МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальними апаратами

**Курсова робота**

з дисципліни: «Об’єктно-орієнтоване проектування СУ»

(назва дисципліни)

на тему: «Методи обробки відеозображень в системах управління з технічним зором»

Виконав: студент \_2\_ курсу групи № \_322\_

Напряму підготовки (спеціальності)

272 «Авіаційний транспорт»\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва напряму підготовки (спеціальності))

Марковский А. О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

Прийняв: канд. техн. наук, доц. на каф. 301

Гавриленко О. В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_\_\_

Оцінка: ECTS\_\_\_\_\_

Харків – 2024

Реферат

Листів 31, ілюстрацій 6, джерел 7, додатків 2.

Дана курсова робота присвячена дослідженню та аналізу можливостей обробки відеоданих з використанням бібліотеки OpenCV у середовищі програмування Python. Робота складається з декількох розділів, в яких розглядаються основні методи обробки відео, такі як виділення меж, фільтрація та реалізація класу для обробки відеоданих. Кожен розділ містить опис методів, їх реалізацію на Python та аналіз результатів їх застосування.

КЮЧОВІ СЛОВА: OpenCV, Python, обробка відеоданих, виділення меж, фільтрація, об'єктно-орієнтоване програмування.

# Зміст

[Зміст 3](#_Toc163081679)

[Перелік використаних позначень 4](#_Toc163081680)

[Вступ 5](#_Toc163081681)

[1 Огляд проблеми обробки зображень в системах управління з технічним зором 6](#_Toc163081682)

[2 Методи та засоби отримання відеоДАНИХ 8](#_Toc163081683)

[2.1 Завантаження відео з файлу 8](#_Toc163081684)

[2.2 Захват відео з веб-камери 8](#_Toc163081685)

[3 Геометричні перетворення відео-зображень 9](#_Toc163081686)

[3.1 Можливості бібліотеки OpenCV з геометричних перетворень 9](#_Toc163081687)

[3.2 Метод повороту для завантаженого відео з файлу. 9](#_Toc163081688)

[3.3 Реалізація на Python й аналіз результатів 10](#_Toc163081689)

[4 Кольорові перетворення відео-зображень 11](#_Toc163081690)

[4.1 Можливості бібліотеки OpenCV з кольорових перетворень 11](#_Toc163081691)

[4.2 Метод зміни кольорового простору на RGB для завантаженого відео з файлу. 11](#_Toc163081692)

[4.3 Реалізація на Python й аналіз результатів 12](#_Toc163081693)

[5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ 13](#_Toc163081694)

[5.1 Можливості бібліотеки OpenCV для виконання операцій 13](#_Toc163081695)

[5.2 Метод виділення меж з різними порогами Sobel для відео з файлу. 13](#_Toc163081696)

[5.3 Реалізація на Python й аналіз результатів 14](#_Toc163081697)

[6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ 15](#_Toc163081698)

[6.1 Можливості бібліотеки OpenCV з фільтрації 15](#_Toc163081699)

[6.2 Метод Фільтр НЧ з різними масками для відео 15](#_Toc163081700)

[6.3 Реалізація на Python й аналіз результатів 16](#_Toc163081701)

[7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ 18](#_Toc163081702)

[7.1 Поняття класу та його створення на Python 18](#_Toc163081703)

[7.2 Реалізація класу VideoProcessor і використання його в інтерфейсі користувача 18](#_Toc163081704)

[Висновки 20](#_Toc163081705)

[Перелік використаних джерел 21](#_Toc163081706)

[Додаток А 22](#_Toc163081707)

[Додаток Б 30](#_Toc163081708)

# Перелік використаних позначень

ООП – обʼєктно орієнтоване програмування;

НЧ – Напрямлене зниження частот;

СТВ – Системи управління з технічним зором;

# Вступ

Сучасні технології систем управління, які базуються на технічному зорі, відіграють ключову роль у багатьох галузях, від виробництва до медицини. Вони автоматизують процеси та оптимізують їх, аналізуючи величезний потік візуальної інформації. Одним з найважливіших аспектів таких систем є обробка відеозображень, яка дозволяє отримувати, аналізувати та використовувати дані для ефективного управління.

Технічний зір, відомий також як машинний зір, це галузь, що досліджує створення систем, які можуть сприймати та обробляти візуальну інформацію аналогічно людському зору. Вони застосовуються у різних сферах, таких як автомобільна промисловість, медицина, безпека та інші.

Обробка відеозображень - це складний процес аналізу та модифікації відеоданих для отримання корисної інформації. Вона включає в себе такі операції, як фільтрація, шумопониження, виявлення об'єктів, відстеження руху та інші.

Мета цього дослідження полягає у докладному вивченні різних методів обробки відеозображень в контексті систем управління з технічним зором. У подальших розділах роботи будуть детально розглянуті основні принципи технічного зору, різні методи обробки відеоданих та їх застосування в сучасних системах управління.

# 1 Огляд проблеми обробки зображень в системах управління з технічним зором

Системи управління з технічним зором (СТВ) стрімко набувають популярності в багатьох сферах, від автоматизації виробництва до медичної діагностики. Їх основою є обробка зображень, яка постійно розвивається завдяки досягненням в галузі інформаційних технологій та цифрових методів.

