Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ
Назва теми

КвРКІ 200104.20.01.03 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр. назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр. назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-1 _____ О. А. Василевич
Підпис. Ініціали, прізвище

Керівник _____ О. В. Іванов
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І. О. Засорнова
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«14» _червня_ 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"10" 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Василевичу Олексію Андрійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ
Керівник проекту (роботи) Іванов О.В., к.т.н., доцент.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Теоретичні основи досліджувальної проблеми
Аналіз існуючих рішень для оптимізації розмірів відеофайлів
Практична реалізація

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Алгоритм роботи кодека H.264
Алгоритм роботи кодувальника та декодера H.264
Алгоритм роботи програми

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Засорнова І. О., доцент кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	01.02.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.02.2024	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	23.02.2024	виконано
4	Робота над розділом 2 – аналіз існуючих рішень для вирішення задачі	29.03.2024	виконано
5	Робота над розділом 3 – проектування та розробка підсистеми для оптимізації параметрів відеозбереження	06.05.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	30.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ О. А. Василевич
Підпис Ініціали, прізвище

Керівник роботи _____ О. В. Іванов
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 200104.20.01.03 ПЗ	Пояснювальна записка	61		
			Графічні матеріали			
2		КвРКІ 200104.20.01.03 Е8	Алгоритм роботи	1		
			кодека Н.264			
3		КвРКІ 200104.20.01.03 Е8	Алгоритм роботи	1		
			кодувальника та декодера			
			Н.264			
4		КвРКІ 200104.20.01.03 Е8	Алгоритм роботи	1		
			програми			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 200104.20.01.03 ВП			
Розробив		Василевич		13.06	Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ	Літера	Аркуш	Аркушів
Перевір.		Іванов		13.06		У	1	1
						ХНУ, КІ2-20-1		
Н. контр.		Засорнова		13.06				
Затв.		Говорущенко		14.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ».

Автор роботи: Василевич Олексій Андрійович.

Керівник роботи: Іванов Олексій Валентинович.

Пояснювальна записка: 64 с., 26 рис., 7 табл., 3 дод., 40 джерел.

Графічна частина: 3 креслення.

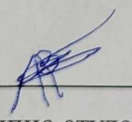
АЛГОРИТМ РОБОТИ КОДЕКА H.264, АЛГОРИТМ РОБОТИ КОДУВАЛЬНИКА ТА ДЕКОДЕРА H.264, АЛГОРИТМ РОБОТИ ПРОГРАМИ.

Метою дипломної роботи є визначення найкращий параметрів відеозображення для оптимізації та мінімізації дискового простору, а також розробка програми з можливістю використання цих параметрів.

Об’єктом дослідження процес оптимізації параметрів відеозображення для зменшення розміру відеофайлів без втрати якості.

Предметом дослідження є підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.



Підпис студента

30.05.2024

Дата

ЗМІСТ

Зм	Арк.	№докум.	Підпис	Дата	КвРКІ.200104.20.01.03 ПЗ		
Виконав		Василевич О. А.		13.06	Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ	Літера	Аркуш 2 / Аркушів 64
Перевір.		Іванов О. В.		13.06		У	
Н контр.		Засорнова І.О		13.06		ХНУ КІ2-20-1	
Затвер.		Говорущенко Т.О.		14.06			

ВСТУП

У сучасному світі перегляд різноманітного відеоконтенту став невід'ємною складовою нашого життя. Відеоконтент належить не лише до засобів вираження, але і є надзвичайно важливим об'єктом для збереження великої кількості інформації.

Доволі часто для збереження відеофайлів використовують різноманітні відеохостинги, які є зручними платформами, на яких власники відеоматеріалу можуть ділитися та зберігати свої творіння. Багато з них можна використовувати як відеосховище з невеликою кількістю переглядів, ділитися контентом лише з певною аудиторією, або викладати у загальний доступ.

З кожним днем у світі створюється все більше і більше нового контенту, який займає певне місце на дисковому просторі та займає як і місце на пристроях звичайних користувачів, так і на дисковому просторі великих серверів, тому проблема щодо оптимізації розмірів відеофайлів набуває великого значення.

Зменшення розміру цих відеофайлів є основним аспектом у вирішенні великої кількості задач:

1) ефективне зберігання даних: при зменшеному розмірі відеофайлу — зменшується об'єм використаного ним сховища, що сильно зменшує витрати на зберігання цих файлів та є особливо критичним чинником для великих сервісів із великою кількістю контенту;

2) економія пропускної здатності: з меншим розміром відеофайлів можна зекономити пропуску здатність мережі, що є особливо критичним чинником при перегляді відео онлайн для користувачів із обмеженим підключенням до інтернету;

3) покращення продуктивності під час відтворення: відтворення великих відеофайлів вимагає більше ресурсів процесора та оперативної пам'яті, тому менші за розміром файли будуть краще відтворюватися на пристроях з обмеженими ресурсами;

4) мобільна сумісність: із зменшенням розмірів відеофайлів покращується їхня мобільна сумісність, що у свою чергу, покращує їх сумісність з іншими пристроями та дозволяє зручно переглядати їх вміст [1-2].

Для вирішення проблеми оптимізації розмірів відеофайлів було розроблено багато різноманітних рішень, від зменшення параметрів якості відеофайлів, до використання різноманітних сучасних алгоритмів стиснення та використання нейромереж.

Великі відеохостинги, такі як YouTube, використовують різні технології, що саме і робить їх зручними та ефективними для користувачів.

Метою даної роботи є аналіз існуючих стратегів та алгоритмів та стратегій стиснення відеофайлів для визначення найкращої та найоптимальнішої з них для довготривалого збереження відеофайлів.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАЛЬНОЇ ПРОБЛЕМИ

1.1 Використання відеохостингів YouTube та Vimeo для збереження пізнавального відеоконтенту

Відеохостинги — це важливий інструмент для збереження та викладання пізнавального та навчального контенту. Вони можуть знадобитися та бути дуже ефективними для викладачів, тренерів, їх учнів та різних експертів, які створюють свій контент для обмеженої аудиторії та роблять фокус на якісному вивчені.

Основні переваги відеохостингів:

1) доступність: контент є доступним для перегляду в будь-який час та з будь-якого місця;

2) взаємодія з аудиторією: можливість взаємодіяти з аудиторія через лайки та коментарі, що полегшує обговорення тем;

3) аналітика: платформи мають інструменти для аналітики відео, що допомагає вам відстежувати взаємодію аудиторії з вашим контентом;

4) контроль доступу та приватність: відеохостинги надають різноманітні функції для контролю доступу до відео та його конфіденційності, такі як доступ до відео тільки за посиланням, а такий відеохостинг як Vimeo дозволяє встановити пароль на ваше відео [3].

YouTube та Vimeo є двома найбільш зручними платформами для зберігання відеоконтенту. Проте вони відрізняються один від одного (таблиця 1.1).

Таблиця 1.1 — Основні відмінності платформ YouTube та Vimeo [4]

Ознака для порівняння	YouTube	Vimeo
1	2	3
Сховище	Має необмежене сховище для усіх користувачів.	Залежить від тарифного плану: безкоштовний тариф пропонує 500МБ на тиждень, платні тарифи до 1ТБ.

Кінець таблиці 1.1

1	2	3
Якість відео	Підтримує потокове відео з якістю до 8К, з нижчим бітрейтом, ніж Vimeo.	Підтримує потокове відео з якістю до 8К, з вищим бітрейтом, ніж YouTube.
Конфіденційність	Пропонує менше можливостей конфіденційності, за замовчування усі відео є загальнодоступними, проте дозволяє встановити доступ за посиланням, або для конкретних користувачів.	Пропонує більше можливостей конфіденційності, ніж YouTube, наприклад, можливість захистити відео паролем.
Якість звуку	Підтримує високоякісні аудіоформати, такі як FLAC, Opus.	Підтримує високоякісні формати, такі як FLAC, Apple Lossles.
Спільнота	Має дуже велику та різноманітну спільноту, з широким спектром зацікавленості у контенті.	Має значно меншу, але більш активну спільноту.
Монетизація	Пропонує різні варіанти монетизації, такі як реклама, платне членство в каналах, суперчат.	Пропонує різноманітні варіанти монетизації, такі як плани підписки, банки з порадами, платне завантаження відео.
Аналітика	Пропонує безкоштовну та розширену статистику для усіх користувачів.	Пропонує базові метрики безкоштовно, але пропонує додаткові можливості за підпискою.
Підтримка та обслуговування клієнтів	В основному покладається на форуми та спільноти і пропонує лише обмежену підтримку.	Пропонує персоналізовану підтримку для платних користувачів.
Модерація контенту	Можна завантажувати майже будь-який контент, якщо в ньому немає відвертих та дуже жорстоких сцен.	Має суворішу модерацію контенту та не дозволяє завантажувати контент, який не має в собі сенсу та може бути спамом.

Для користувачів та авторів контенту може більше підходити та чи інша платформа, в залежності від їхніх цілей та потреб.

Vimeo може бути кращим для різноманітних підприємств через його конфіденційність, професійний підхід та аудиторії, яка орієнтована на якість, або для більш нішових авторів, які орієнтовані на якість свого контенту та пошуку зацікавленої аудиторії.

YouTube більше підходить для малого та середнього бізнесу, тому що завдяки великому охопленню він допоможе швидше привернути увагу аудиторії, або для авторів, які знімають більш загальний контент, орієнтований на ширшу аудиторію. Також YouTube потенційно може принести набагато більшу монетизацію.

