УДК 004.4
DOI https://doi.org/10.32782/tnv-tech.2025.1.15

АНАЛІЗ ФУНКЦІОНАЛЬНО-ОПТИМІЗАЦІЙНИХ МЕТОДІВ ПРИ РОБОТІ З ВІДЕО У OPENCV

Полоневич Д. В. *– здобувач Державного університету «Житомирська політехніка»*
ORCID ID: 0009-0006-2811-1994

Вакалюк Т. А. *– доктор педагогічних наук, професор, завідувач кафедри інженерії програмного забезпечення Державного університету «Житомирська політехніка»*
ORCID ID: 0000-0001-6825-4697

Стаття присвячена аналізу сучасних методів обробки відео з використанням бібліотеки функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення OpenCV. Дослідження зосереджене на аналізі функціональних можливостях бібліотеки та можливих способах оптимізації обробки відеоконтенту.

У роботі висвітлюється актуальність використання OpenCV при роботі з відео, зокрема розглядається модульна структура бібліотеки, до якої входять модулі core, imgproc, video, highgui, features2d, calib3d та stitching. Основна увага в роботі приділяється методам оптимізації, зокрема використанню пірамід зображень pyrUp та pyrDown, змішуванню зображень за допомогою методу addWeighted, конвертації кольорів у градації сірого з використанням методу cv2.cvtColor, швидкому переміщенню по кадрах вихідного відеофайлу через метод Seek та обрізанню лишніх відеофрагментів через метод Clip.

Ключовими результатами дослідження є аналіз ефективності різних функціонально-оптимізаційних методів, особливо в контексті обробки зображень, їх змішування та створення ефектів переходу між відеокадрами, конвертації в градації сірого, навігації у відеопотоці без значних затримок, а також обрізання зайвих областей відеофайлів. Ці методи довели свою ефективність, підвищуючи швидкодію обробки відео.

Дослідження підтверджує високу ефективність використання OpenCV для задач відеообробки та оптимізації. Впровадження функціонально-оптимізаційних методів дозволяє значно покращити продуктивність відеоредакторів та знизити навантаження на обчислювальні ресурси.

Подальші дослідження можуть бути спрямовані на інтеграцію OpenCV з технологіями машинного навчання та штучного інтелекту для автоматизованої обробки відео. Така інтеграція дозволить створювати більш розумні та ефективні процеси обробки відео, відкриваючи нові можливості для реального часу та складних відеоефектів.

Ключові слова: *обробка відео, OpenCV, функціонально-оптимізаційні методи, піраміди зображень, продуктивність відеоредакторів.*

Polonevych D. V., Vakaliuk T. A. The analysis of functional optimization methods for video processing in OpenCV

The article is devoted to analysing modern video processing methods using the library of functions and algorithms for computer vision, image processing and general-purpose numerical algorithms OpenCV. The study analyses the library's functionality and possible ways to optimise video content processing.

The paper highlights the relevance of using OpenCV when working with video; in particular, the modular structure of the library, which includes the core, imgproc, video, highgui, features2d, calib3d and stitching modules, is considered. The primary attention in the paper is paid to optimisation methods, in particular, the use of image pyramids pyrUp and pyrDown, image blending using the addWeighted method, colour conversion to grayscale using the cv2.cvtColor method, fast movement through the frames of the source video file using the Seek method and trimming excess video fragments using the Clip method.

The key results of the study are an analysis of the effectiveness of various functional optimisation methods, especially in the context of image processing, their mixing and creating

transition effects between video frames, conversion to grayscale, navigation in the video stream without significant delays, as well as cropping unnecessary areas of video files. These methods have proven their effectiveness, increasing the speed of video processing.

The study confirms the high effectiveness of using OpenCV for video processing and optimisation tasks. Implementing functional optimisation methods allows video editors to significantly improve their performance and reduce the load on computing resources.

Further research can aim to integrate OpenCV with machine learning and artificial intelligence technologies for automated video processing. Such integration will allow for more intelligent and efficient video processing processes, opening up new opportunities for real-time and complex video effects.

