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 № \_321\_

Напряму підготовки (спеціальності)

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

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

(шифр і назва напряму підготовки (спеціальності))

Новіков О. С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(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 32, ілюстрацій 7, джерел 9, додатків 2.

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

КЛЮЧОВІ СЛОВА: OPENCV, PYTHON, ОБРОБКА ВІДЕО, КОМПʼЮТЕРНИЙ ЗІР, ВІДЕОАНАЛІЗ

# Зміст

[Зміст 3](#_Toc163164249)

[Перелік використаних позначень 4](#_Toc163164250)

[Вступ 5](#_Toc163164251)

[1 Огляд проблеми обробки зображень в системах управління з технічним зором 7](#_Toc163164252)

[2 Методи та засоби отримання відеоДАНИХ 10](#_Toc163164253)

[2.1 Завантаження відео з файлу 10](#_Toc163164254)

[2.2 Захват відео з веб-камери 10](#_Toc163164255)

[3 Геометричні перетворення відео-зображень 11](#_Toc163164256)

[3.1 Можливості бібліотеки OpenCV з геометричних перетворень 11](#_Toc163164257)

[3.2 Метод геометричного перетворення для вертикального зсуву зображення 11](#_Toc163164258)

[3.3 Реалізація на Python й аналіз результатів 12](#_Toc163164259)

[4 Кольорові перетворення відео-зображень 13](#_Toc163164260)

[4.1 Можливості бібліотеки OpenCV з кольорових перетворень 13](#_Toc163164261)

[4.2 Метод зміни кольорового простору на XYZ 14](#_Toc163164262)

[4.3 Реалізація на Python й аналіз результатів 14](#_Toc163164263)

[5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ 15](#_Toc163164264)

[5.1 Можливості бібліотеки OpenCV для виконання операцій 15](#_Toc163164265)

[5.2 Метод виділення меж з різними порогами canny. 16](#_Toc163164266)

[5.3 Реалізація на Python й аналіз результатів 16](#_Toc163164267)

[6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ 17](#_Toc163164268)

[6.1 Можливості бібліотеки OpenCV з фільтрації 17](#_Toc163164269)

[6.2 Метод фільтрації зображень з ефектом зсуву з різними параметрами 18](#_Toc163164270)

[6.3 Реалізація на Python й аналіз результатів 19](#_Toc163164271)

[7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ 20](#_Toc163164272)

[7.1 Поняття класу та його створення на Python 20](#_Toc163164273)

[7.2 Реалізація класу VideoProcessor і використання його в інтерфейсі користувача 20](#_Toc163164274)

[Висновки 22](#_Toc163164275)

[Перелік використаних джерел 23](#_Toc163164276)

[Додаток А 24](#_Toc163164277)

[Додаток Б 32](#_Toc163164278)

# Перелік використаних позначень

OpenCV – Open Source Computer Vision Library

CIE – Corpus Inscriptionum Etruscarum

ООП – обʼєктно орієнтоване програмування;

# Вступ

У сучасному світі системи управління з технічним зором відіграють ключову роль у різноманітних сферах, від промисловості до медицини та безпеки. Обробка відеозображень у таких системах виявляється критичною для забезпечення точності та ефективності їх функціонування. Вона дозволяє аналізувати та інтерпретувати великі обсяги відеоданих, виявляти об'єкти, розпізнавати патерни та приймати рішення в реальному часі, що забезпечує автоматизацію процесів та підвищує рівень безпеки та ефективності систем управління.

Актуальність теми полягає в необхідності вдосконалення сучасних технологічних рішень шляхом застосування методів обробки відеозображень, особливо в контексті систем управління з технічним зором. Поринувши у цей контекст, ми вбачаємо значну потребу у розробці та впровадженні інноваційних методів обробки відеоданих для забезпечення ефективної функціональності та забезпечення потреб сучасного ринку.

Основною метою дослідження є аналіз інструментів та методів обробки відеозображень з використанням Python та OpenCV з метою оптимізації та підвищення продуктивності систем управління з технічним зором. Серед конкретних завдань роботи визначається аналіз існуючих методів обробки відеоданих, розробка нових алгоритмів та їхнє порівняння для визначення оптимальних рішень.

Об'єктом дослідження є системи управління з технічним зором, а предметом - методи обробки відеозображень.

Практичне значення проведеного дослідження полягає у висвітленні потенційних можливостей застосування отриманих результатів у практиці. Розроблені методи обробки відеозображень можуть бути впроваджені для покращення функціональності та ефективності систем управління з технічним зором. Це відкриває шлях до оптимізації процесів, підвищення рівня безпеки та забезпечення кращої адаптації до змінних умов.

Структура курсової роботи передбачає детальний огляд та аналіз проблеми обробки зображень у системах управління з технічним зором у першому розділі. Наступні розділи присвячені методам та засобам отримання відеоданих, геометричним та кольоровим перетворенням, операціям та фільтрації зображень. Завершальний розділ присвячений реалізації класу для обробки відеоданих, що узагальнює отримані результати та дозволяє їх застосування у практичних проектах. Кожен розділ взаємопов'язаний із іншими, утворюючи цілісну структуру дослідження, яка дає змогу зрозуміти проблему та знаходити шляхи для її вирішення.

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

Теоретичні основи обробки зображень є фундаментальною частиною систем управління з технічним зором. Цей підрозділ розглядає ключові концепції, які дозволяють зрозуміти та виконувати обробку зображень на різних рівнях.

Моделі зображень є важливими для розуміння того, як зображення представлені в системах. Вони можуть бути аналоговими або цифровими, і кожна модель має свої особливості та обмеження. Цифрові зображення, наприклад, представлені у вигляді матриці пікселів, де кожен піксель має певне значення яскравості та кольору¹.