Висока точність та швидкість обробки зображень стають ключовими викликами при розробці СТВ. Це потребує створення ефективних алгоритмів, здатних швидко опрацьовувати великі обсяги даних без втрати якості.

Надійність систем також має виняткове значення, особливо в умовах мінливого середовища, яке може негативно впливати на якість зображень. В цьому випадку морфологічна обробка зображень може суттєво покращити їх якість та забезпечити точніший аналіз.

Впровадження штучного інтелекту, зокрема нейронних мереж, значно спрощує процес обробки зображень та робить його більш ефективним в різних сценаріях застосування. Наприклад, штучний інтелект може автоматично виявляти та класифікувати об'єкти на зображеннях, що робить його незамінним в медицині та системах безпеки.

Швидкий розвиток штучного інтелекту відкриває широкі можливості для подальшого вдосконалення СТВ, що дозволить створювати більш інтелектуальні та ефективні рішення для різних галузей, від промисловості до медицини.

Збільшення обсягів візуальної інформації, що генерується та збирається, робить актуальним питання ефективного зберігання та обробки цих даних. Здешевлення камер, комп'ютерів та інших компонентів робить СТВ більш доступними для широкого кола користувачів. СТВ знаходять нове застосування в таких сферах, як автономні транспортні засоби, віртуальна та доповнена реальність, робототехніка.

Всі ці фактори свідчать про те, що СТВ мають значний потенціал для розвитку та вдосконалення в найближчі роки. Це робить їх перспективною технологією, яка може суттєво вплинути на різні аспекти нашого життя.

Розвиток систем управління з технічним зором (СТВ) відбувається на фоні постійних викликів і можливостей. Одним із таких викликів є забезпечення приватності та безпеки оброблюваних зображень. З огляду на те, що деякі зображення можуть містити конфіденційну інформацію, необхідно розробляти ефективні методи анонімізації та захисту даних. Технології шифрування та анонімізації можуть бути використані для забезпечення конфіденційності зображень під час їх обробки та передачі.

Крім того, інтеграція систем управління з технічним зором з іншими технологіями, такими як інтернет речей (IoT), також відкриває нові можливості та виклики. За допомогою IoT, СТВ можуть отримувати дані з сенсорів та пристроїв у реальному часі, що поліпшує їхню здатність до аналізу та реагування на зміни в навколишньому середовищі. Проте це також ставить питання щодо захисту від кібератак та забезпечення безпеки підключених пристроїв.

Крім того, розвиток технологій аугментованої реальності (AR) та віртуальної реальності (VR) відкриває нові можливості для застосування СТВ в різних сценаріях, від навчання до розваг. Інтеграція обробки зображень з AR та VR може допомогти створити іммерсивні та відчутні досвіди для користувачів.

Загалом, розвиток СТВ вимагає поєднання інноваційних алгоритмів обробки зображень з врахуванням викликів у сферах безпеки, приватності та інтеграції з іншими технологіями, щоб максимально використовувати їх потенціал у різних галузях.

# 2 Методи та засоби отримання відеоДАНИХ

## 2.1 Завантаження відео з файлу

Використовуючи бібліотеку OpenCV у Python, можна працювати з відеофайлами. Для цього використовується клас cv2.VideoCapture, який дозволяє відкривати відеофайли. Щоб завантажити відео з файлу, потрібно передати шлях до цього файлу. Кожен кадр можна зчитати з відеопотоку за допомогою методу read(). Після зчитування кадру його можна відобразити у вікні, використовуючи imshow(). Щоб зупинити відображення відео та завершити програму, потрібно натиснути клавішу 'q'. При завершенні програми необхідно звільнити ресурси відеопотоку методом release() та закрити всі вікна OpenCV за допомогою destroyAllWindows().

## 2.2 Захват відео з веб-камери

Для захоплення відео з вебкамери використовується клас cv2.VideoCapture, в який передається параметр 0 (нуль). Кожен кадр можна зчитати з відеопотоку за допомогою методу read(). Після зчитування кадру його можна відобразити у вікні, використовуючи imshow(). При натисканні клавіші 'q' відображення відео припиняється, і програма завершується. На завершення програми необхідно звільнити ресурси відеопотоку методом release() та закрити всі вікна OpenCV за допомогою destroyAllWindows().

Лістинг програми, яка демонструє завантаження відео з файлу та з вебкамери (лістинг А.1).

Екранні форми виконання програми, яка завантажує відео з файлу та з вебкамери (рисунок Б.1).

# 3 Геометричні перетворення відео-зображень

## 3.1 Можливості бібліотеки OpenCV з геометричних перетворень

OpenCV - це потужна бібліотека комп'ютерного зору з відкритим вихідним кодом, яка відкриває широкі можливості для обробки зображень та відео. Завдяки своїм функціям, вона стала невід'ємною частиною багатьох проектів у сфері розробки програмного забезпечення, особливо в галузі розпізнавання облич, відстеження об'єктів та розпізнавання руху.

