 255)

        # v = v.astype('uint8')

        #---------------------
        # v = v.astype('int32')
        # v[v < (0 + value)] = 0 + value
        # v -= value
        # v = v.astype('uint8')

            # Зворотнє формування зображення з урахуванням скорегованого Value
каналу
        return cv2.merge((h, s, v))


    def equalize_value_channel(image_bgr):
        """Еквалізація Value каналу HSV зображення"""
        # Переводимо зображення до формату HSV
        image_hsv = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2HSV)
```

```
        # Відокремлюємо канали
        h, s, v = cv2.split(image_hsv)
        # Еквалізуємо канал яскравості Value (V)
        equalized_v = cv2.equalizeHist(v)
        # Відновлюємо HSV зображення з еквалізованним каналом яскравості
        equalized_hsv = cv2.merge((h, s, equalized_v))
        # Переводимо до формату RGB
        equalized_rgb = cv2.cvtColor(equalized_hsv, cv2.COLOR_HSV2BGR)

        return equalized_rgb


    import cv2
    import numpy as np
    from Scripts.preprocessing import brightness_hsv, brightness_change, resize


    class AverageFrameBrightnessClass:
        """Клас для попередньої обробки відео-даних з метою стабілізації яскравості"""
        def __init__(self):
            self.T_sec = 5    # розмір динамічного вікна для усереднення яскравості
            self.T_frame = int(self.T_sec * 30)  # розмір динамічного вікна (у кількості кадрів) для усереднення яскравості
            self.dynamic_window = []  # значення динамічного вікна
            self.size_multiplier = 1  # множник масштабного коефіцієнта

        def set_current_video(self, fps):
            """Метод для встановлення зв'язку з поточним відео"""
            # Визначаємо розмір окремого вікна у фреймах
            self.T_frame = int(self.T_sec * fps)
            # Онуляємо динамічне вікно
            self.dynamic_window = []

        def set_size_multiplier(self, value):
            """Метод для установки значення масштабного коефіцієнта"""
            leftover = value % 0.05
            if leftover < 10**-5:  # Якщо значення є кратним 0.05
                self.size_multiplier = value
                return value
            else:  # Якщо кратність не виконується, повертаємо найближче кратне
                self.size_multiplier = value - leftover
                return value - leftover

        def refresh_window(self):
```

```
    """Метод для очищення динамічного вікна"""
    self.dynamic_window = []

def get_image_for_brightness_stab(self, frame):
    # Робота з окремими фреймами
    # Перехід до простору HSV та визнаення середньої яскравості
    original_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # Зменшення робочого кадру для подальшої обробки
    resized_hsv = resize(original_hsv, self.size_multiplier)

    b_avg = brightness_hsv(resized_hsv)
    # Оновлення динамічного вікна
    self.dynamic_window.append(b_avg)
    dynamic_window_size = len(self.dynamic_window)

    if dynamic_window_size > self.T_frame:
        self.dynamic_window.pop(0)

