Міністерство освіти і науки України
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»
Факультет систем управління літальними апаратами
Кафедра систем управління літальних апаратів

Пояснювальна записка

до дипломної роботи
магістра
(освітньо-кваліфікаційний рівень)
на тему: Розробка алгоритмів читання та запису відео з використанням бібліотеки OpenCV

ХАІ.301.362.24О.151.183019 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”

Содилєв Н. Р.
(прізвище та ініціали студента)
Керівник Краснов Л. О.
(прізвище та ініціали)
Рецензент Трубчанінова К. А.
(прізвище та ініціали)

м. Харків – 2024 рік

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

Факультет систем управління літальними апаратами
Кафедра систем управління літальних апаратів (301)
Рівень вищої освіти другий (магістерський)
Галузь знань 15 «Автоматизація та приладобудування»
Спеціальність 151 Автоматизація та комп'ютерно - інтегровані технології
(шифр і назва)
Освітня програма Інженерія мобільних додатків

ЗАТВЕРДЖУЮ
Завідувач кафедри
систем управління ЛА
к.т.н., доц. К. Ю. Дергачов

"____" _____2024 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Содилєву Нікіті Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка алгоритмів читання та запису відео з використанням бібліотеки OpenCV
керівник роботи Краснов Л.О., к.т.н., доцент каф. 301,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 06.11.2023 року №1968-уч
2. Строк подання студентом роботи: 08.01.2024 року
3. Вихідні дані до роботи: програмне забезпечення для обробки та збереження відеозображень.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Огляд літератури предметної області, аналіз шляхів та способів рішення завдання, опис та розробка алгоритму, розробка програмного забезпечення, аналіз результатів моделювання системи, розрахунок собівартості виготовлення програмного забезпечення
5. Перелік графічного матеріалу:
13 слайдів формату А3 для презентації та захисту роботи(див. додаток С)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Стан проблеми	Доцент каф. 301 Краснов Л. О.	12.09.23	08.01.24
Аналіз і синтез СУ	Доцент каф. 301 Краснов Л. О.	12.09.23	08.01.24
Математичний опис СУ	Доцент каф. 301 Краснов Л. О.	12.09.23	08.01.24
Дослідницька частина	Доцент каф. 301 Краснов Л. О.	12.09.23	08.01.24
Експ.-практ. частина	Доцент каф. 301 Краснов Л. О.	12.09.23	08.01.24
Економічне обґрунтування	Доцент каф. 301 Краснов Л. О.	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	Прим.
1.	Початок переддипломної практики	2.10.2023	
2.	Формулювання теми роботи. Розробка технічного завдання	06.11.2023	
3.	Опис системи управління. Аналіз і синтез моделі. Проведення експериментальних досліджень	30.10.2023	
4.	Конструкторська частина роботи. Дослідницька частина роботи. Експериментально-практична частина. Економічне обґрунтування розробки.	06.11.2023	
5.	Оформлення розрахунково-пояснювальної записки і графічного матеріалу	11.11.2023	
6.	Попередній захист і рецензування роботи	10.12.2023	

Здобувач _____ ___Нікита СОДИЛЄВ
(підпис) (прізвище та ініціали)

Керівник роботи _____ _Леонід КРАСНОВ
(підпис) (прізвище та ініціали)

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»
Кафедра 301

«ЗАТВЕРДЖУЮ»
Завідуючий кафедрою
к.т.н., с.н.с., доцент
К.Ю. Дергачов
«_____»_____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на дипломне проектування

Содилєва Нікити Руслановича

1 Тема роботи: Розробка алгоритмів читання та запису відео з використанням бібліотеки OpenCV

затверджена наказом по університету від «06» Листопада 2023 р. №1968-уч.

2 Строк здачі студентом закінченої роботи «08» січня 2024 р.

3 Область застосування розробки: обробка відеозображення.

4 Початкові дані для розроблювальної системи

4.1 Призначення і мета створення системи: Обробка відеозображення, запис у окремий файл.

4.2 Загальні відомості; ознайомлення з рядом методів які використовуються для обробки та збереження відеозображення.

5 Технічні вимоги до каналів системи управління

5.1 Питання, що підлягають розробці: встановити залежності і фактори, які впливають на обробку, і точність 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 : точність визначення не менше 90%.

5.4 Вимоги до конфігурації обчислювальної техніки: 64-бітний двохядерний процесор з тактовою частотою від 1,2 ГГц, оперативна пам'ять від 1 ГБ DDR2 SDRAM, цифровий відеовихід: HDMI.

6 Умови експлуатації системи

6.1 Вимоги до програмної та інформаційної сумісності: кросплатформенність.

6.2 Вимоги до зовнішніх збурень: температура середовища (-10℃…+40℃), вологість середовища 20-80%, хімічно активні компоненти відсутні.

6.3 Характер роботи системи (безперервної, циклічний, одноразового дії): безперервний.

7 Обсяг виконуваних розроблювачем робіт

7.1 Етапи проведення роботи: 1) вибір теми; 2) розробка ТЗ на проектування ПЗ; 3) Оцінка стану проблеми, аналіз ТЗ; 4)Опис системи. Дослідження системи; 5) проектування ПЗ; 6) експериментальна частина; 7) оформлення записки; 8) захист дипломного проекту

8 Вимоги до захисту інформації й надійності: Не передбачені

9 Порядок контролю й приймання системи: система повинна досягати максимальної точності вимірювання даних.

10 Дослідницька частина: опис програми, вирішення питання швидкості роботи системи для обробки потокового відео з вебкамери.

11 Експериментально-практична частина: проведення тестування системи, знаходження похибок.

12 Економічна частина

12.1 Розробити (розрахувати, одержати): розрахувати собівартість і ціну розробки системи;

12.2 Умови і вимоги: річна програма випуску не менше 140 штук;

12.3 Очікуваний результат: виробництво даної системи повинно виходити на точку беззбитковості.

13 Перелік графічних матеріалів із зазначенням форматів: 13 слайдів формату А3 для презентації та захисту роботи(див. додаток С).

Керівник проектування
_____Краснов Л.О._____
(П.І.Б.)
_____
« » _____ 2023 р.

Прийняв до виконання
_____Содилєв Н. Р._____
(П.І.Б. студента)
_____
« » _____ 2023 р.

Погоджено з питань:

проектування
_____Краснов Л.О._____
(П.І.Б.)
_____
« » _____ 2023 р.

дослідницької частини
_____Краснов Л.О._____
(П.І.Б.)
_____
« » _____ 2023 р.

економіки
_____Краснов Л.О._____
(П.І.Б.)
_____
« » _____ 2023 р

РЕФЕРАТ

Розрахунково-пояснювальна записка: 74 сторінки, 17 рис., 7 табл., 15 джерел.
Мета роботи: Розробка програми для наглядного аналізу та вивчення обробки та збереження фото та відео матеріалів

КЛЮЧОВІ СЛОВА: OpenCV, Python, ОБРОБКА ВІДЕОЗОБРАЖЕНЬ, ВІДЕО, ЗОБРАЖЕННЯ.

Предмет роботи: Система обробки фото та відео зображень
Розроблена програма обробки на мові програмування Python.
В експериментальній частині був проведений експеримент, встановлені залежності і фактори, що впливають на точність та відображення відео файлу
В економічній частині проведено розрахунок собівартості і ціни виготовлення системи.

ЗМІСТ

ВСТУП 11
1. ОГЛЯД ЛІТЕРАТУРИ І ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ 12
1.1 Мета роботи та підходи до її вирішення 12
1.2 Аналіз методів рішення задач 12
1.3 Існуючі методи обробки та збереження відеозображень 16
1.4. Постановка задачі проектування 17
2. ДОСЛІДЖЕННЯ ПРОБЛЕМИ ТА СИНТЕЗ МОДЕЛІ 18
2.1 Концепція та класифікація зображень 18
2.2 Представлення зображень у пам’яті комп’ютера 19
2.3 Зчитування, вивід та збереження зображень 20
2.4 Базові операції над зображеннями 21
2.5 Запис та збереження відео 22
2.6. Бінаризація та обробка відео 23
2.7 Обробка зображень для аналізу 24
2.8 Компресія та оптимізація відеоданих 25
2.9 Підвищення швидкості обробки 26
2.10 Оптимізація алгоритмів 28
2.11 Приклади оптимізації алгоритмів 28
2.12 Колірний простір YCbCr 29
2.13 Модель стиснення відео стандарту MPEG 32
2.14 Дискретне косинусне перетворення 32
2.15 Квантування 34
3. ПРОЕКТУВАННЯ СИСТЕМИ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ OPENCV 36
3.1. Алгоритм реалізації моделі 36
3.1.1. Ініціалізація та Створення Графічного Інтерфейсу 36
3.1.2. Кнопки та елементи керування 36
3.1.3. Функціональні Чекбокси 36
3.1.4. Відображення Відео 36
3.2.1. Відтворення та зупинка відео 37
3.2.2. Оновлення відеофреймів 37
3.2.3. Обробка та відображення кадрів 37
3.3.1. Функція "Grayscale" 38
3.3.2. Функція "HSV Conversion" 38
3.4 Висновки за розділом 3 39
4. РЕАЛІЗАЦІЯ СИСТЕМИ 40
4.1 Дослідження системи 40
4.2 Інтерфейс програми 41
4.3 Висновки за розділом 4 45
5. ЕКОНОМІЧНА ЧАСТИНА 46
5.1. Мета економічної частини 46
5.2. Опис виробу 46
5.2.1. Управління рухом в робототехніці 46
5.2.2. Розпізнавання образів 46
5.2.3. Оптимізація алгоритмів 46
5.2.4. Практичне застосування 46
5.3. Сегментування ринку 47
5.3.1. Визначення сегментів 47
5.3.2. Оцінка ринкової ємності 47
5.3.3. Аналіз і вибір сегментів 47

5.3.4. Цільове позиціонування 47
5.4. Аналіз конкурентоспроможності.....48
5.5. Розрахунок собівартості та ціни виготовлення установки.....49
5.6 Висновки 54
ЗАКЛЮЧЕННЯ.....55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 56
ДОДАТОК А.....58
ДОДАТОК В.....63

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення;

СУ- система управління;

ПК – персональний комп’ютер;

КЗ – комп’ютерний зір;

OpenCV - Open Source Computer Vision;

ВСТУП

В сучасному світі зростання доступності великого обсягу даних та швидкість розвитку технологій відкривають нові перспективи для дослідження та розробки різноманітних інноваційних рішень. Однією з ключових областей, що привертає увагу вчених та інженерів, є обробка відеоданих. Відео, як основний елемент мультимедійного змісту, знаходить широке застосування в різних галузях, включаючи медицину, науку, індустрію, інформаційні технології та багато інших.

Існує безліч завдань, пов'язаних із відео, які вимагають глибокого розуміння та вдосконалення обробки відеоданих. В одному з напрямків цієї проблематики виникає питання про розробку програмного забезпечення для обробки та запису відео, що є важливим етапом в обробці візуальної інформації. У цьому контексті бібліотека OpenCV, яка є потужним інструментом для роботи з відеоданими, стає ключовим інструментом для розробників та дослідників.

Мета цієї магістерської роботи полягає у дослідженні та розробці ефективних методів обробки відеоданих з використанням бібліотеки OpenCV, а також у створенні програмного забезпечення для запису та обробки відео з метою визначення нових можливостей та покращення існуючих технологій. У даній роботі буде розглянуто широкий спектр завдань, таких як фільтрація, аналіз руху, обробка кольору та використання різноманітних алгоритмів машинного навчання для досягнення високоякісних результатів в обробці та аналізі відеоданих.

Завдяки поєднанню теоретичного підходу до аналізу та реалізації програмних алгоритмів у середовищі OpenCV, магістерська робота має на меті внести свій внесок у вдосконалення технік та методів обробки відеоданих, сприяючи подальшому розвитку області комп'ютерного бачення та аналізу відеоінформації.

У подальших розділах магістерської роботи буде проведено аналіз існуючих підходів до обробки відеоданих, описані основні принципи роботи бібліотеки OpenCV та наведені результати експериментів, що підтверджують ефективність розроблених методів та програмного забезпечення.

1. ОГЛЯД ЛІТЕРАТУРИ І ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Мета роботи та підходи до її вирішення

Магістерська робота на тему "Розробка алгоритмів читання та запису відео з використанням бібліотеки OpenCV[17]" має на меті розгляд аспектів обробки відеоданих за допомогою популярного інструментарію для комп'ютерного зору – OpenCV. Завдання включає в себе ретельний аналіз та розробку алгоритмів для обробки відеозаписів з використанням функціоналу OpenCV.

Основні етапи виконання роботи включають:

Огляд існуючих рішень: Дослідження поточних підходів та методів, що використовують OpenCV для аналізу та запису відеозображень.

Розробка алгоритмів обробки відео: Розгляд створення ефективних та точних алгоритмів для виявлення об'єктів, відстеження руху та аналізу вмісту відеозаписів.

Інтеграція OpenCV: Проведення практичних експериментів з використанням OpenCV для обробки вхідних відеоданих та вибір найбільш ефективних функцій бібліотеки. [11]

Реалізація запису відео: Розробка функціоналу для запису оброблених відеозображень з використанням OpenCV, включаючи параметри якості та форматування виведеного відео.

Оцінка та порівняння результатів: Проведення аналізу ефективності розроблених алгоритмів та функціоналу з врахуванням різних сценаріїв та типів відеоданих.

Документація та висновки: Написання звіту, який включатиме опис розроблених рішень, їхню ефективність та можливі напрямки подальших досліджень у цій області. Ця магістерська робота відкриває можливості для подальшого розвитку методів обробки відеоданих з використанням OpenCV та внесення вагомого вкладу в галузь комп'ютерного зору та комп'ютерного бачення.