Популярні операції з геометричних перетворень, такі як зсув, масштабування, обертання та зміна перспективи, стають доступними завдяки OpenCV. Це дозволяє розробникам легко маніпулювати зображеннями для вирішення різноманітних завдань у сфері комп'ютерного зору та обробки зображень.

Бібліотека також надає зручний інтерфейс для виконання геометричних трансформацій як з окремими зображеннями, так і з потоковим відео. Це дозволяє вирішувати завдання, такі як виправлення спотворень в перспективі та роботу з об'єктами у складних умовах зйомки.

OpenCV відкриває безліч можливостей для реалізації інноваційних додатків, які використовують комп'ютерний зір для розв'язання різних завдань в області обробки зображень і відеоаналізу.

## 3.2 Метод повороту відео.

Одним з ключових методів обробки зображень є геометричне перетворення для повороту. Воно використовується для зміни орієнтації зображення шляхом обертання. Для цього потрібно вказати кут обертання та центр обертання. Математичні основи цього методу використовують матричні операції та тригонометричні функції. Обертання зображення на певний кут здійснюється за допомогою матриці обертання. Таким чином, застосовуючи цю матрицю до кожного пікселя на зображенні, отримуємо змінене зображення з відповідною орієнтацією.

Цей метод є дуже ефективним та широко використовується у комп'ютерному зорі та обробці зображень. Він знадобляється для різних завдань, таких як виправлення перспективи, аугментація даних чи обробка медичних зображень.

## 3.3 Реалізація на Python й аналіз результатів

Метод геометричного перетворення для повороту зображень використовує матрицю обертання для зміни положення пікселів. Основні параметри цього методу - кут обертання та центр обертання. Щоб обернути зображення на певний кут відносно центру обертання, використовується функція cv2.getRotationMatrix2D, яка приймає координати центру та кут обертання. Після цього застосовується функція cv2.warpAffine, яка здійснює обертання, передаючи на вхід зображення, матрицю обертання та розміри зображення.

У Python з використанням OpenCV можна реалізувати цей метод за допомогою функції rotate, яка приймає зображення та кут обертання. Функція викликає cv2.getRotationMatrix2D для обчислення матриці обертання та cv2.warpAffine для застосування обертання. Результатом є повернуте зображення.

У лістингу програми, що демонструє поворот відео на заданий кут, цей метод використовується для обертання кадрів відеопотоку на 30 градусів. Зчитуються кадри, застосовується обертання та відображення результату. Для виходу з програми потрібно натиснути клавішу 'q'.

Лістинг програми, яка демонструє поворот відео (лістинг А.2).

У вказаному коді реалізується обертання кадрів відеопотоку на 30 градусів за допомогою функції rotate. Для цього використовується цикл, який зчитує кадри з відеопотоку, застосовує до них обертання та відображає результат. Для виходу з програми необхідно натиснути клавішу q.

Екранні форми виконання програми, яка виконує поворот відео (рисунок Б.2).

# 4 Кольорові перетворення відео-зображень

## 4.1 Можливості бібліотеки OpenCV з кольорових перетворень

Кольорові перетворення - це ключова складова в обробці зображень, оскільки вони дозволяють змінювати представлення кольорів зображення, що може бути корисним для багатьох завдань, включаючи покращення якості зображень, виявлення об'єктів та аналіз даних.

Однією з основних можливостей OpenCV є здатність виконувати перетворення між різними кольоровими просторами. Кольорові простори - це спосіб представлення кольору в числовій формі, такі як RGB, HSV, Lab тощо. OpenCV надає функції для перетворення зображень між цими просторами, що відкриває широкі можливості для аналізу та обробки зображень.

Ще однією важливою можливістю є фільтрація та підсилення кольорів у зображеннях. OpenCV надає різноманітні фільтри, такі як фільтри морфологічної обробки, згладжування та розмиття, які можна використовувати для поліпшення якості зображень, а також для виділення або приховування певних деталей.

Крім того, OpenCV має можливості для виявлення та витягування кольорів у зображеннях. Це може бути корисним для різних завдань, таких як виявлення об'єктів на зображеннях або аналіз певних аспектів зображень на основі їхнього кольору.

Додатково, OpenCV дозволяє виконувати геометричні перетворення зображень, такі як зміна розміру, обрізка та обертання. Ці перетворення можуть бути важливими при обробці зображень з точки зору кольорів, оскільки вони можуть впливати на співвідношення кольорів та їхній вигляд.

Загалом, можливості бібліотеки OpenCV у кольорових перетвореннях вкрай широкі та різноманітні. Вони включають у себе здатність виконувати перетворення між кольоровими просторами, фільтрацію та підсилення кольорів, виявлення та витягування кольорів, а також геометричні перетворення зображень. Ці можливості роблять OpenCV потужним інструментом для роботи з кольоровими зображеннями в різних доменах, від комп'ютерного зору до обробки зображень у реальному часі.

4.2 Метод зміни кольорового простору на gray

Метод зміни кольорового простору на відео відображається через перетворення кольорів на відтінки сірого. Це часто використовується для спрощення аналізу відеоданих та візуалізації, оскільки кольорова інформація перетворюється на яскравість пікселів без втрати змісту.