YouTube може виявитися більш кращим вибором, навіть, для викладання навчального та пізнавального контенту та використання його як відеосховища з невеликою кількістю переглядів з кількох причин.

По-перше, YouTube має значно більшу користувацьку базу та має набагато ширше охоплення, що на перших етапах допоможе краще знайти свою аудиторію.

По-друге, YouTube пропонує набагато більше інструментів для аналітики та для зрозуміння своєї аудиторії, що може бути значно кориснішим для взаємодії зі своїми глядачами та для оптимізації контенту.

Узагальнюючи, YouTube може бути привабливішим вибором для викладання та зберігання великої кількості пізнавального контенту, якщо брати до уваги його велику аудиторію, різноманітні інструменти для аналізу цих відео та повністю безкоштовне сховище для зберігання.

1.2 Основи алгоритмів стиснення відеофайлів

Без використання відеокодеків не було б ні Ютубу, ні відеострімінгів, ні відеоконференцій: уся справа в тому, що відеофайл займає дуже багато місця.

Наприклад, якщо взяти 90 хвилинне відео у форматі 4К * 24 кадри в секунду — воно буде важити більше 3 тб (біля 36 гб на 1 хвилину відео). Сучасні відеокодеки дозволяють стиснути таке відео до 5 гб (приблизно у 650 раз). Для цього використовуються спеціальні алгоритми стиснення інформації [1, 3].

Існує 3 основних підходи, які використовуються для стиснення відеофайлів: внутрішньокадровий, міжкадровий та передбачення [5, 6].

При внутрішньокадровому стиснені алгоритм намагається позбавитися лишньої інформації всередині кожного кадру, тут використовується велика кількість методів по викиданню лишньої інформації, яку людське око не помічає. Наприклад, людське око чудово помічає перепади яскравості, але погано розрізняє перепади кольору, тому в кодеках активно використовується метод кольорової субдискретизації: береться відео із трьома каналами кольору RGB, по суті це 3 змінні, які визначають зображення, а це означає, що їх можна перетворити в інший формат, у якому 3 інші змінні YCrCb (Y – канал для яскравості, Cr та Cb – 2 канали для кольору) [7].

Приклад перетворення простору кольорів з RGB в YCrCb зображено на рисунку 1.1.

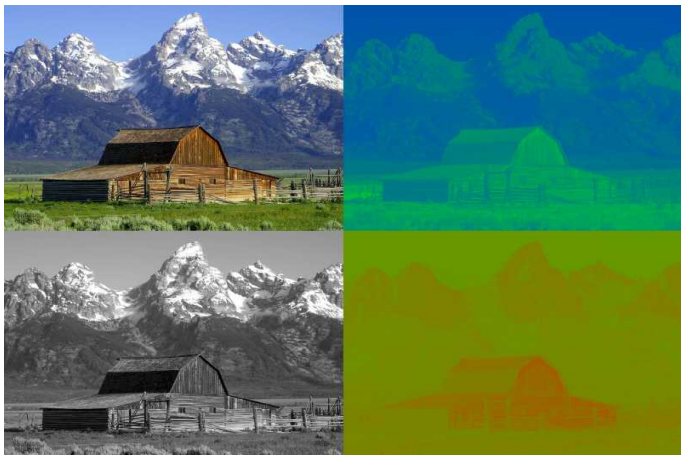


Рисунок 1.1 — Перетворення а простору кольорів з RGB в YCrCb

Після цього кольори та яскравості — це окремі сутності, можна зменшити канали кольору за дозволом в 2 рази та накласти все на канал яскравості. Після

цього різниця майже непомітна для ока, а кількість інформації майже в двічі зменшилась.

Такий варіант субдискретизації позначається трьома цифрами 4:2:0, що розшифровується як на 4 пікселі яскравості припадає лише 1 кольоровий, саме в цьому форматі в інтернеті відображається найбільша кількість відео.

Другий варіант для внутрішньокадрового стиснення використовує дискретне косинусне перетворення. Дискретне косинусне перетворення — це метод, що використовується для аналізу і обробки сигналів. Якщо розбити фрагмент зображення на маленькі блоки, все зображення можна представити як накладання косинусів один на одного, оскільки накладаючи блоки з різним ступенем прозорості один на одного — можна отримати будь-яке зображення.

Зображення прикладів блоків для дискретного косинусного перетворення зображено на рисунку 1.2.

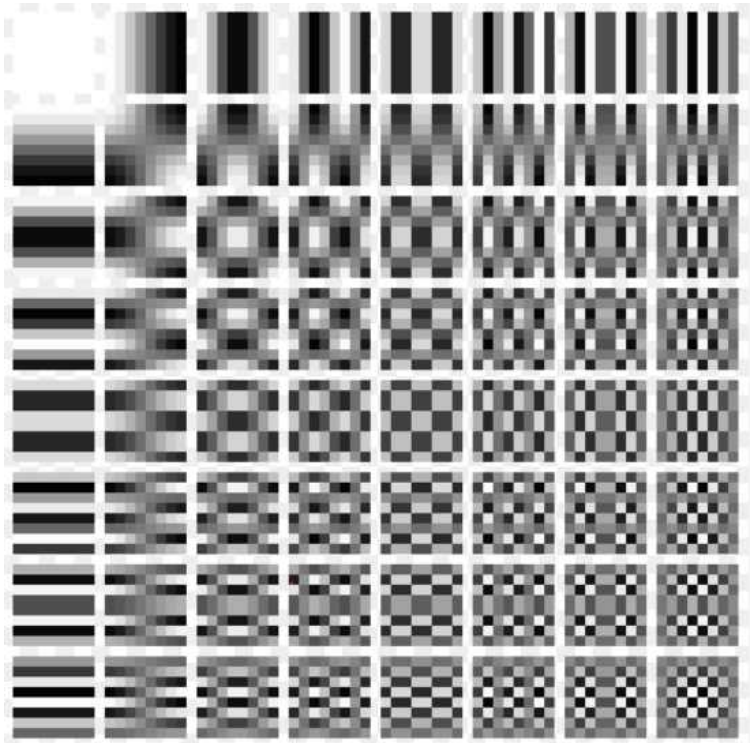


Рисунок 1.2 — Блоки для дискретного косинусного перетворення

Такий метод перетворює вхідні дані, зазвичай послідовності дійсних чисел, в суму косинусних функцій різних частот. Таке перетворення є важливим компонентом в алгоритмах стиснення даних з втратами. Графічне зображення можна розглядати як сукупність просторових хвиль, причому осі X і Y збігаються з шириною і висотою зображення, а по осі Z відкладається значення кольору відповідного пікселя зображення. Зміни у спектральному поданні можуть впливати на якість зображення, дозволяючи зберегти баланс між якістю та ступенем стиснення.

Внутрішньокадрові алгоритми дуже добре справляються зі своєю задачею та допомагають стискати відео в десятки раз.

Міжкадрові підходи до стиснення відеофайлів являються більш сучасними та допомагають стиснути відео навіть в декілька сотень раз.

В основі до міжкадрового підходу стиснення лежить доволі проста ідея: якщо дуже уважно подивитися на відеоролик, то можна побачити, що, зазвичай, різниця між двома сусідніми кадрами дуже маленька і виникає закономірне питання: “навіщо кодувати кожен окремий кадр?”.

Приклад двох схожих кадрів на відео із невеликим зміщенням об’єктів зображено на рисунку 1.3.

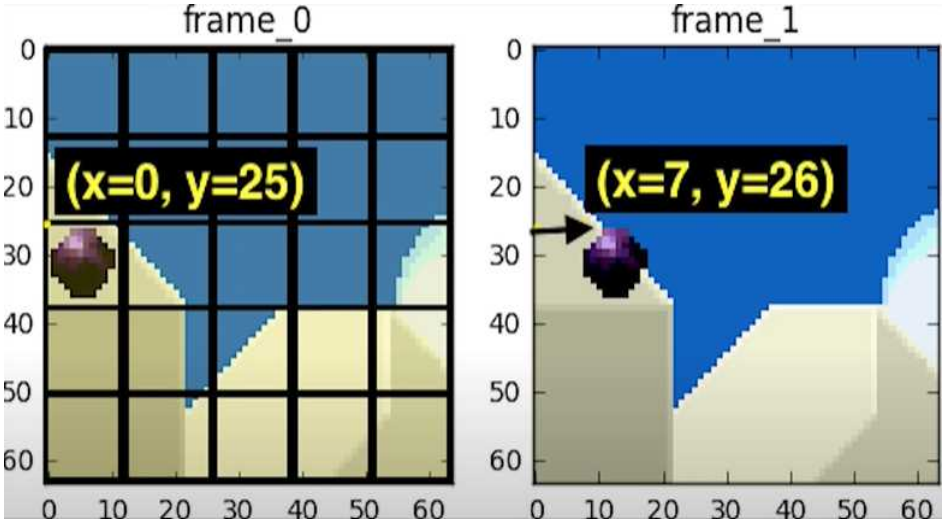


Рисунок 1.3 — Приклад двох схожих кадрів

При міжкадровому підході перший кадр називається ключовим кадром та повністю кодується, другий кадр називається проміжним та кодується на основі першого, тобто кодується лише та інформація, яка змінилася.