    base_value = np.mean(self.dynamic_window)
    # Виконання усереднення яскравості
    diff = int(base_value - b_avg)
    changed_image = brightness_change(original_hsv, diff)
    changed_image_rgb = cv2.cvtColor(changed_image, cv2.COLOR_HSV2BGR)
    return changed_image_rgb
```

ДОДАТОК В

Graph_plotter.py

Клас побудови графіків по показникам якості алгоритмів

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time

AVERAGE_FILTER_WINDOW_SIZE = 7

def equalize_hsv(image_hsv):
    """Еквалізація каналу яскравості зображення у форматі HSV.
    Вхідні дані: image_hsv - зображення у форматі HSV
    Вихідні дані: зображення після еквалізації каналу яскравості"""
    # Отримуємо окремі канали
    h, s, v = cv2.split(image_hsv)
    # Еквалізуємо канал яркості
    v = cv2.equalizeHist(v)
    return cv2.merge((h, s, v))

def histogram(input_images, xlabel, ylabel, colors, to_show=True, to_save=False,
dpi=300):
    """Побудова гістограм. Вхідні дані:
    input_images - список вхідних зображень для побудови гістограм
    xlabel - підпис вісі x
    ylabel - підпис вісі y
    colors - список кольорів для побудови гістограм (повинен відповідати
розмірності input_data
    to_show - логічна зміна для відобраення гістограми (True за замовченням)
    to_save - логічна зміна для збереження гістограми на пристрої (False за
замовченням)
```

```
    dpi - значення dpi при збереженні на пристрої (300 за замовченням)
    """
    fig = plt.figure()
    for index, data in enumerate(input_images):
        max_value = int(np.max(data))
        min_value = int(np.min(data))
        # створюємо пустий словник для майбутніх значень гістограми
        tmp_hist = [0]*256
        height, width = data.shape  # розміри зображення
        for i in range(height):  # прохід по строкам
            for j in range(width):  # прохід по стовпцям
                tmp_value = data[i][j]  # яскравість поточного пікселя
                tmp_hist[tmp_value] += 1
        # визначаємо кількість пікселей
        num = height * width
        # нормалізуємо гістограму відповідно до загальної кількості пікселей
        for i in range(256):
            # розрахунок відсотку для кожного значення яркості
            tmp_hist[i] = tmp_hist[i] / num * 100
        # побудова гістограми
        plt.bar(range(min_value, max_value + 1), tmp_hist[min_value:max_value +
1], width=0.4, linewidth=2, color=colors[index])
    plt.grid()
    plt.xlabel(xlabel, fontsize=14)
    plt.ylabel(ylabel, fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    # Збереження та відображення результату
    if to_save:
        time_struct = time.gmtime(time.time())
        save_index = "{}.{}.{}//{}:{}:{}".format(time_struct[2], time_struct[1],
time_struct[0], time_struct[3], time_struct[4], time_struct[5])
        plt.savefig('hist{}.png'.format(save_index), dpi=dpi)
        save_index += 1
    if to_show:
        plt.show()
```

```
    else:
        plt.close(fig)


def basic_plot(x, y, xlabel, ylabel):
    """Побудова графіка двох змінних"""
    plt.plot(x, y)
    # сітка
    plt.grid()
    # підпис кожної вісі
    plt.xlabel(xlabel, fontsize=14)
    plt.ylabel(ylabel, fontsize=14)
    # збільшення розміру тексту на підписах кожної вісі
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    # демонстрація графіку
    plt.show()


def multi_plot(xs: list, ys: list, xlabel, ylabel, title, colors: list, legend:
list):
    """Побудова накладенних графіків двох змінних"""
    plt.figure()
    for i, tmp_color in enumerate(colors):
        plt.plot(xs[i], ys[i], color=tmp_color)
    # сітка
    plt.grid()
    # підпис кожної вісі
    plt.xlabel(xlabel, fontsize=14)
    plt.ylabel(ylabel, fontsize=14)
    # збільшення розміру тексту на підписах кожної вісі
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.title(title, fontsize=14)
    plt.legend(legend)
    # демонстрація графіку
    plt.show()
```

```
def full_plot(data_dict_1, data_dict_2):

    """Побудова усіх необхідних графіків на основі двух словників даних"""

    # Залежність FPS першого відео від часу

    xs = [data_dict_1.get("Current Frame"), data_dict_1.get("Current Frame")[:-
AVERAGE_FILTER_WINDOW_SIZE]]