Для реалізації цього методу з використанням бібліотеки OpenCV спочатку потрібно завантажити відео з файлу. Це можна зробити за допомогою відповідної функції OpenCV, яка приймає шлях до відеофайлу як вхідний аргумент. Після завантаження відео вказаної функцією ми отримуємо об'єкт відеопотоку, з яким можна взаємодіяти.

Наступним кроком є ітерація через кожен кадр відео. Для кожного кадру ми застосовуємо операцію перетворення кольорового простору на відтінки сірого. Це досягається за допомогою функції OpenCV, яка приймає кожен кадр як вхід та повертає його відображення у відтінках сірого. Цей новий кадр можна потім відобразити або зберегти.

Завершальним кроком є відображення або збереження зміненого відео. Це може бути здійснено шляхом використання функції OpenCV для відображення відеопотоку у вікні або для запису відеопотоку у новий відеофайл.

Отже, метод зміни кольорового простору на відтінки сірого для завантаженого відео з файлу включає завантаження відео, ітерацію через кожен кадр, застосування операції перетворення кольорового простору та відображення або збереження зміненого відео.

## 4.3 Реалізація на Python й аналіз результатів

Для реалізації зазвичай використовується функція cv2.cvtColor бібліотеки OpenCV для зміни кольорового простору кожного кадру. Параметри цієї функції включають вхідне зображення, початковий та кінцевий кольорові простори. Після зміни кольорового простору кадр відображається або зберігається для подальшого використання.

Лістинг програми, яка демонструє зміну кольорового простору на grey (лістинг А.3).

Екранні форми виконання програми, яка виконує зміну кольорового простору на grey (рисунок Б.3).

# 5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ

## 5.1 Можливості бібліотеки OpenCV для виконання операцій

OpenCV - це потужний інструмент для обробки зображень та відео, який надає широкі можливості для розв'язання різноманітних завдань у сфері комп'ютерного зору, обробки зображень, відеоаналізу та машинного навчання.

Серед основних можливостей OpenCV - виділення меж за допомогою алгоритму Canny, розмивання зображень різними методами, зміна розміру та обрізка зображень, фільтрація та видалення шуму.

Додатково, OpenCV дозволяє працювати з відео: читати та записувати відеопоток, відтворювати та відображати відео, а також виконувати операції над кадрами відео, наприклад, виділення меж або розмивання.

## 5.2 Метод детектування кутів Ши-Томасі з різними парметрами.

Метод детектування кутів Ші-Томасі є одним із ключових алгоритмів у комп'ютерному зорі та обробці зображень, використовуваним для виявлення особливих точок, які можуть служити орієнтирами для подальшого аналізу зображень. Цей метод базується на використанні мінімального власного числа матриці в грубці ковзної автокореляції, щоб визначити області з великою зміною яскравості.

У розділі буде розглянуто різні параметри методу детектування кутів Ші-Томасі та їх вплив на результати. Один з параметрів - розмір ядра фільтрації. Збільшення розміру ядра може призвести до зменшення чутливості методу до малих деталей, але збільшити точність виявлення більших кутів. Інший параметр - мінімальне значення власного числа. Це поріг, який використовується для відсіювання слабких кутів. Зменшення цього порогу може призвести до збільшення кількості виявлених кутів, включаючи менш значущі.

Далі розглядається вплив розміру вікна у матриці автокореляції. Збільшення розміру вікна може дозволити виявити кути на більш широких ділянках зображення, але це також може призвести до втрати точності та збільшення обчислювальних витрат. Іншим важливим параметром є коефіцієнт якості кутоутворення. Цей параметр визначає, наскільки кут повинен відрізнятися від своїх сусідів, щоб бути визнаним як кут. Зміна цього коефіцієнту може вплинути на точність та кількість виявлених кутів.

Нарешті, розділ також включає порівняльний аналіз різних комбінацій параметрів та їх вплив на ефективність методу детектування кутів Ші-Томасі. Цей аналіз може допомогти визначити оптимальні значення параметрів для конкретного завдання або типу зображень. Загалом, розділ надасть читачеві розуміння та інсайти щодо використання методу детектування кутів Ші-Томасі з різними параметрами у практичних застосуваннях комп'ютерної зорі та обробки зображень.

## 5.3 Реалізація на Python й аналіз результатів

У функції corner\_detector, призначеної для виявлення кутів на зображенні, спочатку створюється копія вихідного зображення. Далі, з огляду на те, що метод детектування кутів Ші-Томасі працює з відтінками сірого, вихідне зображення перетворюється у відтінки сірого за допомогою функції cv2.cvtColor бібліотеки OpenCV.

Після цього застосовується метод cv2.goodFeaturesToTrack, що дозволяє знаходити кути на зображенні. Цей метод приймає відтінки сірого зображення, максимальну кількість кутів для виявлення, якість кутів та мінімальну відстань між ними. В результаті отримуємо координати виявлених кутів у форматі Numpy масиву.

Далі проходить ітерація через отримані координати кутів, під час якої для кожного кута створюється коло з визначеним радіусом, що позначає місце виявлення кута на зображенні. Кола наносяться на копію вихідного зображення, щоб показати результат виявлення кутів.