1.2 Аналіз методів рішення задач

Можна розглядати різні методи, які забезпечують ефективне відтворення та зберігання відео зображень.[12]

- Кодеки відеостискання**:** Використання різних кодеків для стискання та зберігання відеоданих. Наприклад, H.264, H.265, або MPEG. Аналізується ефективність стискання, якість зображення та ресурсо-витратність.

- Регулювання параметрів зображення**:** Визначення впливу різних параметрів, таких як роздільна здатність, кадрова швидкість, та бітовий потік на якість та розмір файлу відеозапису.
- Робота з камерами та стрімінгом**:** Вивчення можливостей OpenCV для роботи з камерами, а також для обробки та зберігання відеострімів у реальному часі.
- Використання розширених функцій OpenCV**:** Дослідження можливостей OpenCV для додаткового обробки відео, таких як виявлення об'єктів, відстеження руху, або аналіз елементів на зображенні.
- Робота з аудіо супроводом**:** Вивчення можливостей синхронізації відео та аудіо, а також методів для зберігання аудіодоріжки у відеофайлі.

Розглянемо кожен з пунктів більш докладно. Використання кодеків у відеоопрацюванні є важливою складовою роботи з відеоданими в бібліотеці OpenCV. Кодеки відеостиснення дозволяють стискувати та декомпресувати відеодані, зменшуючи їхній розмір і полегшуючи їхню передачу та зберігання.

Кодеки відеостискання:

- Вибір оптимального кодеку**:** Одним із ключових завдань є вибір кодеку, який відповідає вимогам магістерської роботи. Різні кодеки мають різні рівні стиснення та якості зображення. Наприклад, H.264 часто використовується для зберігання високоякісних відео з невеликим розміром файлу.
- Параметри кодеку**:** Розуміння та встановлення параметрів кодеку є важливим етапом. Налаштування, такі як бітовий потік, профілі та рівні стиснення, впливають на якість та розмір відеозапису.
- Взаємодія з OpenCV**:** OpenCV надає зручний інтерфейс для роботи з кодеками. Функції, такі як `cv2.VideoWriter()`, дозволяють визначити тип кодеку та його параметри при збереженні відео.
- Стискання та втрати якості**:** Аналіз впливу стиснення на якість відеозображення є необхідним для забезпечення оптимального балансу між розміром файлу та якістю. Необхідно досліджувати рівні стиснення, щоб уникнути значущих втрат у якості відображення.
- Підтримка апаратного прискорення**:** Врахування можливостей апаратного прискорення, яке надається деякими кодеками та платформами, для оптимізації продуктивності та зменшення навантаження на обчислювальні ресурси.

- Робота з різними форматами**:** Розгляд можливостей роботи з різними форматами відео (наприклад, AVI, MP4, MKV) та вибір формату, який найбільше відповідає вимогам проекту.

Регулювання параметрів зображення:

- Роздільна здатність (резолюція): Аналіз різних рівнів роздільної здатності відеозображення дозволяє визначити оптимальний баланс між деталізацією та розміром файлу. Висока роздільна здатність може покращити якість, але при цьому може збільшити розмір файлу.
- Кадрова швидкість**:** Вивчення впливу кадрової швидкості на відеозапис. Визначення оптимальної кадрової швидкості для конкретного застосування, з урахуванням обчислювальних ресурсів та сприйняття людського ока.
- Бітовий потік: Аналіз впливу бітового потоку на якість та розмір відеофайлу. Встановлення оптимальних значень бітового потоку для забезпечення відповідності вимогам якості та об'єму файлу.
- Компресія та формати зображення**:** Розгляд можливостей різних методів компресії та вибір формату зображення (наприклад, JPEG, PNG) для збереження відео зображення з врахуванням вимог до якості та займаного місця.
- Контрастність та яскравість**:** Визначення оптимальних значень контрастності та яскравості для поліпшення візуального враження при відтворенні відео.
- Колірні фільтри та насиченість**:** Дослідження можливостей використання різних колірних фільтрів та насиченості для досягнення бажаного візуального ефекту.
- Гама-корекція та налаштування освітлення**:** Врахування впливу гама-корекції та параметрів освітлення на відображення відео та вибір оптимальних налаштувань для конкретних умов.
- Адаптивне регулювання параметрів**:** Розгляд можливостей автоматизованого адаптивного регулювання параметрів зображення в реальному часі в залежності від змінюваних умов (освітлення, об'єкти на зображенні).

Робота з камерами та стрімінгом:

- Захоплення відео з камери**:** Використання OpenCV для ініціалізації та захоплення відеосигналу з різних джерел, таких як вбудована веб-камера, зовнішня USB-камера або інші відео пристрої.

- Контроль над параметрами камери**:** Аналіз можливостей OpenCV для управління різними параметрами камери, такими як роздільна здатність, кадрова швидкість, експозиція та інші, для досягнення оптимальних умов захоплення.
- Обробка в реальному часі**:** Розгляд можливостей обробки відеосигналу в реальному часі за допомогою OpenCV, що включає виявлення об'єктів, фільтрацію, відстеження руху та інші техніки.
- Збереження відеопотоку**:** Аналіз методів для збереження обробленого відеопотоку у вигляді відеофайлу або передачі його по мережі в режимі стрімінгу.
- Використання мережевих камер**:** Дослідження можливостей роботи з мережевими камерами та IP-камерами, включаючи протоколи передачі даних та взаємодію OpenCV з цими пристроями.
- Стрімінг в мережі: Розгляд можливостей для передачі відеопотоку по мережі за допомогою OpenCV, включаючи вибір протоколів (HTTP, RTSP) та оптимізацію для мінімізації затримок.
- Обробка аудіосигналу**:** У разі потреби аналіз вивчення можливостей обробки аудіосигналу разом з відеосигналом та синхронізація аудіо та відео.

Використання розширених функцій OpenCV:

OpenCV[16] - це потужна бібліотека для обробки зображень і відео. Вона містить широкий спектр функцій, які можна використовувати для вирішення різних завдань, таких як виявлення об'єктів, розпізнавання лиць, відстеження руху та багато іншого.

Основні функції OpenCV[15]

- `cv2.imread()` ця функція використовується для завантаження зображення з файлу або з камери.
- `cv2.imshow()` ця функція використовується для відображення зображення на екрані.
- `cv2.imwrite()`ця функція використовується для збереження зображення у файл.
- `cv2.cvtColor()` ця функція використовується для перетворення формату кольору зображення.
- `cv2.threshold()`ця функція використовується для порогового регулювання зображення.

- `cv2.findContours()` ця функція використовується для виявлення контурів на зображенні.

OpenCV також містить широкий спектр додаткових функцій, які можна використовувати для вирішення більш складних завдань. Ось деякі приклади:

- Алгоритми розпізнавання лиць - ці алгоритми використовуються для розпізнавання осіб на зображеннях або відео.
- Алгоритми відстеження руху - ці алгоритми використовуються для відстеження переміщення об'єктів на зображеннях або відео.

Робота з аудіо супроводом:

- Синхронізація аудіо та відео: Важливим етапом є точна синхронізація аудіо та відео. OpenCV надає можливість використання функцій для визначення точного відповідного моменту відео та аудіо.
- Обробка та фільтрація аудіосигналу**:** Аналіз методів для обробки аудіосигналу, таких як фільтрація шуму, еквалайзинг та інші техніки для покращення якості звуку.
- Збереження аудіо у відео-файлі: Вивчення можливостей OpenCV для об'єднання аудіо та відео в одному файлі, включаючи вибір формату файлу та кодеку для забезпечення оптимального збереження.
- Аудіо стрімінг**:** Визначення можливостей стрімінгу аудіо разом з відео по мережі. Розгляд аспектів безперервного відтворення та передачі аудіоданих через мережу.
- Виявлення та аналіз аудіо-подій: Використання OpenCV для виявлення та аналізу аудіо-подій, таких як голосові команди, звукові ефекти або інші аспекти, що впливають на відтворення відео.
- Інтеграція аудіо з іншими джерелами**:** Розгляд можливостей інтеграції аудіо з іншими джерелами, такими як музичні файли чи аудіозаписи з інших джерел.
- Дослідження просторового аудіо**:** У випадку використання технологій просторового звуку, аналіз можливостей для відтворення аудіо у відповідних аудіоканалах.

1.3 Існуючі методи обробки та збереження відеозображень

Обробка та збереження відеозображень займає центральне місце в інформаційних технологіях та мультимедійних системах. Розпізнавання об'єктів на відео — це ключовий аспект, який використовує такі алгоритми як YOLO та SSD для точного визначення та відстеження об'єктів у реальному часі.

Додатково, методи обробки включають в себе фільтрацію для покращення якості відеозображень, зміну роздільної здатності для оптимізації передачі та зменшення обсягу даних. Зокрема, алгоритми компресії, такі як H.264 та H.265, використовуються для ефективної збереження та передачі великих обсягів відеоінформації.

За останні роки відзначається зростаючий інтерес до використання нейромереж та алгоритмів глибокого навчання для складніших завдань обробки та аналізу відеоданих, що відкриває нові перспективи у розробці мультимедійних систем та технологій розпізнавання образів.

1.4. Постановка задачі проектування

Проект націлено на створення системи для відтворення та зберігання відеозображень, використовуючи бібліотеку OpenCV[17] в мові програмування Python. Мета полягає в розробці ефективного інструменту для оброблення та збереження великих обсягів відеоматеріалів.

Визначено потребу у створенні системи, яка забезпечить точне захоплення відеосигналу з камери та інших джерел. Основні функціональні вимоги включають обробку відео з використанням функціоналу OpenCV[17], такого як виявлення об'єктів та руху, а також налаштування параметрів зображення, таких як роздільна здатність, кадрова швидкість, бітовий потік та інші.

Також визначено, що важливо забезпечити синхронізацію та об'єднання аудіо та відео, а також реалізувати методи збереження відео в різних форматах та кодеках. З технічної точки зору, проект передбачає використання мови програмування Python та бібліотеки OpenCV[17].

Етапи реалізації включають в себе огляд функціоналу OpenCV[17] для захоплення відео та обробки зображень, створення інтерфейсу для налаштування параметрів відеозображення, інтеграцію аудіо та відео, а також визначення методів збереження відео в різних форматах та кодеках.

Проект планується піддати тестуванню для визначення його ефективності та надійності. Здійснюватиметься оцінка роботи системи з точки зору виконання та використання ресурсів. Очікується розробка оптимізованої системи, яка задовольнить визначені потреби.

2. ДОСЛІДЖЕННЯ ПРОБЛЕМИ ТА СИНТЕЗ МОДЕЛІ

2.1 Концепція та класифікація зображень

Що таке зображення? На це питання можна відповісти по різному. Найпростішим і найширшим визначенням цього поняття є таке: Зображення - це те, що ми бачимо.

Інше визначення: Зображення - це ін формація, придатна для візуального сприйняття. В залежності від походження, умовно можна виділити наступні типи зображень:

- Рисоване або друковане (джерелом є художник, поліграфія, принтер);
- Оптичне (розподіл інтенсивності електромагнітного поля, що створюється оптичним прибором у деякій області простору (області локалізації), наприклад, на сітківці ока, на екрані при проектуванні, у площині приймача об’єктиву фотоапарата);
- Фотографічне (оптичне зображення, зареєстроване на фотоматеріалі у результаті хімічного процесу);
- Електронне або цифрове (оптичне зображення, зареєстроване з допомогою електронного приймача, наприклад ПЗЗ-матриці (прилад із зарядовим зв’язком), сканера).

Електронним також називають зображення, що відображається на екрані монітора. Легко бачити, що цей поділ є умовним. Зображення із одного типу відразу переходить у інший. Ланцюжок цих перетворень у більшості випадків закінчується зображенням на сітківці ока і образом у мозку людини. Поняття зображення можна формалізувати, описати математично і маніпулювати ним для досягнення певних цілей. Ці маніпуляції назвемо обробкою зображень.

Метою оброки зображень може бути:

- Зміна (спотворення) зображення з метою досягнення тих чи інших ефектів (художнє покращення);
- Image Processing - візуальне (помітне оку) покращення якості зображення (корекція яскравості і контрасту, корекція кольорів та ін.); об’єктивне покращення якості зображення (усунення спотворень зображення різних типів);
- Image Analysis - проведення вимірів на зображенні (аналіз інтерферограм, гартманограм та ін.);
- Image Understanding - розпізнавання образів (розпізнавання символів, відбитків пальців, обличь та ін.).

2.2 Представлення зображень у пам’яті комп’ютера

Для перетворення аналогового сигналу у цифровий, у сучасних фотоапаратах використовуються спеціальні сенсори, за допомогою яких ми отримуємо дискретну матрицю $n \times m$. Чим більша кількість світлочутливих елементів, тим кращу якість фотографій можна отримати. Це визначається параметрами матриці.

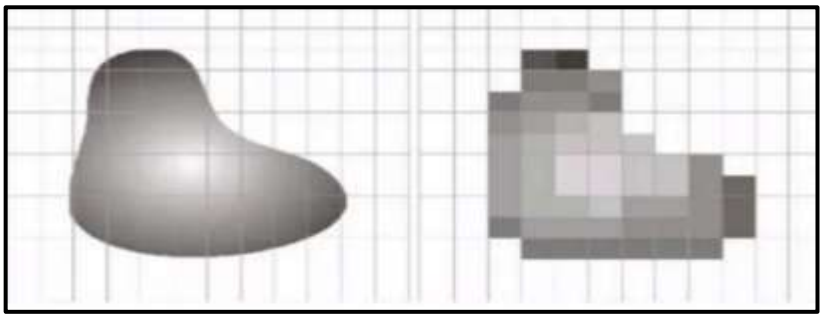


Рисунок 2.1 – Переклад аналогового зображення у цифрове

Як показано на рис. 2.1 наведено процес перекладу аналогового зображення у цифрове, що вказує на технологічний аспект конвертації візуальної інформації.