    ys = [data_dict_1.get("Current FPS"),
average_values(data_dict_1.get("Current FPS"), AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Current FPS", "FPS vs. frame count
(video 1)", ['steelblue', 'r'], ["base value", "average value"])

    # Залежність FPS другого відео від часу

    xs = [data_dict_2.get("Current Frame"), data_dict_2.get("Current Frame")[:-
AVERAGE_FILTER_WINDOW_SIZE]]

    ys = [data_dict_2.get("Current FPS"),
average_values(data_dict_2.get("Current FPS"), AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Current FPS", "FPS vs. frame count
(video 2)", ['steelblue', 'r'], ["base value", "average value"])

    # Порівняння графіків залежності FPS від часу

    xs = [data_dict_1.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE],
data_dict_2.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE]]

    ys = [average_values(data_dict_1.get("Current FPS"),
AVERAGE_FILTER_WINDOW_SIZE), average_values(data_dict_2.get("Current FPS"),
AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Current FPS", "FPS vs. frame count",
['steelblue', 'r'], ["video 1", "video 2"])



    # Залежність швидкісних показників роботи першого відео від часу

    xs = [data_dict_1.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE]] * 3

    ys = [average_values(data_dict_1.get("Frame Reading Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_1.get("Frame Preprocessing Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_1.get("Frame Displaying Time"),
AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Time, sec", "Operations time (video 1)",
['steelblue', 'r', 'm'],

               ["Reading time", "Preprocessing time", "Displaying time"])

    # xs = [data_dict_1.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE]] * 4

    # ys = [

    # average_values(data_dict_1.get("Frame Reading Time"),
AVERAGE_FILTER_WINDOW_SIZE),
```

```
    # average_values(data_dict_1.get("Frame Reading Time"),
AVERAGE_FILTER_WINDOW_SIZE),

    # average_values(data_dict_1.get("Frame Preprocessing Time"),
AVERAGE_FILTER_WINDOW_SIZE),

    # average_values(data_dict_1.get("Frame Displaying Time"),
AVERAGE_FILTER_WINDOW_SIZE)

    # ]

    # multi_plot(xs, ys, "Video Frames", "Time, sec", "Operations time (video
1)", ['r', 'steelblue', 'g', 'm'], ["1280x720", "1920x1080","2560x1440",
"3840x2160"])


    # Залежність швидкісних показників роботи другого відео від часу

    xs = [data_dict_2.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE]] * 3

    ys = [average_values(data_dict_2.get("Frame Reading Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_2.get("Frame Preprocessing Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_2.get("Frame Displaying Time"),
AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Time, sec", "Operations time (video 2)",
['steelblue', 'r', 'm'],

               ["Reading time", "Preprocessing time", "Displaying time"])

    # Порівняння графіків швидкості обробки відео

    xs = [data_dict_1.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE],

          data_dict_2.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE]]

    ys = [average_values(data_dict_1.get("Frame Preprocessing Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_2.get("Frame Preprocessing Time"),
AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Time, sec", "Frame Preprocessing Time",
['steelblue', 'r'], ["video 1", "video 2"])

    # Порівняння графіків швидкості відображення відео

    xs = [data_dict_1.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE],

          data_dict_2.get("Current Frame")[:-AVERAGE_FILTER_WINDOW_SIZE],]

    ys = [average_values(data_dict_1.get("Frame Displaying Time"),
AVERAGE_FILTER_WINDOW_SIZE),

          average_values(data_dict_2.get("Frame Displaying Time"),
AVERAGE_FILTER_WINDOW_SIZE)]

    multi_plot(xs, ys, "Video Frames", "Time, sec", "Frame Displaying Time",
['steelblue', 'r'], ["video 1", "video 2"])

def average_values(values: list, window_size):

    """Формування списку з усередненими даними"""
```

```
    average_y = []
    for ind in range(len(values) - window_size):
        average_y.append(np.mean(values[ind:ind + window_size]))
    return average_y
```