Нарешті, функція повертає зображення з нанесеними на нього колами, які вказують на місця розташування кутів.

Після виконання функції, для аналізу результатів можна розглянути зображення, що повернула функція, з нанесеними на нього колами. Кількість кутів, які були виявлені, може бути контрольована зміною параметрів max\_corners, quality\_level та min\_dist. Значення цих параметрів можна експериментально підібрати для кожного конкретного випадку використання для отримання найкращих результатів.

Лістинг програми, яка демонструє виділення меж з різними порогами sobel (лістинг А.4).

Екранні форми виконання програми, яка виконує виділення меж з різними порогами sobel (рисунок Б.4).

# 6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ

## 6.1 Можливості бібліотеки OpenCV з фільтрації

OpenCV багатофункціональна бібліотека, яка надає широкі можливості для обробки зображень і відео. Вона включає в себе різноманітні типи фільтрів, такі як гаусівський, медіанний, та фільтр усереднення, які призначені для поліпшення якості та розмиття зображень, а також для видалення шуму. Крім цього, бібліотека має інші корисні фільтри, такі як білатеральний та адаптивний, які зберігають різкість країв зображення та автоматично адаптуються до характеристик зображення відповідно. Використання цих фільтрів може покращити якість зображень, знизити рівень шуму, та виправити зміни в контрасті та освітленості. Завдяки цим можливостям, OpenCV стає незамінним інструментом для обробки зображень та відео у різних сферах, таких як комп'ютерний зір, медичне обладнання, та аналіз зображень.

## 6.2 Метод фільтрації з ефектом зсуву

Метод фільтрації з ефектом зсуву є важливим інструментом в обробці зображень, який дозволяє створювати цікаві ефекти та покращувати якість зображень шляхом зміни їхнього положення. Цей метод полягає в застосуванні фільтра до зображення з наступним зсувом пікселів у певному напрямку. Він може бути використаний для різних цілей, включаючи створення мистецьких ефектів, виправлення зісковзування зображень та виявлення руху.

Під час застосування методу фільтрації з ефектом зсуву спочатку обирається фільтр, який буде використовуватися для обробки зображення. Це може бути, наприклад, гаусівський фільтр для розмиття або фільтр Собеля для виявлення країв. Після цього фільтр застосовується до кожного пікселя зображення.

Після застосування фільтра до кожного пікселя виконується зсув зображення у заданому напрямку. Цей зсув може бути горизонтальним, вертикальним або діагональним, в залежності від обраного ефекту та вимог завдання.

Після зсуву зображення може бути відобразжено, збережено або використано для подальшого аналізу. Результат може бути дуже різним в залежності від обраного фільтра, напрямку та величини зсуву.

Одним з прикладів використання методу фільтрації з ефектом зсуву є створення панорамних зображень. У цьому випадку кожне зображення зсувається в потрібному напрямку, щоб вирівняти їхні області перекриття перед їхнім об'єднанням.

Також метод фільтрації з ефектом зсуву може бути використаний для створення ефекту руху в анімаціях чи відео, додавання текстури до зображень або навіть для застосування ефекту панорамної камери в мобільних додатках.

## 6.3 Реалізація на Python й аналіз результатів

У функції motion\_blur, реалізованій на мові Python, виконується фільтрація з ефектом зсуву на зображенні. Цей метод застосовує рухливий розмив до зображення, що створює відчуття руху або швидкості відтворення.

Починаючи зі створення ядра для рухливого розмиву, функція спочатку ініціалізує ядро з нулів. Потім в центральному рядку ядра встановлюються значення 1, створюючи ефект "зсуву" пікселів у напрямку горизонталі. Розміри ядра можуть бути налаштовані за замовчуванням або передані як аргумент функції.

Далі виконується нормалізація ядра, розділяючи кожний елемент на суму всіх елементів у ядрі. Це забезпечує правильну інтенсивність пікселів на вихідному зображенні після фільтрації.

Застосування отриманого ядра до вхідного зображення виконується за допомогою функції cv2.filter2D бібліотеки OpenCV. Ця функція використовує зазначене ядро для виконання згортки зображення, що призводить до ефекту руху або розмивання вказаного напрямку.

Після застосування фільтру до зображення, функція повертає результат, який можна використовувати для подальшого аналізу або візуалізації.

Лістинг програми, яка демонструє фільтрації з ефектом зсуву (лістинг А.5).

Екранні форми виконання програми, яка виконує фільтрацію з ефектом зсуву (рисунок Б.5).

# 7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ

## 7.1 Поняття класу та його створення на Python

Клас є фундаментальною концепцією в об'єктно-орієнтованому програмуванні, яка дозволяє створювати абстракції та моделювати реальний світ у вигляді об'єктів. У мові програмування Python класи визначаються за допомогою ключового слова class, за яким слідує ім'я класу та блок, що містить атрибути і методи.

Атрибути класу представляють дані, що належать до класу, і можуть бути доступні через крапку, яка розділяє ім'я класу та ім'я атрибуту. Методи класу, які визначають поведінку об'єктів, також включаються у визначення класу. Перший аргумент методу, зазвичай self, вказує на конкретний об'єкт класу.

