РЕФЕРАТ

Листів 32, ілюстрацій 8, джерел 8, 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
Перелік використаних позначень 4
Вступ 5
1 Огляд проблеми обробки зображень в системах управління з технічним зором 6
2 Методи та засоби отримання відеоДАНИХ 8
2.1 Завантаження відео з файлу 8
2.2 Захват відео з веб-камери 8
3 Геометричні перетворення відео-зображень 9
3.1 Можливості бібліотеки OpenCV з геометричних перетворень 9
3.2 Метод повороту для завантаженого відео з файлу. 9
3.3 Реалізація на Python й аналіз результатів 10
4 Кольорові перетворення відео-зображень 11
4.1 Можливості бібліотеки OpenCV з кольорових перетворень 11
4.2 Метод зміни кольорового простору на RGB для завантаженого відео з файлу. 11
4.3 Реалізація на Python й аналіз результатів 12
5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ 13
5.1 Можливості бібліотеки OpenCV для виконання операцій 13
5.2 Метод виділення меж з різними порогами Sobel для відео з файлу. 13
5.3 Реалізація на Python й аналіз результатів 14
6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ 15
6.1 Можливості бібліотеки OpenCV з фільтрації 15
6.2 Метод Фільтр НЧ з різними масками для відео 15
6.3 Реалізація на Python й аналіз результатів 16
7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ 18
7.1 Поняття класу та його створення на Python 18
7.2 Реалізація класу VideoProcessor і використання його в інтерфейсі користувача 18
Висновки 20
Перелік використаних джерел 21
Додаток А 22
Додаток Б 30

ПЕРЕЛІК ВИКОРИСТАНИХ ПОЗНАЧЕНЬ

ООП – об’єктно орієнтоване програмування;
НЧ – Напрямлене зниження частот;

ВСТУП

Системи управління з технічним зором мають важливе значення в сучасному технологічному середовищі, охоплюючи широкий спектр галузей від промислового виробництва до медичної діагностики. Вони забезпечують автоматизацію та оптимізацію процесів за допомогою аналізу великої кількості візуальної інформації. Одним із невід'ємних компонентів таких систем є обробка відеозображень, яка дає можливість отримувати, аналізувати та використовувати дані для прийняття рішень та ефективного керування.

Технічний зір, часто відомий як машинний зір, є областю науки та технології, що досліджує розробку систем, здатних сприймати та обробляти візуальну інформацію на кшталт людського зору. Застосування таких систем широко поширюється у різних галузях, включаючи автомобільну промисловість, медицину, безпеку, та інші.

Обробка відеозображень – це складний процес аналізу та модифікації відеоданих з метою отримання корисної інформації. Вона охоплює різноманітні операції, такі як фільтрація, видалення шуму, виявлення об'єктів, відстеження руху та багато інших.

Метою цього дослідження є ретельне вивчення різних методів обробки відеозображень у контексті систем управління з технічним зором. У подальших розділах роботи будуть докладно розглянуті основні принципи технічного зору, різні методи обробки відеоданих, а також їхнє застосування в сучасних системах управління.

1 ОГЛЯД ПРОБЛЕМИ ОБРОБКИ ЗОБРАЖЕНЬ В СИСТЕМАХ УПРАВЛІННЯ З ТЕХНІЧНИМ ЗОРОМ

Обробка зображень є ключовою складовою систем управління з технічним зором, які використовуються в широкому спектрі застосувань, від промислового автоматизування до медичної діагностики. Розвиток інформаційних технологій сприяє активному розвитку цифрових методів обробки відеозображень, що включає в себе збільшення обсягів доступних даних та покращення алгоритмів обробки.

Однією з основних проблем, яка стоїть перед розробниками систем технічного зору, є забезпечення високої точності та швидкості обробки зображень. Це вимагає розробки ефективних алгоритмів, які можуть швидко обробляти великі обсяги даних без втрати якості зображення. Методи цифрової обробки зображень включають поелементні перетворення, лінійне та зональне контрастування, еквалізацію гістограм та геометричну корекцію. [3]

Іншою важливою проблемою є забезпечення надійності систем управління з технічним зором у різних умовах експлуатації. Системи повинні бути стійкими до змін освітлення, коливань температури та інших зовнішніх факторів, які можуть впливати на якість зображення. Морфологічна обробка зображень, яка включає в себе операції ерозії та диляції, допомагає покращити якість зображень, що обробляються, та забезпечити більш точну інтерпретацію даних. [4]

Завданням дослідників у цій галузі є не лише розробка нових методів обробки, але й оптимізація існуючих підходів, щоб забезпечити їх ефективність у реальних умовах. Це включає в себе використання штучного інтелекту та машинного навчання для автоматизації процесів обробки та аналізу зображень, що відкриває нові можливості для покращення систем управління з технічним зором.

Продовжуючи обговорення проблем обробки зображень у системах технічного зору, важливо відзначити, що сучасні дослідження в цій галузі активно використовують потенціал штучного інтелекту (ШІ). ШІ може значно покращити якість та швидкість обробки зображень завдяки своїй здатності до глибокого навчання та аналізу великих даних. Наприклад, нейронні мережі здатні виявляти складні взаємозв’язки в даних, що дозволяє їм розпізнавати об’єкти та властивості на зображеннях з високою точністю. [5]

Одним із напрямків застосування ШІ в обробці зображень є розробка алгоритмів для автоматичного виявлення та класифікації об’єктів. Це особливо актуально для медичної діагностики, де ШІ може допомогти в розпізнаванні патологій на медичних знімках, забезпечуючи більш швидке та точне діагностування. Також ШІ використовується в системах безпеки для аналізу відео з камер спостереження, де він може виявляти підозрілу поведінку або ідентифікувати осіб. [5]

Ще однією важливою областю є використання ШІ для оптимізації процесів обробки зображень в реальному часі, що має велике значення для автоматизованого виробництва та робототехніки. ШІ може аналізувати зображення з високою швидкістю, виявляючи дефекти виробів на виробничих лініях або навіть керуючи роботами в складних умовах. [5]

Враховуючи швидкий розвиток технологій ШІ, можна очікувати, що в майбутньому його роль у системах технічного зору стане ще більш значущою. Це відкриває нові перспективи для розробки більш ефективних та інтелектуальних систем, здатних адаптуватися до змінних умов та вимог.