ДОДАТОК Г

Main.py

```
import sys
from Scripts.main_window_script import MainWindow
from Scripts.main_window_script import LaunchWindow
from PyQt5.QtWidgets import QApplication

if __name__ == "__main__":
    app = QApplication(sys.argv)
    player = LaunchWindow()
    player.show()
    sys.exit(app.exec_())
```

Additional_info.py

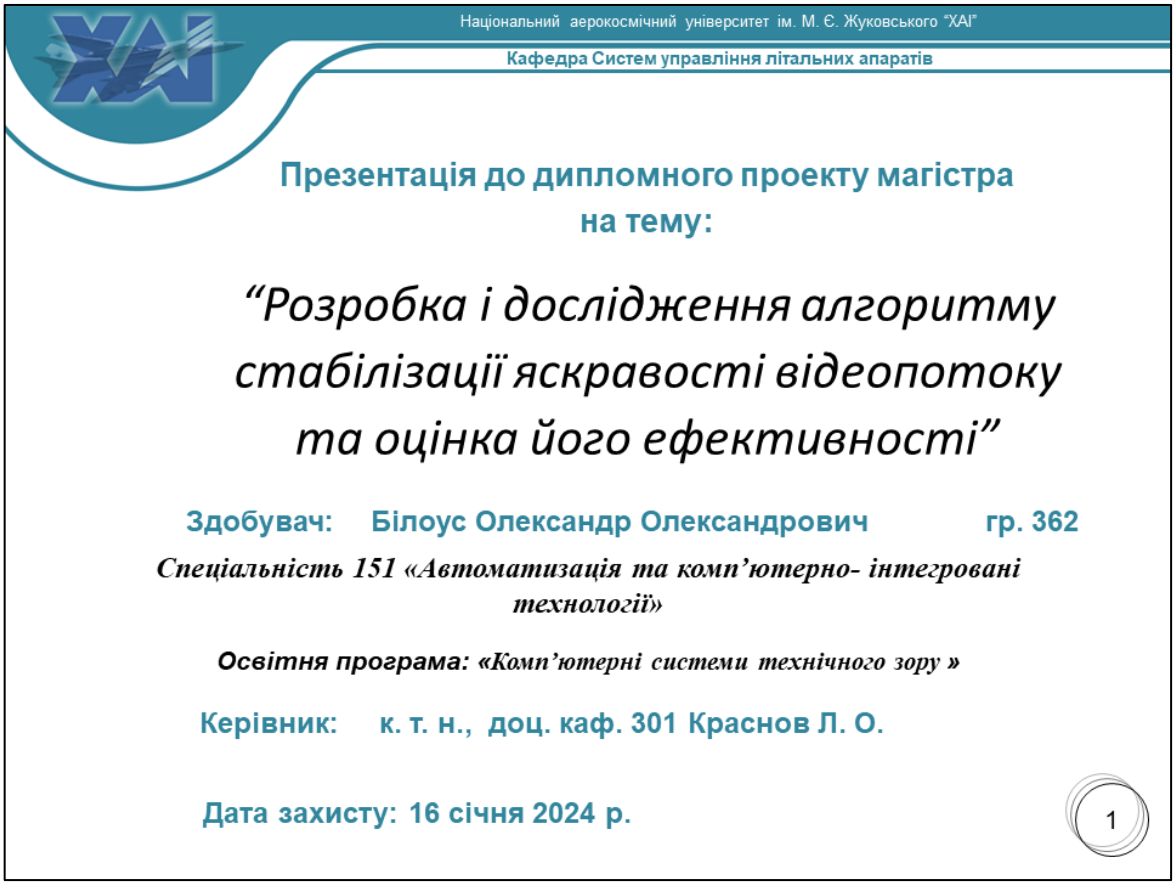
Клас запису текстових даних

```
class AdditionalInfo:
    """Клас для зберігання текстової інформації"""
    def __init__(self):
        self.info_dict = {"Initial FPS": [], "Initial Shape": [], "Frame Reading
Time": [],
                          "Frame Preprocessing Time": [], "Frame Displaying
Time": [], "Total time": [],
                          "Current FPS": [], "Current Frame": []}  # словник
даних
        self.keys = ["Initial FPS", "Initial Shape", "Frame Reading Time",
"Frame Preprocessing Time",
                     "Frame Displaying Time", "Total time", "Current FPS",
"Current Frame"]

    def get_keys(self):
        """Метод для повернення списку ключів словника"""
        return self.keys
```

```
    def get_data(self):
        """Метод для повернення збережених даних"""
        return self.info_dict

    def clear_data(self):
        """Метод для видалення даних"""
        self.info_dict = {"Initial FPS": [], "Initial Shape": [], "Frame Reading
Time": [],
                          "Frame Preprocessing Time": [], "Frame Displaying
Time": [], "Total time": [],
                          "Current FPS": [], "Current Frame": []}

    def set_current_data_iteration(self, tmp_dict):
        """Метод для заповненния ітерації даних за словником tmp_dict"""
        # Перевіряємо, що у словнику є усі необхідні дані ітерації
        for key in self.keys:
            if key in tmp_dict:
                self.info_dict.get(key).append(tmp_dict.get(key))
            else:
                self.info_dict.get(key).append(-1)

    def display_data(self):
        """Метод для формування текстової репрезентації даних"""
        if len(self.info_dict.get(self.keys[0])) > 0:
            inital_fps = self.info_dict.get(self.keys[0]).pop()
            inital_shape = self.info_dict.get(self.keys[1]).pop()
            reading_time = self.info_dict.get(self.keys[2]).pop()
            preprocessing_time = self.info_dict.get(self.keys[3]).pop()
            displaying_time = self.info_dict.get(self.keys[4]).pop()
            total_time = self.info_dict.get(self.keys[5]).pop()
            current_fps = self.info_dict.get(self.keys[6]).pop()
            current_frame = self.info_dict.get(self.keys[7]).pop()
            return f"{self.keys[1]}: {inital_shape[1]}x{inital_shape[0]}\n\n" \
                   f"{self.keys[2]}: {reading_time:.5f} sec\n" \
                   f"{self.keys[3]}: {preprocessing_time:.5f}
sec\n{self.keys[4]}: {displaying_time:.5f} sec\n" \
```

```
                    f"{self.keys[5]}: {total_time:.5f} sec\n\n{self.keys[7]}:
{current_frame}\n\n" \
                    f"{self.keys[0]}: {inital_fps:.2f}\n{self.keys[6]}:
{current_fps:.2f}\n"
        else:
            return None



if __name__ == "__main__":
    test = AdditionalInfo()
    print(test.display_data())
```

ДОДАТОК Д


Національний аерокосмічний університет ім. М. Є. Жуковського "ХАІ"
Кафедра Систем управління літальних апаратів
Презентація до дипломного проекту магістра
на тему:
"Розробка і дослідження алгоритму стабілізації яскравості відеопотоку та оцінка його ефективності"
Здобувач: Білоус Олександр Олександрович гр. 362
Спеціальність 151 «Автоматизація та комп'ютерно- інтегровані технології»
Освітня програма: «Комп'ютерні системи технічного зору »
Керівник: к. т. н., доц. каф. 301 Краснов Л. О.
Дата захисту: 16 січня 2024 р.
1