Конструктор класу, метод \_\_init\_\_, викликається при створенні нового об'єкта і використовується для ініціалізації атрибутів об'єкта. Це важливий момент, оскільки дозволяє задати початковий стан для нового об'єкта.

Інкапсуляція в об'єктно-орієнтованому програмуванні полягає у обмеженні доступу до атрибутів і методів класу ззовні. Це досягається шляхом використання атрибутів та методів з модифікаторами доступу, такими як private чи protected, щоб контролювати доступ до них.

Наслідування є ще однією важливою концепцією в об'єктно-орієнтованому програмуванні, яка дозволяє класу успадковувати атрибути та методи іншого класу. Це сприяє перевикористанню коду та створенню ієрархії класів, що спрощує розробку програм та забезпечує більшу гнучкість.

## 7.2 Реалізація класу Cv2Demo

Клас Cv2Demo є універсальним інструментом для роботи з відео та зображеннями з використанням бібліотеки OpenCV в мові програмування Python. Він надає можливість захоплення відео з різних джерел, обробки кадрів в реальному часі, а також застосування різноманітних фільтрів та операцій над зображеннями.

Метод capture\_video відповідає за захоплення відео з вказаного джерела. Він створює потік кадрів, який можна обробляти в реальному часі. Метод process\_video обробляє кожен кадр відео з використанням вказаної функції обробки, такої як рухливий розмив, виявлення кутів або зміна кольорового простору.

Клас також надає низку методів для реалізації різних операцій над зображеннями. Наприклад, метод motion\_blur застосовує рухливий розмив до зображення з вказаним розміром ядра, що дозволяє створювати ефект руху. Метод corner\_detector використовує алгоритм Ші-Томасі для виявлення кутів на зображенні. Інші методи, такі як colorspace\_change та rotate, виконують зміну кольорового простору та обертання зображення відповідно.

Метод menu надає простий текстовий інтерфейс для вибору різних опцій обробки відео. Користувач може вибрати захоплення відео з файлу або веб-камери, а також застосувати різні фільтри та операції до кадрів відео.

Цей клас дозволяє легко експериментувати з обробкою відео та зображень у реальному часі, надаючи широкі можливості для аналізу та обробки мультимедійних даних.

Лістинг класу Cv2Demo (лістинг А.6).

## Екранні форми використання класу Cv2Demo (рисунок Б.6).

# Висновки

OpenCV – це потужний інструмент для обробки зображень та відео, який надає розмаїття функцій, включаючи розмиття, виявлення країв, фільтрацію та інші. Ця бібліотека є незамінним інструментом для багатьох завдань у сфері комп'ютерного зору та обробки відео. Різні методи обробки зображень та відео, такі як виявлення країв за допомогою оператора Собеля або фільтрація для зменшення шуму, можуть бути використані для різних сценаріїв обробки відеоданих.

Створення класу Cv2Demo дозволяє дослідити можливості OpenCV та ефективно виконувати операції з відео, такі як відтворення, застосування фільтрів, виявлення країв тощо. Цей клас може автоматизувати обробку відеоданих у Python, роблячи процес зручним та ефективним. Він дозволяє використовувати потужні можливості OpenCV для аналізу та обробки відеоматеріалів.

Загалом, використання OpenCV та створення відповідних класів для обробки відеоданих є важливими кроками у сфері комп'ютерного зору та аналізу відео. Це може бути корисно у багатьох галузях, включаючи медицину, безпеку, мультимедіа та інші. Розуміння та використання таких інструментів допомагає зробити обробку та аналіз відеоданих більш ефективними та продуктивними.

# Перелік використаних джерел

1. Основы построения современных мобильных систем технического зрения [Текст]: учеб. пособие (часть 2). / Л. А. Краснов, К. Ю. Дергачев, С. В. Багинский – Х.: Нац. аэрокосм. ун-т им. Н. Е. Жуковского «Харьк. авиац. ин-т», 2018. – 92 с.
2. Электронный ресурс: http://docs.opencv.org – Документация по библиотеке OpenCV.
3. ­­­­­­­­­­­­­­­­­­Методи цифрової обробки зображень [Текст]: навч. посібник. /   
   О.А. Кобилін, І.С. Творошенко: ХНУРЕ, 2021. – 124 с.
4. Аналіз методів обробки відеозображень з вимірювальною інформацією, отриманих від тепловізора / спектральної камери [Текст]: навч. посібник. / Ю.О. Подчашинський, О.А. Лугових, Л.О. Чепюк: Державний університет «Житомирська політехніка», 2020. – 150 с.
5. Цифрова обробка зображення за допомогою штучного інтелекту [Текст]: стаття / Ю.О. Подчашинський, О.А. Лугових, Л.О. Чепюк: Державний університет «Житомирська політехніка», 2020. – 1 с.
6. Електроний ресурс: https://viso.ai/computer-vision/opencv/ – What is OpenCV? The Complete Guide (2024)
7. Електроний ресурс: https://metanit.com/python/tutorial/7.1.php – ООП класи і обʼєкти