При переводі аналогового зображення у цифрове ми розбиваємо на частини не тільки область нашого малюнку, а і квантуємо кольори. Якщо у нас хороша камера, то зображення розбивається на стільки маленьких частинок пікселів (від англ. picture element), що наше око не помічає різниці.

На якість зображення (чорно-білого) впливає як кількість пікселів, виділених на це зображення, так і кількість біт, виділених на один піксель. Так, якщо на один піксель виділено 8 біт, то у ньому можна представити 256 відтінків. При цьому 0 буде позначати чорний колір, а 255 - білий.

Якщо для представлення чорно-білого малюнку потрібна одна матриця, то для кольорових малюнків ми використовуємо три матриці - червону, зелену і синю (RGB).

За типом даних зображення поділяють на:

- бітові (булевські, логічні);
- байтові (зі знаком і без знаку);
- цілочисельні (зі знаком і без знака);
- дійсні (з фіксованою і плаваючою крапкою);
- кольорові (спеціальний тип даних);
- векторні (піксель є масивом або списком чисельних значень).

2.3 Зчитування, вивід та збереження зображень

Розглянемо основні операції для роботи із зображеннями. Перш за все, потрібно підключити необхідні бібліотеки.

```
Import cv2
Import numpy
From matplotlib import pyplot as plt
Img = cv2.imread('image.jpg')
print(img.shape)
img
```

Для зчитування зображень використовується функція `cv2.imread`. Дана функція має два параметри:

- path - шлях до файлу, який необхідно відно відкрити;
- flag - опція, що вказує, яким чином слід відкрити файл. Функція `cv2.imread` повертає масив numpy. Якщо файл не буде знайдено, то буде повернено порожній масив. Параметр flag може приймати наступні значення:
- `cv2.IMREAD_COLOR`: завантажуване зображення розглядається як кольорове. Будь-яку прозорість зображення буде проігноровано. Це є значенням параметру за замовчуванням. Замість цього значення можна також передати значення 1.
- `cv2.IMREAD_GRAYSCALE`:завантажуване зображення розглядається як кольорове. Альтернативним варіантом задання цього параметру є число 0.
- `cv2.IMREAD_UNCHANGED`:показує, що завантажуване зображення містить альфа канал (тобто містить інформацію не тільки про значення кольорових каналів, наприклад інформацію про прозорі частини зображення). Альтернативним варіантом задання цього параметру є число - 1.

Для виведення зображення на екран існує два можливі варіанти:

1)Вивід у окремому вікні

2)Вивід в комірці PyCharm

```
window_name = "Image"
cv2.imshow(window_name,img)
cv2.waitKey(0)
cv2.dwstroyAllWindows()
```

У цьому випадку для виводу використано функцію `cv2.imshow`. Після появи віконця із зображенням, слід викликати функцію `cv2.waitKey(0)` для паузи, та `cv2.destroyAllWindows()` для закриття усіх вікон.

2.4 Базові операції над зображеннями

Оскільки зображення представляються у вигляді матриць (або масивів матриць, як у випадку із кольоровими зображеннями), то над операції над матрицями можуть бути застосовані і до зображень. Так, можна змінювати пікселі зображень, виділяти окремі області зображень і т.д. Аналогічно ми можемо змінювати значення пікселя та виділяти окремі області зображення. Також ми можемо розділяти канали зображення, склеювати їх і т.д.

Над зображеннями також можна використовувати арифметичні операції. Якщо додати число до усіх елементів зображення, то збільшиться інтенсивність (зображення стане світліше), якщо відняти - зменшиться.

Самі зображення також можна додавати. При цьому використовується формула:

$$g(x)=(1-\alpha)f0(x)+\alpha f1(x). \quad (2.1)$$

Змінюючи α від нуля до одиниці, отримуємо плавний перехід між зображеннями. Для додавання зображень використовується функція `cv2.addWeighted()`, яка додає зображення за формулою:

$$img=\alpha\cdot img1+\beta\cdot img2+\gamma. \quad (2.2)$$

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

Колірна модель - абстрактна модель опису представлення кольорів у вигляді кортежів (наборів) чисел, зазвичай з трьох або чотирьох значень, званих колірними компонентами або колірними координатами.

Разом з методом інтерпретації цих даних (наприклад, визначення умов відтворення та / або перегляду - тобто завдання способу реалізації), множина кольорів колірної моделі визначає колірний простір.

Також під колірною моделлю необхідно розуміти спосіб відображення колірної гами в дискретному вигляді, для представлення її в обчислювальних, цифрових системах.

В OpenCV ми здебільшого матимемо справу із трьома колірними моделями: grayscale (чорно-білі зображення),BGR та HSV (або HSB). Розглянемо ці моделі детальніше:

- Greyscale - це модель, що зводить усю колірну інформацію до відтінків сірого, тобто до яскравості окремих частин зображення. Ця модель дуже корисна, коли інформація про яскравість є достатньою (наприклад при розпізнаванні обличь). Зазвичай кожен піксель представляється 8-бітним числом, від 0 (чорне), до 255 (біле).

- BGR - у цій моделі кожен піксель представляється трійкою значень, кожне з яких відповідає за окремий канал: синій, зелений та червоний. В даній моделі трійка (0,0,0) позначає чорний колір, (255,0,0) - синій, (0,255,0) - зелений, (0,0,255) - червоний.
- HSV (також HSB) - дана колірна модель, заснована на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value), який також називають яскравістю (Brightness). Тон змінюється у проміжку [0,179], насиченість у [0,255], а значення кольору у проміжку [0,255]. Різні програми використовують різні шкали, тож при обробці зображень в OpenCV може виникнути необхідність нормалізації значень.

Для конвертації між різними колірними моделями в OpenCV використовується функція `cv2.cvtColor(input_image, flag)`, де flag визначає тип конвертації. Для переведення BGR у Grayscale використовується значення прапорця `cv2.COLOR_BGR2GRAY`. Аналогічно, для переведення BGR у HSV, ми використовуємо прапорець `cv2.COLOR_BGR2HSV`.

2.5 Запис та збереження відео

У сучасному світі відео стає все більш популярним форматом для обміну інформацією та взаємодії з користувачами. Програмісти та розробники випереджають цей тренд, створюючи додатки та проекти, які використовують відео для різноманітних цілей. Мова програмування Python, завдяки своїй простоті та розширюваності, часто використовується для обробки відеоданих.

Для запису відео використовуються різні бібліотеки, але однією з найпопулярніших є OpenCV. OpenCV (Open Source Computer Vision) надає функціонал для роботи з відео та зображеннями. Процес запису відео розпочинається з визначення параметрів камери чи відеозаписувача, налаштування формату виводу та фреймрейту. Приклад такого коду наведений далі.

```
import cv2
video_writer = cv2.VideoWriter('output_video.mp4',
cv2.VideoWriter_fourcc(*'mp4v'), 30, (640, 480))
while True:
    ret, frame = capture.read()
    if not ret:
        break
    video_writer.write(frame)
video_writer.release()
```

Щоб зберегти відео, необхідно забезпечити правильні налаштування кодеку, роздільної здатності, та швидкості кадрів. OpenCV використовує чотири символи для визначення кодеку. У прикладі використовується 'mp4v', але інші можливі варіанти - 'XVID', 'MJPG', тощо.

```
import cv2
video_capture = cv2.VideoCapture('input_video.mp4')
video_writer = cv2.VideoWriter('output_video.mp4',
cv2.VideoWriter_fourcc(*'mp4v'), 30, (640, 480))
while True:
    ret, frame = video_capture.read()
    if not ret:
        break
    video_writer.write(frame)
video_writer.release()
video_capture.release()
```

Запис та збереження відео в Python відкриває широкі можливості для розробників. Це може використовуватися у сферах комп'ютерного зору, машинного навчання, розпізнавання обличчя, створення відеоконтенту та інших областях. За допомогою бібліотек, таких як OpenCV, розробники можуть легко маніпулювати відеоданими та створювати нові застосування, що використовують відео як ключовий компонент. З плином часу можна очікувати розвиток нових технологій та підходів, що ще більше полегшать і розширять можливості роботи з відео в середовищі Python.

2.6. Бінаризація та обробка відео

Бінаризація у відео — це техніка обробки зображень, що полягає в перетворенні кольорового або відтінкового зображення в чорно-білий формат, де кожен піксель або певна область отримує значення чорного або білого в залежності від заданого порогу. Цей процес є важливим етапом у візуальному аналізі та обробці відео з метою виділення або приховання певних об'єктів, спрощення даних або підвищення ефективності обробки в реальному часі.

Основна ідея бінаризації у відео полягає у встановленні порогового значення, яке розділяє пікселі на дві групи: ті, що мають значення менше порогу, та ті, що мають значення рівне або вище порогу. Такий підхід дозволяє визначати наявність або відсутність певних характеристик у зображенні.

Наприклад, якщо ми маємо відео у кольоровому форматі, можна провести бінаризацію для виділення об'єктів певного кольору або яскравості, встановивши порогове значення для відповідного каналу (червоного, зеленого, або синього). Чорні пікселі в результаті вказуватимуть на відсутність цих характеристик, а білі - на їхню наявність.

Застосування:

- Обробка зображень для аналізу: Бінаризація дозволяє виділити об'єкти або області інтересу для подальшого аналізу. Наприклад, в реальному часі може проводитися відстеження руху або визначення наявності об'єктів на відеозаписі.
- Компресія та оптимізація даних: Бінаризація може використовуватися для зменшення обсягу відеоданих, особливо у випадку, коли важливо лише визначити наявність певних об'єктів чи областей.
- Підвищення швидкості обробки: В деяких випадках, замість опрацювання повного спектру кольорів, можна обмежитися аналізом лише чорного та білого, що може покращити швидкість обробки в реальному часі.

Рисунок 2.2 – Приклад бінаризації

На рисунку 2.2 наведено приклад бінаризації. Вона використовує алгоритми порогового визначення, такі як глобальний поріг, локальний поріг, адаптивна бінаризація тощо. Вони можуть враховувати специфічні характеристики зображення для оптимального вибору порогу в кожному випадку.

2.7 Обробка зображень для аналізу

Обробка зображень для аналізу є ключовою галуззю у сфері комп'ютерного зору і обробки візуальної інформації. Цей процес включає в себе використання різних технік та алгоритмів для витягнення корисної інформації з зображень з метою подальшого розпізнавання об'єктів, вимірювання характеристик, виявлення шаблонів і багато іншого. Важливим етапом в цьому процесі є попередня обробка зображень, яка може включати в себе такі етапи, як підготовка даних, фільтрація, видалення шуму, а також бінаризацію, яку ми обговорювали раніше.

Основні етапи обробки зображень для аналізу:

- Завантаження та підготовка даних: Перший крок - це завантаження зображення і підготовка даних для подальшого аналізу. Це може включати вирівнювання, зміну розміру, конвертацію кольорів і інші операції, які допомагають підготувати зображення для подальшого обробку.

- Фільтрація та видалення шуму**:** Зображення може містити різноманітний шум, який може виникати внаслідок різних факторів, таких як нестабільність освітлення чи апаратні артефакти. Фільтри та методи видалення шуму використовуються для зменшення впливу цього шуму на подальший аналіз.
- Визначення контурів та контурних ознак**:** Алгоритми визначення контурів виділяють зображення об'єктів та їхніх меж. Це важливий етап для подальшого розпізнавання об'єктів, адже контурні ознаки можуть містити інформацію про форму, розмір та інші характеристики об'єктів на зображенні.
- Бінаризація**:** Як вже зазначалося, бінаризація конвертує кольорові або відтінкові зображення в чорно-білий формат, визначаючи поріг значень пікселів. Це дозволяє визначити наявність чи відсутність певних характеристик на зображенні.
- Витягнення ознак та дескрипторів**:** На цьому етапі використовуються різні алгоритми для витягнення ознак та дескрипторів з областей зображення. Це може включати в себе використання локальних багатовимірних дескрипторів, які служать для унікальної ідентифікації різних об'єктів.
- Класифікація та розпізнавання**:** На останньому етапі використовуються алгоритми машинного навчання для класифікації об'єктів чи розпізнавання певних патернів на зображенні. Застосування навчених моделей дозволяє автоматизувати процес розпізнавання та класифікації.

Обробка зображень для аналізу відіграє ключову роль у багатьох сферах, таких як медицина, автоматизація, безпека та багато інших. Вона дозволяє витягати інформацію з візуального контенту та робити це ефективно та автоматизовано.

2.8 Компресія та оптимізація відеоданих

Компресія та оптимізація[18] даних є важливими аспектами обробки та зберігання інформації. Вони мають велике значення в різних сферах, включаючи передачу даних через мережі, зберігання на носіях, оптимізацію роботи програм та багато іншого. Давайте розглянемо основні принципи та призначення обох цих концепцій.

Компресія даних - це процес зменшення обсягу даних з метою зменшення їхнього обсягу або економії місця при зберіганні та передаванні. Це досягається за рахунок вилучення зайвої інформації або використання спеціальних алгоритмів, які дозволяють зберігати ту саму інформацію за менше місця.

Основні методи компресії:

- Безвтратна компресія: Безвтратна компресія дозволяє відновити оригінальні дані без втрати якості. Це важливо для деяких типів даних, де навіть незначна втрата інформації неприйнятна. Прикладами є алгоритми Хаффмана, LZ77, архіватори ZIP, RAR.
- Втратна компресія: Втратна компресія призводить до втрати певної інформації, але це допустимо в тих випадках, коли важлива лише загальна структура даних, а не кожен окремий елемент. Прикладами є стандарти зображень, такі як JPEG для фотографій та MPEG для відео.

Оптимізація даних включає в себе ряд методів та стратегій для поліпшення ефективності обробки та використання ресурсів, зокрема, мінімізацію використання пам'яті та часу обчислень.