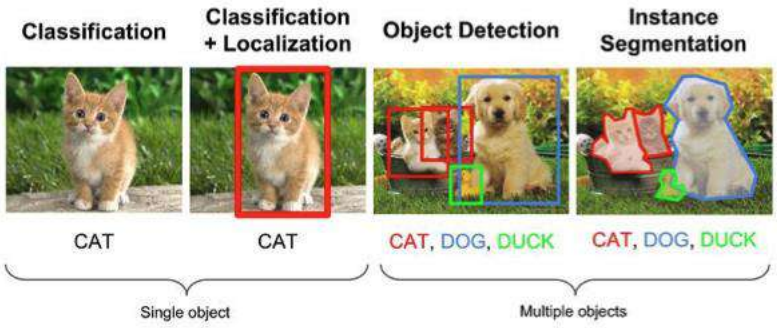

Рисунок 1.1 – Розпізнавання об’єктів на фото штучним інтелектом

2 МЕТОДИ ТА ЗАСОБИ ОТРИМАННЯ ВІДЕОДАНИХ

2.1 Завантаження відео з файлу

Для роботи з відеофайлами у Python використовується бібліотека OpenCV. Відеофайл відкривається за допомогою класу cv2.VideoCapture(video_src : str). Для завантаження відео с файлу необхідно передати шлях до цього файлу. Кожен кадр зчитується з відеопотоку за допомогою методу cap.read(). Після зчитування кадру він відображається у вікні за допомогою cv2.imshow(). При натисканні клавіші q відображення відео припиняється, і програма завершується. На завершення програми звільняються ресурси відеопотоку за допомогою cap.release() та закриваються всі вікна OpenCV за допомогою cv2.destroyAllWindows().

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

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

2.2 Захват відео з веб-камери

Захват відео з відеокамери виконується за допомогою класу в який передається 0 в якості параметра cv2.VideoCapture(0). Кожен кадр зчитується з відеопотоку за допомогою методу cap.read(). Після зчитування кадру він відображається у вікні за допомогою cv2.imshow(). При натисканні клавіші q'відображення відео припиняється, і програма завершується. На завершення програми звільняються ресурси відеопотоку за допомогою cap.release() та закриваються всі вікна OpenCV за допомогою cv2.destroyAllWindows().

Лістинг програми, яка демонструє завантаження відео з веб камери (лістинг А.2).

Екранні форми виконання програми, яка захоплює відео з веб камери (рисунок Б.2).

3 ГЕОМЕТРИЧНІ ПЕРЕТВОРЕННЯ ВІДЕО-ЗОБРАЖЕНЬ

3.1 Можливості бібліотеки OpenCV з геометричних перетворень

OpenCV - це бібліотека комп'ютерного зору з відкритим вихідним кодом, яка надає широкий спектр можливостей з обробки зображень та відео. У більшості випадків OpenCV використовується для вирішення завдань обробки зображень, таких як виявлення об'єктів, визначення контурів, аналіз кольорів і текстури, а також вирішення задач розпізнавання облич, відстеження об'єктів і розпізнавання руху. [6]

Що стосується геометричних перетворень, OpenCV надає можливості для зміни розміру зображень, обертання, зсуву, зміни перспективи та інших трансформацій. Зокрема, вона включає функціонал для перетворення координат точок, зміни розміру та форми об'єктів, а також вирівнювання зображень.

Популярними операціями з геометричних перетворень є зсув, масштабування, обертання та зміна перспективи. За допомогою OpenCV можна виконувати ці операції як з окремими зображеннями, так і з потоковим відео. Це корисно для таких завдань, як виправлення спотворень в перспективі, розпізнавання об'єктів у складних умовах зйомки та вирішення багатьох інших завдань у сфері комп'ютерного зору.

Загалом, завдяки можливостям OpenCV з геометричних перетворень, розробники можуть легко маніпулювати зображеннями для вирішення різноманітних завдань комп'ютерного зору та обробки зображень. [2]

3.2 Метод повороту для завантаженого відео з файлу.

Метод геометричного перетворення для повороту зображень є одним з основних методів обробки зображень та використовується для зміни орієнтації зображення шляхом обертання. Його параметрами є кут обертання та центр обертання.

Основним математичним апаратом, що лежить в основі цього методу, є матричні операції та тригонометричні функції. Для обертання зображення на певний кут використовується матриця обертання.

Отже, для виконання повороту зображення необхідно обчислити матрицю обертання та застосувати її до кожного пікселя на зображенні. Після цього отримуємо змінене зображення з відповідною орієнтацією.

Цей метод є дуже ефективним та широко використовується в області комп'ютерного зору та обробки зображень для вирішення різноманітних завдань, таких як виправлення перспективи, аугментація даних, обробка медичних зображень тощо.

3.3 Реалізація на Python й аналіз результатів

Метод геометричного перетворення для повороту зображень полягає у використанні матриці обертання для зміни положення пікселів на зображенні. Параметрами цього методу є кут обертання та центр обертання.

При обертанні зображення на кут angle відносно центру обертання, використовується матриця обертання. Ця матриця обчислюється функцією cv2.getRotationMatrix2D, яка приймає координати центру обертання та кут обертання. Потім за допомогою функції `cv2.warpAffine` здійснюється саме обертання, де на вхід передаються зображення, матриця обертання та розміри зображення.

Для реалізації цього методу у Python за допомогою OpenCV створена функція rotate, яка приймає зображення та кут обертання. Ця функція викликає функцію cv2.getRotationMatrix2D для обчислення матриці обертання та функцію cv2.warpAffine для застосування обертання до зображення. Результатом роботи функції є обернене зображення.

Лістинг програми, яка демонструє поворот відео на заданий кут (лістинг А.3).

У вказаному коді реалізується обертання кадрів відеопотоку на 90 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 на заданий кут (рисунок Б.3).

4 КОЛЬОРОВІ ПЕРЕТВОРЕННЯ ВІДЕО-ЗОБРАЖЕНЬ

4.1 Можливості бібліотеки OpenCV з кольорових перетворень

Бібліотека OpenCV надає широкі можливості з кольорових перетворень, що дозволяють здійснювати різноманітні операції з кольорами на зображеннях та відео. Основні можливості включають зміну кольорових просторів, корекцію кольорів, фільтрацію за кольором, а також виявлення та видалення окремих кольорів.

Однією з ключових можливостей OpenCV є зміна кольорових просторів. Бібліотека підтримує широкий спектр кольорових просторів, таких як RGB, BGR, HSV, HLS, LAB тощо. Це дозволяє здійснювати перетворення між різними просторами кольорів, що корисно для різноманітних завдань обробки зображень та аналізу кольорів. [6]

Крім того, OpenCV надає інструменти для корекції кольорів, такі як збільшення чи зменшення насиченості, зміна яскравості, контрастності та інтенсивності кольорів. Ці операції дозволяють покращити якість зображення та забезпечити більш точне відтворення кольорів.

За допомогою OpenCV також можна застосовувати фільтри за кольором, такі як фільтри для виявлення об'єктів певного кольору або для видалення шуму у певних кольорах. Ці фільтри можуть бути ефективними в різних застосуваннях, включаючи виявлення об'єктів, сегментацію зображень та видалення непотрібних елементів.

Загалом, завдяки розширеним можливостям кольорових перетворень, OpenCV є потужним інструментом для роботи з кольорами на зображеннях та відео. Вона дозволяє виконувати різноманітні операції з кольорами, що робить її незамінним інструментом у сфері комп'ютерного зору, обробки зображень та машинного навчання. [2]

4.2 Метод зміни кольорового простору на RGB для завантаженого відео з файлу.