Якщо під час перегляду відео в інтернеті ви бачили різноманітні збої, коли відео стає зеленим або фіолетовим і часто малюється поверх іншого кадру — це якраз і є наслідки міжкадрового стиснення, просто ключовий кадр під час стрімінгу втратився.

Проте, найсучаснішими та найбільш економними з точки зору збереження дискового простору є саме алгоритми передбачення, які також працюють за доволі простим принципом.

Наприклад, якщо людині показати 2 кадри, на першому кадрі об’єкт буде зліва, на другому — справа, та запитати, що відбувалось між цими двома кадрами, то людина інтуїтивно домалює траєкторію руху. За точно таким принципом і працюють алгоритми передбачення, вже вшиті в відеокодеки. Прораховуючи вектори руху, вони припускають якими можуть бути проміжні кадри. Якщо припущення алгоритму співпадає з реальним рухом у кадрі, то виходить, що цей кадр можна просто не передавати, алгоритм сам його домалює. У випадку, коли щось не співпадає — можна не просто сам рух, а різницю між рухом у кадрі та тим, що показав алгоритм. Під час такого стиснення кадри поділяються на 3 типи — це I, P та B. Вони відрізняються за наступними характеристиками:

1) I-кадри — кадри з найменшим ступенем стиснення та для декодування не потребують додаткових кадрів;

2) P-кадри — кадри, які можуть використовувати дані з попередніх кадрів для розгортання та являються більш стиснутими, ніж I-кадри (кадри прямого прогнозування);

3) B-кадри — кадри, які можуть використовувати як і попередні, так і наступні кадри для посилання на дані, щоб отримати максимальний ступінь стиснення даних (кадри двонаправленого прогнозування)[8].

Послідовність відеокадрів, що складається з двох ключових кадрів, одного кадру прямого прогнозування та одного кадру двонаправленого прогнозування зображена на рисунку 1.4.

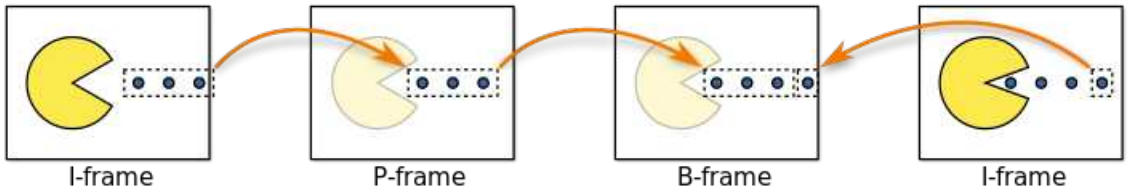


Рисунок 1.4 — Послідовність відеокадрів

Крім того, за допомогою алгоритмів передбачення, можна передбачати не тільки сусідні кадри, але і сусідні блоки всередині кадру, наприклад, якщо всередині певного блоку у нас одні сині пікселі, можна припустити, що і сусідній блок буде складатися з синіх пікселів.

Приклад такого передбачення сусідніх блоків та різницю між реальним та передбаченим значенням зображено на рисунку 1.5.

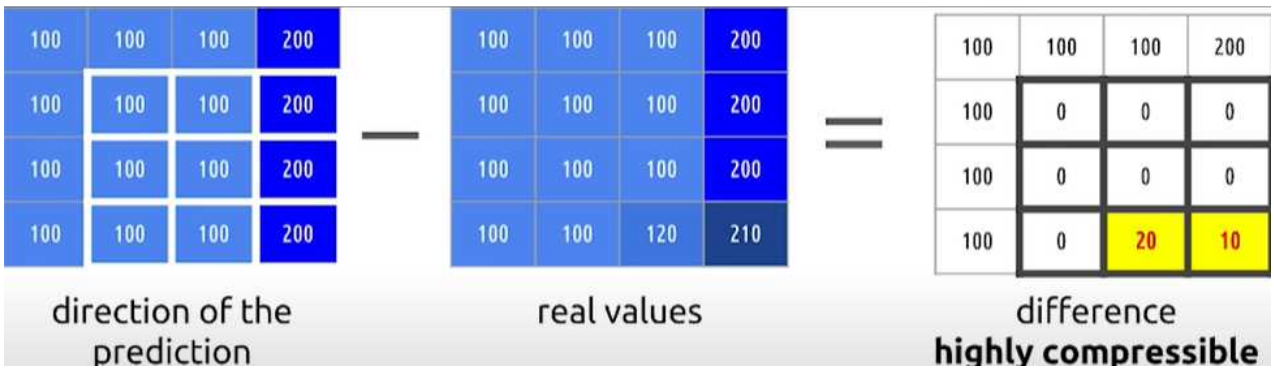


Рисунок 1.5 — Приклад передбачення кольору сусіднього блоку

Чим кращою буде модель передбачення, тим кращим буде кодек, який її використовує і саме в цих алгоритмах розробники відеокодеків зараз змагаються. Не останню роль у цих розробках зараз займають сучасні генеративні нейронні мережі.

1.3 Огляд бібліотеки FFmpeg

FFmpeg – це набір вільних бібліотек із відкритим вихідним кодом, завдяки яким можна проводити різні маніпуляції з відео та аудіо файлами: записувати, передавати, кодувати, конвертувати, фільтрувати, відтворювати, чи конвертувати їх у різноманітних форматах.

Проект був заснований у 2000 році. FFmpeg розроблений під операційну систему на основі GNU Linux, проте може бути скомпільованим на інших операційних системах, включаючи Windows та MacOS. Назва FFmpeg складається з: швидкого перемотування (англ. Fast forward – FF) та з назви групи стандартів відео (MPEG).

За весь період свого існування FFmpeg постійно розвивався та стабільно випускалися нові версії проекту. Сьогодні FFmpeg є невід'ємною складовою багатьох процесів обробки відео та аудіо через свою універсальність та потужність. Майже всі програми, які взаємодіють з мультимедійними файлами, незалежно від їх мети – чи то для створення, редагування, або відтворення файлів – використовують функціонал FFmpeg під капотом. Це свідчить про широке визнання його надзвичайної надійності та ефективності серед розробників усього світу. Завдяки своїй гнучкості, широким можливостям та високій продуктивності, FFmpeg став серйозним та важливим інструментом для обробки мультимедійних даних у різних галузях, від відео виробництва до розробки програмного забезпечення [1].

Основний функціонал FFmpeg включає в себе:

1) конвертація форматів: FFmpeg дозволяє конвертувати (транскодувати) відеофайли з одного формату в інший, шляхом налаштування різних форматів, таких як бітрейт, частота кадрів, кодувальник, розширення відео, підтримується дуже широкий список вхідних та вихідних форматів, завдяки чому можна з легкістю працювати з різними типами мультимедіа.

Загальний процес конвертації для кожного відеофайлу в FFmpeg можна описати завдяки наступної схеми поданої на рисунку 1.6.

Рисунок 1.6 — Загальний процес конвертації

Наприклад, перетворити відео формату .avi в .mp4 можна за допомогою простої команди у командному рядку:

ffmpeg -i input.avi output.mp4

Для того, щоб встановити бажаний бітрейт для вихідного файлу (наприклад 64 кбіт/с) необхідно прописати:

ffmpeg -i input.avi -b:v 64k -bufsize 64k output.mp4

Для того, щоб встановити бажану частоту кадрів для вихідного файлу (наприклад 24 кадри в секунду) необхідно прописати:

ffmpeg -i input.avi -r 24 output.mp4;

2) стиснення: бібліотека FFmpeg дозволяє стискати відеофайли, при цьому зберігаючи якість їх відтворення, для цього використовуються різноманітні кодеки. Найпопулярнішими та найкращими відеокодеками для стиснення відеофайлів без втрати якості є: H.264, H.285 та VP9.

3) редагування відеофайлів: FFmpeg дозволяє обрізати відеофайли, об'єднувати декілька файлів в один, вирізати певні фрагменти, додавати підписи та робити інші, більш складні, маніпуляції з відеофайлами [9-12].

Наприклад, вирізати певний фрагмент з відеофайлу можна за допомогою наступної програми:

ffmpeg -i input.mp4 -ss 00:05:30 -t 00:10:30 -c:v copy-c:a copy output.mp4

Вищенаведена команда візьме з вхідного відео та виріже 10 хвилин починаючи з 00:05:30 (5 хвилин 30 секунд), тобто вихідне відео буде тривати 10 хвилин з 00:05:30 до 00:15:30.

Для того, щоб об'єднати декілька відеофайлів в один, необхідно створити звичайний текстовий файл в папці з відеофайлами та перерахувати усі файли, які ми хочемо об'єднати. Наприклад створимо файл videos.txt та пропишемо у ньому всі файли:

file 'input1.mp4'

file 'input2.mp4'

file 'input3.mp4'

Далі необхідно відкрити командний рядок у заданій папці та прописати у ньому команду:

ffmpeg -f concat -i videos.txt -c copy output.mp4

На виході отримуємо файл output.mp4, який складається з усіх вищеперерахованих файлів [13].

FFmpeg є кросплатформеним та підтримується на різних операційних системах, включаючи Windows, Linux, macOS.

Загалом FFmpeg є невід'ємною частиною роботи з мультимедійними даними завдяки своїй потужності, гнучкості та повністю безкоштовній доступності, саме тому далі під час виконання даної роботи будемо спиратися на FFmpeg та різноманітні програми та бібліотеки створені на його основі.