Основні аспекти оптимізації даних:

- Алгоритмічна оптимізація**:** Вибір ефективних алгоритмів для обробки та зберігання даних може значно поліпшити продуктивність. Оптимізовані алгоритми виконують завдання за менший час та з меншим обсягом використання ресурсів.
- Оптимізація використання пам'яті: Ефективне управління пам'яттю грає важливу роль в оптимізації. Це може включати в себе кешування, використання стиснених форматів даних або стратегії управління пам'яттю.
- Паралельність та розподілені системи**:** Використання паралельних обчислень та розподілених систем може допомогти розподілити завдання та прискорити обробку великих обсягів даних.
- Кешування та попередні обчислення**:** Використання кешування результатів попередніх обчислень може заощаджувати час та ресурси при повторних запитаннях або використанні даних.
- Оптимізація мережевої взаємодії**:** Для передачі даних через мережі важливо оптимізувати протоколи та використовувати стиснення для зменшення обсягу передаваємих даних.

Об'єднання компресії та оптимізації даних дозволяє досягати ефективного використання ресурсів, зменшення обсягу зберігання та прискорення обробки, що є ключовими аспектами в багатьох областях інформаційної технології.

2.9 Підвищення швидкості обробки

Підвищення швидкості обробки є важливим аспектом в багатьох областях, таких як комп'ютерний зір, обробка сигналів, штучний інтелект, аналіз даних та інші. Цей процес спрямований на те, щоб забезпечити більш ефективну роботу алгоритмів та програм, що обробляють великі обсяги інформації в реальному

часі. Давайте розглянемо деякі ключові принципи та методи підвищення швидкості обробки.

- Оптимізація алгоритмів**:** Вибір оптимальних алгоритмів є ключовим етапом для підвищення швидкості обробки. Деякі алгоритми мають кращу ефективність у порівнянні з іншими при роботі з конкретними видами даних чи завдань. Оптимізація алгоритмів може включати в себе вибір більш ефективних структур даних, використання апаратних оптимізацій та інше.
- Використання паралелізму та розподілених систем**:** Розділення завдань та використання паралельних обчислень дозволяє використовувати різні обчислювальні ресурси одночасно, що призводить до збільшення швидкості обробки. Можливості паралельного виконання можуть бути реалізовані за допомогою багатоядерних процесорів або груп обчислювальних вузлів в розподілених системах.
- Кешування та попередні обчислення: Зберігання проміжних результатів обчислень у кеші або використання попередніх обчислень може допомогти уникнути зайвих повторних обчислень та зменшити час виконання. Ефективне використання кеш-пам'яті може значно підвищити швидкість обробки.
- Апаратне прискорення: Використання спеціалізованих апаратних засобів, таких як графічні процесори (GPU), тензорні процесори (TPU) або спеціальні процесори для обробки сигналів (DSP), може значно підвищити швидкість обробки певних видів даних або завдань.
- Стиснення та оптимізація даних: Зменшення обсягу даних шляхом стиснення чи використання оптимізованих форматів даних може покращити час передачі та обробки інформації. Це особливо важливо в мережевому взаємодії чи зберіганні великих обсягів даних.
- Компіляція та інтерпретація: Використання ефективних компіляторів, оптимізаторів та використання віртуалізації для покращення виконання програм.
- Оптимізація мережевої взаємодії: При передачі даних через мережу важливо використовувати ефективні протоколи та стратегії, щоб уникнути зайвої затримки та збільшити швидкість передачі.
- Конкурентність та асинхронність: Використання конкурентних та асинхронних програмних підходів може покращити реактивність систем та використовувати час ефективно.

Підвищення швидкості обробки - це багатоплановий підхід, який враховує аспекти алгоритмів, апаратного забезпечення, паралелізму та оптимізації даних. Застосування цих стратегій дозволяє досягти оптимальної продуктивності в областях, де швидкість обробки має ключове значення.

2.10 Оптимізація алгоритмів

Оптимізація алгоритмів — це процес вдосконалення роботи алгоритмів для зменшення часу виконання, оптимізації використання пам'яті та збільшення ефективності в цілому. Оптимізація може включати в себе виправлення швидкості, роботи з даними, структур даних та використання апаратного забезпечення. Ось деякі основні принципи та приклади оптимізації алгоритмів:

- Вибір ефективних структур даних: Використання оптимальних структур даних може драматично покращити продуктивність алгоритмів. Наприклад, для швидкого пошуку елементів чи вставки/видалення може бути вигідніше використовувати хеш-таблиці або дерева пошуку, залежно від конкретного завдання.
- Приклад**:** У заміжжій (balanced) бінарній кучі пошук/вставка/видалення елементів відбувається за час O(log n), що робить її ефективною для великих обсягів даних.
- Оптимізація циклів та ітерацій: Мінімізація кількості ітерацій у циклах може значно зменшити час виконання.
- Приклад**:** Замість подвійного циклу можна використовувати один, якщо відомо, що пари елементів не повторюються.
- Мемоізація**:** Збереження результатів попередніх обчислень та їх використання в подальших викликах може уникнути повторних обчислень.
- Приклад**:** У динамічному програмуванні можна використовувати мемоізацію для збереження результатів попередніх викликів функцій та уникнення зайвих обчислень.
- Використання бінарного пошуку: Бінарний пошук ефективний для відсортованих даних та може значно зменшити час пошуку елементів.

2.11 Приклади оптимізації алгоритмів

- Оптимізація алгоритмів обробки зображень: У візуальних задачах, таких як розпізнавання обличчя, важливо мати швидкі алгоритми обробки зображень. Наприклад, використання алгоритмів розпізнавання обличчя, які використовують швидкі та ефективні методи визначення ключових точок обличчя, може суттєво покращити продуктивність системи в реальному часі.

- Бінаризація зображень для відокремлення об'єктів: При аналізі зображень для виявлення об'єктів часто використовується бінаризація для перетворення зображення в чорно-білий формат. Використання оптимізованих алгоритмів бінаризації, таких як адаптивна бінаризація або використання порогових значень, може поліпшити швидкість обробки.
- Оптимізація алгоритмів відслідковування об'єктів в відео: У задачах відслідковування об'єктів в відео важливо використовувати швидкі та точні алгоритми. Наприклад, алгоритми, які використовують фільтр Калмана або методи оптимізованого відслідковування, можуть допомогти в покращенні продуктивності системи.
- Оптимізація алгоритмів зменшення шуму в зображеннях: Під час обробки зображень часто важливо зменшити шум для отримання чіткіших та якісніших результатів. Використання ефективних алгоритмів фільтрації, таких як медіанний фільтр або фільтр Гаусса, може поліпшити якість обробки та знизити час виконання.
- Оптимізація алгоритмів розпізнавання об'єктів на зображеннях: В розпізнаванні об'єктів важливо використовувати ефективні моделі та алгоритми класифікації. Використання оптимізованих нейронних мереж, таких як MobileNet чи EfficientNet, може забезпечити високу швидкість розпізнавання об'єктів на зображеннях.

Ці приклади вказують на те, як оптимізація алгоритмів може суттєво поліпшити швидкість та ефективність обробки зображень та відео у візуальних завданнях.

2.12 Колірний простір YCbCr

Колірний простір YCbCr складається з трьох компонент: компоненти яскравості Y і двох компонент колірності Cb і Cr. Такий розподіл обумовлений тим, що людське зорове сприйняття має велику чутливість до яскравості, ніж до кольору об'єкта . У зв'язку з цим компоненти колірності Cb і Cr можна зберігати з меншою роздільною здатністю, що дозволяє зменшити обсяг збережених або передаваних даних . Також колірний простір YCbCr часто використовується в стеганографічних алгоритмах для роботи з чорно-білим зображенням, використовуючи канал яскравості.

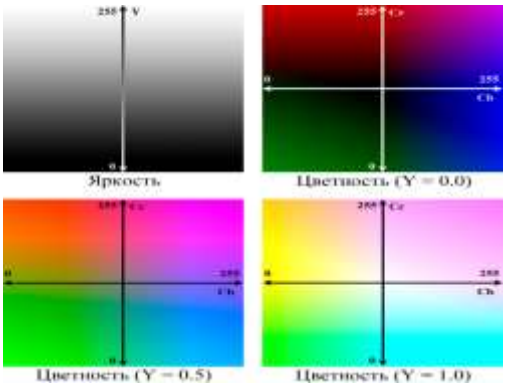


Рисунок 2.3 – Кольоровий простір YCbCr

На рис. 2.3 відображений кольоровий простір YCbCr, що представляє собою метод кодування кольорів, де яскравість (Y) та дві колірні складові (Cb та Cr) розділені, сприяючи ефективній обробці та передачі кольорової інформації у цифрових системах.

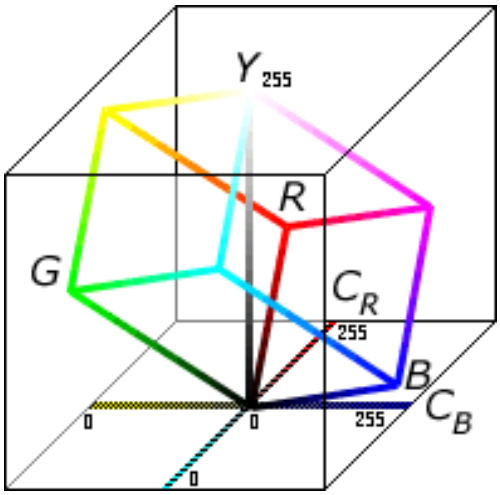


Рисунок 2.4 - 3D представлення колірного простору YCbCr

На рис. 2.4 представлене 3D подання колірного простору YCbCr, що демонструє тривимірну організацію яскравості та колірних компонентів. Це графічне зображення сприяє кращому розумінню просторових відношень між різними кольоровими елементами, що важливо для обробки та аналізу візуальної інформації.

Переклад зображення з колірного простору RGB в простір YCbCr можна виконати за допомогою формули, яку використовує формат JPEG . У цій формулі компоненти простору RGB та YCbCr знаходяться в інтервалі [0;255].

$$\begin{cases} Y = 0.299R + 0.597G + 0.114B \\ Cb = 128 - 0.1687R - 0.3313G + 0.5B \\ Cr = 128 + 0.5R - 0.4187G - 0.0813B \end{cases} \quad (2.3)$$

Обернене перетворення з простору кольору YCbCr в простір RGB можна виконати за допомогою формули, в якій компоненти обох просторів також повинні знаходитися в інтервалі [0;255].

$$\begin{cases} R = Y + 1.402(C_r - 128) \\ G = Y - 0.34414(C_b - 128) - 0.71414(C_r - 128) \\ B = Y + 1.772(C_b - 128) \end{cases} \quad (2.4)$$

Двійкові циклічні коди входять до різних типів кодів, завдяки яким процеси кодування та декодування легко описати і реалізувати:

- Лінійні коди: Коди цього типу дозволяють описати процеси кодування та декодування за допомогою лінійної алгебри.
- Блокові коди: Ці коди є кодами фіксованої довжини. Під час кодування інформаційне слово W довжини k перетворюється в кодове слово C(W) довжини n.
- Двійкові коди: Ці коди використовують модулярну арифметику за модулем 2, яка оперує символами 0 і 1.
- Систематичні коди: Кодові слова цього типу можна розділити на дві частини: інформаційне (вихідне) слово і перевірочне (додаткове) слово.
- Циклічні коди: Кожен циклічний зсув кодового слова даного підмножини також є кодовим словом.

При визначенні конкретного двійкового циклічного коду використовують три параметри:

- n – довжина кодового слова;
- k – довжина інформаційного (вихідного) слова;
- g(X) – породжуючий поліном коду.

$$g(X) = g_0X^0 + g_1X^1 + \cdots + g_rX^r \quad (2.5)$$

g(X) – це породжуючий многочлен (n, k)-коду, де:

- g_i ∈ {0,1}, що вказує на коефіцієнти многочлена, які можуть бути 0 або 1,

- r = n-k, що представляє різницю між загальною довжиною кодового слова (n) і довжиною інформаційного слова (k),

- X^i – позначення місця відповідної компоненти в кодовому слові (векторі), де i ∈ [0, r].

Многочлен g(X) можна представити у вигляді двійкового слова g_r g_(r-1)…g_1 g_0. Породжуючим многочленом g(X) циклічного (n, k)-коду може бути будь-який многочлен, який ділить многочлен X^n+1 без залишку. Він називається породжуючим через те, що в буквальному сенсі породжує циклічний код, тобто перетворює інформаційне слово в кодове.

2.13 Модель стиснення відео стандарту MPEG

MPEG (Motion Picture Experts Group) - це група експертів, яка розробляє, оновлює та підтримує стандарти стиснення цифрової аудіо та відео інформації . Група MPEG створила багато стандартів, серед яких найважливішими є MPEG-1, MPEG-2 та MPEG-4. MPEG-1 - це перший стандарт, розроблений для стиснення аудіо та відео об'єктів з подальшим записом на компакт-диск. Стандарт MPEG-2 використовується у телевізійному мовленні та відео на DVD. Стандарт MPEG-4 має вищий ступінь стиснення порівняно з MPEG-2 та дозволяє працювати з об'єктами (зображення, тривимірні моделі, текстові дані).
Високий рівень стиснення в стандартах MPEG досягається за допомогою методів стиснення інформації з втратами. Ці методи усувають значну кількість між-кадрової та внутрішньо-кадрової зайвості.