Представлення кольору в цифрових зображеннях зазвичай виконується за допомогою кольорових моделей, таких як RGB (червоний, зелений, блакитний), CMYK (циан, маджента, жовтий, чорний) або HSL (відтінок, насиченість, яскравість). Кожна модель має свої застосування, наприклад, RGB часто використовується в екранних технологіях, тоді як CMYK використовується в поліграфії².

Основні операції з зображеннями включають фільтрацію, яка може бути лінійною або нелінійною, та використовується для покращення зображення або виділення певних характеристик. Інші операції, такі як морфологічні перетворення, допомагають в аналізі форми та структури об'єктів на зображенні. Сегментація зображення дозволяє виділити та аналізувати окремі об'єкти, тоді як класифікація використовується для розпізнавання та категоризації цих об'єктів³.

Ці основні теоретичні поняття є відправною точкою для глибшого розуміння та розробки методів обробки зображень, які можуть бути застосовані в різних областях, від медичної візуалізації до автоматизованого виробництва. Вони також є критично важливими для розробки алгоритмів машинного зору, які можуть імітувати людське сприйняття та підтримувати прийняття рішень в реальному часі.

Алгоритми обробки зображень відіграють ключову роль у системах технічного зору, дозволяючи покращити якість зображень, виявити краї, сегментувати та класифікувати об’єкти. Ці процеси є фундаментальними для аналізу та інтерпретації візуальної інформації.

Покращення якості зображень зазвичай досягається за допомогою фільтрації, яка може включати згладжування для усунення шуму та підсилення для підкреслення деталей. Фільтри, такі як Гауссівський фільтр або медіанний фільтр, використовуються для згладжування, тоді як фільтри Собеля та Прюітта застосовуються для підсилення країв3.

Виявлення країв є важливим етапом у обробці зображень, оскільки краї часто відображають межі об’єктів. Алгоритм Кенні є одним з найпопулярніших методів виявлення країв, оскільки він ефективно виявляє слабкі та сильні краї та зменшує кількість помилкових країв2.

Сегментація дозволяє виділити та аналізувати окремі об’єкти в зображенні. Це може бути здійснено за допомогою порогової обробки, де пікселі, що перевищують певний поріг, виділяються, або за допомогою кластеризації, де пікселі групуються на основі їх схожості3.

Класифікація об’єктів використовується для ідентифікації та категоризації об’єктів у зображенні. Методи машинного навчання, такі як нейронні мережі, зокрема конволюційні нейронні мережі (CNN), є дуже ефективними для цього завдання, оскільки вони можуть вчитися розпізнавати складні шаблони та характеристики об’єктів3.

Кожен з цих алгоритмів та методів має свої особливості та області застосування, і їх вибір залежить від конкретних завдань та вимог системи технічного зору. Вони разом формують основу для розуміння та обробки візуальної інформації, що є критично важливим для різноманітних застосувань, від автоматичного виробництва до медичної діагностики.

Застосування методів обробки зображень у системах технічного зору є широким та різноманітним, охоплюючи різні галузі, від робототехніки до медичної діагностики. У цьому підрозділі ми розглянемо декілька конкретних прикладів, які демонструють потенціал та важливість обробки зображень.

У робототехніці, системи технічного зору використовуються для навігації, маніпуляції об'єктами та взаємодії з навколишнім середовищем. Розпізнавання об'єктів, виявлення країв та сегментація зображень дозволяють роботам виконувати складні завдання, такі як збірка компонентів або навігація у складних умовах4.

У виробничих лініях, системи технічного зору використовуються для контролю якості, сортування деталей та автоматизації складних процесів. Обробка зображень дозволяє точно ідентифікувати та класифікувати вироби, забезпечуючи високу точність та ефективність виробничих процесів5.

У медицині, обробка зображень відіграє ключову роль у діагностиці та лікуванні. Методи, такі як комп'ютерна томографія (КТ) та магнітно-резонансна томографія (МРТ), залежать від складних алгоритмів обробки зображень для створення детальних візуалізацій внутрішніх структур тіла, що дозволяє лікарям точно діагностувати захворювання4.

У технології доповненої реальності, обробка зображень використовується для інтеграції віртуальних об'єктів у реальний світ. Це вимагає точного розпізнавання та відстеження реальних об'єктів, щоб віртуальні елементи могли бути правильно розміщені та взаємодіяти з реальним середовищем5.

Кожен з цих прикладів підкреслює важливість обробки зображень у сучасних системах технічного зору, демонструючи її вплив на різні аспекти нашого життя та роботи.

# 2 Методи та засоби отримання відеоДАНИХ

## 2.1 Завантаження відео з файлу

Для завантаження відео файлу використовується бібліотека OpenCV, створюється об’єкт VideoCapture, у параметри якого передається шлях до файлу, для зчитування відео з нього. Перевіряється, чи файл відкрився успішно, і якщо так, то зчитуються кадри у циклі. Кожен кадр відображається у вікні, і якщо користувач натисне клавішу ‘q’, програма завершиться.

Лістинг коду – А. Лістинг коду для зчитування з файлу

Екранні форми виконання програми – рисунок Б.1.

## 2.2 Захват відео з веб-камери

Для захвату відео з веб-камери використовується бібліотека OpenCV, створюється об’єкт VideoCapture, у параметри якого передається 0, для зчитування відео з веб-камери. Перевіряється, чи підключення до камери успішно, і якщо так, то зчитуються кадри у циклі. Кожен кадр відображається у вікні, і якщо користувач натисне клавішу ‘q’, програма завершиться.