# Додаток А

Лістинг коду

Лістинг А.1 – Зчитування відео з файлу та відео

# підключення необхідних бібліотек

import cv2

sources = {'video1': "Hourglass.mp4", "web": 0}

def main():

# зчитування даних з відеофайлу

cap\_video = cv2.VideoCapture(sources.get('video1'))

while cap\_video.isOpened():

# Запис фреймів

ret, frame = cap\_video.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

# Відображення результату

cv2.imshow('frame', frame)

if cv2.waitKey(25) == ord('q'):

break

# Завершуємо запис у кінці роботи

cap\_video.release()

# зчитування даних з відеокамери

cap\_web = cv2.VideoCapture(sources.get('web'))

# Перевірка готовності веб-камери

while cap\_web.isOpened():

# Запис фреймів

ret, frame = cap\_web.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

# Відображення результату

cv2.imshow('frame', frame)

if cv2.waitKey(25) == ord('q'):

break

cv2.destroyAllWindows()

# при запуску як головного файлу

if \_\_name\_\_ == '\_\_main\_\_':

main()

Лістинг А.2 – Поворот відео

import cv2

sources = {'video1': "Hourglass.mp4", "web": 0}

# Функція поворту відео

def rotate(video, angle):

num\_rows, num\_cols = video.shape[:2]

rotation\_matrix = cv2.getRotationMatrix2D((num\_cols / 2, num\_rows / 2), angle, 1)

img\_rotation = cv2.warpAffine(video, rotation\_matrix, (num\_cols, num\_rows))

return img\_rotation

def main():

cap = cv2.VideoCapture(sources.get('video1'))

# Перевірка готовності веб-камери

while cap.isOpened():

# Запис фреймів

ret, frame = cap.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

frame\_rotate = rotate(frame, 30)

# Відображення результату

cv2.imshow('frame', frame\_rotate)

if cv2.waitKey(25) == ord('q'):

break

# Завершуємо запис у кінці роботи

cap.release()

cv2.destroyAllWindows()

# при запуску як головного файлу

if \_\_name\_\_ == '\_\_main\_\_':

main()

Лістинг А.3 – Зміна кольорового простору на GREY

# підключення необхідних бібліотек

import cv2

sources = {'video1': "Hourglass.mp4", "web": 0}

colorspaces = {'Gray': cv2.COLOR\_BGR2GRAY, 'XYZ': cv2.COLOR\_BGR2XYZ, 'LAB': cv2.COLOR\_BGR2LAB,

'YUV': cv2.COLOR\_BGR2YUV, 'HSV': cv2.COLOR\_BGR2HSV}

def colorspace\_change(input\_frame):

return cv2.cvtColor(input\_frame, cv2.COLOR\_BGR2GRAY)

def main():

cap = cv2.VideoCapture(sources.get('video1'))

# Перевірка готовності веб-камери

while cap.isOpened():

# Запис фреймів

ret, frame = cap.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

# Зміна колірного простору зображення (фрейму)

frame\_gray = colorspace\_change(frame, 'Gray')

# Відображення результату

cv2.imshow('frame Grayscale', frame\_gray)

if cv2.waitKey(25) == ord('q'):

break

# Завершуємо запис у кінці роботи

cap.release()

cv2.destroyAllWindows()

# при запуску як головного файлу

if \_\_name\_\_ == '\_\_main\_\_':

main()

Лістинг А.4 – Детектування кутів Ши-Томасі з різними параметрами

import cv2

import numpy as np

sources = {'video1': "Hourglass.mp4", "web": 0}

# функція детектування кутів Ши-Томасі

def corner\_detector(image, max\_corners=5, quality\_level=0.01, min\_dist=20):

new\_image = image.copy()

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

corners = cv2.goodFeaturesToTrack(gray\_image, max\_corners, quality\_level, min\_dist)

corners = np.float32(corners)

for item in corners:

x, y = item[0]

cv2.circle(new\_image, (int(x), int(y)), 5, 255, -1)

return new\_image

def main():

cap = cv2.VideoCapture(sources.get('video1'))

# Перевірка готовності веб-камери

while cap.isOpened():

# Запис фреймів

ret, frame = cap.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

frame\_changed = corner\_detector(frame, max\_corners=20, quality\_level=0.01, min\_dist=50)

# Відображення результату

cv2.imshow('frame\_changed', frame\_changed)

if cv2.waitKey(25) == ord('q'):

break

# Завершуємо запис у кінці роботи

cap.release()

cv2.destroyAllWindows()

# при запуску як головного файлу

if \_\_name\_\_ == '\_\_main\_\_':

main()

Лістинг А.5 – Фільтр з ефектом зсуву

# підключення необхідних бібліотек

import cv2

import numpy as np

sources = {'video1': "Hourglass.mp4", "web": 0}

# Фільтрація з ефектом зсуву

def motion\_blur(image, kernel\_size=3):

kernel\_motion\_blur = np.zeros((kernel\_size, kernel\_size))

kernel\_motion\_blur[int((kernel\_size - 1) / 2), :] = np.ones(kernel\_size)

kernel\_motion\_blur = kernel\_motion\_blur / kernel\_size

return cv2.filter2D(image, -1, kernel\_motion\_blur)

def main():

cap = cv2.VideoCapture(sources.get('video1'))