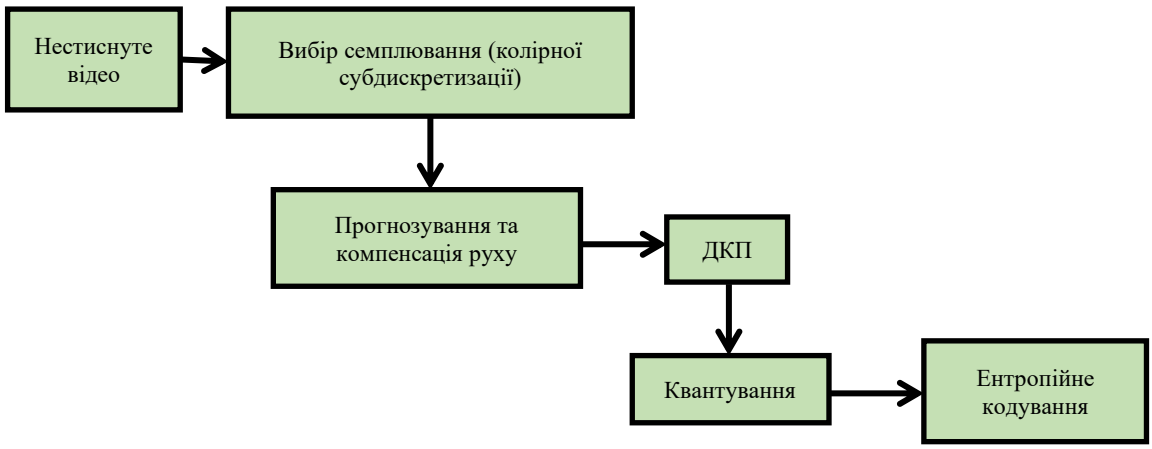


Рисунок 2.5 – Модель сжимання GPEG

На рис. 2.5 наведена модель стиснення GPEG (Joint Photographic Experts Group), що вказує на використання цього стандарту для зменшення розміру та збереження візуальної якості цифрових зображень.

2.14 Дискретне косинусне перетворення

Дискретне косинусне перетворення (ДКП)[14] відоме тим, що пікселі на зображенні корелюють зі своїми сусідами, оскільки значення конкретного пікселя можна передбачити за його сусідами. ДКП зменшує цю зайвість між пікселями. Воно перетворює вихідну матрицю даних в матрицю некорельованих

величин, використовуючи суми косинусів на різних частотах. ДКП має обернене перетворення.

$$\mathbf{F}[u,v] = a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{F_{DCT}}[x,y]\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad (2.12)$$

$$\mathbf{F_{DCT}}[x,y] = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}a(u)a(v)\mathbf{F}[x,y]\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad (2.13)$$

де $\mathrm{F_{DCT}}$ – початкова матриця,

F – матриця коєфіцієнтів ДКП,

N – розмір матриці F та $\mathrm{F_{DCT}}$,

$$a(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{если } k = 0 \\ \sqrt{\frac{2}{N}}, & \text{если } k > 0 \end{cases}, \quad (2.14)$$

$k = u$, або $k = v$

$u, v, x, y \in [0, N)$.

Перетворення відбувається таким чином, що коефіцієнти матриці ДКП впорядковані за частотою. Спочатку йдуть низькочастотні коефіцієнти, потім середньочастотні і високочастотні. Низькочастотні коефіцієнти містять найважливішу інформацію для відновлення вихідних даних, і їх зміна призводить до сильного спотворення даних після застосування оберненого перетворення. Високочастотні коефіцієнти можна відсікти (занулити) без сильного впливу на дані після застосування оберненого перетворення, що відбувається на етапі квантування.

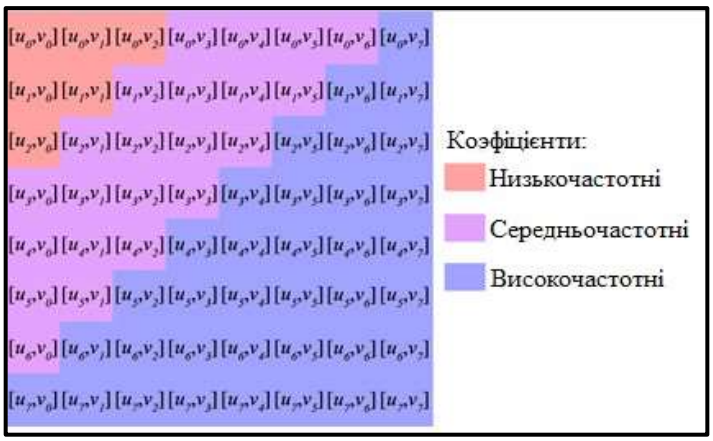


Рисунок 2.7 – Коєфіцієнти квантування

На рис. 2.7 відображені коєфіцієнти квантування, що використовуються у процесі обробки та стиснення зображень.

У моделі стиснення MPEG в якості даних для матриці беруться числові значення компонент простору кольору YCbCr. Для цього компоненти Y, Cb і Cr розбиваються на блоки розміром 8x8 або 4x4, і потім до кожного з блоків застосовується ДКП.

2.15 Квантування

На етапі квантування матриця коефіцієнтів ДКП перетворюється в нову матрицю з зменшеною областю значень. Перетворення виконується послідовним діленням кожного коефіцієнта матриці ДКП на значення кроку квантування і округленням отриманого значення .

$$\mathbf{F_Q}[u,v] = round\left(\frac{\mathbf{F}[u,v]}{Q_{step}}\right) \tag{2.15}$$

де F_Q – квантована матриця коєфіцієнтів ДКП,

F – початкова матриця коєфіцієнтів ДКП,

Q_{step} – крок квантування,

$u, v \in [0, N)$,

N – розмір матриці F та F_Q,

$round$ – операція округлення.

Нижче наведено приклад квантування матриці коефіцієнтів ДКП з кроком 5.

$$\begin{bmatrix} 52.4 & -8.05 & 1.446 & 3.7 \\ 12.79 & 4.64 & -9.46 & -0{,}1 \\ -4.83 & 2.04 & -5.13 & 1.813 \\ 1.71 & -3.644 & 2.201 & 0.868 \end{bmatrix} \xrightarrow{Q_{step}=5} \begin{bmatrix} 10 & -2 & 0 & 1 \\ 3 & 1 & -2 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}$$

Через операцію округлення втрачається частина даних, і під час застосування зворотного квантування при декодуванні отримана матриця буде в певній мірі відрізнятися від початкової.

$$\mathbf{F}'[u,v] = \mathbf{F_Q}[u,v] * Q_{step} \tag{2.16}$$

де F' – матриця коєфіцієнтів ДКП, що пройшла квантування і зворотне квантування,

F_Q – квантована матриця коєфіцієнтів ДКП,

Q_{step} – крок квантування,

$u, v \in [0, N)$,

N – розмір матриці F та F_Q.

Нижче наведено приклад зворотного квантування матриці коефіцієнтів ДКП з кроком 5.

$$\begin{bmatrix} 10 & -2 & 0 & 1 \\ 3 & 1 & -2 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \xrightarrow{Q_{step}=5} \begin{bmatrix} 50 & -10 & 0 & 5 \\ 15 & 15 & -10 & 0 \\ -5 & 0 & -5 & 0 \\ 0 & -5 & 0 & 0 \end{bmatrix}$$

2.16 Висновки

У даному розділі було проведено аналіз поняття "зображення" та визначено його різні типи залежно від походження. Зображення охарактеризовано як інформацію, придатну для візуального сприйняття, що може бути представлене різними методами, такими як рисоване, оптичне, фотографічне чи електронне. Визначено, що поділ цих типів є умовним, оскільки зображення може переходити від одного типу до іншого.

Крім того, обговорено мету обробки зображень, яка включає в себе зміну, аналіз та розпізнавання образів. Зазначено, що обробка зображень може бути спрямована на досягнення різних цілей, таких як художнє покращення, візуальне та об'єктивне поліпшення якості зображення.

3. ПРОЕКТУВАННЯ СИСТЕМИ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ OPENCV

3.1. Алгоритм реалізації моделі

3.1.1. Ініціалізація та Створення Графічного Інтерфейсу

Ініціалізація та створення графічного інтерфейсу відбуваються при створенні об'єкту класу VideoPlayer. Цей процес включає в себе налаштування основних елементів, таких як вікно, кнопки, функціональні чекбокси та поле для відображення відео.

Головне вікно Tkinter створюється за допомогою `tk.Tk()`. Це вікно виступає в ролі контейнера для інших елементів інтерфейсу. Ми встановлюємо назву вікна ("Video Player") та його розмір ('1080x700').

```
self.root = tk.Tk()
self.root.title("Video Player")
self.root.geometry('1080x700')
```

3.1.2. Кнопки та елементи керування

Додавання кнопок для керування відтворенням та іншими функціями відбувається за допомогою класу `tk.Button`. Кнопки "Play", "Stop", "Load Video" та інші додаються до вікна інтерфейсу. Кожна кнопка пов'язана з відповідною функцією, яка буде викликатися при натисканні.

```
self.btn_play = tk.Button(self.root, text="Play",
command=self.play_video)
self.btn_play.pack(side=tk.LEFT)
```

3.1.3. Функціональні Чекбокси

Функціональні чекбокси (Checkbutton) використовуються для активації або деактивації різних фільтрів обробки відео. Ці елементи дозволяють користувачеві вибирати, які трансформації будуть застосовані до відеопотоку.

```
self.gray_checkbox = tk.Checkbutton(self.root, text="Gray",
command=self.toggle_gray)
self.gray_checkbox.pack(side=tk.LEFT)
```

3.1.4. Відображення Відео

Для відображення відео використовується `tk.Canvas`, який створюється заздалегідь визначеним розміром. На цьому полі зображення буде виводитися у вигляді відеокадрів.

```
    self.canvas = tk.Canvas(self.root, width=650, height=400)
    self.canvas.pack()
```

Ці елементи створюють інтуїтивний та користувацьки орієнтований графічний інтерфейс для взаємодії з відеопрогравачем. Вони надають користувачеві зручні інструменти для управління відтворенням відео та зміною його візуальних властивостей.

3.2.1. Відтворення та зупинка відео

Кнопки "Play" та "Stop" відповідають за управління відтворенням відео. Метод `play_video` встановлює флаг `is_playing`, який сигналізує про активне відтворення. Метод `stop_video` зупиняє відтворення, встановлюючи флаг в значення False.

```
def play_video(self):
    self.is_playing = True

def stop_video(self):
    self.is_playing = False
```

3.2.2. Оновлення відеофреймів

Оновлення відеофреймів відбувається в методі update. Цей метод викликається кожні 10 мілісекунд, і він визначає, чи відтворення активне. Якщо так, він зчитує новий кадр відео за допомогою OpenCV, обробляє його відповідно до встановлених параметрів та відображає на відповідному Canvas.

```
def update(self):
    if self.is_playing:
        ret, frame = self.cap.read()
        # ... обробка та відображення кадру ...
    self.root.after(10, self.update)
```

3.2.3. Обробка та відображення кадрів

Кожен кадр відео піддається обробці в методах `update_frame` та `update`. Вони викликаються в різних контекстах: `update_frame` викликається при ручному переміщенні слайдера, а `update` викликається в режимі відтворення.

```
def update_frame(self, value):
    frame_position = int(value)
    self.cap.set(cv2.CAP_PROP_POS_FRAMES, frame_position)
    ret, frame = self.cap.read()
```

Кожен кадр піддається операціям, таким як конвертація кольорового простору, зеркальне відображення, а також можливість перетворення у відтінки сірого або інші опції відповідно до вибору користувача. Зображення потім відображається на Canvas.

Ці функціональності надають користувачеві повний контроль над відтворенням та обробкою відео в реальному часі через інтуїтивний графічний інтерфейс.

3.3.1. Функція "Grayscale"

Опис алгоритму**:**

- Для кожного пікселя відеозображення виконується перетворення відтінку кольору на його яскравість (якщо відтінок взяти зі спектру кольорів).
- Отриманий відтінок кольору використовується для створення відтінків сірого, де 0 представляє чорний, а 255 - білий.

Безпосереднє застосування**:**

- Розглянемо RGB-піксель: (R, G, B).
- Яскравість (яка відповідає відтінку в градаціях сірого) обчислюється за формулою: brightness = 0.299 * R + 0.587 * G + 0.114 * B.
- Всі три компоненти (brightness, brightness, brightness) використовуються для нового відтінку кольору пікселя.

3.3.2. Функція "HSV Conversion"

Функція "HSV Conversion" виконує конвертацію кольорового простору з RGB до HSV (відтінок, насиченість, значення). Це дозволяє користувачу працювати з кольорами в більш інтуїтивно зрозумілому форматі, що може бути корисним при виборі конкретних відтінків для аналізу або обробки.

Опис алгоритму**:**

- Відтінок (Hue)**:** Представляє кольоровий тон і вимірюється у градусах (0° - 360°). Зазвичай зображується на колі, де 0° та 360° - це червоний колір.
- Насиченість (Saturation)**:** Вимірюється відсотками та вказує на "яскравість" кольору. Насиченість 0% - відтінок відтворює відтінки сірого.
- Значення (Value)**:** Також вимірюється відсотками і вказує на яскравість кольору. Значення 0% представляє чорний колір.

  Безпосереднє застосування**:**

- Розглянемо RGB-піксель: (R, G, B).
- Конвертація у HSV простір: `hsv_pixel=cv2.cvtColor((R, G, B), cv2.COLOR_BGR2HSV)`.
- Відтінок (Hue), насиченість (Saturation), та значення (Value) пікселя можна отримати як hsv_pixel[0], hsv_pixel[1], hsv_pixel[2] відповідно.

Ці функції взаємодіють зі зображенням, яке отримано з відеопотоку, та використовують можливості OpenCV для ефективної обробки та трансформації

кадрів. Кожна функція активується або деактивується за допомогою відповідного флажка на графічному інтерфейсі, надаючи користувачеві гнучкість у виборі параметрів обробки в режимі реального часу.

3.4 Висновки за розділом 3

У даному розділі детально розглянуто алгоритм реалізації моделі системи, яка використовує бібліотеку OpenCV. Зокрема, описано процес ініціалізації та створення графічного інтерфейсу, що відбувається при створенні об'єкту класу VideoPlayer. Цей процес включає в себе конфігурування основних елементів, таких як вікно, кнопки, функціональні чекбокси та поле для відображення відео.