Лістинг коду – А. Лістинг коду для зчитування з веб камери

Екранні форми виконання програми – рисунок Б.2.

# 3 Геометричні перетворення відео-зображень

## 3.1 Можливості бібліотеки OpenCV з геометричних перетворень

Бібліотека OpenCV є потужним інструментом для роботи з комп'ютерним зором та машинним навчанням. Вона містить багато оптимізованих алгоритмів, включаючи різноманітні геометричні перетворення.

Геометричні перетворення в OpenCV дозволяють здійснювати такі операції, як масштабування, обертання, зсув та викривлення зображень. Ці перетворення можуть бути корисними для попередньої обробки зображень перед подальшим аналізом або для покращення якості зображень.

Масштабування є одним з найпростіших перетворень і полягає у зміні розміру зображення. В OpenCV це можна зробити за допомогою функції resize.

Обертання зображення на певний кут можна виконати за допомогою функції getRotationMatrix2D, яка створює матрицю обертання, та warpAffine, яка застосовує матрицю до зображення.

Зсув зображення виконується за допомогою функції warpAffine з відповідною матрицею перекладу.

Афінні перетворення дозволяють комбінувати масштабування, обертання та переклад, і виконуються за допомогою функції getAffineTransform , яка створює матрицю афінного перетворення, та warpAffine.

Перспективні перетворення або викривлення зображень здійснюються за допомогою функції getPerspectiveTransform, яка створює матрицю перспективного перетворення, та warpPerspective , яка застосовує цю матрицю.

Використання цих перетворень в OpenCV може значно розширити можливості аналізу та обробки зображень, дозволяючи розробникам створювати більш гнучкі та потужні комп'ютерні зорові системи.78

## 3.2 Метод геометричного перетворення для вертикального зсуву зображення

Геометричне перетворення зображення, таке як вертикальний зсув, є одним із фундаментальних операцій у обробці зображень. Для виконання вертикального зсуву зображення використовується афінне перетворення, яке дозволяє змінювати положення пікселів зображення без зміни їх взаємного розташування.

Афінне перетворення можна представити у вигляді матриці. Для вертикального зсуву ця матриця має вигляд:

де *Ty* - це величина зсуву по вертикалі.

Коли ми застосовуємо цю матрицю до координат пікселів зображення, кожен піксель переміщується на величину (*Ty*) вверх або вниз, залежно від знаку *T\_y*. Якщо *T\_y* позитивне, зсув відбувається вниз; якщо негативне - вгору

## 3.3 Реалізація на Python й аналіз результатів

У програмуванні на Python з бібліотекою OpenCV цю операцію можна виконати за допомогою функції cv2.warpAffine(), яка приймає вхідне зображення, матрицю афінного перетворення та розміри вихідного зображення. Матриця створюється за допомогою бібліотеки numpy.

Лістинг коду – А. Лістинг коду для вертикального здвигу

Екранні форми виконання програми – рисунок Б.3.

# 4 Кольорові перетворення відео-зображень

## 4.1 Можливості бібліотеки OpenCV з кольорових перетворень

Бібліотека OpenCV пропонує широкий спектр можливостей для роботи з кольором у зображеннях, що є ключовим аспектом комп'ютерного зору. Кольорові перетворення в OpenCV дозволяють змінювати колірні простори зображень, що може бути корисним для різних завдань, таких як виявлення об'єктів, обробка зображень та машинне навчання.

Колірні простори – це різні способи представлення кольорів у зображеннях. Найбільш відомим є RGB (Червоний, Зелений, Блакитний), але OpenCV підтримує багато інших, включаючи HSV (Відтінок, Насиченість, Значення) та CIELab.

Перетворення кольорових просторів в OpenCV виконуються за допомогою функції cvtColor , яка дозволяє легко перетворювати зображення з одного колірного простору в інший. Це може бути корисно для виділення певних характеристик зображення, які можуть бути менш помітними в стандартному RGB просторі.8

Канали кольору в зображенні можуть бути розділені та аналізовані окремо. В OpenCV це можна зробити за допомогою функції split , що дозволяє отримати індивідуальні канали (наприклад, R, G, B) зображення для подальшої обробки.

Гістограми кольор є потужним інструментом для аналізу кольорової інформації в зображенні. Вони дозволяють визначити розподіл інтенсивності кольорів у зображенні та можуть бути використані для корекції кольору або порівняння зображень.

Баланс білого та корекція кольору можуть бути досягнуті за допомогою різних методів у OpenCV, що дозволяє покращити візуальну якість зображень та готувати їх до подальшого аналізу.

Використання цих інструментів та методів відкриває безліч можливостей для розробників та дослідників у сфері комп'ютерного зору, дозволяючи створювати більш точні та ефективні системи обробки зображень.7

## 4.2 Метод зміни кольорового простору на XYZ

Зміна кольорового простору зображення на CIE 1931 XYZ є важливою операцією в обробці зображень, оскільки цей кольоровий простір служить як універсальний мовник кольорів, що дозволяє точно визначати кольори незалежно від пристрою. Простір XYZ був створений Міжнародною комісією з освітлення (CIE) і є основою для багатьох інших кольорових просторів.

Перетворення зображення з одного кольорового простору в інший вимагає застосування математичних операцій до кольорових компонентів кожного пікселя. Для перетворення RGB в XYZ використовується лінійне перетворення, яке можна виразити через матрицю перетворення. Ця матриця залежить від конкретного RGB простору, який використовується, оскільки різні пристрої можуть мати різні RGB простори.