Key words: *video processing, OpenCV, functional optimization methods, image pyramids, video editor performance.*

Актуальність. Сьогодні, коли відеоконтент стає одним із основних способів комунікації та передачі інформації, ефективність та зручність інструментів для його редагування набувають особливого значення [1]. Однією з головних проблем є оптимізація швидкості обробки відео, зокрема, забезпечення швидкої та плавної роботи при редагуванні високоякісного відео. Робота з відеоефектами, трансформування зображення, колажування та інші графічні прийоми, що напряму працюють з відео потоком, є доволі вимогливими функціональними можливостями у ресурсах для машини кожного користувача [2]. З огляду на ці проблеми, необхідно вдосконалювати методи обробки відео для зменшення затримок і підвищення ефективності програмного забезпечення. Перспективними напрямками є використання мультипоточності, апаратного прискорення та адаптивних алгоритмів. Важливо інтегрувати нові технології, наприклад, в існуючі відеоредактори.

Оптимізація процесів покращує продуктивність, знижує витрати часу та ресурсів, що важливо для відеомонтажерів. Водночас, для розробників важливо зберігати зручність і швидкість інтерфейсу.

Використання ефективних методів, таких як розподіл навантаження на ядра процесора або графічні процесори, допомагає швидко обробляти великі відеофайли з високою якістю. Подальші дослідження повинні орієнтуватися на адаптацію до різних платформ та інтеграцію штучного інтелекту для оптимізації процесів редагування.

Огляд останніх досліджень та публікацій. Питання роботи з обробкою вихідних зображень за допомогою OpenCV є досить актуальним у науковій літературі. Так, у [3] обґрунтоване використання OpenCV через здатність забезпечувати ефективну обробку зображень у реальному часі для задач, пов'язаних з комп'ютерним зором, таких як виявлення кольорів і осіб [3]. Автори досліджують використання відеообробки в Unity для створення симуляційної платформи моніторингу людей за допомогою безпілотників. Оскільки Unity спочатку розроблявся для ігор, автори вивчають можливості інтеграції OpenCV через два підходи: використання DLL на C++ та зовнішнього модуля на Python із MQTT-комунікацією [3].

У [4] описано систему автоматичного розпізнавання заповненості медичних декларацій пасажирів у митному контролі, яка поєднує алгоритм опорних векторів (SVM) і можливості комп'ютерного зору OpenCV. Автори вважають, що використання OpenCV є ключовим, оскільки ця бібліотека дозволяє ефективно обробляти зображення документів, виділяти необхідні елементи та аналізувати їхній вміст. Завдяки цьому система може швидко й точно визначати, чи заповнені всі обов'язкові поля, що не тільки спрощує роботу митників, а й значно прискорює процес перевірки для пасажирів. У контексті контролю поширення епідемій це особливо важливо, адже мінімізує фізичний контакт і знижує ризики для персоналу та подорожуючих [4].

У [5] описано створення платформи розпізнавання тексту (OCR), заснованої на OpenCV, для автомобільної індустрії. Автори використовують алгоритми OpenCV для аналізу та обробки зображень, виділення номерних знаків і символів, які потім маркуються та використовуються для тренування моделей SVM і ANN. Окрім розпізнавання номерних знаків, система може також ідентифікувати VIN-коди, номери водійських посвідчень та інші текстові дані. Платформа розроблена на Java з використанням Spring Boot, що забезпечує сумісність з іншими програмними рішеннями галузі. OpenCV тут відіграє критичну роль, оскільки дозволяє ефективно обробляти великі обсяги зображень, витягувати необхідні символи та підвищувати точність розпізнавання [5].