Метод зміни кольорового простору на RGB для завантаженого відео з файлу передбачає перетворення кожного кадру відеопотоку з одного кольорового простору в RGB. Це може бути корисно для аналізу, обробки або відтворення

відео в стандартному кольоровому просторі RGB, який є найбільш поширеним і зрозумілим для більшості програм та пристроїв.

Математичний апарат, що лежить в основі цього методу, полягає у використанні матриці перетворення кольорових просторів. Кожен кольоровий простір може бути представлений у вигляді лінійної комбінації основних кольорів, а зміна простору зазвичай включає в себе обчислення нових значень кольорів на основі старих значень.

У випадку зміни кольорового простору на RGB, перетворення здійснюється шляхом множення кожного пікселя кадру на відповідну матрицю перетворення. Ця матриця розраховується з урахуванням особливостей кольорового простору, з якого виконується перетворення, та кольорового простору RGB.

Отже, метод зміни кольорового простору на RGB для завантаженого відео з файлу полягає у перетворенні кожного кадру відео з початкового кольорового простору в RGB за допомогою відповідної матриці перетворення. Це дозволяє працювати з відео в стандартному форматі RGB, що спрощує подальшу обробку та аналіз.

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 виділення меж з різними порогами sobel (лістинг А.4).

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

5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ

5.1 Можливості бібліотеки OpenCV для виконання операцій

Бібліотека OpenCV надає різноманітні можливості для обробки зображень та відео. Ці можливості включають в себе виділення меж за допомогою алгоритму Canny, розмивання зображень різними методами (гаусівське розмиття, медіанне розмиття, розмиття за допомогою фільтра усереднення), розтягування та стискання зображень, зміну розміру зображень, обрізку зображень, фільтрацію зображень за різними критеріями, а також виявлення та видалення шуму на зображеннях. [6]

Додатково, OpenCV надає можливості для обробки відео, такі як читання та запис відеопотоку, відтворення та відображення відео, виконання операцій над кадрами відео (наприклад, виділення меж, розмивання), а також аналіз руху на відео.

Ці можливості роблять OpenCV потужним інструментом для розв'язання різноманітних завдань у сфері комп'ютерного зору, обробки зображень, відеоаналізу та машинного навчання. Використовуючи ці функції, можна створювати складні та ефективні алгоритми обробки зображень та відео з мінімальними зусиллями. [2]

5.2 Метод виділення меж з різними порогами Sobel для відео з файлу.

Метод виділення меж з різними порогами Sobel використовується для виявлення різкості змін в інтенсивності пікселів на зображенні, що може вказувати на наявність границь або контурів об'єктів. Для цього використовується фільтр Собеля, який є лінійним фільтром, застосованим до зображення для обчислення градієнту інтенсивності пікселів у напрямку x та y.

Метод Собеля використовує дві матриці ядра, які називаються ядрами Собеля, для обчислення градієнтів відповідно до напрямків x та y. При застосуванні цих ядер до зображення виконується згортка, що призводить до отримання відповідних градієнтів. Після цього використовуються отримані градієнти для обчислення градієнту з обох напрямків та оцінки загальної інтенсивності границь.

Застосування методу Собеля для виділення меж вимагає визначення порогових значень, які визначають, які пікселі будуть вважатися границями. Це дозволяє регулювати чутливість алгоритму та вибирати найвідповідніші границі зображення в залежності від вимог застосування.

Отже, метод виділення меж з різними порогами Sobel є ефективним інструментом для виявлення границь на зображеннях та відео, що забезпечує можливість регулювання чутливості та вибору найкращих параметрів порогів для конкретного завдання обробки зображення.

5.3 Реалізація на Python й аналіз результатів

Метод використання оператора Собеля для виявлення країв відображається у функції sobel_edge_detection. Ця функція приймає зображення та розмір ядра для оператора Собеля. Спочатку зображення перетворюється у відтінки сірого для спрощення обробки. Потім застосовуються два фільтри Собеля: один для визначення горизонтальних границь (по горизонталі) і другий для вертикальних границь (по вертикалі). Результати обох фільтрів комбінуються для отримання кінцевого зображення, яке містить виявлені краї.

У вказаному коді відеопотік зчитується з файлу, і для кожного кадру використовується метод sobel_edge_detection для виявлення країв. Результати цього операції відображаються на екрані у реальному часі. Параметр k_size визначає розмір ядра фільтра Собеля, що впливає на чутливість виявлення країв.

Отримані результати виявлення країв можуть варіюватися залежно від значення параметрів та властивостей вхідного відеопотоку. Зміна розміру ядра фільтра Собеля може впливати на чутливість та розмір виявлених країв. Результати виявлення країв можуть використовуватися для подальшого аналізу зображень, такого як сегментація об'єктів, виявлення контурів або для створення візуальних ефектів.

Лістинг програми, яка демонструє виділення меж з різними порогами sobel (лістинг А.5).

Екранні форми виконання програми, яка виконує виділення меж з різними порогами sobel (рисунок Б.5).

6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ

6.1 Можливості бібліотеки OpenCV з фільтрації

Бібліотека OpenCV пропонує різноманітні можливості для фільтрації зображень та відео. Ці можливості включають в себе застосування різних типів фільтрів, таких як гаусівський фільтр, медіанний фільтр, фільтр усереднення та інші, для покращення якості та чіткості зображень.

Гаусівський фільтр застосовується для розмиття зображень шляхом зменшення впливу високочастотних компонентів і підсилення низькочастотних компонентів. Це допомагає зменшити шум та покращити різкість зображення.

Медіанний фільтр використовується для розмиття зображень шляхом заміни значення пікселя на медіану значень пікселів у вікні фільтра. Цей тип фільтра особливо ефективний для видалення шуму на зображеннях без значного втручання у деталі. [2]

Фільтр усереднення обчислює середнє значення пікселів у вікні фільтра, що допомагає зменшити шум та розмити зображення.

Крім того, OpenCV пропонує інші типи фільтрів, такі як білатеральний фільтр, який зберігає різкість країв зображення, та адаптивний фільтр, який автоматично адаптується до характеристик зображення.

Ці фільтри можуть бути використані для покращення якості зображень, видалення шуму, виправлення змін у контрасті та освітленості, а також для підготовки зображень для подальшої обробки та аналізу. Завдяки цим можливостям OpenCV є потужним інструментом для обробки зображень та відео у багатьох областях, включаючи комп'ютерний зір, медичне зображення, аналіз зображень та багато інших. [6]

6.2 Метод Фільтр НЧ з різними масками для відео

Метод фільтрації НЧ (напрямленого зниження частот) з різними масками для відео використовується для зниження високочастотного шуму та видалення дрібних деталей зображення, залишаючи при цьому низькочастотні складові, такі як основні структури та об'єкти.

Основна ідея полягає в застосуванні різних масок (які також відомі як ядра або фільтри) до кожного кадру відеопотоку. Ці маски визначають, які частоти слід підсилити (низькочастотні) та які частоти слід приглушити (високочастотні). Зазвичай використовуються фільтри з різними розмірами та формами, які дозволяють досягти різного ступеня зниження високочастотних складових.