## 4.3 Реалізація на Python й аналіз результатів

У Python з бібліотекою OpenCV цю операцію можна виконати за допомогою функції cv2.cvtColor(), вказавши відповідний код перетворення. Наприклад, для перетворення зображення з простору BGR (стандартний у OpenCV) в XYZ, використовується код cv2.COLOR\_BGR2XYZ.

Лістинг коду – А. Лістинг коду для зміни кольорового простору на XYZ

Екранні форми виконання програми – рисунок Б.4.

# 5 ОПЕРАЦІЇ З ВІДЕО-ЗОБРАЖЕННЯМИ

## 5.1 Можливості бібліотеки OpenCV для виконання операцій

Бібліотека OpenCV надає потужні інструменти для роботи з відео, що відкриває широкі можливості для розробників та дослідників у галузі комп'ютерного зору. Ось деякі з основних функцій та можливостей OpenCV, які можуть бути застосовані до відео:

- Захоплення відео : OpenCV дозволяє захоплювати відеопотоки з різних джерел, включаючи відеофайли та камери в реальному часі, за допомогою класу VideoCapture .

- Запис відео : За допомогою класу VideoWriter можна записувати оброблені відеопотоки у файл, що дозволяє створювати відео після проведення різних операцій обробки.

- Обробка кадрів : OpenCV пропонує функції для обробки окремих кадрів відео, такі як фільтрація, перетворення кольорів, виявлення країв, тощо.

- Виявлення та відстеження об'єктів : Завдяки алгоритмам машинного навчання та комп'ютерного зору, OpenCV може виявляти та відстежувати об'єкти у відеопотоках.

- Аналіз руху : Можливості аналізу руху в OpenCV дозволяють виявляти та відстежувати рухомі об'єкти, що є корисним для систем відеоспостереження та інтерактивних застосунків.

- Розпізнавання образів : OpenCV містить інструменти для розпізнавання образів та класифікації, які можуть бути застосовані до відео для ідентифікації осіб, предметів або сцен.

- 3D реконструкція: З використанням стереозору та інших технік, OpenCV дозволяє виконувати 3D реконструкцію сцен з відео.

Ці та багато інших функцій роблять OpenCV незамінним інструментом для розробки сучасних комп'ютерних зорових систем, які використовують відео для різноманітних цілей, від простого захоплення та обробки відео до складних завдань аналізу та розуміння візуальної інформації78.

## 5.2 Метод виділення меж з різними порогами canny.

Метод Canny для виділення меж зображення є одним із найбільш використовуваних алгоритмів у комп’ютерному зорі. Цей метод використовує багатоступеневий підхід для виявлення широкого спектру країв у зображенні. Основні кроки алгоритму Canny включають згладжування зображення за допомогою Гауссового фільтра, знаходження градієнтів зображення, застосування немаксимального придушення, подвійне порогове фільтрування та відстеження країв за допомогою гістерезису.

Для початку, зображення згладжується для зменшення впливу шуму на результат виділення меж. Це досягається за допомогою конволюції зображення з Гауссовим ядром. Після згладжування виконується пошук градієнтів зображення, що визначають інтенсивність та напрямок зміни кольору. Градієнти обчислюються за допомогою операторів Собеля або інших операторів градієнтів.

Немаксимальне придушення допомагає відокремити справжні краї від шуму, зберігаючи лише найсильніші градієнти, що вказують на краї. Подвійне порогове фільтрування використовує два пороги: високий і низький. Сильні краї, які перевищують високий поріг, вважаються справжніми краями, тоді як слабкі краї, які перевищують низький поріг, вважаються потенційними краями. Краї, інтенсивність яких знаходиться між двома порогами, аналізуються на предмет їх зв’язку з сильними краями за допомогою процесу, відомого як гістерезис.

## 5.3 Реалізація на Python й аналіз результатів

Для виділення меж з різними порогами за допомогою методу Canny на Python з використанням OpenCV використовується функція cv2.Canny, через параметри зображення у відтінках сірого, оскільки метод Canny працює з одноканальними зображеннями. Потім передаються через аргументи нижній та верхній пороги для методу Canny і він застосовується до зображення. Результатом є зображення, на якому виділені межі, яке ми можемо відобразити для перегляду. Метод Canny ефективний для виділення меж у зображеннях з різними рівнями контрасту.

Лістинг коду – А. Лістинг коду для виділення меж з різними порогами canny.

Екранні форми виконання програми – рисунок Б.5.

# 6 ФІЛЬТРАЦІЯ ВІДЕО-ЗОБРАЖЕНЬ

## 6.1 Можливості бібліотеки OpenCV з фільтрації

Бібліотека OpenCV пропонує розширені можливості для фільтрації зображень, що є важливим елементом у передобробці та аналізі зображень у комп'ютерному зорі. Фільтрація дозволяє покращити якість зображення, виділити важливі особливості або зменшити шум. Ось деякі ключові аспекти фільтрації в OpenCV:

- Лінійна фільтрація: Використовуючи функції, такі як blur для простого розмиття та GaussianBlur для гауссового розмиття, можна досягти згладжування зображень.

- Нелінійна фільтрація: Функції, такі як medianBlur , застосовують медіанний фільтр для усунення "соль-перець" шуму без значної втрати країв об'єктів.

- Фільтри країв: За допомогою функцій Sobel , Scharr та Laplacian можна виявити краї об'єктів у зображенні, що є корисним для виявлення контурів.

- Морфологічні операції: Функції, такі як erode , dilate , morphologyEx , дозволяють виконувати ерозію, дилятацію та інші морфологічні перетворення для форми аналізу.