Інші автори розглядають оптимізацію алгоритмів відстеження об'єктів у бібліотеці OpenCV, зосереджуючись на використанні паралельних обчислень на GPU [6]. Науковці досліджують алгоритми виявлення об'єктів за допомогою Haar/LBP-класифікаторів та оптичного потоку Farneback, виявляючи основні фактори, що впливають на їхню продуктивність. Вони пропонують удосконалення, які дозволяють більш ефективно використовувати ресурси GPU, зокрема оптимізують завантаження обчислювальних блоків і зменшують накладні витрати, пов'язані з викликами OpenCL API. Запропоновані методи випробувані на прискореному процесорному блоці (APU) та дискретному GPU, що дозволило досягти значного приросту продуктивності: швидкість роботи алгоритмів зросла до 73% і 86% відповідно, а усунення зайвих викликів ядер дало додатковий приріст у 10%. Автори також аналізують вплив оптимізацій на рівні апаратного забезпечення, доводячи ефективність своїх підходів [6].

Виклад основного матеріалу. OpenCV (англ. Open Source Computer Vision Library) – це бібліотека з відкритим кодом для комп'ютерного зору та обробки зображень і відео. Вона надає набір інструментів для вирішення різноманітних задач у цих сферах і активно використовується в багатьох галузях, від науки та медицини до розваг і робототехніки. Важливою особливістю OpenCV є її багаторівнева архітектура, яка дозволяє розробникам використовувати різні модулі бібліотеки для досягнення бажаних результатів, забезпечуючи високу продуктивність та гнучкість у роботі. Розберемо модульну структуру бібліотеки (див. рис. 1).

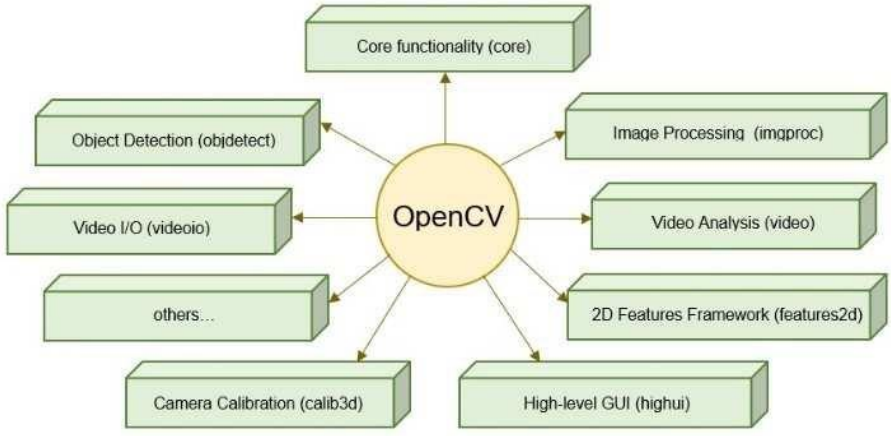

Рис. 1. Модульна структура бібліотеки OpenCV

Архітектура OpenCV побудована таким чином, щоб забезпечити зручність у використанні та можливість масштабування. Вона включає декілька основних модулів, кожен з яких відповідає за конкретну задачу. Наприклад, модуль *core* містить базові функції для роботи з матрицями (які є основною структурою даних в OpenCV), обробки зображень і виконання математичних операцій. Це основа бібліотеки, на якій базуються всі інші модулі [7]. Модуль *imgproc* відповідає за обробку зображень: фільтрацію, зміну розміру, контрастність, обертання, вирізання, корекцію кольору та багато іншого. Ці функції є надзвичайно корисними для створення складних відео та графічних ефектів [7].

Ще одним важливим модулем є *video*, який забезпечує можливість роботи з відеофайлами та виконання операцій у реальному часі. Завдяки цьому модулю можна реалізувати такі функції, як детекція руху, стабілізація відео та відстеження об'єктів у відеопотоці. Це відкриває можливості для розробки відеоредакторів, здатних обробляти та аналізувати відео в реальному часі, що є критично важливим для створення інтерактивних програм і систем безпеки [7].

Модуль *highgui* відповідає за графічний інтерфейс, забезпечуючи виведення зображень та відео на екран, а також взаємодію з користувачем. Він дозволяє створювати прості вікна для демонстрації результатів обробки, а також інтегрувати інтерфейси для управління програмами. Це дає змогу швидко розробляти інструменти для візуалізації оброблених даних [7].