Головне вікно Tkinter використовується як контейнер для інших елементів інтерфейсу, а його параметри, такі як назва та розмір, налаштовуються відповідно до вимог. Детально розглянуто додавання кнопок для керування відтворенням та іншими функціями за допомогою класу tk.Button. Кожна кнопка пов'язана з певною функцією, що викликається при натисканні.

Також розглянуто використання функціональних чекбоксів (Checkbutton), які дозволяють користувачеві активувати або деактивувати різні фільтри обробки відео. Ці елементи інтерфейсу надають користувачеві можливість вибору трансформацій, які будуть застосовані до відеопотоку.

Загальний висновок розділу підкреслює успішну інтеграцію функціоналу, який надає бібліотека OpenCV, у графічний інтерфейс, що сприяє ефективній роботі та зручності використання створеної системи.

4. РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Дослідження системи

У цьому розділі розглянута конкретна реалізація розробленої системи, яка базується на використанні бібліотеки OpenCV та включає в себе важливі аспекти її функціональності. Описано алгоритм роботи програми та показано результати взаємодії системи з вебкамерою.

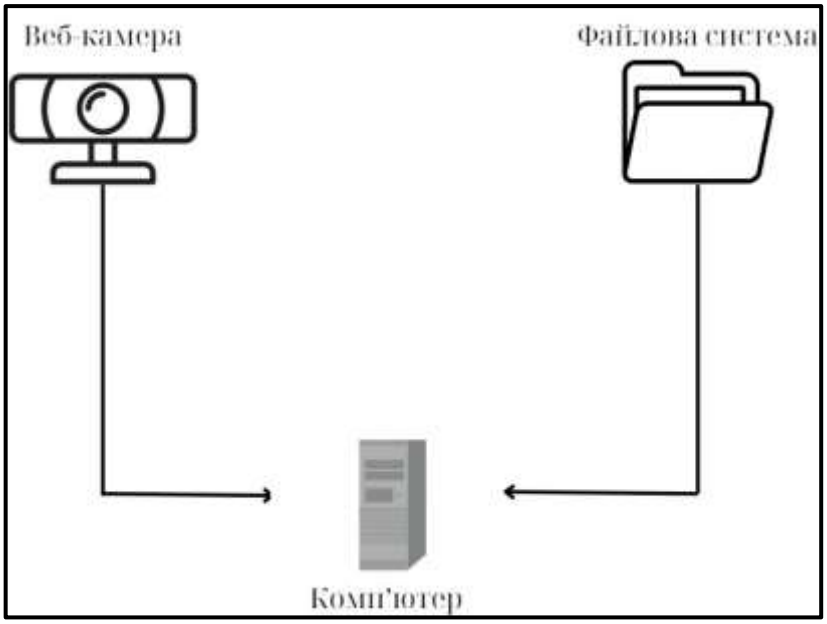


Рисунок 4.1 – Структурна схема роботи програми

На рис. 4.1 представлена структурна схема роботи програми, що надає узагальнений огляд взаємозв'язків та функціональних блоків системи.

Рисунок 4.2 - Зовнішній вигляд макета

Характеристики макета:

1) Відстань 30-40см;

2) USB камера 1stPlayer 1ST-WC01FHD;

3) Фокусна відстань 4.8мм;
4) Матриця 2 МП;
5) Тип сенсора : CMOS;
6) Роздільна здатність: 1920x1080;
7)Максимальна частота кадрів: 30Гц;
8) Глибина кольору: 24 біт;
9) Інтерфейс: USB 2.0.
Мінімальна вимога до персонального компютера:
1) Предвстановлена операційна система Windows,MacOS, Linux;
2) Процесор: 64-бітний двоядернийз тактовою частотою від 1.2 ГГц;
3) Оперативна пам’ять: від 1 ГБ DDR2;
4) Порти USB кількістю від 2 шт.

4.2 Інтерфейс програми

На рис. 4.3 –представлений зовнішній вигляд програми. Це візуальне представлення може включати в себе графічні елементи, кнопки управління, чекбокси та інші компоненти, які дозволяють користувачеві взаємодіяти з програмою для обробки відеоданих або взаємодії з відеопотоком.

Рисунок 4.3 – Зовнішній вигляд програми

На рис. 4.4 представлено відеопотік з вебкамери, яке відображає реальний час роботи вебкамери.

Рис. 4.4 – Відображення роботи вебкамери

На рис. 4.5 наведено приклад роботи сірого фільтру на вебкамері показано ефект відображення в реальному часі роботи сірого фільтру на відеопотоці з вебкамери.

Рисунок 4.5 – Відображення роботи сірого фільтру на вебкамері

На рис. 4.6 показано приклад роботи HSV фільтру на вебкамері" показано результат застосування HSV (відтінок, насиченість, значення) фільтру до відеопотоку з вебкамери.

Рисунок 4.6 – Відображення роботи HSV фільтру на вебкамері

На рис. 4.7 наведено приклад роботи відзеркалювання вебкамери" представлено візуальний результат застосування ефекту відзеркалення до відеопотоку з вебкамери.

Рисунок 4.7 – Приклад роботи відзеркалювання вебкамери

На рис. 4.8 – показано приклад роботи сірого фільтру на відео", представлено відтворення ефекту сірого фільтру на відео. Цей приклад демонструє конвертацію кольорового відеосигналу у відтінки сірого, що призводить до відсутності кольорового контрасту та виокремлення лише яскравості об'єктів на відеозапису.

Рисунок 4.8 – Приклад роботи сірого фільтру на відео

На рис. 4.9 зображено приклад роботи HSV фільтру на відео" відображено візуалізацію застосування HSV (відтінок, насиченість, значення) фільтру до відео. Цей приклад ілюструє вплив фільтрації на кольоровий простір, виокремлюючи та виділяючи різні відтінки та насиченості в обраному відеоматеріалі.

Рисунок 4.9 – Приклад роботи HSV фільтру на відео

На рис. 4.10 – демонструється приклад відзеркалювання, видно об'єкт, який відображено в дзеркалі, в результаті чого створюється дублюючий ефект та інверсія образу. Це ілюструє явище відзеркалення та підкреслює взаємозв'язок між оригіналом та його відображенням у дзеркалі.

Рисунок 4.10 – Приклад відзеркалювання

4.3 Висновки за розділом 4

У цьому розділі представлено результати реалізації розробленої системи. Фотографії роботи програми і зображення стенду, на якому встановлено вебкамеру, демонструють ефективність та стабільність роботи системи в реальному часі. Взаємодія програми з вебкамерою відображена на зазначених зображеннях, і це підкреслює успішну інтеграцію розробленого рішення.

Структурна схема роботи програми, яка також включена у цей розділ, надає загальний огляд архітектури системи та взаємозв'язку між її ключовими компонентами. Цей етап розгляду підкреслює логічну організацію та оптимізацію програмного забезпечення, спрямовану на досягнення поставлених завдань.

5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Мета економічної частини

Метою даного розділу у магістерському дипломному проекті є економічне обгрунтування розробки макету для візуалізації роботи алгоритму, що базується на мові програмування Python та бібліотеці OpenCV. Здійснення цієї мети вимагає вирішення таких завдань:

1) Проведення сегментації ринку для пристрою.

2) Визначення конкурентоспроможності даної розробки.

3) Розрахунок трудомісткості виконання робіт.

4) Складання кошторисів витрат на розробку.

5) Розрахунок заробітної плати, визначення вартості об’єкта та розрахунок очікуваного прибутку від реалізації комплексу.

6) Розрахунок точки беззбитковості та побудова графіка беззбитковості.

5.2. Опис виробу

У контексті застосування бібліотеки OpenCV для обробки та збереження відеозображень, можна відзначити важливі можливості та переваги даного інструменту. OpenCV дозволяє ефективно виконувати операції обробки відео, включаючи аналіз та взаємодію з відеозображеннями.

5.2.1. Управління рухом в робототехніці

Використання OpenCV для визначення та відстеження об'єктів у відеопотоці. Аналіз руху об'єктів за допомогою бібліотеки для ефективного управління рухом роботів.

5.2.2. Розпізнавання образів

Застосування OpenCV для виявлення та розпізнавання об'єктів на відеозображеннях. Використання алгоритмів обробки зображень для точного визначення характеристик об'єктів.

5.2.3. Оптимізація алгоритмів

- Використання OpenCV для ускладнених операцій обробки відеоданих.
- Оптимізація алгоритмів для прискорення роботи при використанні сучасних обчислювальних ресурсів.

5.2.4. Практичне застосування

- Врахування тенденцій у розвитку OpenCV та його практичне впровадження в різноманітних областях, включаючи науку, промисловість та розваги.

- За допомогою бібліотеки OpenCV та її функціоналу можливо створити ефективні інструменти для обробки та збереження відеозображень з урахуванням сучасних вимог до швидкодії та точності.

5.3. Сегментування ринку

Сегментування ринку для розробки комплексу з обробки та збереження відеозображень за допомогою бібліотеки OpenCV визначається рядом факторів, які враховують особливості та потреби потенційних споживачів.

5.3.1. Визначення сегментів

- Виявлення ключових принципів та факторів сегментації для даного продукту на ринку.
- Визначення основних критеріїв, які впливають на сприйняття та використання продукту в різних групах споживачів.

5.3.2. Оцінка ринкової ємності

- Розрахунок кількості підприємств-споживачів у кожному сегменті.
- Визначення середньорічної програми виробів для кожного сегменту, що підлягає постачанню продукту.
- Використання формули для розрахунку повної ємності ринку.

5.3.3. Аналіз і вибір сегментів

- Аналіз інформації про сегменти, визначення їх привабливості та важливості для бізнесу.
- Вибір найбільш перспективних сегментів для подальшого аналізу та розробки.

5.3.4. Цільове позиціонування

- Визначення стратегії позиціонування продукту в обраному сегменті.
- Оптимізація маркетингових комунікацій та виведення продукту на ринок з урахуванням особливостей кожного сегменту.

Споживачами розробленого комплексу є університети, підприємства України та ближнього зарубіжжя, а також студенти та учні, зацікавлені в інноваційних розробках у галузі обробки та збереження відеозображень. Сегментування ринку дозволяє ефективно взаємодіяти з цільовими групами, забезпечуючи їм продукт, який відповідає їхнім унікальним потребам та очікуванням.

Розрахунок повної ємності ринку будемо проводити за формулою:

$$S_{\text{повн. }1} = N_1 \cdot Q_1 \cdot m_1 \quad (5.1)$$

де N_1 - кількість підприємств споживачів виробу в і-тому сегменті; Q_1 - середня річна програма виробів в і-тому сегменті, для яких буде поставлятися розглянутий товар; m_1 - кількість виробів, що йдуть в один виріб-споживач (1шт.).

5.4. Аналіз конкурентоспроможності

Товар вважається конкурентоспроможним, коли він відповідає вимогам обраного ринку за комерційними, технічними та економічними показниками, надаючи можливість успішного реалізації на цьому ринку. Конкурентоспроможність визначається характеристиками, що роблять даний товар вигідно вирізняються серед товарів-конкурентів.

Оцінка конкурентоспроможності розробленої системи передбачає аналіз характеристик системи порівняно з аналогічними виробами за рядом параметрів. Для цього використовується метод комплексних показників якості, що враховує узагальнені показники якості з використанням гіпотетичного варіанту.

Основними показниками якості системи є швидкість визначення об'єкту, точність визначення відстані та швидкість роботи програми. Ці характеристики безпосередньо впливають на ефективність всієї системи управління, визначаючи найкращі техніко-економічні параметри.

Визначимо абсолютне значення і-х показників j варіантів p_{ij} у балах. Показникам якості присвоюємо коефіцієнті вагомості b_i:

$$\sum_{i=1}^{n} b_i = 1 \text{ і } b_1 < 0, i = 1, \quad (5.2)$$

Визначені показники якості розподілені на дві категорії: мінімізовані та максимізовані, і використовуються для формування гіпотетичного (еталонного) варіанту. Для кожного варіанту j розраховані відносні значення індикаторів (k_ij), порівнюючи p_ij з p_i ГИП (з урахуванням умови k_ij≤1).

Для всіх розглянутих варіантів отримали узагальнені показники k_j^0. Тепер проводимо розрахунок рівнів якості нового комплексу порівняно з комплексами конкурентів.

$$Y_{j-b} = \frac{k_j^0}{k_k^0}, \quad (5.3)$$

де k_j^0 - узагальнений показник приладу-конкуренту.

5.5. Розрахунок собівартості та ціни виготовлення установки

Собівартість продукції складається з ряду найменувань витрат. Сюди входять: витрати на основні матеріали, на великі комплектуючі вироби, пряма і додаткова заробітна плата, витрати на утримання та експлуатацію обладнання, утримання транспорту, а також цілий ряд загальнодержавних податків і відрахувань.

Для проектування установки необхідна участь наступних робочих: розробник, керівник, інженер. Тривалість робочого місяця в середньому приймемо 22 дня.

Далі необхідно обчислити основну заробітну плату (далі - ОЗП), з урахуванням трудовитрат, кількості виконавців і середньоденна заробітна плата (далі - ЗП).

$$\text{ОЗП} = \sum N_i * \text{ЗП}_{\text{ср}}, \quad (5.4)$$

Де N_i- кількість днів, відпрацьоване і-ми виконавцями за стадіями;
$\text{ЗП}_{\text{ср}}$- денні оклади і-х виконавців.