Математично метод фільтрації НЧ відображається через згортку між вихідним зображенням та маскою, де кожен піксель вихідного зображення замінюється на ваговану суму пікселів вихідного зображення та відповідних ваг кожного пікселя в масці.

Параметри методу фільтрації НЧ включають в себе вибір конкретної маски (ядро) для фільтрації, розмір ядра (визначає ступінь розмиття або зниження частот) та форму маски (наприклад, кругову, квадратну, або варіанти Гаусіана).

Отже, метод фільтрації НЧ з різними масками для відео дозволяє підсилити низькочастотні складові зображення та приглушити високочастотні, що допомагає поліпшити якість зображення та зменшити шум.

6.3 Реалізація на Python й аналіз результатів

Метод, представлений у наведеному коді, використовує низькочастотний фільтр (НЧ фільтр) для обробки кадрів відеопотоку. Низькочастотний фільтр дозволяє проходити низькочастотні компоненти сигналу, приглушаючи високочастотні складові. У контексті зображень, це означає, що дрібні деталі та шум будуть приглушені, залишаючи головні структурні елементи зображення.

У функції apply_low_pass_filter використовується функція cv2.getGaussianKernel, щоб створити маску для низькочастотного фільтра. Параметр kernel_size визначає розмір маски, який впливає на ступінь розмиття та зниження частот. Більший розмір маски призведе до більшого зниження високочастотних складових, але також може спричинити втрату деякої деталізації.

У циклі обробляється кожен кадр відеопотоку, застосовуючи низькочастотний фільтр до кожного кадру за допомогою функції apply_low_pass_filter. Результат обробки відображається у реальному часі за допомогою функції cv2.imshow.

Значення параметра kernel_size може бути налаштоване залежно від конкретних вимог задачі та характеристик вхідного відеопотоку. Після застосування фільтра до кожного кадру відеопотоку, отриманий результат може бути використаний для подальшого аналізу, обробки або візуалізації, залежно від конкретних потреб користувача.

Лістинг програми, яка демонструє виділення меж з різними порогами sobel (лістинг А.6).

Екранні форми виконання програми, яка виконує виділення меж з різними порогами sobel (рисунок Б.6).

7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ

7.1 Поняття класу та його створення на Python

Клас є основною концепцією об'єктно-орієнтованого програмування (ООП). Він визначає структуру об'єктів, що представляють реальні сутності або абстракції. У Python клас визначається ключовим словом class, за яким слідує ім'я класу та блок визначення атрибутів і методів.

Атрибути класу - це змінні, які зберігають дані, що належать до класу. До них можна звертатися через крапку, яка розділяє ім'я класу та ім'я атрибуту.

Методи класу - це функції, що визначають поведінку об'єктів класу. Перший аргумент методу, зазвичай self, посилається на конкретний об'єкт класу.

Конструктор класу, метод __init__, викликається при створенні нового об'єкта. Він ініціалізує атрибути об'єкта.

Інкапсуляція полягає в обмеженні доступу до атрибутів і методів класу ззовні. Наслідування дозволяє класу успадковувати атрибути та методи іншого класу, що дозволяє перевикористовувати код та створювати ієрархію класів. [7]

7.2 Реалізація класу VideoProcessor і використання його в інтерфейсі користувача

VideoProcessor - це класс, спрямований на обробку відеоданих залежно від різноманітних параметрів та умов, що визначаються користувачем. Цей клас виконує різні завдання, пов'язані з обробкою відео, такі як геометричні трансформації, колірні перетворення, фільтрація шумів, виявлення країв та інші.

Одним із головних атрибутів цього класу є video_src, який представляє шлях до відеофайлу. Це важливий параметр, який визначає джерело відеоданих для обробки. Другий атрибут, media, представляє собою тип Media, що визначає джерело відео: відео може завантажуватися з файлу або використовувати вебкамеру.

Крім того, клас має атрибут mode, який визначає режим обробки відео. Це дозволяє користувачеві вибрати певний тип обробки, наприклад, геометричну трансформацію, колірне перетворення, фільтрацію шумів тощо.

Клас також містить ряд методів для роботи з відеоданими. Метод __init__ є конструктором класу, який приймає шлях до відеофайлу та ініціалізує інші атрибути. Методи set_mode та set_media встановлюють режим та джерело відео відповідно. Метод show відтворює відео та застосовує обробку до кожного кадру. Метод __process() використовується для обробки кадру в залежності від обраного режиму, а методи __rotate(), __apply_low_pass_filter() та __sobel_edge_detection() реалізують відповідно геометричну трансформацію, низькочастотну фільтрацію та виявлення країв за допомогою оператора Собеля.

Цей клас надає гнучкий та потужний засіб для обробки відеоданих за допомогою різних методів та параметрів, що визначаються користувачем.

Лістинг реалізації класу VideoProcessor (лістинг А.7).

Клас VideoProcessor використовується в графічному інтерфейсі користувача. Він починається зі спілкування з юзером, який обирає джерело відео та режим його обробки. Користувачеві надається можливість обрати відеофайл для обробки або використати вебкамеру як джерело відеоданих.

Користувач вибирає джерело відео, вказуючи відповідні цифрові коди: "1" для відео з файлу або "2" для вебкамери. Після цього, якщо обране джерело - файл, користувач обирає конкретний відеофайл зі списку доступних варіантів.

Після вибору джерела відео, користувач обирає режим обробки відео, вибираючи відповідний номер зі списку доступних режимів.

Після введення всіх необхідних параметрів програма створює екземпляр класу `VideoProcessor`, передаючи шлях до відеофайлу, вибраний режим та джерело відео. Після цього викликається метод `show()`, який починає відтворення відео та застосовує обрану обробку до кожного кадру.

Ця програма надає зручний та інтуїтивно зрозумілий інтерфейс для вибору джерела відео та режиму його обробки, дозволяючи користувачеві легко працювати з відеоданими та отримувати необхідні результати обробки.

Лістинг використання класу VideoProcessor в GUI (лістинг А.8).

Екранні форми використання класу VideoProcessor (рисунок Б.7).

ВИСНОВКИ

Бібліотека OpenCV виявляється потужним інструментом для обробки зображень та відео. Вона надає широкий спектр функцій, включаючи розмиття, виявлення країв, фільтрацію, аналіз руху та інші, що робить її корисним інструментом для великої кількості завдань у сфері комп'ютерного зору та обробки відео.

Розглянуті різні методи обробки зображень та відео, такі як виявлення країв за допомогою оператора Собеля, фільтрація низьких частот для зниження шуму, розмиття та інші. Кожен з цих методів має свої унікальні особливості та застосування, що робить їх корисними для різних сценаріїв обробки відеоданих.

Описано створення класу VideoProcessor, який є засобом для обробки відеоданих. Цей клас дозволяє виконувати різні операції з відео, такі як відтворення, застосування фільтрів, виявлення країв та інші. Він є потужним інструментом для автоматизації обробки відеоданих за допомогою Python.