- Фільтрація з використанням просторових та частотних перетворень : OpenCV дозволяє застосовувати фільтри в просторовій та частотній областях, що включає перетворення Фур'є для аналізу частотних компонентів зображення.

Ці інструменти фільтрації відкривають широкі можливості для покращення якості зображень та виділення необхідної інформації, що робить OpenCV незамінним інструментом для розробників та дослідників у сфері комп'ютерного зору78.

## 6.2 Метод фільтрації зображень з ефектом зсуву з різними параметрами

Метод фільтрації зображень з ефектом зсуву використовується для покращення якості зображень шляхом усунення шуму, зберігаючи при цьому важливі деталі та краї. Основними параметрами, які впливають на процес фільтрації, є розмір ядра фільтра та параметри, що визначають ступінь згладжування.

Для зсуву зображення використовуються різні математичні операції, такі як конволюція з ядром фільтра. Ядро фільтра може бути різного розміру та форми, залежно від бажаного ефекту. Наприклад, Гауссівське ядро часто використовується для згладжування зображень, оскільки воно має природній "розмитий" ефект.

Математичний апарат, що лежить в основі методу фільтрації з ефектом зсуву, включає в себе такі поняття, як:

- Конволюція:

Конволюція - це математична операція, яка застосовується до двох функцій та виробляє третю функцію, яка є модифікованою версією однієї з оригінальних функцій. У контексті обробки зображень, одна функція зазвичай є зображенням, а інша - ядром фільтра.

- Функція імпульсної відгуки (ФІВ):

ФІВ ядра фільтра визначає, як кожен піксель впливає на оточуючі пікселі під час процесу фільтрації.

- Частотний аналіз:

Частотний аналіз дозволяє аналізувати зображення на предмет присутності шуму на різних частотах та відповідно фільтрувати ці частоти.

Приклад математичного виразу для конволюції зображення f з ядром фільтра h:

де g(x, y) - це фільтроване зображення, f(x-i, y-j) - значення пікселя оригінального зображення, а h(i, j) - значення ядра фільтра.

Цей метод може бути адаптований для різних типів зображень та шумів, використовуючи специфічні ядра та налаштування параметрів. Важливо знайти баланс між усуненням шуму та збереженням деталей зображення.2

## 6.3 Реалізація на Python й аналіз результатів

Для реалізації методу фільтрації зображень з ефектом зсуву спочатку ініціалізується ядро фільтра, яке представляє собою двовимірний масив. Для створення ефекту зсуву, у визначеному рядку ядра встановлюються одиниці. Після цього ядро нормалізується, щоб сума його елементів була рівною одиниці, що дозволяє отримати середнє значення пікселів зображення. Функція filter2D з бібліотеки OpenCV застосовує ядро до зображення, що призводить до створення ефекту руху або зсуву. Параметри функції включають вхідне зображення, бажану глибину зображення та ядро фільтра. Завдяки цьому методу можна ефективно симулювати рух або розмиття об'єктів на зображенні.

Лістинг коду – А. Лістинг коду для фільтру з ефектом зсуву з різними параметрами

Екранні форми виконання програми – рисунок Б.6.

# 7 РЕАЛІЗАЦІЯ КЛАСУ ДЛЯ ОБРОБКИ ВІДЕОДАНИХ

## 7.1 Поняття класу та його створення на Python

Клас у програмуванні – це спосіб об'єднання даних та методів, які працюють з цими даними, в єдину структуровану одиницю. У Python класи є основними будівельними блоками для об'єктно-орієнтованого програмування.

Створення класу в Python починається з ключового слова class, за яким слідує назва класу та двокрапка. Тіло класу містить визначення методів та атрибутів. Методи класу – це функції, які визначають поведінку об'єктів класу, тоді як атрибути – це дані, пов'язані з цими об'єктами.

Кожен клас у Python має спеціальний метод під назвою \_\_init\_\_, який викликається автоматично, коли створюється новий об'єкт класу. Цей метод часто використовується для ініціалізації атрибутів об'єкта.

Об'єкти класу створюються шляхом виклику назви класу з дужками, і можуть містити різні атрибути та методи, які визначають їх стан та поведінку. Важливою особливістю класів в Python є наслідування, яке дозволяє одному класу успадковувати атрибути та методи іншого класу, тим самим спрощуючи повторне використання коду.9

## 7.2 Реалізація класу VideoProcessor і використання його в інтерфейсі користувача

Клас VideoProcessor призначений для обробки відео з використанням різних методів, таких як вертикальний зсув, виявлення країв за допомогою алгоритму Canny, зміна кольорового простору на XYZ та фільтрація з ефектом зсуву. Клас містить приватні методи для кожного з цих процесів, а також методи для взаємодії з користувацьким інтерфейсом та обробки відео.

Приватні методи класу реалізують різні варіанти обробки зображень. Наприклад, метод \_\_vertical\_shift здійснює вертикальний зсув зображення, метод \_\_canny\_edge\_detection використовує алгоритм Canny для виявлення країв, метод \_\_to\_xyz\_colorspace перетворює зображення у колірний простір XYZ, а метод \_\_motion\_blur застосовує фільтрацію з ефектом зсуву.

Клас також має приватні методи для ініціалізації графічного інтерфейсу \_\_loadGui, встановлення шляху до відео \_\_set\_video\_file\_path та початку запуску відео \_\_start\_processing. Графічний інтерфейс, який реалізований з використанням бібліотеки Tkinter дозволяє користувачеві вибрати тип обробки відео та джерело відео (веб-камера або файл), а також вибрати файл відео для обробки та почати процес обробки.