Додаткова заробітня плата (ДЗП) розрахується за формулою:

$$\text{ДЗП} = \frac{20\% * \text{ОЗП}}{100\%} \quad (5.5)$$

Таблиця 6.1 - Склад виконавців роботи

Посада	Посадові оклади, грн.	
	За місяць	За день
Розробник	12000	545
Керівник	20000	909
Інженер	10000	454

Проведемо розрахунок трудомісткості робіт. Результати розрахунків наведені в таблиці 6.3

Таблиця 6.2 - Розрахунок трудомісткості робіт

	Тривалість	Трудомісткість	Виконавці		
			Керівник	Розробник	Інженер
1	2	3	4	5	6
Постановка задачі	1	1	+	-	-

Продовження таблиці 6.2

Розробка тз	1	1	-	+	-
Погодження та затвердження ТЗ	1	3	+	+	+
Закупка товарів	2	2	-	-	+
Виробничі роботи	2	2	-	-	+
Розробка алгоритмів	5	5	-	+	-
Розробка програми	5	5	-	-	+
Налагодження продукту	2	4	-	+	+
Випробування і здача продукції в експлуатацію	2	6	+	+	+
Разом	21	29	5	10	14

Обчислимо основну заробітну плату розробників установки, з урахуванням трудовитрат, кількості виконавців і середньоденний ЗП. Для цього кількість днів, відпрацьованих окремими виконавцями, множать на їх денні оклади:

$$ОЗП = 545*10 + 909*5+454* 14 = 16351 \text{ грн} \quad (5.6)$$

Додаткова заробітна плата обчислюється за формулою

$$ДЗП=20\%*ОЗП =16351*20\%=3270\text{грн}. \quad (5.7)$$

Розрахуємо вартість покупних елементів, необхідних для виготовлення установки. Перелік покупних елементів складається з урахуванням переліку блоків функціональної схеми (див. таблицю 6.4). Ціни вказані в гривнях.

Таблиця 6.3 – Перелік покупних елементів

№	Елементи	Кількість	Ціна, грн.	Вартість
1	Веб-камера	1	600,00	600,00
Загальна вартість				600

Відрахування в єдиний соціальний фонд складають 22% від основної заробітної плати і додаткової заробітної плати:

$$З_{відр} = \frac{ОЗП+ДЗП}{100} \cdot 22 = \frac{16351 + 3270}{100} \cdot 22 = 4316{,}62 \text{ грн.} \quad (5.8)$$

Таблиця 6.4 – Перелік обладнання

Найменування	Кількість	Ціна, грн
Стіл	1 шт	3000
Стілець	1 шт	4000
Комп'ютер	1 шт	22000
Інструменти	1 шт	1000
Сума		30000

Таблиця 6.5 - Розрахунок собівартості і ціни виробу за статтями

№	Статті	Сума, грн	Примітка
1	Основна заробітна плата (ОЗП)	16351	
2	Додаткова ЗП (ДЗП)	3270	20% від ОЗП
3	Єдиний соціальний фонд	4316.62	22%*(ОЗП+ДЗП)
4	Матеріали й куплені вироби ($С_м$)	600	Таблиця 6.4
5	Амортизація	537	$Ам = \frac{ОС * 0{,}25 * 21}{12 \cdot 22}$
6	Витрати на утримання обладнання	3000	10% від табл.6.5
7	Додаткові витрати	6540	40% від ОЗП

Продовження таблиці 6.5

8	Виробнича собівартість (С)	31000	п.1+п.2+п.3+…+п.7
9	Адміністративні витрати	7357	45% від ОЗП
10	Витрати на збут	520,5	2,5% від п.8
11	Собівартість власних робіт	29489	п.1+п.2+п.3+…+п.10
12	Прибуток (П)	5097	20% від п.11
13	Ціна без НДС	30586	П+п.11
14	НДС	8217	20% від ціни без ПДВ
15	Ціна з НДС	42230	п.13+п.14

Річна норма амортизаційних відрахувань (Ам) розраховується як 25% від вартості обладнання. Так, як трудомісткість становить 21 день, то амортизація обчислюється за формулою:

$$\text{Ам} = \frac{\text{ОС}*0{,}25*21}{12 \cdot 22} = \frac{27020 \cdot 0{,}25 \cdot 21}{12 \cdot 22} = 537 \text{ грн.} \qquad (5.9)$$

Таким чином, собівартість власних робіт проекту на розроблення лабораторного практикуму складає 29537,5 грн, а ціна проекту (з НДС) – 36730грн.

Кількість замовлених екземплярів має бути не менше ніж 140 штук. Виробничу собівартість одного екземпляра програмного продукту (ВС) Визначається за формулою:

$$\text{ВС}_0 = \frac{29537}{140} = 210.9 \qquad (5.10)$$

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:

$$\text{СП}_0 = \text{ВС}_0 + \text{АВ}_0 + \text{Вз}_0 \qquad (5.11)$$

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

$$АВ_0 = \frac{АВ}{КПП} = \frac{7357}{140} = 52.5 \text{ грн.} \quad (5.12)$$

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

$$ВЗ_0 = \frac{ВЗ}{КПП} = 6,4 \text{ грн.} \quad (5.13)$$

Таким чином, $СП_0$= 210.9+52.5+6,4= 269.9 грн.

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):

$$Р_п = \frac{Ц - ВС}{ВС} * 100\% = \frac{42230 - 29489}{29489} * 100\% = 45\% \quad (5.14)$$

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

$$ЦПП = СП * (1 + Р_п /100) = 269.9*(1+45/100) = 391.35 \quad (5.15)$$

Де $Р_п$ - коефіцієнт рентабельності.

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

$$ДР = ЦПП * КПП = 391.35 * 140 = 54789 \text{ грн.} \quad (5.16)$$

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

$$РКП = \frac{РП0СТВ}{(ЦПП - З_м В_0)} = \frac{15467.5}{313.49 - 95,5} = 71 \text{ шт.} \quad (5.17)$$

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

$$РП_{ост}В = ВОУ + А_м + ДВ + АВ + ВЗ = 4152 + 655.2 + 8810.4 + 8077 + 1268 = 22962 \text{ грн} \quad (5.18)$$

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

$$РЗ_м В = ВМ + ФОП + ЄСФОП = 1945 + (16195 + 3273) + 4152 = 25565 \text{ грн.} \quad (5.19)$$

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

продукту:

$$З_м В_0 = \frac{РЗ_м В}{КПП} = \frac{25565}{140} = 182.6 \text{ грн.} \quad (5.20)$$

Річний дохід в точці беззбитковості:

$$Д_{БЗ} = 316 * 259 = 81844 \text{грн.} \quad (5.21)$$

5.6 Висновки

В даному розділі був проведений розрахунок ціни і собівартості для системи обробки відеозображення. Ціна установки з НДС склала 42230. Розрахунок зроблений з урахуванням всіх необхідних трудовитрат, НДС склала 8217 гривень, відрахування в єдиний соціальний фонд – 4316 гривень.

Розроблена установка та система управління може скласти доволі високу конкуренцію на ринку збуту, так як собівартість виробництва і конструювання не дорожча аналогів на ринку.

ЗАКЛЮЧЕННЯ

Розроблений VideoPlayer є високофункціональним інструментом для відтворення та обробки відеофайлів та відеопотоків з вебкамери. Завдяки інтуїтивно зрозумілому графічному інтерфейсу, користувач може з легкістю вибирати, відтворювати та обробляти відеоматеріали.

Основні можливості програми включають в себе відтворення та зупинку відео, завантаження відеофайлів, а також запуск відеопотоку з вебкамери. Є можливість застосування різноманітних фільтрів та трансформацій, таких як чорно-білий, HSV, BGR, що дозволяє користувачеві експериментувати з візуальним виглядом відео.

Додатково, програма дозволяє використовувати кольорові фільтри LUV та LAB, що розширює мозаїку можливостей для обробки кадрів. Зручна опція збереження оброблених відеокадрів у форматі MP4 забезпечує користувачеві зручний інструмент для архівації та подальшого використання відеоматеріалів.

Загальний висновок полягає в тому, що розроблений VideoPlayer є ефективним та зручним засобом для взаємодії з відео та вебкамерою, надаючи широкий функціонал для відтворення, обробки та збереження відеоматеріалів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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).

12. 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.
13. 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.
14. 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.
15. https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/
16. https://docs.opencv.org/4.x/d4/da8/group__imgcodecs.html
17. https://docs.opencv.org/4.x/index.html
18. https://docs.opencv.org/4.x/d4/da8/group__imgcodecs.html

ДОДАТОК А
Лістинг коду програми

```
import cv2
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk

class VideoPlayer:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Video Player")
        self.root.geometry('1080x500')

        self.video_source = 0
        self.cap = cv2.VideoCapture(self.video_source)

        #self.frame_slider = tk.Scale(self.root, from_=0,
to=self.cap.get(cv2.CAP_PROP_FRAME_COUNT), orient=tk.HORIZONTAL,
command=self.update_frame)
        #self.frame_slider.pack(fill=tk.X, padx=10, pady=5)

        self.canvas = tk.Canvas(self.root, width=650, height=400)
        self.canvas.pack()

        self.btn_play = tk.Button(self.root, text="Play",
command=self.play_video)
        self.btn_play.place(x='10', y='10')

        self.btn_stop = tk.Button(self.root, text="Stop",
command=self.stop_video)
        self.btn_stop.place(x='70', y='10')

        self.btn_load = tk.Button(self.root, text="Load Video",
command=self.load_video)
        self.btn_load.place(x='22', y='40')

        self.btn_load_webcam = tk.Button(self.root, text="Load
Webcam", command=self.load_webcam)
        self.btn_load_webcam.place(x='13', y='100')

        self.gray_checkbox = tk.Checkbutton(self.root,
text="Gray", command=self.toggle_gray)
```

```
        self.gray_checkbox.place(x='905', y='10')
        self.is_gray = False

        self.hsv_checkbox = tk.Checkbutton(self.root, text='HSV',
command=self.toggle_HSV)
        self.hsv_checkbox.place(x='990', y='10')
        self.is_HSV = False

        self.lab_checkbox = tk.Checkbutton(self.root, text='LAB',
command=self.toggle_LAB)
        self.lab_checkbox.place(x='905', y='40')
        self.is_LAB=False

        self.bgr_checkbox = tk.Checkbutton(self.root, text='bgr',
command=self.toggle_bgr)
        self.bgr_checkbox.place(x='990', y='40')
        self.is_bgr = False

        self.mirror_checkbox = tk.Checkbutton(self.root,
text="Mirror", command=self.toggle_mirror)
        self.mirror_checkbox.place(x='955', y='75')
        self.is_mirrored = False

        self.btn_save = tk.Button(self.root, text="Save as MP4",
command=self.save_video)
        self.btn_save.place(x='18', y='70')

        self.is_playing = False
        self.update()
        self.root.mainloop()

    def play_video(self):
        self.is_playing = True

    def stop_video(self):
        self.is_playing = False

    def load_video(self):
        self.clear_canvas()
        file_path = filedialog.askopenfilename(filetypes=[("Video
files", "*.mp4;*.avi;*.mkv")])
        if file_path:
            self.set_video_source(file_path)
```

```
    def load_webcam(self):
        self.clear_canvas()
        self.set_video_source(0)
        self.play_video()

    def set_video_source(self, source):
        self.video_source = source
        if self.cap.isOpened():
            self.cap.release()
        self.cap = cv2.VideoCapture(self.video_source)

#self.frame_slider.config(to=self.cap.get(cv2.CAP_PROP_FRAME_COUNT
))

    def toggle_gray(self):
        self.is_gray = not self.is_gray

    def toggle_LAB(self):
        self.is_LAB=not self.is_LAB
    def toggle_HSV(self):
        self.is_HSV = not self.is_HSV

    def toggle_bgr(self):
        self.is_bgr = not self.is_bgr

    def toggle_mirror(self):
        self.is_mirrored = not self.is_mirrored

    def mirror_frame(self, frame):
        return cv2.flip(frame, 1)

    def update_frame(self, value):
        frame_position = int(value)
        self.cap.set(cv2.CAP_PROP_POS_FRAMES, frame_position)
        ret, frame = self.cap.read()
        if ret:
            frame = self.process_frame(frame)
            self.display_frame(frame)

    def update(self):
        if self.is_playing:
            ret, frame = self.cap.read()
```

```
            if ret:
                frame = self.process_frame(frame)
                self.display_frame(frame)
            else:
                self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        self.root.after(10, self.update)

    def process_frame(self, frame):
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        if self.is_mirrored:
            frame = self.mirror_frame(frame)
        if self.is_gray:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2RGB)
        elif self.is_HSV:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
            frame = cv2.cvtColor(frame, cv2.COLOR_HSV2RGB)
        elif self.is_LAB:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
            frame = cv2.cvtColor(frame, cv2.COLOR_LAB2RGB)
        elif self.is_bgr:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        frame = cv2.resize(frame, (650, 400))
        return frame

    def display_frame(self, frame):
        self.clear_canvas()
        self.photo =
ImageTk.PhotoImage(image=Image.fromarray(frame))
        self.canvas.create_image(0, 0, anchor=tk.NW,
image=self.photo)

    def save_video(self):
        file_path =
filedialog.asksaveasfilename(defaultextension=".mp4",
filetypes=[("MP4 files", "*.mp4")])
        if file_path:
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Choose
codec according to your video format
            out = cv2.VideoWriter(file_path, fourcc, 20.0, (650,
400))  # 20.0 - frames per second, (650, 400) - frame size
            self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
```

```
            while True:
                ret, frame = self.cap.read()
                if not ret:
                    break
                frame = self.process_frame(frame)
                out.write(frame)
            out.release()

    def clear_canvas(self):
        self.canvas.delete("all")

    def __del__(self):
        if self.cap.isOpened():
            self.cap.release()

if __name__ == "__main__":
    player = VideoPlayer()
```

ДОДАТОК В


Національний аерокосмічний університет ім. М. Є. Жуковського "ХАІ"
Кафедра Систем управління літальних апаратів
Презентація до дипломного проекту магістра
на тему:
Розробка алгоритмів читання та запису відео з використанням бібліотеки OpenCV
Здобувач: Содилєв Нікита Русланович Гр. 352
Спеціальність: 151 "Автоматизація та комп'ютерно-інтегровані технології"
Освітня програма: « Інженерія мобільних додатків »
Керівник: к.т.н., доц. каф.301 Краснов Л. О.
Дата захисту: 17 січня 2024 р.