1.4 Програми для запису відео з екрану

Сьогодні існує багато різних для запису екрану комп'ютера. Серед них є доволі прості програми та програми з розширеними можливостями. Найпопулярнішими з них є HyperCam та OBS.

HyperCam – це програма для запису екрану, створена компаніями Hyperionics та Solveig Multimadia в 1997. Вона призначена екрану у ОС Windows та зберігає усі дані в форматі .avi. Програма пропонує базовий набір для запису, редагування та може використовуватися для створення різноманітних навчальних відео, запису презентацій, геймплею. У HyperCam присутній доволі простий інтерфейс, що значно полегшує її у використанні з іншими конкурентами, проте має більш обмежений функціонал у порівнянні з більш потужними інструментами на ринку.

OBS (Open Broadcaster Software) – це відкрите програмне забезпечення для запису відео з екрану та стрімінгу в реальному часу, розроблена незалежним товариством розробників у 2012 році. У програмі є розширені можливості налаштування різних параметрів відео та редагування відео під час стрімінгу та запису. Однак програма є більш складною та має функції, які можуть бути просто непотрібними для багатьох користувачів [14].

Таблиця 1.2 — Основні відмінності HyperCam та OBS

Ознака	HyperCam	OBS
1	2	3
Ліцензія	Пропрієтарне програмне забезпечення, проте присутня безкоштовна пробна версія.	Вільна програмна з відкритим вихідним кодом.
Запис екрану та стрімінг	Програмна для запису екрану, яка дозволяє захоплювати відео та аудіо з екрану.	Дозволяє захоплювати відео та аудіо з екрану. Дозволяє проводити стрімінг у реальному часі.

Кінець таблиці 1.2

1	2	3
Інтерфейс	Простий інтерфейс, зрозумілий навіть початківцям.	Більш складний та просунутий інтерфейс.
Можливості редагування та запису	Пропонує базовий набір функцій для запису чи редагування відео.	Має більший вибір налаштувань та контролю якості відео під час запису.
Підтримка форматів	.avi, mp4	.mp4, .flv, .mov, .mkv, .ts, .m3u8
Підтримка відеокодеків	ffmpeg libx264, Mjpeg, x264	Nvenc, x264, QSV

Щодо вибору між цими двома програмами, необхідно враховувати потреби користувача. HyperCam – може бути більш кращим вибором для простих завдань та тим людям, які шукають просте рішення для запису екрану в ОС Windows та збереження цього відео у форматі .avi. OBS – буде кращим варіантом для тих, хто шукає інструменти для стрімінгу та професійного запису відео [15].

Після вибору програми для запису екрану, необхідно обрати формат, у якому буде записуватись відео.

Формат відео для запису з екрану залежить від мети використання відео, але найкращим, найпопулярнішим та найоптимальнішим варіантом буде саме формат mp4.

Формат mp4 є найпопулярнішим форматом відео, який підтримується практично всіма платформами та пристроями, з цим форматом найкраще використовуються різні сучасні кодеки, які саме забезпечують хороший баланс між якість відео та його стисненням.

Для кодування відео під час запису можна використовувати апаратні або програмні відеокодеки, які по різному впливають на запис відео, навантаження на процесор під час запису та використовуються у різних ситуаціях.

Таблиця 1.3 — Порівняння кодувальників

Ознака для порівняння	Апаратні кодеки	Програмні кодеки
1	2	3
Використання апаратури	Використовують спеціалізовані апаратні компоненти, такі як графічні процесори, або спеціальні чіпи для кодування та декодування	Використовують центральний процесор для кодування та декодування та не потребують спеціалізованого обладнання.
Швидкість	Завдяки використанню апаратури, апаратні кодеки можуть забезпечити високу швидкість і продуктивність під час кодування та декодування відео	Оскільки програмні кодеки покладаються на процесор, вони можуть бути менш продуктивними та швидкими в порівнянні з апаратними кодеками.
Енергоефективність	Апаратні кодеки, як правило, більш енергоефективні, оскільки вони використовують спеціалізовані апаратні компоненти, які споживають менше енергії порівняно з загальними процесорами.	Кодування та декодування відео за допомогою програмних кодеків може займати багато процесорних ресурсів, що може впливати на продуктивність системи.
Підтримка форматів	Апаратні кодеки часто підтримують обмежену кількість форматів кодування, таких як H.264 або HEVC (H.265), залежно від специфікацій обладнання.	Програмні кодеки зазвичай підтримують більшу кількість форматів та налаштувань, що надає більше гнучкості в процесі кодування та декодування
Основні переваги	Апаратне кодування зменшує навантаження на процесор та дозволяє швидше кодувати відео, зокрема у високій якості. Також, зазвичай, такі відео займають менше місця	Більш широкий спектр налаштувань, підвищена якість та гнучкість у порівнянні з апаратним кодуванням. Дозволяє точно контролювати параметри відео

Кінець таблиці 1.3

1	2	3
Основні недоліки	Залежить від моделі відеокарти або обладнання, що впливає на якість кодування та функціональність	Може сповільнити роботу системи, особливо при запису у високій роздільній здатності. Також, як правило, такі відео займають більше місця

Отже, апаратне кодування відео рекомендується використовувати для швидкого запису або потокового мовлення з мінімальним навантаженням на систему, особливо при високій роздільній здатності. Програмне кодування надає більшу гнучкість та точний контроль, тому підходить для ситуацій, коли вам потрібно максимально контролювати якість відео [16-17].

Найкращими варіантами для запису відео апаратним кодування є: NVENC (Nvidia Encoder) та QSV (Quik Sync Video).

Найкращими варіантами для запису відео програмним кодування є: libx264, x264, x265, Theora.

1.5 Висновки

У першому розділі роботи проведений ретельний аналіз принципів роботи основних підходів стиснення відеофайлів, які представляють собою основу кваліфікаційної роботи про стиснення відеофайлів, оскільки усі алгоритми стиснення базуються на цих підходах.

Була розглянута бібліотека FFmpeg та основні принципи роботи з цією бібліотекою. FFmpeg – вільна бібліотека з відкритим вихідним кодом, яка пропонує найзручніший та найякісніший доступ до оптимізації та редагування

відеофайлів, це підтверджується тим, що майже усі програми, які проводять певні дії з відеофайлами використовують саме FFmpeg.

У подальшому буде розглянуто декілька варіантів стиснення відеофайлів за допомогою програм, які працюють з FFmpeg та за допомогою FFmpeg напряму. Також уся робота буде базуватися на використанні бібліотек FFmpeg.

Крім цього, було проведено аналіз основного програмного забезпечення для запису відеофайлів з екрану та аналіз і порівняння кодувальників, які використовуються під час запису відео.

Додатково було проведено огляд платформ, які можна використовувати для збереження відеоінформації.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОПТИМІЗАЦІЇ РОЗМІРУ ВІДЕОФАЙЛІВ

2.1 Використання відеокодеків для мінімізації розміру відеофайлів

Сучасні відеокодеки у своїй роботі використовують усі 3 підходи для зменшення розміру відеофайлів, а саме: внутрішньокадровий, міжкадровий та передбачення.

Одним з найпопулярніших відеокодеків для стиснення відеофайлів на сьогоднішній день є H.264 або AVC (Advanced Video Coding). Цей кодек існує ще з 2003 року та все ще залишається популярним, в першу чергу саме через те, що він підтримується усіма браузерами, операційними системами та процесами. H.264 має невелику складність обчислення, відносно новіших кодеків для стиснення відеофайлів, та потребує значно менше обчислювальних ресурсів для кодування та декодування відео, ніж стандарти майбутніх поколінь.

H.264 чудово справляється зі стисненням відеофайлів у форматі HD та Full HD, проте він не підходить для відео з більшою якістю.

Все це відбувається тому, що максимальний розмір блоків, на які розбивається зображення під час обробки відео, у H.264 всього 16*16 пікселів і це дуже мало для 4K і тим паче 8K контенту. Наприклад, у кадрів може бути велика частина неба, яку можна розбити на великі сині блоки, але H.264 так не може, тому відео у таких великий дозволах будуть займати набагато більше місця.

Також до мінусів цього кодеку можна віднести те, що він не вміє відновлювати відео з проміжних кадрів, а це означає, що відео може часто розпадатись на блоки та погано підходить для стрімінгу в реальному часі та відеовикликів.

Основний алгоритм роботи H.264 включає наступні етапи:

1) розділення кадру: відеофайл розділяється на невеликі блоки, так звані макроблоки або частини;

2) прогнозування руху: для кожного блоку в кадрі проводиться прогнозування руху. Це означає, що алгоритм визначає, як частини зображення переміщуються від одного кадру до іншого;

3) компенсація руху: для покращення ефективності стиснення кодек використовує дані про рух, використовуючи компенсацію руху. Це дозволяє зменшити обсяг інформації, необхідної для кодування руху в кадрі;

4) просторова стиснення: після прогнозування руху та компенсації руху використовуються методи стиснення для зменшення розміру файлу, такі як дискретне косинусне перетворення та квантування;

5) кодування та стиснення: застосовуються алгоритми стиснення, такі як адаптивне кодування довжин блоків (AVC), для подальшого зменшення розміру файлу без втрати якості зображення;

6) передача даних: остаточно стиснені дані передаються або зберігаються у вигляді відеофайлу [11-12, 18-19].

Загалом, H.264 ще не втратив свою популярність та активно використовується для кодування різноманітних відео у форматі Full HD.