Після створення об'єкту класу MediaTools можна викликати метод open(), який відкриває графічний інтерфейс для вибору параметрів та обробки відео.

Лістинг коду – А. Лістинг коду класу MediaTools

Екранні форми виконання програми – рисунок Б.7.

# Висновки

Одна з найвідоміших бібліотек в сфері компʼютерного зору OpenCV відіграє ключову роль у розвитку технологій обробки зображень та відео. Її висока продуктивність та широкий спектр функціональних можливостей роблять її незамінним інструментом для розробників, які прагнуть реалізувати складні проекти у цій галузі.

OpenCV надає можливість виконувати такі операції, як розмиття зображень для зниження рівня шуму та підвищення якості візуальних даних, виявлення країв за допомогою оператора Canny для точного визначення контурів об’єктів, а також фільтрацію із зсувом, яка дозволяє виділяти або приховувати специфічні особливості зображення. Перетворення кольорового простору відкриває нові можливості для адаптації візуальної інформації під конкретні завдання, а вертикальний здвиг є важливим інструментом для стабілізації та корекції відео.

Створено клас MediaTools на базі OpenCV для автоматизації процесів обробки відеоданих.

# Перелік використаних джерел

1. Електронний ресурс: <https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D1%8C> – Обробка зображень — Вікіпедія.
2. Електронний ресурс:   
   <https://electronics.lnu.edu.ua/wp-content/uploads/2020-TSOZ-NAVCHALNA.pdf>. – Цифрова обробка та класифікація зображень.
3. Електронний ресурс: <https://elartu.tntu.edu.ua/bitstream/123456789/17887/1/Samostijna_robota_OSZ_6_170101.pdf> – Обробка сигналів та зображень.
4. Електронний ресурс: <https://cyberleninka.ru/article/n/zastosuvannya-mashinnogo-zoru-ta-metodiv-obrobki-zobrazhennya-na-virobnitstvi> – Застосування машинного зору та методів обробки зображення на виробництві
5. Електронний ресурс: <https://nure.ua/metodi-normalizacii-rozpiznavannja-analizu-ta-obrobki-zobrazhen-u-sistemah-komp-juternogo-zoru> – Методи нормалізації, розпізнавання, аналізу та обробки зображень у системах комп’ютерного зору
6. Основы построения современных мобильных систем технического зрения [Текст]: навч. посібник (частина 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.
7. Електронний ресурс: <http://docs.opencv.org> – Документация по библиотеке OpenCV.
8. Електронний ресурс: <https://old-zdia.znu.edu.ua/gazeta/Kozachenko_conf.pdf>. – Дослідження можливості використання бібліотеки OpenCV
9. Електронний ресурс: <https://realpython.com/python3-object-oriented-programming/> – Object-Oriented Programming (OOP) in Python 3

# Додаток А

Лістинг коду

Лістинг коду для зчитування з файлу

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

# Відкриття відеофайлу з вказанням шляху до файлу

cap = cv2.VideoCapture("course-work/videos/Video Test 2K Samsung.mp4")

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Відображення обробленого кадру у вікні з назвою "Processed Frame"

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду для зчитування з веб камери

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

# Захват відео з веб камери за допомогою параметра 0

cap = cv2.VideoCapture(0)

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Відображення обробленого кадру у вікні з назвою "Processed Frame"

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду для зсуву

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

import numpy as np # Імпорт бібліотеки numpy для роботи з матрицями

# Відкриття відеофайлу для читання за допомогою методу VideoCapture з вказанням шляху до файлу

cap = cv2.VideoCapture("course-work/videos/Video Test 2K Samsung.mp4")

# Функція вертикального зсуву зображення

def vertical\_shift(frame, top=200):

# Отримання розмірів кадру (висоти та ширини)

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

# Створення матриці зсуву зображення за допомогою зміщення у вертикальному напрямку

translation\_matrix = np.float32([[1, 0, 0], [0, 1, top]])

# Застосування зсуву до зображення за допомогою афінного перетворення

shifted\_frame = cv2.warpAffine(frame, translation\_matrix, (num\_cols, num\_rows))

return shifted\_frame

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Вертикальний зсув кадру

frame = vertical\_shift(frame)

# Відображення обробленого кадру

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду для зміни кольорового простору на XYZ

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

# функція зміни кольорового простору на XYZ

def to\_xyz\_colorspace(frame):

return cv2.cvtColor(frame, cv2.COLOR\_BGR2XYZ)

# Відкриття відеофайлу для читання за допомогою методу VideoCapture з вказанням шляху до файлу

cap = cv2.VideoCapture("course-work/videos/Video Test 2K Samsung.mp4")

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Зміни кольорового простору на XYZ

frame = to\_xyz\_colorspace(frame)

# Відображення обробленого кадру у вікні з назвою "Processed Frame"

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду для виділення меж з різними порогами canny

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

# функція виділення меж canny

def canny\_edge\_detection(frame, t\_lower=250, t\_upper=250):

# Перетворення зображення у відтінки сірого

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

# Використання алгоритму Canny для знаходження країв у зображенні

frame\_edges = cv2.Canny(gray\_image, t\_lower, t\_upper)

return frame\_edges

# Відкриття відеофайлу для читання за допомогою методу VideoCapture з вказанням шляху до файлу

cap = cv2.VideoCapture("course-work/videos/Video Test 2K Samsung.mp4")

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Виділення меж методом Canny

frame = canny\_edge\_detection(frame)