Загалом, описно важливість використання бібліотеки OpenCV та створення відповідних класів для ефективної обробки відеоданих, що може бути корисним у різних областях, таких як комп'ютерний зір, аналіз відео та інші.

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 об’єкти

ДОДАТОК А

Лістинг коду

Лістинг А.1 – Зчитування відео з файлу

```
import cv2
from Constants import HOURGLAS_VIDEO_SRC

# Відкриття відеопотоку за допомогою відеофайлу
cap = cv2.VideoCapture(HOURGLAS_VIDEO_SRC)

while cap.isOpened():
    # Запис кадру з відеопотоку
    ret, frame = cap.read()

    # При завешенні відображеня виходимо з циклу
    if not ret:
        break

    # Відображення відеопотока з відео
    cv2.imshow('frame', frame)

    # Вихід з програми натисканням клавіши q
    if cv2.waitKey(25) == ord('q'):
        break

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
cap.release()
cv2.destroyAllWindows()
```

Лістинг А.2 – Зчитування відео з веб камери

```
import cv2

# Відкриття відеопотоку з вбудованої вебкамери (за замовчуванням 0)
vid = cv2.VideoCapture(0)

while vid.isOpened():
    # Зчитування кадру з відеопотоку
    ret, frame = vid.read()

    # Перевірка на успішне зчитування кадру
    if not ret:
        break  # Якщо кадр не було зчитано (наприклад, відеопотік завершився),
вийти з циклу

    # Відображення кадру на екрані
    cv2.imshow('frame', frame)

    # Вихід з програми натисканням клавіши q
```

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

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
vid.release()
cv2.destroyAllWindows()
```

Лістинг А.3 – Розворот відео

```
import cv2
from Constants import HOURGLAS_VIDEO_SRC

# Функція повороту кадру на заданий кут
def rotate(frame, angle):
    # Отримання висоти та ширини кадру
    num_rows, num_cols = frame.shape[:2]
    # Створення матриці повороту
    rotation_matrix = cv2.getRotationMatrix2D((num_cols / 2, num_rows / 2), angle,
1)
    # Поворот кадру за допомогою отриманої матриці
    img_rotation = cv2.warpAffine(frame, rotation_matrix, (num_cols, num_rows))
    return img_rotation


# Відкриття відеопотоку за допомогою відеофайлу
cap = cv2.VideoCapture(HOURGLAS_VIDEO_SRC)

while cap.isOpened():
    # Зчитування кадру з відеопотоку
    ret, frame = cap.read()

    # Перевірка на успішне зчитування кадру
    if not ret:
        break

    # Поворот фрейму на 90 градусів
    frame = rotate(frame, 90)

    # Відображення кадру з відео
    cv2.imshow('Rotated Frame', frame)

    # Очікування натискання клавіші з кодом ASCII 'q' (для виходу з програми)
    if cv2.waitKey(25) == ord('q'):
        break

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
cap.release()
cv2.destroyAllWindows()
```

Лістинг А.4 – Зміна кольорового простору на RGB

```
import cv2
from Constants import HOURGLAS_VIDEO_SRC

# Відкриття відеопотоку за допомогою відеофайлу
cap = cv2.VideoCapture(HOURGLAS_VIDEO_SRC)

while cap.isOpened():
    # Зчитування кадру з відеопотоку
    ret, frame = cap.read()

    # Перевірка на успішне зчитування кадру
    if not ret:
        break

    # Перетворення кольорового простору в RGB
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Відображення кадру з відео
    cv2.imshow('RGB', frame_rgb)

    # Очікування натискання клавіші з кодом ASCII 'q' (для виходу з програми)
    if cv2.waitKey(25) == ord('q'):
        break

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
cap.release()
cv2.destroyAllWindows()
```

Лістинг А.5 – Виділення меж з різними порогами sobel

```
import cv2
from Constants import HOURGLAS_VIDEO_SRC

# Функція для застосування оператора Собеля для виявлення країв
def sobel_edge_detection(image, k_size):
    # Перетворення зображення у відтінки сірого
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Застосування оператора Собеля по горизонталі
    sobel_x = cv2.Sobel(gray_image, -1, 1, 0, k_size)
    # Застосування оператора Собеля по вертикалі
    sobel_y = cv2.Sobel(gray_image, -1, 0, 1, k_size)
    # Об'єднання результатів за допомогою апроксимації величини градієнту
    sobel_xy = sobel_x + sobel_y

    return sobel_xy


# Відкриття відеопотоку за допомогою відеофайлу
cap = cv2.VideoCapture(HOURGLAS_VIDEO_SRC)
```

```
while cap.isOpened():
    # Зчитування кадру з відеопотоку
    ret, frame = cap.read()

    # Перевірка на успішне зчитування кадру
    if not ret:
        break

    # Перетворення для виявлення країв оператором Собеля
    frame = sobel_edge_detection(frame, 10)

    # Відображення кадру з відео
    cv2.imshow('Sobel', frame)

    # Очікування натискання клавіші з кодом ASCII 'q' (для виходу з програми)
    if cv2.waitKey(25) == ord('q'):
        break

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
cap.release()
cv2.destroyAllWindows()
```

Лістинг А.6 – Фільтр НЧ з різними масками

```
import cv2
from Constants import HOURGLAS_VIDEO_SRC

# Функція для застосування низькочастотного фільтра до кадру
def apply_low_pass_filter(frame, kernel_size):
    # Створення маски для низькочастотного фільтра
    kernel = cv2.getGaussianKernel(kernel_size, 0)
    # Застосування фільтра до кадру
    filtered_frame = cv2.filter2D(frame, -1, kernel)

    return filtered_frame

# Відкриття відеопотоку за допомогою відеофайлу
cap = cv2.VideoCapture(HOURGLAS_VIDEO_SRC)

while cap.isOpened():
    # Зчитування кадру з відеопотоку
    ret, frame = cap.read()

    # Перевірка на успішне зчитування кадру
    if not ret:
        break

    # Застосування низькочастотного фільтра до кадру
    frame = apply_low_pass_filter(frame, 5)
```

```
    # Відображення відеопотока з фільтром
    cv2.imshow('Filter', frame)

    # Очікування натискання клавіші з кодом 'q' (для виходу з програми)
    if cv2.waitKey(25) == ord('q'):
        break

# Звільнення ресурсів відеопотоку та закриття вікон OpenCV
cap.release()
cv2.destroyAllWindows()
```

Лістинг А.7 – Реалізація класу VideoProcessor

```
import cv2
from enum import Enum

class Media(Enum):
    FROM_FILE = 1   # Завантаження відео з файлу
    WEBCAMERA = 2   # Завантаження відео з вебкамери

class Mode(Enum):
    GEOMETRY_TRANSFORMATION = 1 # Поворот на 45 градусів
    COLOR_TRANSFORMATION = 2    # Колірне перетворення в RGB
    NOISE_FILTRATION = 3        # Низькочастотна фільтрація
    SOBEL = 4                   # Оператора Собеля для виявлення країв
    DEFAULT = 5                 # Виведення звичайного відео

class VideoProcessor:
    # Конструктор
    def __init__(self, video_src):
        self.video_src = video_src      # Шлях до файлу
        self.media = Media.FROM_FILE      # За замовчуванням використовується
завантаження відео з файлу
        self.mode = Mode.DEFAULT        # За замовчуванням використовується режим
без змін