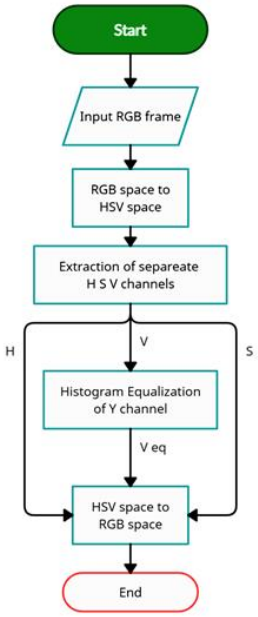
2

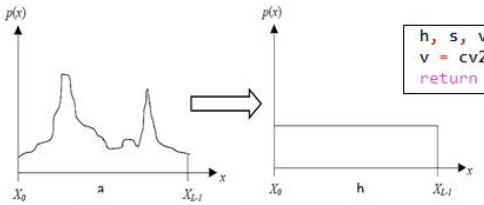
Мета роботи:

- Провести порівняльний аналіз відомих робіт щодо поліпшення контрастності та стабілізації яскравості даних у системах відеоспостереження
- Розробити універсальний критерій для об'єктивної оцінки стану відеоданих за різних умов зйомки та характеру освітленості сцени
- За результатами аналізу створити ефективний лінійний алгоритм стабілізації показника яскравості прийнятого відеопотоку;
- Дати оцінку властивостей відеоданих у результаті їх попередньої обробки за загальноприйнятими статистичними критеріями якості;
- Програмні коди для роботи нового алгоритму написати мовою Python з використанням функцій бібліотеки OpenCV;
- Достовірність отриманих результатів перевірити і підтвердити на реальних записах відеоспостережень.

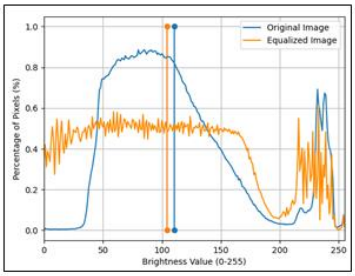
3

ІСНУЮЧІ ТЕХНІЧНІ РІШЕННЯ






```
h, s, v = cv2.split(image_hsv)
v = cv2.equalizeHist(v)
return cv2.merge((h, s, v))
```

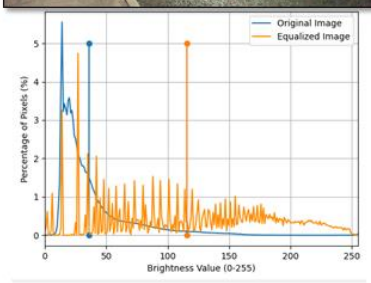


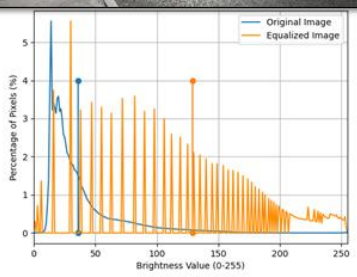
ВПЛИВ КОЛЬОРОВОГО ПРОСТОРУ НА АЛГОРИТМ

4

HSV

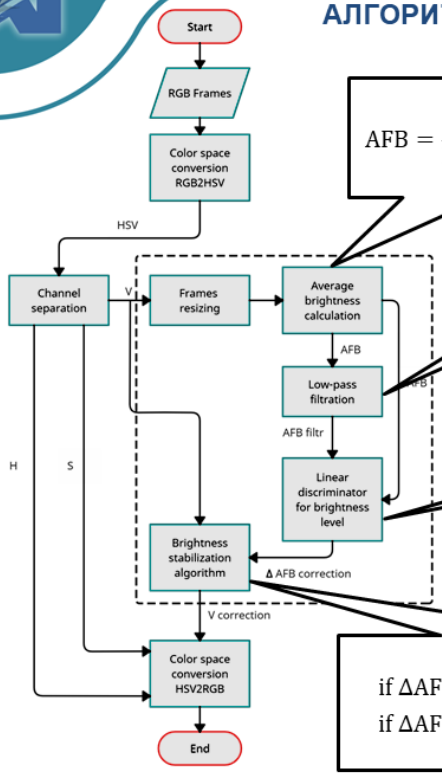
YIQ





АЛГОРИТМ ЛІНІЙНОЇ СТАБІЛІЗАЦІЇ ЯСКРАВОСТІ

5



$$\mathrm{AFB} = \frac{1}{\mathrm{MN}} \sum_{n=1}^{N} \sum_{m=1}^{M} \mathrm{V}(n,m)$$

$$\mathrm{AFB}_{\mathrm{filtr}} = \frac{1}{\mathrm{W}} \sum_{w=1}^{W} \mathrm{AFB}_{w}$$