# Відображення обробленого кадру у вікні з назвою "Processed Frame"

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду для фільтру з ефектом зсуву з різними параметрами

import cv2 # Імпорт бібліотеки OpenCV для обробки відео

import numpy as np # Імпорт бібліотеки numpy для роботи з матрицями

# Функція фільтрації з ефектом зсуву

def motion\_blur(frame, kernel\_size=5):

# Створення пустого ядра для фільтра зсуву

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

# Встановлення одиниць у рядках ядра для створення зсуву

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

# Нормалізація ядра для отримання середнього значення

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

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

# Відкриття відеофайлу для читання за допомогою методу VideoCapture з вказанням шляху до файлу

cap = cv2.VideoCapture("course-work/videos/Video Test 2K Samsung.mp4")

# Безкінечний цикл для читання кадрів з відеофайлу

while True:

# Читання кадру з відеофайлу, ret - змінна, що показує, чи вдалося прочитати кадр

ret, frame = cap.read()

# Перевірка, чи вдалося прочитати кадр; якщо ні, вихід із циклу

if not ret:

break

# Застосовування фільтру з ефектом зсуву

frame = motion\_blur(frame)

# Відображення обробленого кадру у вікні з назвою "Processed Frame"

cv2.imshow("Processed Frame", frame)

# Очікування натискання клавіші 'q' для виходу з циклу

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Звільнення ресурсів, пов'язаних із відеофайлом

cap.release()

# Закриття всіх вікон OpenCV

cv2.destroyAllWindows()

Лістинг коду класу MediaTools

import cv2

import tkinter as tk

from tkinter import filedialog, messagebox

import numpy as np

class MediaTools:

def open(self):

self.\_\_loadGui()

# Константи (можливі варіанти обробки відео)

VERTICAL\_SHIFT = 1

EDGE\_DETECTION\_CANNY = 2

COLOR\_SPACE\_XYZ = 3

SHIFT\_EFFECT\_FILTER = 4

# Метод вертикального зсуву зображення

def \_\_vertical\_shift(self, frame, top=200):

# Отримання розмірів кадру (висоти та ширини)

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

# Створення матриці зсуву зображення за допомогою зміщення у вертикальному напрямку

translation\_matrix = np.float32([[1, 0, 0], [0, 1, top]])

# Застосування зсуву до зображення за допомогою афінного перетворення

shifted\_frame = cv2.warpAffine(frame, translation\_matrix, (num\_cols, num\_rows))

return shifted\_frame

# Метод виділення меж canny

def \_\_canny\_edge\_detection(self, frame, t\_lower=250, t\_upper=250):

# Перетворення зображення у відтінки сірого

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

# Використання алгоритму Canny для знаходження країв у зображенні

frame\_edges = cv2.Canny(gray\_image, t\_lower, t\_upper)

return frame\_edges

# Метод зміни кольорового простору на XYZ

def \_\_to\_xyz\_colorspace(self, frame):

return cv2.cvtColor(frame, cv2.COLOR\_BGR2XYZ)

# Метод фільтрації з ефектом зсуву

def \_\_motion\_blur(self, frame, kernel\_size=5):

# Створення пустого ядра для фільтра зсуву

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

# Встановлення одиниць у рядках ядра для створення зсуву

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

# Нормалізація ядра для отримання середнього значення

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

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

# Метод завантаження графічного інтерфейсу

def \_\_loadGui(self):

self.root = tk.Tk()

self.root.title("Media Tools")

# Обирання варіацій обробки відео

self.tool\_selection\_frame = tk.Frame(self.root)

self.tool\_selection\_frame.pack(pady=10)

self.tools\_var = tk.IntVar()

self.video\_file\_path = ""

tk.Checkbutton(self.tool\_selection\_frame, text="Video Shift", variable=self.tools\_var, onvalue=self.VERTICAL\_SHIFT).grid(row=0, sticky="w")

tk.Checkbutton(self.tool\_selection\_frame, text="Edge Detection (Canny)", variable=self.tools\_var, onvalue=self.EDGE\_DETECTION\_CANNY).grid(row=1, sticky="w")

tk.Checkbutton(self.tool\_selection\_frame, text="Color Space (XYZ)", variable=self.tools\_var, onvalue=self.COLOR\_SPACE\_XYZ).grid(row=2, sticky="w")

tk.Checkbutton(self.tool\_selection\_frame, text="Shift Effect Filter", variable=self.tools\_var, onvalue=self.SHIFT\_EFFECT\_FILTER).grid(row=3, sticky="w")

# Обирання джерела відео

self.video\_source\_frame = tk.Frame(self.root)

self.video\_source\_frame.pack(pady=10)

self.video\_source\_var = tk.StringVar()

self.video\_source\_var.set("File")

tk.Radiobutton(self.video\_source\_frame, text="Webcam", variable=self.video\_source\_var, value="Webcam").grid(row=0, sticky="w")

tk.Radiobutton(self.video\_source\_frame, text="File", variable=self.video\_source\_var, value="File").grid(row=1, sticky="w")

# Обирання шляху відео

tk.Button(self.root, text="Select Video File", command=self.\_\_set\_video\_file\_path).pack(pady=10)

# Запуск відео

tk.Button(self.root, text="Start Processing", command=self.\_\_start\_processing).pack(pady=10)

self.root.mainloop()

# Метод встановлення шляху до відеу за допомогою діалогового вікна обирання файлу

def \_\_set\_video\_file\_path(self):

file\_path = filedialog.askopenfilename(filetypes=[("Video files", "\*.mp4 \*.avi")])

if file\_path:

self.video\_file\_path = file\_path

# Метод обробки та показу відео

def \_\_start\_processing(self):

selected\_tool = self.tools\_var.get()

video\_source = self.video\_source\_var.get()

if video\_source == "File":

if not self.video\_file\_path:

messagebox.showerror("Error", "Please select a video file.")