На заміну кодеку H.264 поступово приходить H.265 або HEVC (High Effeciency Video Coding). Перший стандарт H.265 з’явився у 2012 році.

Ключова відмінність між H.264 та H.265 полягає у тому, що в H.264 розмір блоків, на які розділяються кадри, зазвичай складає 16*16 пікселів, це означає, що кожен кадр розбивається на блоки розміром 16*16 пікселів для подальшого кодування та стиснення.

У H.265 розмір блоків може варіюватися від 4*4 до 64*64 пікселів. Це дозволяє краще адаптуватися до різних областей зображення, забезпечуючи більш точне кодування та стиснення. Завдяки більшому розміру блоків H.265 може ефективніше виявляти і використовувати просторову та часову локальність в зображенні, що дозволяє досягти кращої якості відео при меншому обсязі даних.

Отже, відмінність полягає у можливості H.265 використовувати блоки різного розміру, що дозволяє отримати більшу ефективність стиснення і кращу якість зображення порівняно з H.264.

Саме завдяки цій особливості кодек H.265 став основним стандартом для стиснення відеофайлів з великим дозволом.

Різницю між розмірами блоків, на які поділяється зображення у H.264 та H.265 зображено на рисунку 2.1.

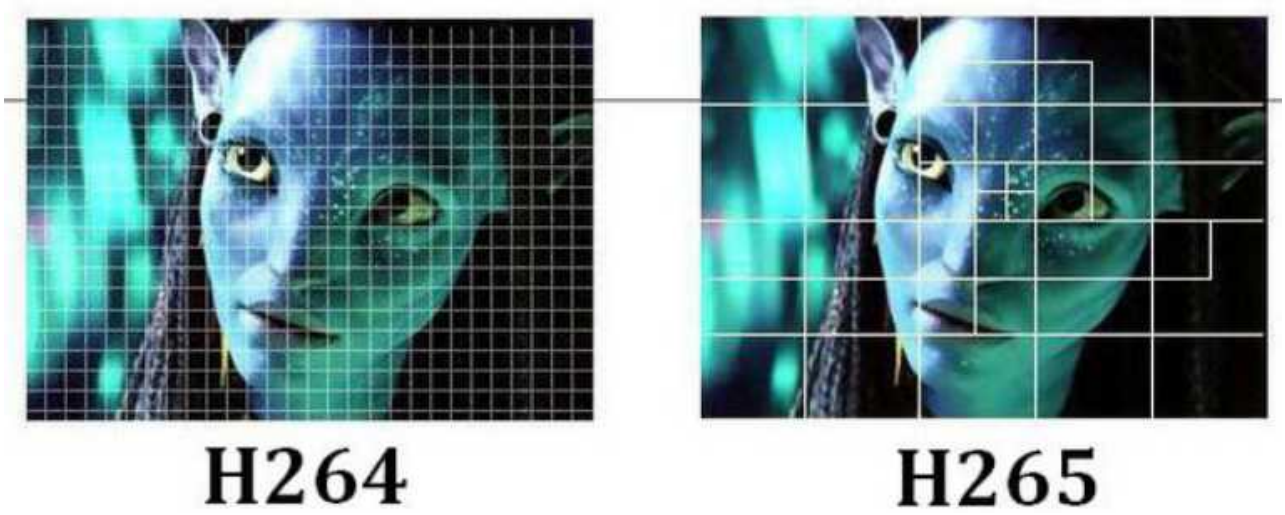


Рисунок 2.1 — Різниця між H.264 та H.265

Важливою функцією саме H.265 є нова функція довільного доступу до зображення: у цьому кодеку немає ключових та проміжних кадрів, тут усі кадри рівні між собою та кожен з них може бути декодованим самостійно, тому відео не розпадаються на блоки та чудово себе ведуть під час стрімінгу в реальному часі та відеовикликів.

Також у H.265 дуже сильно покращили внутрішньокадрове прогнозування, враховується 33 траєкторії руху, замість 9 у H.264.

Основний алгоритм роботи H.265 включає наступні етапи:

1) розділення кадру: подібно до H.264, відеофайл розділяється на невеликі блоки, що дозволяє відстежувати рух і стискати кожен блок окремо;

2) прогнозування руху: кодек H.265 використовує більш продумані методи прогнозування руху, що дозволяє краще використовувати інтеркадрову кореляцію і отримувати більш точні прогнози;

3) компенсація руху: алгоритм H.265 також використовує компенсацію руху, але з більшою точністю і ефективністю, що дозволяє зменшити обсяг даних, необхідних для кодування руху;

4) просторове стиснення: після прогнозування руху та компенсації руху використовуються методи стиснення, такі як дискретне косинусне перетворення та квантування, для зменшення обсягу даних;

5) кодування та стиснення: застосовуються алгоритми стиснення, подібні до тих, що використовуються в H.264, але з більшою ефективністю та здатністю стискати дані;

6) передача даних: остаточно стиснені дані передаються або зберігаються у вигляді відеофайлу [18, 20-22].

Загалом алгоритм роботи кодека H.265 дозволяє досягти значного зменшення розміру файлів відео при збереженні високої якості зображення, що робить його більш ефективним для передачі та зберігання великих обсягів відеоданих.

Основною проблемою кодеку H.265 є дуже складна політика ліцензування, він настільки дорогий, що просто не прижився в інтернеті та його підтримують лише декілька браузерів, не дивлячись на те, що йому вже більше 10 років. Тому основну свою популярність він отримав у сфері фільмів у високій якості та кодування медіафайлів на окремих платформа (таких як IOS).

У 2020 році був опублікований новий стандарт стиснення відео H.266 або VVC (Versatile Video Coding). Цей кодек є універсальним та однаково добре працює з усіма типами контенту: відео, 360-відео, зображення, ігри. Він підтримує змінну частоту кадрів від 0 до 120 Гц та змінне зображення.

Також H.266 чудово підходить для відео у високій якості, оскільки розмір блоків, на які поділяється відео під час обробки може досягати 128*128 пікселів.

Механізм передбачення руху набагато перевищує усі старіші аналоги, оскільки активно використовує сучасні нейронні мережі.

Основний алгоритм роботи H.265 включає наступні етапи:

1) розділення кадру: подібно до H.265, відеофайл розділяється на невеликі блоки, що дозволяє відстежувати рух і стискати кожен блок окремо, кожен блок може досягати розміру від 4*4 до 128*128 пікселів.

2) просторове та часове стиснення: кожен блок кодується за допомогою різних методів стиснення, включаючи передбачення, квантування та кодування змін.

3) параметризація квантування: цей процес визначає, наскільки деталізованими будуть значення кожного пікселя, більш висока квантизація призводить до більшого стиснення, але може призвести до втрати якості зображення.

4) передбачення руху: кодек використовує попередні кадри для передбачення руху об'єктів наступного кадру, зменшуючи тим самим інформацію, яка потрібна для кодування, для передбачення руху використовуються сучасні нейронні мережі, завдяки чому досягаються дуже високі результати стиснення.

5) кодування: кожен блок даних кодується з використанням алгоритмів стиснення, наприклад, алгоритму Хаффмана або адаптивного алгоритму арифметичного кодування (AAC).

6) передача даних: остаточно закодовані дані розкодовуються та зберігаються у вигляді відеофайлу [23].

H.266 із стисненням відеофайлів справляється набагато ефективніше усіх попередніх варіантів, коефіцієнт стиснення приблизно в 1.5 разів більший, ніж у H.265, проте потребує значно більших обчислень для своєї роботи.

2.2 Мінімізація розмірів відеофайлів на YouTube

Найпопулярнішими кодеками в інтернеті є кодеки від Google та Microsoft, а саме: VP8, VP9 та VC1. Основна їхня перевага в тому, що вони повністю безкоштовні, або пропонують дуже прості варіанти для ліцензування.

Ці кодеки дуже швидко отримали підтримку усіх браузерів, пристроїв та процесорів.

При викладанні відео на Youtube, платформа проводить процес кодування та декодування відео за допомогою різноманітних алгоритмів стиснення (компресії), автоматичної оптимізації роздільності та бітрейту відео, використання сучасних аудіо- та відеокодеків.

Такі дії над відео проводяться для того, щоб:

1) зменшити розмір відеофайлу без значної втрати якості, щоб зменшити завантаження дискового простору;

20 зекономити пропускну здатність, що допоможе набагато швидше та ефективніше передавати великі файли через мережу;

3) адаптувати файл до різноманітних пристроїв, оскільки різні пристрої мають різні вимоги щодо розширення та бітрейту відео. Таке кодування дозволяє надати оптимально відео для кожного пристрою;

4) надати можливість використання адаптивної потокової передачі, що дозволяє автоматично або вручну обирати розширення та бітрейт відео, в залежності від швидкості інтернет-з’єднання користувача.

На сьогоднішній день, платформа YouTube найчастіше використовує 2 види кодеків: VC-1 – використовується з менш популярними каналами та значно стискає відео, VP9 – використовується з більш популярними каналами та забезпечує краще зображення, але більший розмір файлу [24].

Кодек VC-1, розроблений у 2006 році компанією Microsoft являється пропрієтарним відео форматом, проте доступний для ліцензування. Він пропонує

доволі непогану ефективність стиснення, але не добре підтримує відеофайли з великим дозволом (як і H.264).