Останні три модулі по значимості це *features2d*, спеціалізується на виявленні та порівнянні ключових точок на зображеннях. Це надзвичайно корисно для завдань, таких як розпізнавання облич, пошук об'єктів на зображеннях або відео, а також для реалізації функцій стабілізації зображення. Модулі calib3d і stitching використовуються для калібрування камер і створення панорам із кількох зображень [7].

Перейдемо до аналізу функціональних можливостей модулів у реальних практичних задачах. Функція «Зміна розміру вихідного файлу» у відеоредакторі дозволяє користувачам змінювати розміри відео під час редагування. Ця функція є незамінною для адаптації відео до різних платформ та пристроїв. Користувачі можуть легко налаштовувати ширину та висоту відео, зберігаючи або змінюючи пропорції за потреби. Швидкодія цієї функції є критично важливою для створення якісного прототипу, тому далі розглянемо готове рішення.

Використання функцій OpenCV *pyrUp()* і *pyrDown()* дозволяє зменшити або збільшити роздільну здатність певного зображення. Зазвичай нам потрібно перетворити зображення на розмір, відмінний від оригінального. Для цього можливі два варіанти: збільшити розмір зображення або зменшити його. Хоча в OpenCV існує функція геометричної трансформації, яка буквально змінює розмір зображення, у цьому розділі ми спочатку проаналізуємо використання пірамід зображень, які широко застосовуються у величезному діапазоні програм по роботі з відео.

Піраміда зображень – це набір зображень, які утворюються з одного оригінального зображення, а дискретизація послідовно знижується, доки не буде досягнуто бажаної точки зупинки. Існує два поширених типи пірамід зображень: піраміда Гауса, що використовується для зменшення дискретизації зображень, та піраміда Лапласа, яка використовується для реконструкції зображення з підвищеною дискретизацією із зображення, розташованого нижче в піраміді (з меншою роздільною здатністю). Уявляємо піраміду як набір шарів, у яких чим вищий шар, тим менший розмір (див. рис. 2).

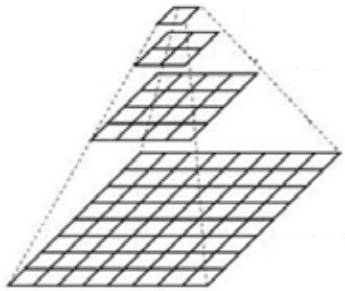

Рис. 2. Представлення піраміди як набір шарів

Кожен шар нумерується знизу вгору, тому шар $(i + 1)$ (позначається як G_{i+1} менший за шар $I(Gi)$. Щоб створити шар $(i + 1)$ у піраміді Гауса, робимо згортання з ядром Гауса (див. рис. 3). Далі видаляємо усі парні рядки та стовпці. Легко помітити, що отримане зображення буде рівно на чверть площі свого попередника. Ітерація цього процесу на вхідному зображенні (оригінальне зображення) створює всю піраміду. Наведена вище процедура була корисною для зменшення дискретизації зображення.

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Рис. 3. Згортання з ядром Гауса

Якщо потрібно зробити його більшим, по-перше, збільшуємо розмір зображення у пропорції двічі до оригіналу в кожному вимірі, додавши нові парні рядки а далі виконуємо згортку з тим самим ядром, що показано вище (помноженим на 4), щоб наблизити значення «відсутніх пікселів». Прикладне демонстрування виконання ефекту на рис. 4.

Рис. 4. Виконання методу pyrDown()

Ці дві процедури (зменшення та збільшення) реалізуються функціями OpenCV *pyrUp()* і *pyrDown()* [7].