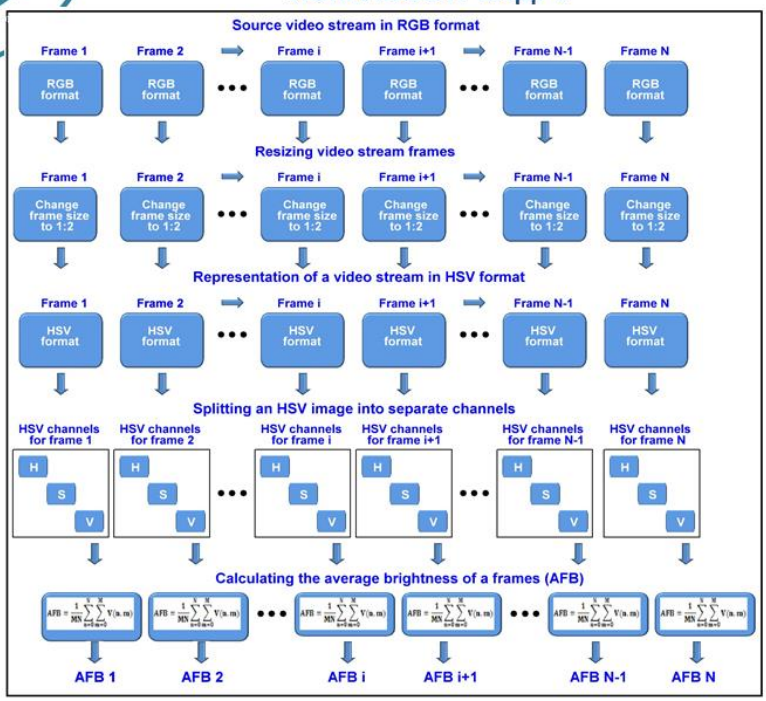
$$\Delta\mathrm{AFB}_{i} = \mathrm{AFB}_{i} - \mathrm{AFB}_{\mathrm{filtri}}$$

$$\text{if } \Delta\mathrm{AFB}_{i} \geq 0 \text{ then } \mathrm{AFB}_{i\ \mathrm{correction}} = \mathrm{AFB}_{i} - \Delta\mathrm{AFB}_{i}$$

$$\text{if } \Delta\mathrm{AFB}_{i} < 0, \text{then } \mathrm{AFB}_{i\ \mathrm{correction}} = \mathrm{AFB}_{i} + \Delta\mathrm{AFB}_{i}$$

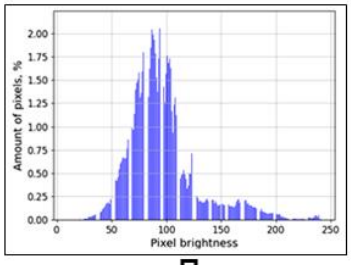
ПОБУДОВА ТРЕНДУ СЕРЕДНЬОЇ ЯСКРАВОСТІ КАДРУ

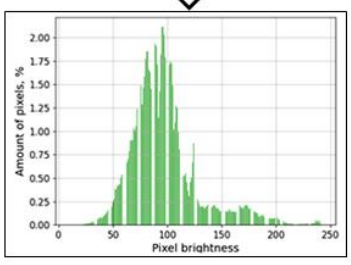
6

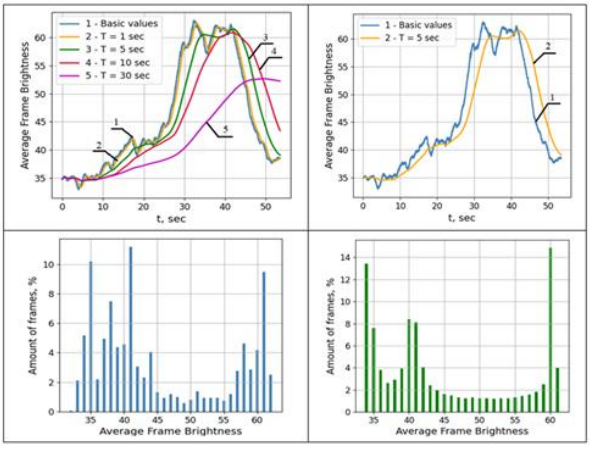


Результати фільтрації даних по середній яскравості кадрів та вплив розмірного коефіцієнту на якість перетворень