Загальний алгоритм роботи VC-1:

1) загальний сигнал перетворюється на цифрову гаму та розбивається на блоки 16*16, кожний з яких складається з 6 вибірок 8*8;

2) розділені блоки піддаються дискретному косинусному перетворенню та після цього квантуються для зменшення обсягу даних;

3) алгоритм прогнозує пересування об’єктів в кадрі за допомогою аналізу блоків;

4) значення яскравості та кольорів кодуються в залежності з прогнозованим пересуванням;

5) використовуються методи стиснення даних, такі як перетворення косинусів, для зменшення об’єму інформації;

6) на стороні приймача виконується декодування відеосигналу, використовуючи декодер VC-1, відтворюючи відео [25].

Відеокодек VC-1 є методом стиснення даних з утратами.

По своїй ефективності VC-1 приблизно як H.264.

Кодек VP9 був розроблений компанією Google в 2012 році та являється проектом з відкритим вихідним кодом. Він пропонує дуже високу ефективність стиснення та забезпечує збереження багатьох деталей, але декодування за допомогою VP9 є дуже сильно вимогливим до ресурсів та має доволі складний процес кодування, що робить його складнішим та більш вимогливим до ресурсів, на відмінно від інших кодеків.

Загальний алгоритм роботи VP9:

1) робота починається з перетворення простору кольорів з RGB в YCrCb (Y – яскравість U та V – кольорова інформація), що дозволяє більш енергоефективно працювати з відеоданими;

2) відеофрагмент ділиться на одиниці кодування (блоки), які мають розмір 64*64 пікселі та називаються суперблоками;

3) усі блоки скануються зліва направо, згори вниз по круговій схемі зі збільшенням довжини від кута (рисунок 2.2);

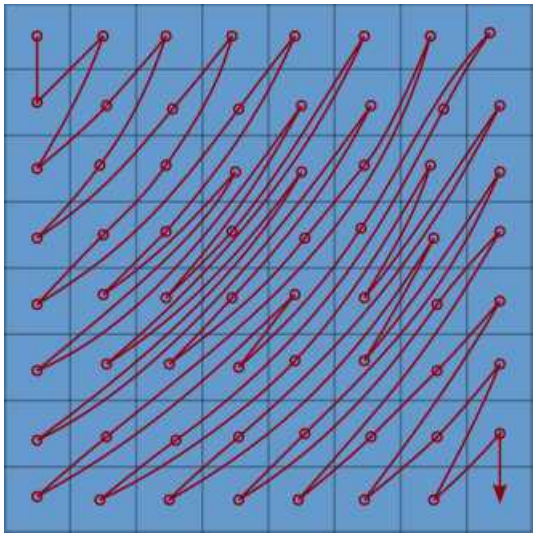


Рисунок 2.2 — Приклад сканування суперблоків

4.) для кожного блоку прогнозується його переміщення відносно попереднього кадру;

5) використовується квантування для зменшення бітової глибини блоків, що призводить до зменшення бітової глибини;

6) використовуються ентропійне кодування для ефективної передачі даних. Головним чином використовується адаптивний код Хаффмана для зменшення загальної довжини бітових повідомлень;

7) на основі оброблених даних формується бітовий потік, який може бути збереженим або переданим для подальшого відтворення;

8) на стороні приймача бітовий потік декодується, використовуючи декодер VP9, відтворюючи відео [24, 26].

Відеокодек VP9 є методом стиснення даних без втрат.

По своїй ефективності VP9 приблизно як H.265.

2.3 Стиснення за допомогою бібліотеки FFmpeg

Стиснення відеофайлів за допомогою бібліотеки FFmpeg є одним із найпоширеніших та ефективніших підходів для стиснення відеофайлів. FFmpeg — потужна відкрита бібліотека, яка включає в себе найширший спектр функцій, щодо: кодування, декодування, мультиплексування, демультиплексування та транскодування аудіо та відео файлів у різноманітних форматах.

Завантажити конкретну FFmpeg для своєї операційної можна з офіційного сайту. Після завантаження усі файли FFmpeg потрібно розмістити у корні системного диску та додати до неї шлях з системного шляху.

В операційній системі Windows це можна зробити за допомогою наступної команди у командному рядку:

setx /m PATH “C:\ffmpeg\bin;%PATH%”

Після цього бібліотека FFmpeg буде коректно працювати у будь-якій директорії на пристрої.

Використання відеокодеків є найрозповсюдженішим на найлегшим рішенням для стиснення відеофайлів без втрати якості. І саме FFmpeg надає зручний доступ до цих рішень [13, 27].

Для ефективного кодування відеофайлу за допомогою відеокодеку H.264 необхідно прописати команду:

ffmpeg -i input.mp4 -c:v libx264 -crf 23 -preset medium -c:a aac -b:a 128k out.mp4

Розберемо команду по кроках:

1) ffmpeg — це команда, яка безпосередньо викликає FFmpeg;

2) -i input.mp4 — це параметр вказує на вхідний відеофайл, який потрібно закодувати, у нашому випадку input.mp4;

3) -c:v libx264 – параметр, який вказує на кодек відео для кодування, в нашому випадку — H.264, libx264 — це вбудований кодек H.264 в FFmpeg;

4) -crf 23 — це параметр, який вказує якість відео, CRF (Constant Rate Factor) — це метод визначення якості зображення. Значення від 0 до 51, де 0 - максимальна якість, 51 — мінімальна, зазвичай використовують значення в діапазоні від 18 до 28, у нашому випадку використовується значення 23;

5) -preset medium — це параметр, який вказує на налаштування швидкості кодування, у нашому випадку medium - це середня швидкість. Існують різні параметри для швидкості, такі як ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow. Обране значення впливає на якість кодування та швидкість стиснення;

6) -c:a aac -b:a 128k — це параметри, які вказують кодек та бітрейт аудіо aac - кодек аудіо, який використовується для кодування аудіо, а 128k - бітрейт аудіо, який, у нашому випадку встановлений на 128 kbps;

7) output.mp4 — це параметр вказує вихідний файл, в який буде записане закодоване відео [28].

Під час кодування відео в командному рядку показуються метадані відео та з’являється інформація про відео: кадр кодування, скільки хвилин відео вже оброблено, розміри закодованого відеофайлу.

Інформацію про відео під час кодування показано на рисунку 2.3.

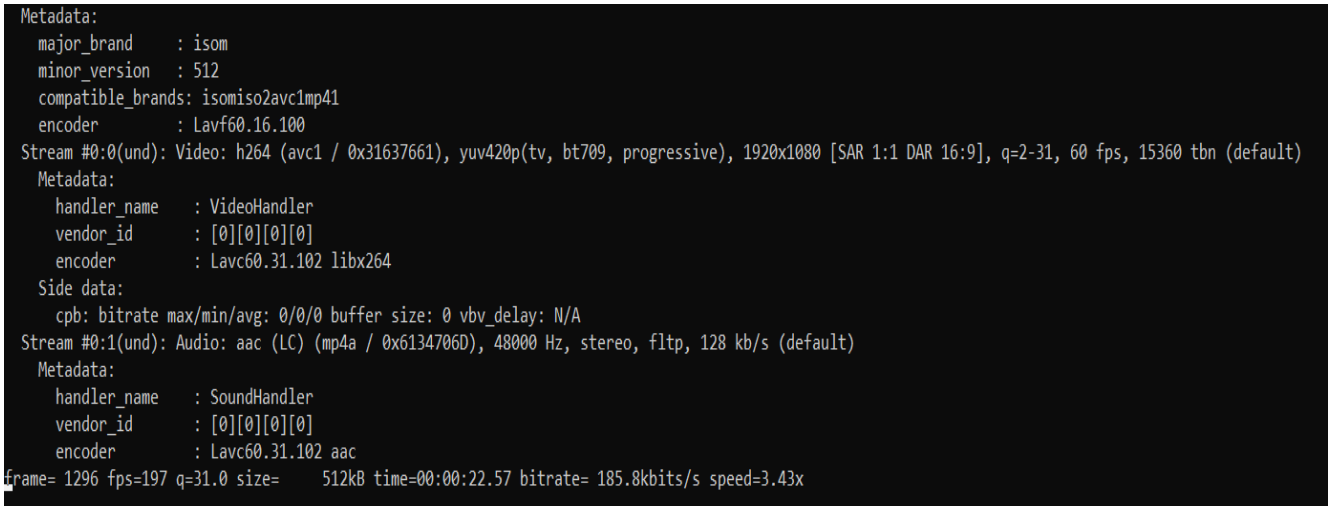


Рисунок 2.3 — Інформація про відео

Для кодування відео можна було лише написати лише: “ffmpeg -i input.mp4 -c:v libx264 out.mp4”, без усіх додаткових параметрів. Проте варто зауважити, що це призведе до використання параметрів за замовчуванням, які часто можуть не відповідати вимогам щодо якості або обсягу відеофайлу. Наприклад, якість відео буде значно нижчою або обсяг виявиться значно більшим.

Для ефективного кодування відеофайлу за допомогою відеокодеку H.265 необхідно прописати команду:

ffmpeg -i input.mp4 -c:v libx265 -crf 23 -preset medium -c:a aac -b:a 128k out.mp4

Перелік команд для кодування з H.264 та H.265 майже однаковий, окрім команди -c:v libx264 (-c:v libx265), оскільки саме він вказує на відеокодек [29].