Національний аерокосмічний університет ім. М. Є. Жуковського "ХАІ"
Кафедра Систем управління літальних апаратів

Мета роботи:

Покращити зчитування та запис відео в програмах через використання OpenCV, що дозволить забезпечити швидкість та ефективність операцій.

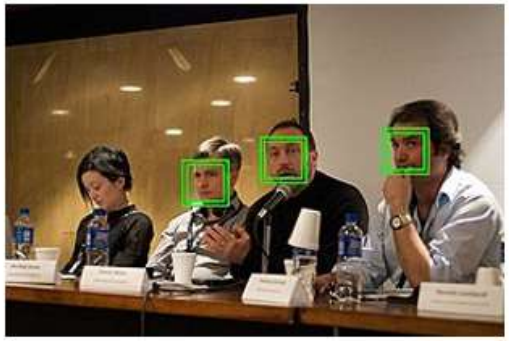
Апаратні та програмні методи для виконання завдання:

Зміст проведених робіт:

- Огляд літератури та методів вирішення задачі
- Вибір та обґрунтування методу читання та запису відео з використанням бібліотеки OpenCV
- Дослідження проблеми та синтез моделі
- Проектування системи за допомогою бібліотеки OpenCV
- Реалізація системи
- Розрахунок оцінки економічної ефективності проекту

OpenCV

OpenCV (Open Source Computer Vision) - це бібліотека комп'ютерного зору з відкритим вихідним кодом, яка надає інструменти для обробки зображень і відео. Вона підтримує різні завдання, такі як розпізнавання обличь, виявлення об'єктів, трасування руху та інші.



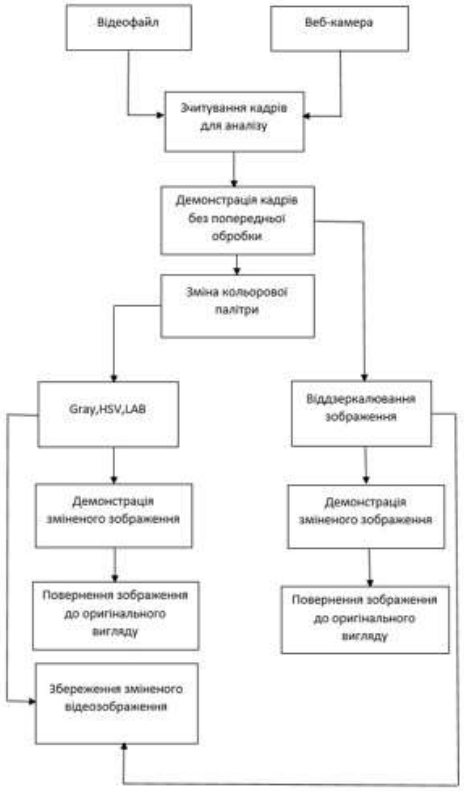
Зчитування та обробка зображень: OpenCV надає ефективні засоби для завантаження та обробки зображень, включаючи читання та запис різних форматів файлів, а також основні операції зображення, такі як зміна розміру, обрізка та фільтрація.

Відстеження об'єктів та руху: За допомогою OpenCV можна реалізувати системи відстеження об'єктів та руху, що є корисним для відображення рухомих об'єктів на відеозаписах або для створення систем відстеження.

Обробка відео та потокова обробка: OpenCV дозволяє легко обробляти відео та відеопотоки, використовуючи різноманітні фільтри та ефекти для покращення зображення.

Структура програми обробки та збереження відеофайлу

Структура алгоритму обробки та збереження відеофайлу



Програмна реалізація основних функцій роботи алгоритму

```
def save_video(self):
    file_path = filedialog.asksaveasfilename(defaultextension=
      ".mp4", filetypes=[("MP4 files", "*.mp4")])
    if file_path:
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out = cv2.VideoWriter(file_path, fourcc, 20.0, (650, 400))
        self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        while True:
            ret, frame = self.cap.read()
            if not ret:
                break
            frame = self.process_frame(frame)
            out.write(frame)
        out.release()
```

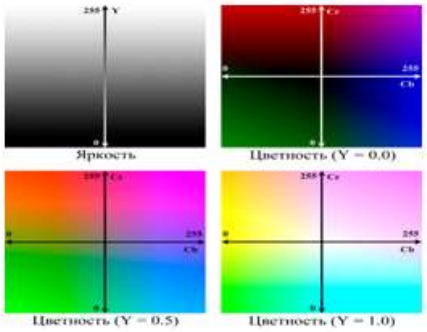
Результати роботи алгоритму

Результат зміни кольору та віддзеркалювання відеозображення з веб-камери

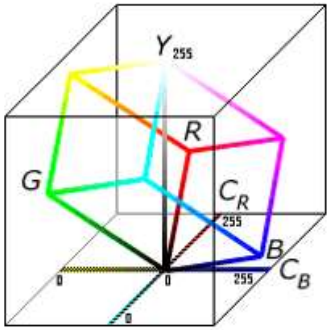
Результат зміни кольору та віддзеркалювання відеозображення з відеофайлу

Перетворення кольорових просторів

Кольоровий простір YCbCr



3D представлення колірного простору YCbCr



Основна формула перетворення відеозображення в кольоровому просторі

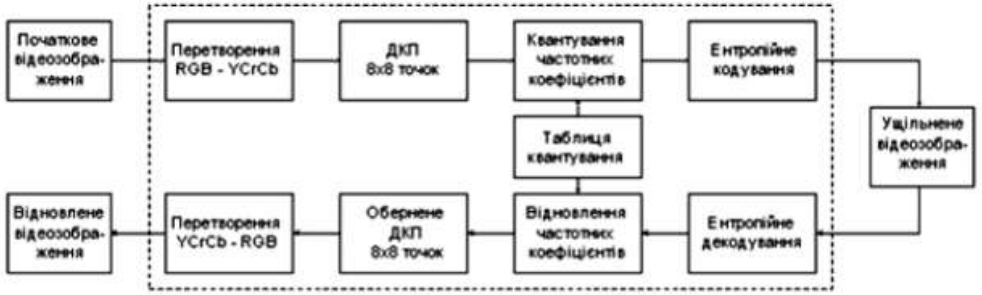
$$Y' = K_R \cdot R' + (1 - K_R - K_B) \cdot G' + K_B \cdot B'$$

$$P_B = \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B}$$

$$P_R = \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_R}$$

Дискретно-косинусне перетворення

Дискретне косинусне перетворення (ДКП або DCT) - це математичний метод, який використовується для перетворення сигналів або зображень з просторової області у частотну. DCT широко використовується в області обробки сигналів і зображень з численними застосуваннями.

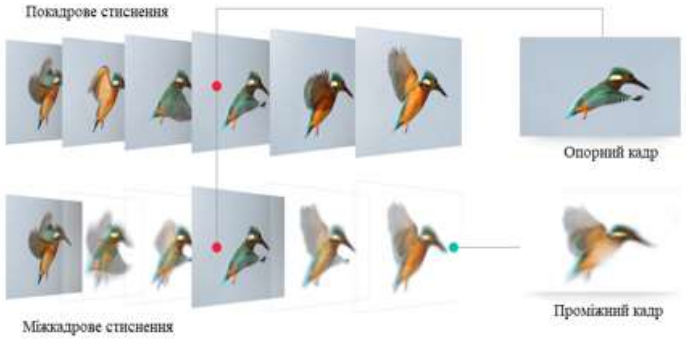


Методи стиснення зображень використовують двовимірне ДКП, яке задається формулою:

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y-0}^{n-1} p_{xy} \cos\left(\frac{(2y+1)j\pi}{2n}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right)$$

Покадрове стиснення

При покадровому стисканні кодеки працюють із кожним кадром окремо. Майже так само стискають звичайні JPEG-зображення. Спершу алгоритм ділить кадр на яскраву та колірну складові, потім знижує деталізацію та виділяє схожі ділянки. За мінімальних втрат якості цей спосіб зменшує розмір файлу в сотні разів.

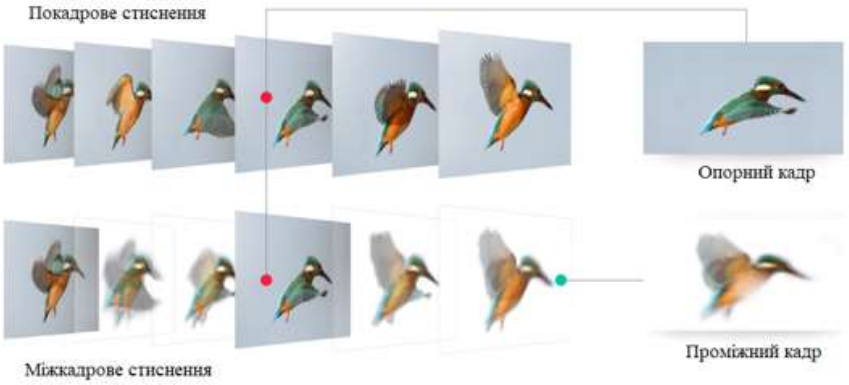


Простота реалізації: Покадрове стиснення є відносно простим методом, особливо порівняно з більш складними техніками стиснення, такими як міжкадрове стиснення. Це може полегшити розробку та впровадження алгоритмів стиснення.

Невеликі обчислювальні витрати при декодуванні: Декодування покадрового стиснення може бути менш витратним з точки зору обчислень, оскільки для відтворення кожного кадру не потрібно враховувати інші кадри.

Гнучкість у роботі з окремими кадрами: Кожен кадр стискається незалежно, що робить можливим легке вставляння, вилучення або редагування конкретних кадрів без впливу на інші частини відео.

Міжкадрове стиснення



Часто сусідні кадри мало відрізняються один від одного, тому їх необов'язково зберігати повністю. Метод міжкадрової різниці ґрунтується на порівнянні кадрів. У підсумковому файлі зберігаються лише відмінності кадрів

В міжкадровому стисненні повністю зберігаються лише опорні кадри, а проміжні добудовуються з них. Щоб побачити особливості міжкадрового стиснення, достатньо поставити на паузу будь-яке відео на момент динамічної сцени. Якщо не потрапити на опорний кадр, в області з об'єктами, що рухаються, зображення буде сильно розмитим. При звичайній швидкості відтворення людське око цього не помічає.

Економічне обґрунтування проекту

Склад виконавців роботи

Посада	Посадові оклади, грн.	
	За місяць	За день
Розробник	12000	545
Керівник	20000	909
Інженер	10000	454

Розрахунок трудомісткості робіт

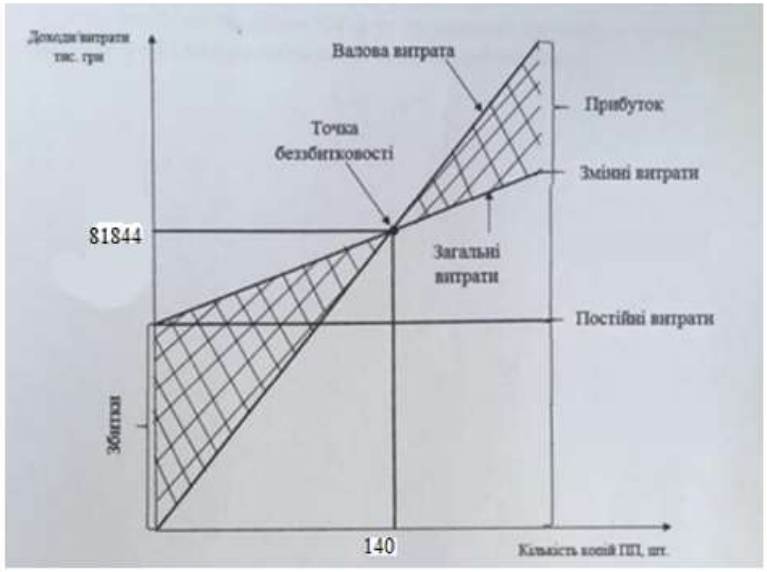
	Тривалість	Трудомісткість	Виконавці		
			Керівник	Розробник	Інженер
1	2	3	4	5	6
Постановка задачі	1	1	+	-	-
Розробка тз	1	1	-	+	-
Погодження та затвердження ТЗ	1	3	+	+	+
Закупка товарів	2	2	-	-	+
Виробничі роботи	2	2	-	-	+
Розробка алгоритмів	5	5	-	+	-
Розробка програми	5	5	-	-	+
Налагодження продукту	2	4	-	+	+
Випробування і здача продукції в експлуатацію	2	6	+	+	+
Разом	21	29	5	10	14

Перелік обладнання

Найменування	Кількість	Ціна, грн
Стіл	1 шт	3000
Стілець	1 шт	4000
Комп'ютер	1 шт	22000
Інструменти	1 шт	1000
Сума		30000

Основні економічні показники

Критична програма випуску продукту



Розрахунок собівартості і ціни виробу

№	Статті	Сума, грн	Примітка
1	Основна заробітна плата (ОЗП)	16351	
2	Додаткова ЗП (ДЗП)	3270	20% від ОЗП
3	Єдиний соціальний фонд	4316.62	22%*(ОЗП+ДЗП)
4	Матеріали й куплені вироби ($С_м$)	600	Таблиця 6.4
5	Амортизація	537	$Ам = \frac{ОС * 0,25 * 21}{12 \cdot 22}$
6	Витрати на утримання обладнання	3000	10% від табл.6.5
7	Додаткові витрати	6540	40% від ОЗП
8	Виробнича собівартість (С)	31000	п.1+п.2+п.3+...+п.7
9	Адміністративні витрати	7357	45% від ОЗП
10	Витрати на збут	520,5	2,5% від п.8
11	Собівартість власних робіт	29489	п.1+п.2+п.3+...+п.10
12	Прибуток (П)	5097	20% від п.11
13	Ціна без НДС	30586	П+п.11
14	НДС	8217	20% від ціни без ПДВ
15	Ціна з НДС	42230	п.13+п.14

Дякую за увагу!