    # Встановлення режиму обробки
    def set_mode(self, mode):
        self.mode = mode

    # Встановлення джерело
    def set_media(self, media):
        self.media = media

    # Відтворення відео та застосування обробки.
    def show(self):
        cap = cv2.VideoCapture(self.video_src)
        while cap.isOpened():
            ret, frame = cap.read()
```

```
            if not ret:
                break

            frame = self.__process(frame)

            cv2.imshow('frame', frame)
            if cv2.waitKey(25) == ord('q'):
                break
        cap.release()
        cv2.destroyAllWindows()

    # Функція обробки кадру в залежності від обраного режиму
    def __process(self, frame):
        if self.mode == Mode.DEFAULT:
            return frame
        if self.mode == Mode.GEOMETRY_TRANSFORMATION:
            return self.__rotate(frame, 45)
        if self.mode == Mode.COLOR_TRANSFORMATION:
            return cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        if self.mode == Mode.NOISE_FILTRATION:
            return self.__apply_low_pass_filter(frame, 5)
        if self.mode == Mode.SOBEL:
            return self.__sobel_edge_detection(frame, 5)

    # Функція повороту кадру на заданий кут
    def __rotate(self, frame, angle):
        # Отримання висоти та ширини кадру
        num_rows, num_cols = frame.shape[:2]
        # Створення матриці повороту
        rotation_matrix = cv2.getRotationMatrix2D((num_cols / 2, num_rows / 2),
angle, 1)
        # Поворот кадру за допомогою отриманої матриці
        img_rotation   =   cv2.warpAffine(frame,   rotation_matrix,   (num_cols,
num_rows))
        return img_rotation

    # Функція для застосування низькочастотного фільтра до кадру
    def __apply_low_pass_filter(self, frame, kernel_size):
        # Створення маски для низькочастотного фільтра
        kernel = cv2.getGaussianKernel(kernel_size, 0)
        # Застосування фільтра до кадру
        filtered_frame = cv2.filter2D(frame, -1, kernel)
        return filtered_frame

    # Функція для застосування оператора Собеля для виявлення країв
    def __sobel_edge_detection(self, frame, k_size):
        # Перетворення зображення у відтінки сірого
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Застосування оператора Собеля по горизонталі
        sobel_x = cv2.Sobel(gray_frame, -1, 1, 0, k_size)
```

```
        # Застосування оператора Собеля по вертикалі
        sobel_y = cv2.Sobel(gray_frame, -1, 0, 1, k_size)
        # Об'єднання результатів за допомогою апроксимації величини градієнту
        sobel_xy = sobel_x + sobel_y
        return sobel_xy
```

Лістинг А.8 – Використання класу VideoProcessor в GUI

```
from VideoProcessor import VideoProcessor, Media, Mode
from Constants import GAME_VIDEO_SRC, HOURGLAS_VIDEO_SRC # Імпорт посилань на відеофайли

def main():
    # Обирання джерела відео
    media_choice = input("Оберіть джерело відео: (1 - з файлу, 2 - веб-камера): ")
    if media_choice == "1":
         # Обирання відеофайлу
        video_choice = input("Оберіть відеофайл: (1 - Hourglass.mp4, 2 - Game.mp4): ")
        if video_choice == "1":
            video_src = HOURGLAS_VIDEO_SRC
        elif video_choice == "2":
            video_src = GAME_VIDEO_SRC
        else:
            print("Невірний вибір. Завершення програми.")
            return
        media = Media.FROM_FILE
    elif media_choice == "2":
        media = Media.WEBCAMERA
        video_src = 0
    else:
        print("Невірний вибір. Завершення програми.")
        return

    # Обирання режиму обробки відео
    mode_choice = input("Оберіть режим обробки відео: (1 - Геометрична трансформація, 2 - Кольорова трансформація, "
                        "3 - Шумозаглушення, 4 - Виявлення країв Собелем, 5 - Звичайне відео): ")
    if mode_choice not in ["1", "2", "3", "4", "5", "6"]:  # Якщо обране невірне значення
        print("Невірний вибір. Завершення програми.")
        return

    # Створення екземпляру класу VideoProcessor
    processor = VideoProcessor(video_src)
    processor.set_media(Media(media)) # Встановленя джерела
    processor.set_mode(Mode(int(mode_choice))) # Встановленя режиму обробки відео2
    processor.show() # Запуск відтворення відео
```