Для ефективного кодування відеофайлу за допомогою відеокодеку VP9 необхідно прописати команду:

ffmpeg -i input.mp4 -c:v libvpx-vp9 -crf 30 -b:v 0 -c:a libopus -b:a 128k output.webm

Розберемо команду детально по кроках:

1) ffmpeg — це стандартна команда, яка безпосередньо викликає FFmpeg, вона використовується завжди при роботі з FFmpeg;

2) -i input.mp4 — це параметр вказує на вхідний відеофайл, який потрібно закодувати, у нашому випадку input.mp4;

3) -c:v libvpx-vp9 — це параметр, який вказує на кодек відео для кодування, в даному випадку — VP9. libvpx-vp9 — це вбудований кодек VP9 в FFmpeg;

4) -crf 30 — це параметр, який визначає якість відео. CRF (Constant Rate Factor) — це метод визначення якості зображення. Для VP9 значення визначаються від 0 до 63, де 0 — максимальна якість, 63 — мінімальна. У цьому випадку використовується значення 30;

5) -b:v 0 — це параметр, який вказує бітрейт відео. У нашому випадку встановлене значення 0, що означає використання змінного бітрейту;

6) -c:a libopus -b:a 128k — це параметри, які вказують кодек та бітрейт аудіо. Libopus – це кодек аудіо, який використовується для кодування аудіо, а 128k – бітрейт аудіо, який в даному випадку встановлений на 128 kbps;

7) out.webm — це параметр, який вказує вихідний файл, в який буде записане закодоване відео [30].

Варто зауважити, що при кодуванні відео за допомогою VP9 слід обрати саме варіант webm, а не mp4, тому що VP9 – це стандарт кодування відео, який розроблений саме для використання на веб-сайтах та відеоплатформах. При кодуванні у формат mp4 відео може відображатись некоректно.

Коефіцієнт стиснення для відео може значно відрізнятися в залежності від багатьох факторів, таких як: частота кадрів, роздільна здатність, якість відео, тощо. Однак, приблизно коефіцієнти стиснення для кодеків H.264, H.265 і VP9 можуть бути такими:

1) H.264 (AVC): зазвичай досягає коефіцієнта стиснення приблизно від 1.5 до 10 разів порівняно з некодованим відео;

2) H.265 (HEVC): може забезпечити коефіцієнт стиснення від 25% до 50% більше, ніж H.264, що означає ще більшу стисліть файлів відео;

3) VP9: також забезпечує високий коефіцієнт стиснення, зазвичай порівнюваний з H.265 або навіть кращий в певних умовах, але кодування за допомогою VP9 використовує значно більше обчислювальних ресурсів та відбувається дуже довго, що і робить його використання недоцільним для звичайного домашнього користування [1, 5, 10].

Також варто зазначити, що коефіцієнти стиснення можуть значно варіюватися в залежності від конкретних параметрів кодування, властивостей відео та налаштувань кодеку. Для отримання точних значень коефіцієнтів стиснення завжди рекомендується провести тести на конкретних відеофайлах з використанням різних кодеків.

2.4 Робота з програмою Video to Video

Video to Video – це безкоштовне програмне забезпечення для конвертування різноманітних відео та аудіо форматів. Дана програма дозволяє конвертувати практично всі нині існуючи формати та підтримує різноманітні рідкісні та специфічні кодеки. Video to Video працює на базі бібліотеки FFmpeg.

Для того, щоб розпочати роботу по конвертації відео, необхідно натиснути на кнопку “Додати” та обрати необхідний відеофайл. Після цього з’явиться діалогове вікно конвертації, де можна обрати необхідний формат. При натисненні на кнопку формату відео програма автоматично завантажує усі налаштування для відео за замовченням, але усі параметри для відео можна налаштувати вручну на сітці, яка відображається справа.

Перелік основних та відео налаштувань, які пропонує програма Video to Video зображено на рисунку 2.4.

Рисунок 2.4 — Перелік основних та відео налаштувань

Програма Video to Video підтримує дуже багато різноманітних параметрів для налаштування відео, серед яких є: відеокодек, розмір відео, Video Aspect, бітрейт відео, частота кадрів, відеосистема та інші.

Відеокодек (Video codec) – налаштування у якому ви обираєте тип компресора для відео та кодек. Кожен формат може підтримувати декілька різних кодувальників. Так наприклад, формат .avi підтримує майже усі існуючи кодеки, а формат .mp4 – підтримує вже меншу кількість.

Кодувальники, які підтримує формат .avi та .mp4 зображені на рисунках 2.5 та 2.6 відповідно.

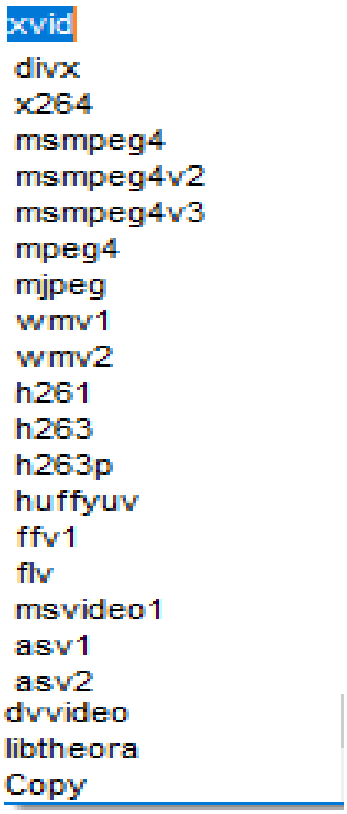


Рисунок 2.5 — Кодувальники .avi

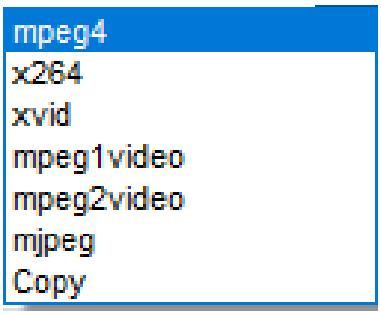


Рисунок 2.6 — Кодувальники .mp4

Розмір відео (Video size) – розмір (ширина та висота кадра) зконвертованого вихідного відеофайлу. За замовченням стоїть “оригінал”, тобто вихідний відеофайл буде такого самого розміру. Якщо цей параметр змінити — програма автоматично перерахує бітрейт відео та змінить його розширення.

Video Aspect – пропорційне співвідношення ширини та висоти відео.

Бітрейт відео (Video Bitrate) – кількість даних, яка оброблюється в одиницю часу в відеофайлі. Більш високий бітрейт означає більш високу якість та більший розмір вихідного файлу.

Частота кадрів відео (Video Framerate) – кількість кадрів в секунду.

Також за допомогою програми Video to Video можна додавати субтитри до своїх відео, працювати з аудіо параметрами та пропонує додаткові розширені опції, які використовуються не так часто, але також можуть знадобитися.

Параметри для роботи з аудіо представлені в блоці Audio Options. Тут є можливість для вибору: аудіо кодеку, аудіо бітрейту, регулювання гучності, вимкнення аудіо та інші параметри.

Представлена можливість для додавання та редагування субтитрів (блок Subtitles).

Додаткові параметри представлені в блоці Advanced.

Keep Aspect (зберегти співвідношення сторін) цей параметр визначає, чи буде вихідне відео зберігати таке саме співвідношення сторін, як і вхідне відео. Якщо співвідношення сторін виявиться не однаковим — конвертер просто додасть до нього чорні доріжки по боках.

Target Filesize дозволяє задати розмір вихідного відеофайлу, проте це може сильно змінити попередні налаштування.

Extra Params (додаткові параметри) — тут можна додати додаткові параметри, оскільки Video to Video працює на базі бібліотеки FFmpeg.

Filters, Crop, Pad, Rotate, Flip – різноманітні відеофільтри.

Кінцевий перелік усіх параметрів представлений на рисунку 2.7.

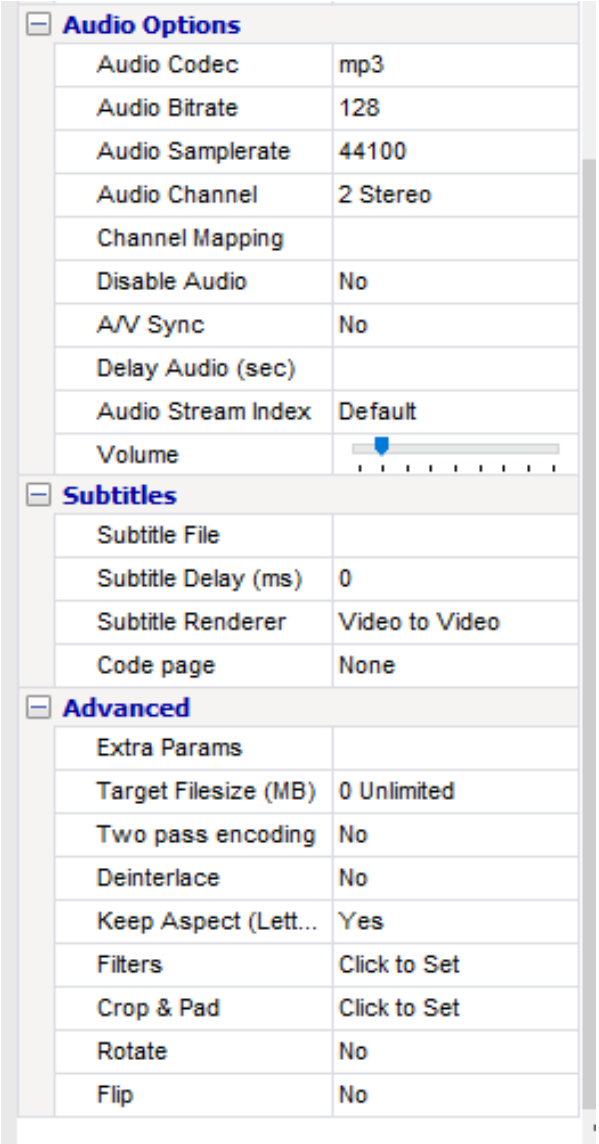


Рисунок 2.7 — Робота з аудіо, субтитрами та додатковими параметрами

При кодуванні відео за допомогою кодеку H.264 (x264) доволі легко можна стиснути в відео 1.5-3 рази без втрати якості. Також можна налаштовувати бітрейт вихідного відеофайлу, щоб краще контролювати якість та розмір файлу.