return

cap = cv2.VideoCapture(self.video\_file\_path)

elif video\_source == "Webcam":

cap = cv2.VideoCapture(0)

else:

messagebox.showerror("Error", "Invalid video source.")

return

while True:

ret, frame = cap.read()

if not ret:

break

# Застосування обраного інструменту обробки відео

if selected\_tool == self.VERTICAL\_SHIFT:

frame = self.\_\_vertical\_shift(frame)

pass

elif selected\_tool == self.EDGE\_DETECTION\_CANNY:

frame = self.\_\_canny\_edge\_detection(frame)

pass

elif selected\_tool == self.COLOR\_SPACE\_XYZ:

frame = self.\_\_to\_xyz\_colorspace(frame)

pass

elif selected\_tool == self.SHIFT\_EFFECT\_FILTER:

frame = self.\_\_motion\_blur(frame)

pass

cv2.imshow("Processed Frame", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

media\_tools = MediaTools()

media\_tools.open()

# Додаток Б

Екранні форми виконання

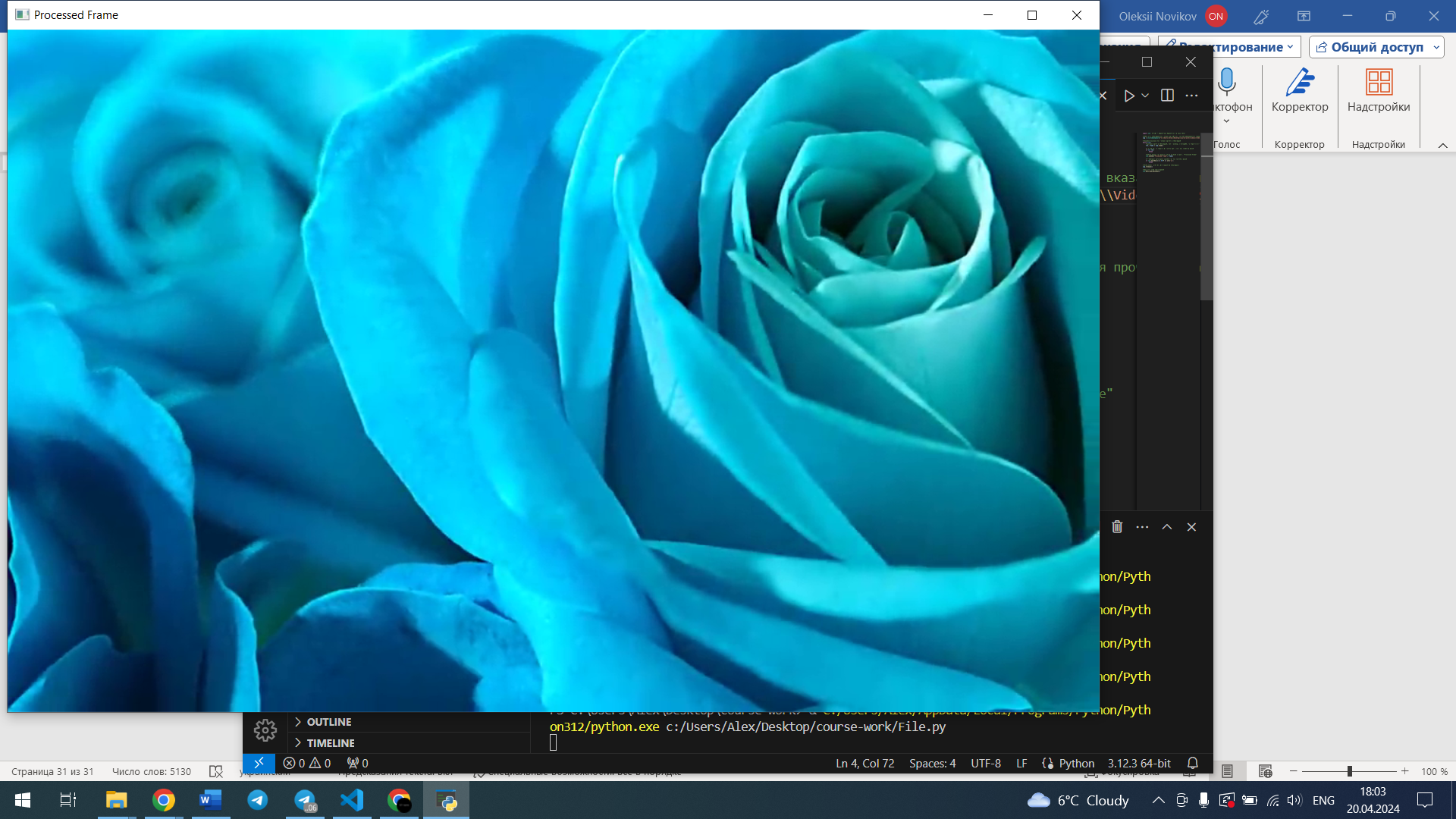


Рисунок Б.1 – Зчитування з файлу

Изображение выглядит как текст, Человеческое лицо, снимок экрана, человек

Автоматически созданное описание

Рисунок Б.2 – Зчитування з веб камери

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок Б.3 – Вертикальний зсув

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок Б.4 – Зміна кольорового простору на XYZ

Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок Б.5 – Виділення меж з різними порогами canny

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок Б.6 – Фільтр з ефектом зсуву з різними параметрами

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок Б.7 – Використання класу MediaTools