7

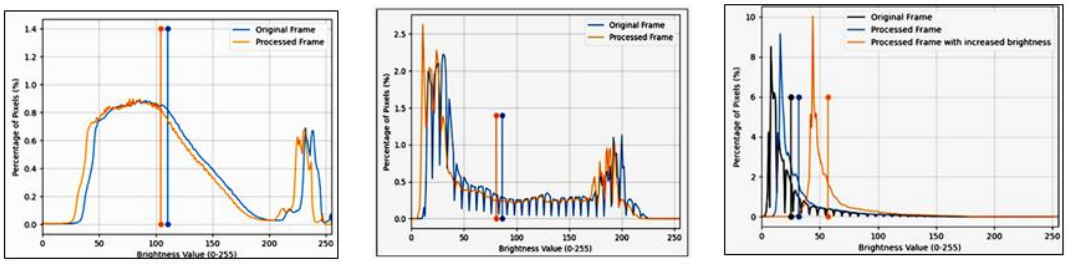






РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМУ ПРИ РІЗНИХ РІВНЯХ ОСВІТЛЕННОСТІ

8



СТАТИСТИЧНІ ПОКАЗНИКИ ЯКОСТІ ПЕРЕТВОРЮВАНЬ

9

$$NCC = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(S(i,j) \times C(i,j)) / \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(S(i,j))^2$$

$$AMBE = \left|AFB_{Ioutput} - AFB_{Iinput}\right|$$

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

$$SSIM(C,S) = \frac{(2\mu_C\mu_S + c_1) \times (2\sigma_{CS} + c_2)}{(\mu_C^2 + \mu_S^2 + c_1) \times (\sigma_C^2\sigma_S^2 + c_2)}$$

$$MSE = \frac{1}{M \times N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(I(m,n) - C(m,n)\right)^2$$

Показники якості еквалізації у просторі HSV

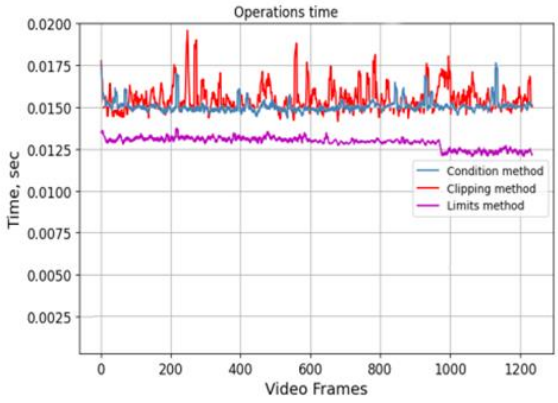
	A	B	C	D	E	F	Average
MSE	86.58	88.27	106.134	93.85	89.56	106.49	95.147
PSNR	28.76	28.67	27.87	28.41	28.61	27.86	28.36
NCC	0.9705	0.9766	0.8352	0.93	0.83	0.90	0.906
SSIM	0.87	0.88	0.35	0.77	0.41	0.58	0.643
AMBE	82.47	88.27	84.93	77.18	47.41	55.15	72.61

Показники якості алгоритму лінійної стабілізації яскравості у просторі HSV

	A	B	C	D	E	F	Average
MSE	16.5710	14.40	18.49	12.85	11.10	15.40	14.80
PSNR	35.94	36.55	35.46	37.04	37.68	36.25	36.48
NCC	0.999	0.997	0.996	0.99	0.997	0.997	0.997
SSIM	1.0	1.1	0.99	1.0	0.95	0.98	0.97
AMBE	4.06	4.24	4.21	3.79	3.85	3.78	3.98

ОЦІНКА ТА ОПТИМІЗАЦІЯ ШВИДКОДІЇ АЛГОРИТМУ

10




```python
if value >= 0:
    # Increase the brightness of V channel by beta, avoiding border values
    v = np.clip(v + value, 0, 255)
else:
    v = np.clip(v - value, 0, 255)
```

```python
    v = v.astype('int32')
    v[v < (0 + value)] = 0 + value
    v -= value
    v = v.astype('uint8')
```

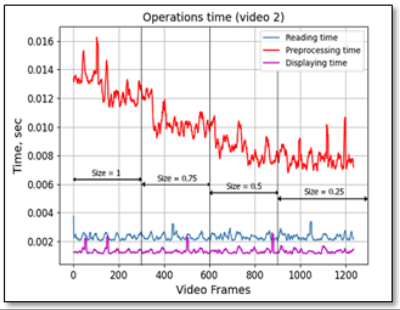
```python
    if value >= 0:
        lim = 255 - value
        v[v > lim] = 255
        v[v <= lim] += value
    else:
        v[v >= -value] -= -value
        v[v < -value] = 0
```