Наступний метод, що активно використовуватиметься у аналізі та розробці це лінійне об'єднання зображень. Метод *addWeighted()* в OpenCV використовується для змішування двох зображень або відео з заданими ваговими коефіцієнтами. Це основний інструмент для створення ефектів плавного переходу (англ. cross dissolve), де одне зображення поступово зникає, а інше з'являється. Основна формула для методу *addWeighted()* виглядає наступним чином:

$$g(x) = \alpha \cdot f0(x) + \beta \cdot f1(x) + \gamma ,$$

де:

– g(x) – результат зображення,
– $f_0(x)$ і $f_1(x)$ – два вхідних зображення,
– α і β – вагові коефіцієнти для кожного зображення,
– γ – додаткова константа, яка додається до результату (опціонально).

Цей метод дозволяє комбінувати два зображення за допомогою вагових коефіцієнтів α і β, які визначають вплив кожного зображення на кінцевий результат. Змінюючи значення α і β, можна керувати прозорістю та інтенсивністю кожного зображення у кінцевому результаті.

У практиці метод *addWeighted()* часто використовується для створення ефектів змішування (рис. 5), таких як плавний перехід між двома кадрами відео або зображеннями в слайд-шоу. Це досягається шляхом поступової зміни вагових коефіцієнтів α і β з 0 до 1 (або навпаки), як було сказано раніше, що призводить до поступового зникнення одного зображення та появи іншого.

Рис. 5. Виконання методу addWeighted ()

Наступний метод стосується корекції кольору зображень. Процес переведення зображення з інших колірних форматів, таких як RGB, CMYK або HSV, у відтінки сірого називається градацією сірого. Це перетворення дозволяє зображенню варіюватися від чисто чорного до чисто білого.

Відтінки сірого мають кілька важливих переваг. По-перше, вони зменшують обсяг даних, оскільки в зображеннях RGB є три кольорові канали, а в градаціях сірого – лише один. Це спрощує обробку інформації. По-друге, зменшення складності моделі дозволяє нейронним мережам легше працювати із зображеннями.

Наприклад, для зображень RGB розміром 10 на 10 пікселів потрібно 300 вхідних вузлів, тоді як для зображень у відтінках сірого – лише 100.

Функціональність цього методу можна продемонструвати за допомогою функції cv2.cvtColor() (рис. 6). Наступний метод присвячений функціоналу «під коробкою», а саме можливість перемотування відео до зазначеного кадру.

Рис. 6. Виконання методу cv2.cvtColor()

Доволі звична дія при роботі з редактором, але саме у таких речах і відчутна оптимізація та швидкодія системи, особливо якщо вихідний файл значної довжини та якості.

Метод *Seek()* служить для швидкого переходу до конкретного місця (кадру) у відео. Цей інструмент є важливим для випадкового доступу, редагування та навігації в медіафайлах. Принцип дії методу полягає в установці поточної позиції для читання або відтворення у відеофайлі. Це дозволяє миттєво перейти до необхідного кадру без потреби переглядати всі попередні кадри послідовно.

У OpenCV для роботи з відео використовується клас *VideoCapture*, який містить метод *set()*. Цей метод дозволяє змінювати різні властивості відео, включаючи позицію кадру. Метод *set()* приймає два параметри: *propId* та *value*. *PropId* є ідентифікатором властивості, яку потрібно змінити; для переходу до конкретного кадру використовується cv2.CAP_PROP_POS_FRAMES. *Value*, в свою чергу, є новим значенням для цієї властивості, а для cv2.CAP_PROP_POS_FRAMES це номер кадру, до якого потрібно переміститися.

Останній метод стосується обрізання відеокадрів – поширеної задачі для користувачів. Наприклад, є фрагмент відео, в якому водяна мітка заважає сприйняттю матеріалу і її потрібно або обрізати або замалювати. Для вирішення цієї проблеми був розроблений метод *clip()*, що виконає перший функціонал.

Метод *clip()* використовується для обрізання частини зображення або відеокадру. Це дуже корисний інструмент для виділення конкретних областей, які можуть бути важливими для подальшої обробки або аналізу. Принцип його роботи полягає в тому, що користувач визначає прямокутну область у зображенні або відеокадрі і зберігає лише цю частину, видаляючи все, що знаходиться за межами. Це особливо корисно для фокусування на певних об'єктах чи ділянках зображення, а також для зменшення розміру зображення перед обробкою або створення спеціальних ефектів.