```
# При запуску як головного файлу
if __name__ == "__main__":
    main()
```

ДОДАТОК Б
Екранні форми виконання

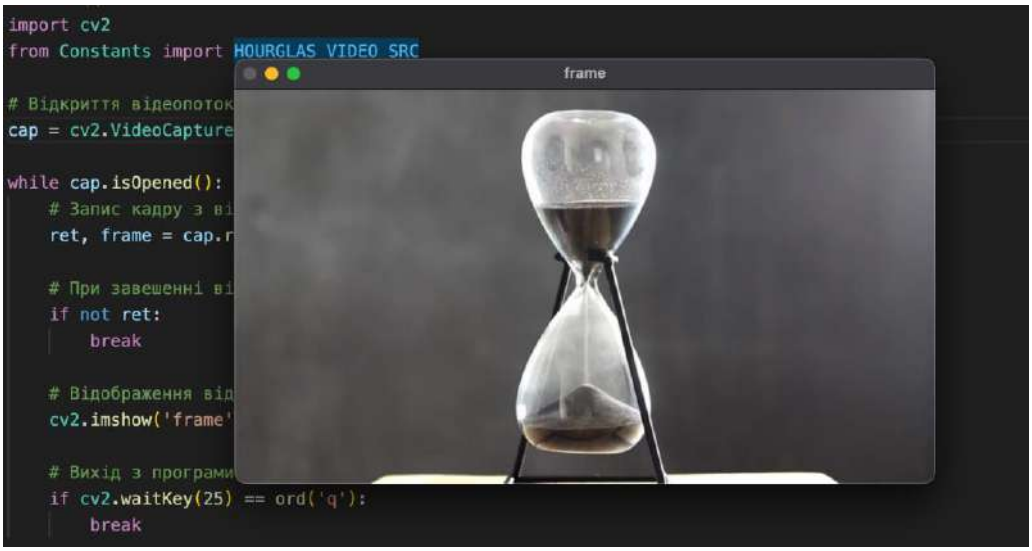

Рисунок Б.1 – Зчитування відео з файлу

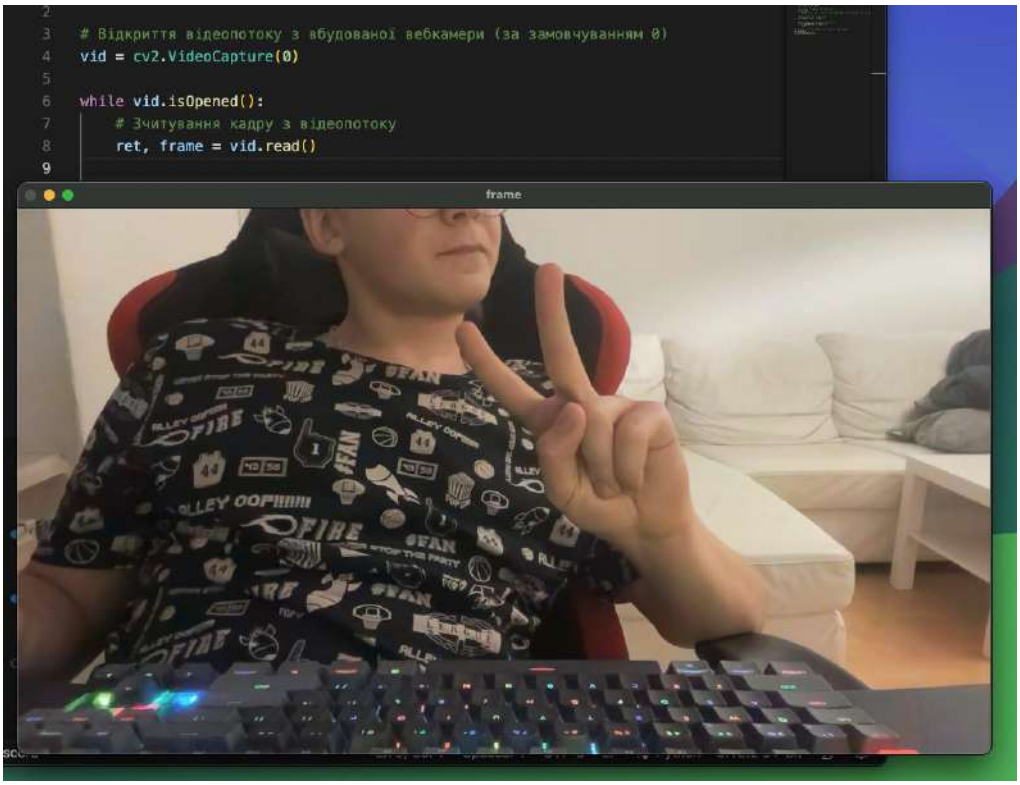

Рисунок Б.2 – Зчитування відео з веб камери

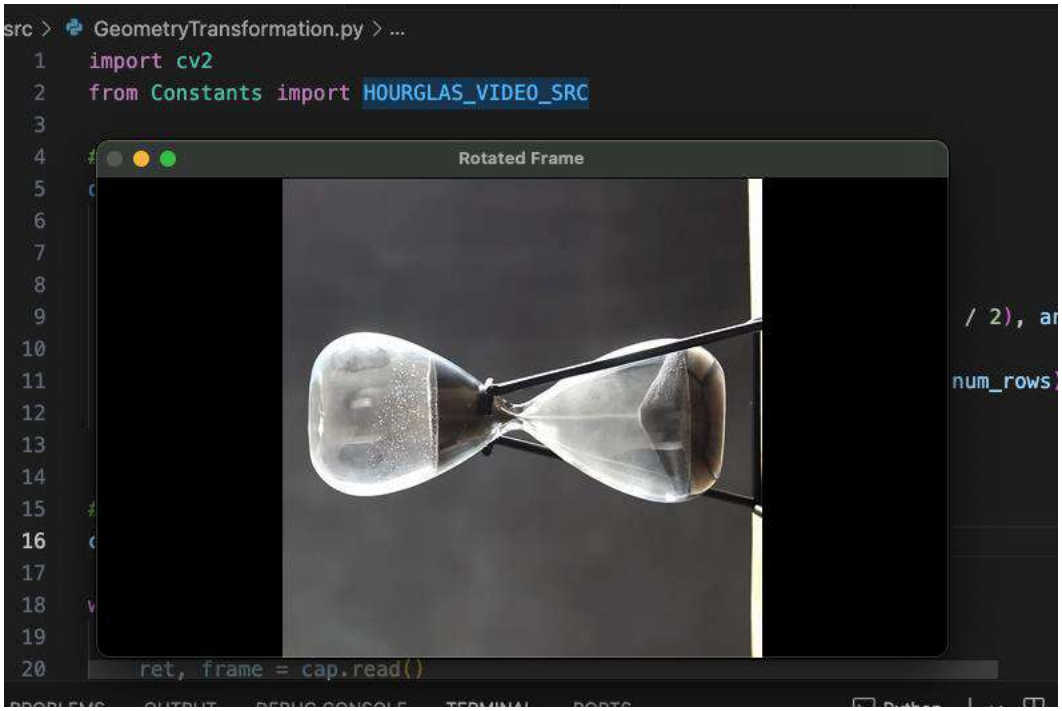