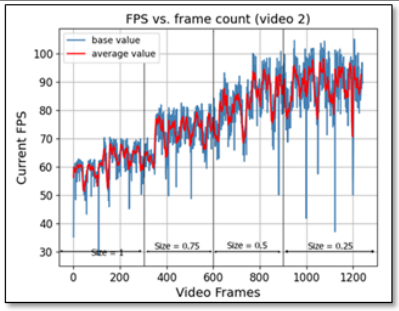
ОПТИМІЗАЦІЯ РОБОТИ АЛГОРИТМУ

11

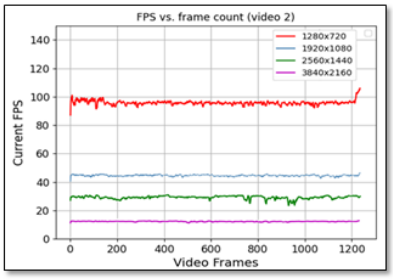
Вплив розмірного коефіцієнту (А) та роздільної здатності кадру (Б) на швидкодію алгоритму

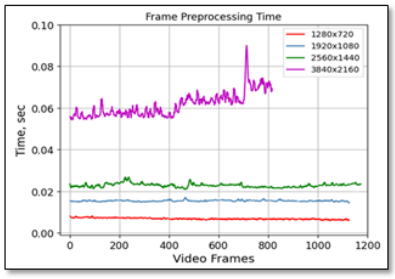
А





Б





ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ПРОЕКТУ

12

Розрахунок собівартості і ціни виробу за статтями

Стаття витрат		Сума, грн.	У відсотках від загальної суми
ФЗП	$З_{осн}$	83 005	47.60
	$З_{дод}$	16 601	9.52
Накладні витрати, $С_{накл}$		49 803	28.56
Соціальне страхування, СС		21 913	12.56
Експлуатаційні витрати	$С_{ЕЕ}$	568	0.32
	$С_{ТО}$	660	0.37
	$А_{год}$	1 411	0.8
Матеріали і комплектуючі, $С_{мик}$		396	0.22
Разом:		174 357	

Коефіцієнти кваліфікації програміста

Досвід роботи	коефіцієнт кваліфікації
До двох років	0.8
2-3 роки	1
3-5 років	1.1 - 1.2
5-7 років	1.3 - 1.4
більше 7 років	1.5 - 1.6

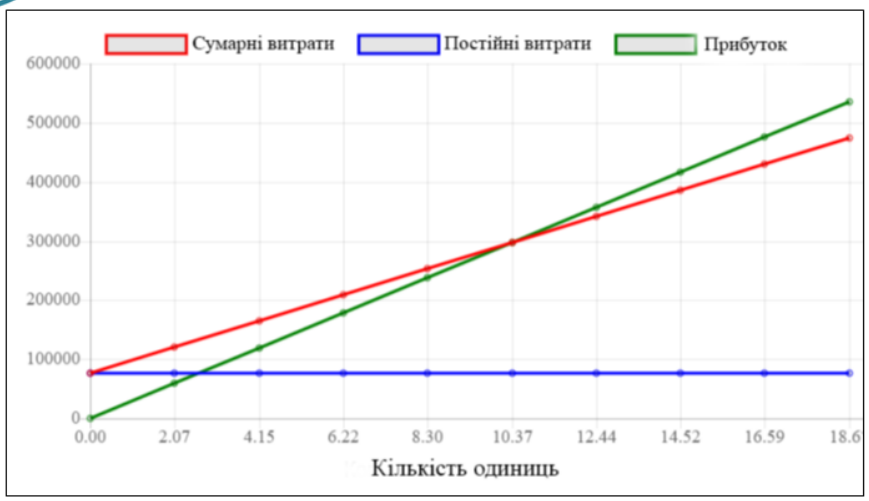
Оцінка часу підготовки опису завдання

q	Tmin	$Т_{нв}$	Tmax
100	10	15	20
500	20	35	50
1 000	25	50	75
1500	30	60	90
2000	40	70	100
2500	50	80	110
5000	70	110	150
10000	100	150	200

Сумарні витрати праці

$$t_c = t_{оп} + t_{ис} + t_{ал} + t_{пр} + t_{отл} + t_{д}$$
$$= 52 + 30 + 88 + 88 + 397 + 198 = 853[\text{люд. год}]$$

13



Критична програма випуску продукту

ДЯКУЮ ЗА УВАГУ