У OpenCV метод *clip()* реалізується через зрізи масиву, що дозволяє легко виділити потрібну область зображення або кадру. Наприклад, можна вказати координати верхнього лівого і нижнього правого кута прямокутника, щоб отримати

бажану частину відео. Це робить процес обрізання простим і швидким, що є важливим аспектом при редагуванні відео та підготовці матеріалів для подальшого використання.

Висновки. Аналіз функціонально-оптимізаційних методів обробки відео в OpenCV підтвердив високу ефективність цієї бібліотеки для роботи з відео. Використання таких методів, як піраміди зображень, змішування кадрів, конвертація кольору та оптимізація навігації у відеопотоці, дозволяє значно підвищити швидкодію обробки та зменшити навантаження на обчислювальні ресурси. Отримані результати свідчать про перспективність подальшого вдосконалення алгоритмів оптимізації для відеоредакторів та систем реального часу.

Перспективним напрямом подальших досліджень є розробка адаптивних алгоритмів обробки відеопотоків у реальному часі з використанням глибоких нейронних мереж.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. The Critical Role of Video Editing Skills in Today's Digital World. URL: https://www.ecgprod.com/the-critical-role-of-video-editing-skills/

2. Video Editing Problems Caused by Software Failures. URL: https://diyvideoeditor.com/debugging-video-editing-software-problems/

3. Bustamante, A.; Belmonte, L.M.; Morales, R.; Pereira, A.; Fernández-Caballero, A. Video Processing from a Virtual Unmanned Aerial Vehicle: Comparing Two Approaches to Using OpenCV in Unity. Appl. Sci. 2022, 12, 5958. https://doi.org/10.3390/app12125958

4. Research on recognizing required items based on opencv and machine learning Qingyun Ma and Xubin Huang SHS Web Conf., 140 (2022) 01016 DOI: https://doi.org/10.1051/shsconf/202214001016

5. Cai, Jianjun, Erxin Sun, and Zongjuan Chen. «OCR Service Platform Based on OpenCV.» *Journal of Physics: Conference Series* 1883, no. 1 (April 1, 2021): 012043. http://dx.doi.org/10.1088/1742-6596/1883/1/012043.

6. Song, J.; Jeong, H.; Jeong, J. Performance Optimization of Object Tracking Algorithms in OpenCV on GPUs. Appl. Sci. 2022, 12, 7801. https://doi.org/10.3390/app12157801

7. OpenCV modules. URL: https://docs.opencv.org/4.x/index.html

REFERENCES:

1. The Critical Role of Video Editing Skills in Today's Digital World. URL: https://www.ecgprod.com/the-critical-role-of-video-editing-skills/

2. Video Editing Problems Caused by Software Failures. URL: https://diyvideoeditor.com/debugging-video-editing-software-problems/

3. Bustamante, A.; Belmonte, L.M.; Morales, R.; Pereira, A.; Fernández-Caballero, A. Video Processing from a Virtual Unmanned Aerial Vehicle: Comparing Two Approaches to Using OpenCV in Unity. Appl. Sci. 2022, 12, 5958. https://doi.org/10.3390/app12125958

4. Research on recognizing required items based on opencv and machine learning Qingyun Ma and Xubin Huang SHS Web Conf., 140 (2022) 01016 DOI: https://doi.org/10.1051/shsconf/202214001016

5. Cai, Jianjun, Erxin Sun, and Zongjuan Chen. «OCR Service Platform Based on OpenCV.» *Journal of Physics: Conference Series* 1883, no. 1 (April 1, 2021): 012043. http://dx.doi.org/10.1088/1742-6596/1883/1/012043.

6. Song, J.; Jeong, H.; Jeong, J. Performance Optimization of Object Tracking Algorithms in OpenCV on GPUs. Appl. Sci. 2022, 12, 7801. https://doi.org/10.3390/app12157801

7. OpenCV modules. URL: https://docs.opencv.org/4.x/index.html