# Перевірка готовності веб-камери

while cap.isOpened():

# Запис фреймів

ret, frame = cap.read()

# При виникненні помилці запису

if not ret:

print("Помилка запису фрейму!")

break

frame\_changed = motion\_blur(frame, kernel\_size=7)

# Відображення результату

cv2.imshow('frame', frame)

cv2.imshow('frame\_changed', frame\_changed)

if cv2.waitKey(25) == ord('q'):

break

# Завершуємо запис у кінці роботи

cap.release()

cv2.destroyAllWindows()

# при запуску як головного файлу

if \_\_name\_\_ == '\_\_main\_\_':

main()

Лістинг А.6 – Реалізація класу Сv2Demo

import cv2

import numpy as np

SOURCES = {'video1': "Hourglass.mp4", "web": 0}

class Cv2Demo:

@staticmethod

def capture\_video(source):

cap = cv2.VideoCapture(source)

while cap.isOpened():

ret, frame = cap.read()

if not ret:

print("Error reading frame!")

break

yield frame

if cv2.waitKey(25) == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

cv2.waitKey(1)

@staticmethod

def process\_frame(frame, process\_func, \*args, \*\*kwargs):

return process\_func(frame, \*args, \*\*kwargs)

@staticmethod

def motion\_blur(image, kernel\_size=3):

kernel\_motion\_blur = np.zeros((kernel\_size, kernel\_size))

kernel\_motion\_blur[int((kernel\_size - 1) / 2), :] = np.ones(kernel\_size)

kernel\_motion\_blur = kernel\_motion\_blur / kernel\_size

return cv2.filter2D(image, -1, kernel\_motion\_blur)

@staticmethod

def corner\_detector(image, max\_corners=5, quality\_level=0.01, min\_dist=20):

new\_image = image.copy()

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

corners = cv2.goodFeaturesToTrack(gray\_image, max\_corners, quality\_level, min\_dist)

corners = np.float32(corners)

for item in corners:

x, y = item[0]

cv2.circle(new\_image, (int(x), int(y)), 5, 255, -1)

return new\_image

@staticmethod

def colorspace\_change(input\_frame):

return cv2.cvtColor(input\_frame, cv2.COLOR\_BGR2GRAY)

@staticmethod

def rotate(video, angle):

num\_rows, num\_cols = video.shape[:2]

rotation\_matrix = cv2.getRotationMatrix2D((num\_cols / 2, num\_rows / 2), angle, 1)

img\_rotation = cv2.warpAffine(video, rotation\_matrix, (num\_cols, num\_rows))

return img\_rotation

@classmethod

def menu(cls):

print("\nChoose an option:")

print("1. Capture Video")

print("2. Capture Webcam")

print("3. Apply Filter")

print("4. Detect Corners")

print("5. Convert to Grayscale")

print("6. Rotate Video")

print("7. Exit")

choice = input("Enter your choice (1-7): ")

if choice == '1':

for frame in cls.capture\_video(SOURCES.get('video1')):

cv2.imshow('frame', frame)

elif choice == '2':

for frame in cls.capture\_video(SOURCES.get('web')):

cv2.imshow('frame', frame)

elif choice == '3':

cls.process\_video(SOURCES.get('video1'), cls.motion\_blur, kernel\_size=7)

elif choice == '4':

cls.process\_video(SOURCES.get('video1'), cls.corner\_detector, max\_corners=20, quality\_level=0.01, min\_dist=50)

elif choice == '5':

cls.process\_video(SOURCES.get('video1'), cls.colorspace\_change)

elif choice == '6':

cls.process\_video(SOURCES.get('video1'), cls.rotate, angle=30)

cv2.destroyAllWindows()

@classmethod

def process\_video(cls, source, process\_func, \*args, \*\*kwargs):

cap = cv2.VideoCapture(source)

while cap.isOpened():

ret, frame = cap.read()

if not ret:

print("Error reading frame!")

break

processed\_frame = cls.process\_frame(frame, process\_func, \*args, \*\*kwargs)

cv2.imshow('frame', frame)

cv2.imshow('processed\_frame', processed\_frame)

if cv2.waitKey(25) == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

cv2.waitKey(1)

if \_\_name\_\_ == "\_\_main\_\_":

Cv2Demo.menu()

# Додаток Б

Екранні форми виконання

A screenshot of a computer

Description automatically generatedA person taking a selfie

Description automatically generated

Рисунок Б.1 – Зчитування відео з файлу і вебкамери

A screenshot of a computer

Description automatically generated

Рисунок Б.2 – Поворот відео

A screenshot of a computer

Description automatically generated

Рисунок Б.3 – Зміна кольорового простору на grey

A screenshot of a computer

Description automatically generated

Рисунок Б.4 – Детектування кутів Ши-Томасі з різними параметрами

A screenshot of a computer

Description automatically generated

Рисунок Б.5 – Фільтр з ефектом зсуву

A screenshot of a computer

Description automatically generated

Рисунок Б.6 – Клас Сv2Demo