Рисунок Б.3 – Розворот відео на 90 градусів

Рисунок Б.4 – Зміна кольорового простору на RGB

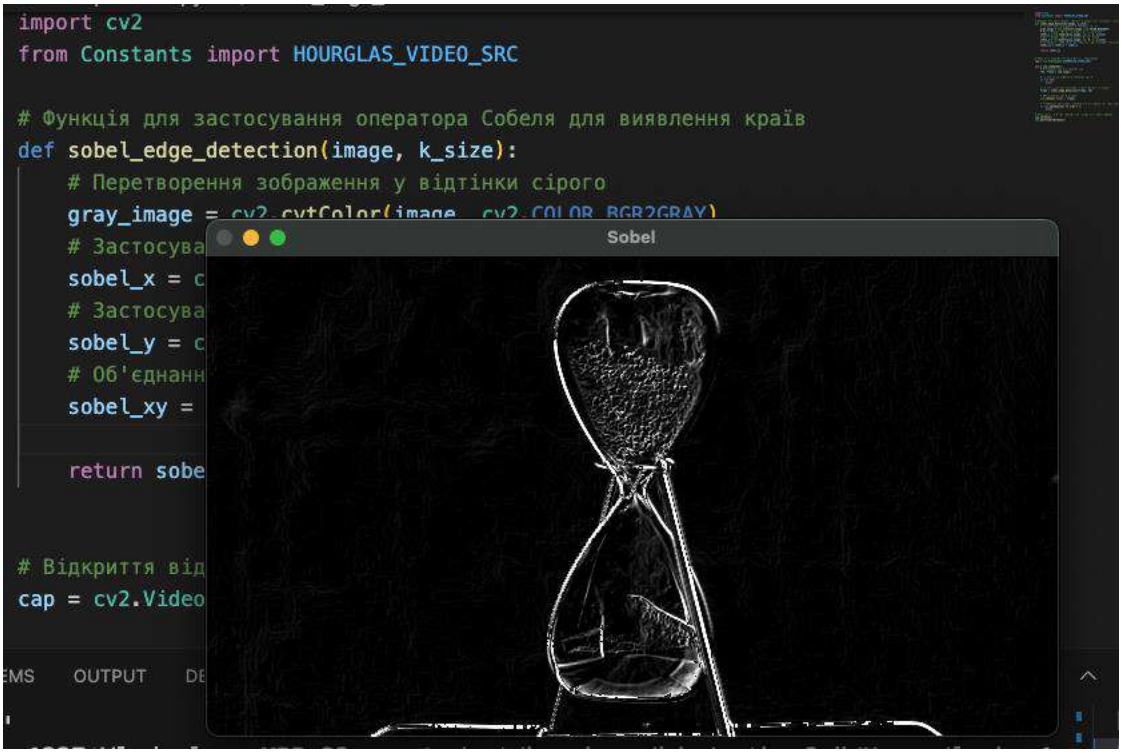

Рисунок Б.5 – Виділення меж з різними порогами sobel

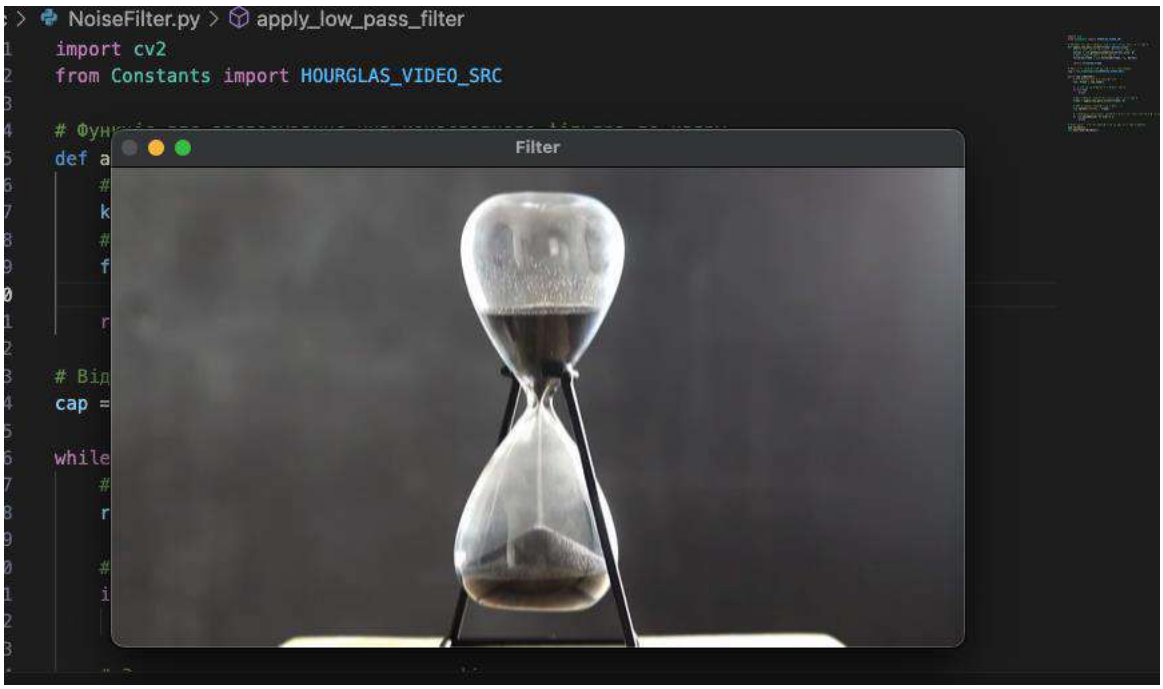

Рисунок Б.6 – Фільтр НЧ з різними масками

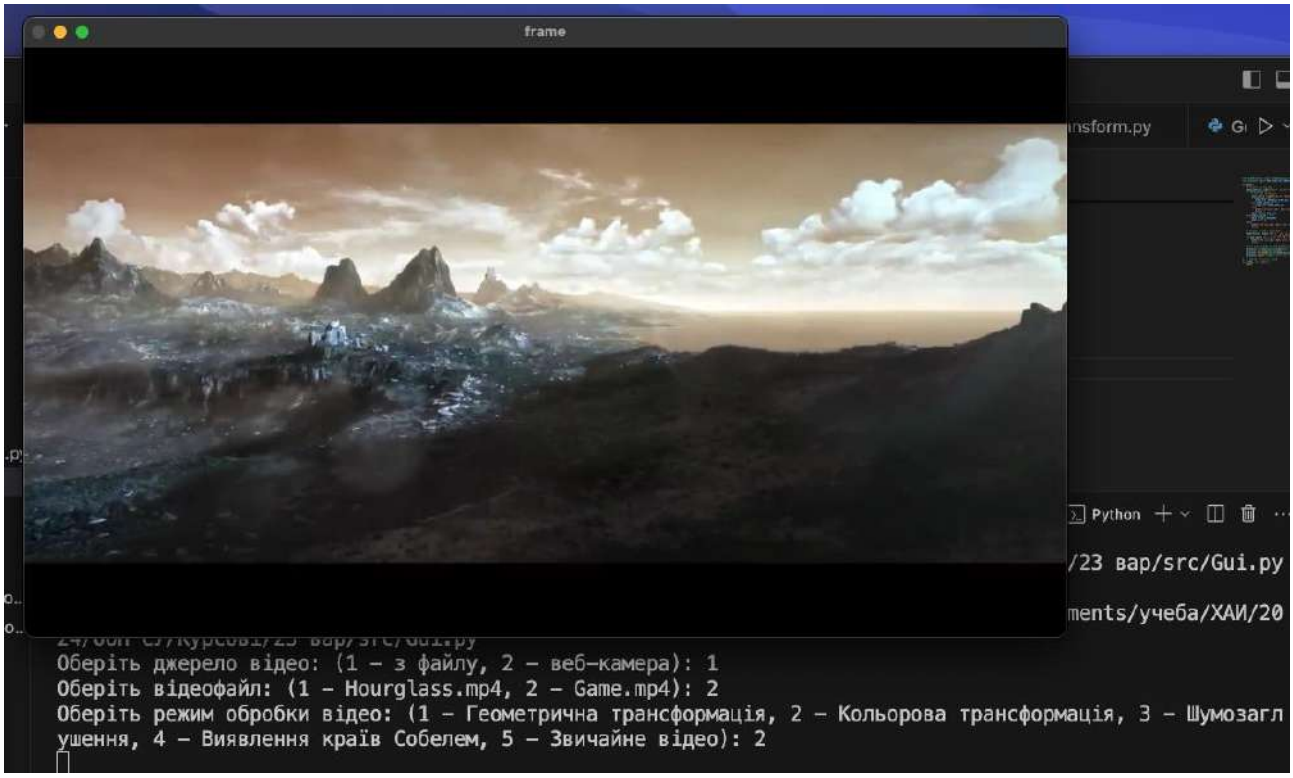

Рисунок Б.7 – Використання класу VideoProcessor