При заданій високій якості та зменшені бітрейту в 1.5 разів відеофайл може зменшитись приблизно у 2 рази.

Відеофайл з 85.87 мб вдалось стиснути до 46.98 мб (майже у 2 рази), без втрати якості.

Налаштування, які використовувалися при кодуванні подано на рисунку 2.8.

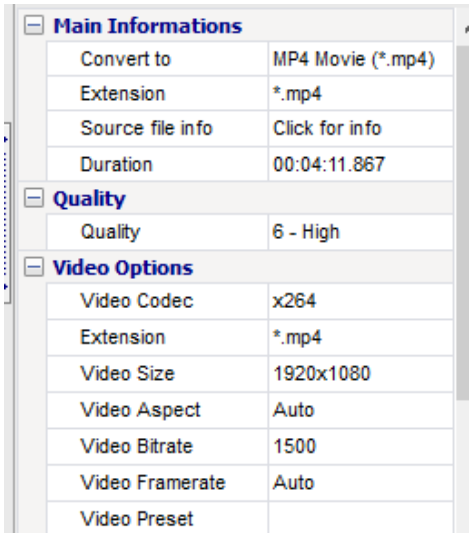


Рисунок 2.8 — Налаштування відеофайлу

Загалом Video to Video є простим та зручним програмним забезпеченням, яке можна використовувати як і для стиснення відеофайлів, для подальшого їх зберігання, так і для конвертації їх у різноманітні формати, які вам необхідні для роботи, оскільки Video to Video підтримує більше 100 різних рідкісних відеоформатів [31].

2.5 Робота з програмою Handbrake

Handbrake – це безкоштовне програмне забезпечення, з відкритим вихідним кодом, для багатопотокового перекодування відеофайлів із одного формату в інший, перша версія якої була випущена у 2003 році спільнотою розробників Handbrake Community.

Основні можливості програми Handbrake:

1) конвертація відеофайлів у інші формати (підтримуються тільки формати: .mp4, .mkv, webm);

2) обрізка та зменшення розміру відео;

3) відновлення старого та неякісно відео;

4) усунення різноманітних артефактів на відео;

5) регулювання гучності відео та його динамічного діапазону для певних типів звуку;

6) робота з субтитрами (збереження, додавання та видалення субтитрів з відео).

Найчастіше Handbrake використовується саме для стиснення відеофайлів та зменшення їх розміру, для такого кодування можуть застосовуватися сучасні та просунуті кодеки, такі як: AV1, H.264, H265, MPEG-2, VP8 та VP9, Theora для відео, AAC, AC-3, FLAC, Vorbis та Opus для роботи зі звуком.

На початку роботи, після вибору необхідного відеофайлу, над яким буде проводитись стиснення, необхідно обрати шаблон (preset), завдяки якому файл буде конвертуватись. За замовченням обирається шаблон Fast 1080p30, який доволі часто є хорошим вибором та сумісний з більшістю сучасних пристроїв.

Шаблони у Handbrake — це група налаштувань, які спеціально призначені для конкретного програмного забезпечення чи пристрою, на якому ви бажаєте відтворювати своє відео.

Handbrake включає в себе ряд офіційних налаштувань шаблонів, які обирають певні налаштування для забезпечення сумісності в Інтернеті, з конкретними пристроями та для загального використовування. При виборі одного з цих шаблонів — параметри відео з більш високим дозволом буде зменшено до максимального дозволу, а більша частота кадрів буде обмежена максимальною частотою кадрів, налаштування звуку та інші налаштування також будуть примусово застосовані.

Як саме шаблони у Handbrake працюють на практиці:

1. При виборі одного з 720p30 шаблонів на відео з якістю 1080p та швидкістю кадрів 60 кадрів на секунду — якість буде зменшена до 720p, а частота кадрів обмежена 30;

2. При виборі одного з 720p30 шаблонів на відео з якістю 480p та швидкістю кадрів 30 кадрів на секунду — якість не буде зменшена та кількість кадрів в секунду не буде обмежена [32-33].

Перелік основних шаблонів у Handbrake зображений на рисунку 2.9.

Рисунок 2.9 — Перелік основних шаблонів

Різні шаблони можуть вплинути на сумісність (чи буде ваше відео коректно працювати з вашими пристроями та програмним забезпеченням). Наприклад, якщо ваше відео буде відтворюватись на платформі Playstation – можна обрати

шаблон Playstation 1080p30 Surround. Загалом у Handbrake доступні спеціальні шаблони для пристроїв, які працюють на базі: Amazon, Apple, Android, Chromecast, Playstation, Roku, та Xbox.

Також присутні спеціальні шаблони для вебсайтів, які використовуються на відео, які будуть завантажуватись на відеохостинги, такі як: Youtube та Vimeo. Зазвичай такі шаблони забезпечують кращу якість відео та звуку, щоб зменшити втрати при перекодуванні в різні формати та роздільну здатність на відеохостингах, які зазвичай виникають під час перекодування.

Наступним важливим кроком при налаштуванні відео перед конвертацією буде вибір відеокодеку, завдяки якому буде проходити весь процес та налаштування якості RF (Reference Frame).

Перелік відеокодеків, які підтримуються у Handbrake зображено на рисунку 2.10.

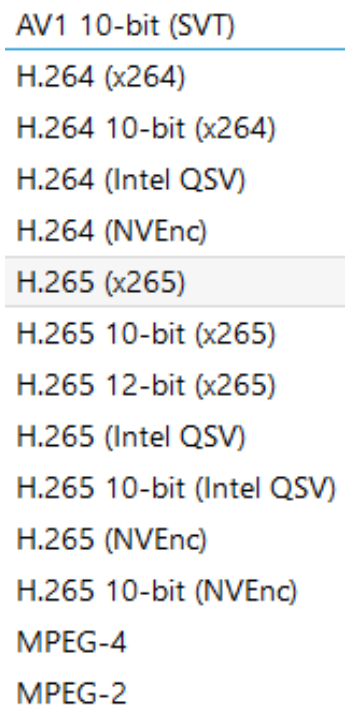


Рисунок 2.10 — Перелік відеокодеків

Найпопулярнішими відеокодеками для стиснення відеофайлів є H.264 (x264) та H.265 (x265).

Reference frame, в контексті кодеків для відео, вказує на зображення, на яких будуть засновані наступні кадри у відеопотоці. Ці кадри використовуються для передбачення та стиснення наступних кадрів, що дозволяє суттєво зменшити розмір файлу при збереженні його якості.

Під час вибору якості відео за допомогою RF, чим вище значення RF, тим вища якість відео (відбувається більш точна передача його зображення), але при цьому збільшується і розмір самого файлу.

Рекомендовані налаштування значень RF для кодеків H.264 (x264) та H.265 (x265):

1. RF 18-22 для стандартного розширення 480p/576p;
2. RF 19-23 для високої якості 720p;
3. RF 20-24 для 1080p Full High Definition;
4. RF 22-28 для 2160p 4K Ultra High Definition [33].

Шкала вибору якості RF зображена на рисунку 2.11.

Рисунок 2.10 — Шкала вибору якості RF

Після вибору шаблону Fast 1080p30, відеокодеку H.264 (x264) та значення RF = 22, відео, яке було записано у формату 1080p, без втрати якості вдалось стиснути з 43.5 мб до 10.9 мб, що майже у 4 рази, але при цьому зменшується бітрейт відео.

Завдяки програми Handbrake різні відеофайли, в залежності від їхнього наповнення можна стиснути у 3-4.5 рази. Під час кодування відео сильно навантажується процесор, але кодування відбувається доволі швидко. Якщо

відбулось дуже сильне навантаження на апаратні ресурси — є можливість поставити відкласти процес та поставити його на паузу.

Загалом Handbrake є простим та інтуїтивно зрозумілим програмним забезпеченням, яке може бути хорошим та зручним варіантом для швидкого стиснення великої кількості відеофайлів для подальшого їх зберігання.

2.6 Висновки

Після аналізу основних застосунків, які використовуються для стиснення відеофайлів та безпосередньої конвертації відеофайлів для їх стиснення, можна виокремити декілька основних стратегій для того, щоб при конвертації відео зменшився його розмір без значної втрати якості:

1) зменшення бітрейту: у такому випадку значно зменшується об’єм даних, але також варто зауважити, що це може сильно вплинути на якість, тому часто треба експериментувати з різноманітними значеннями бітрейту, щоб знайти ідеальний баланс між якістю файлу та його бітрейтом;

2) використання найбільш ефективного кодека: різні кодеки мають різну ступінь стиснення даних та забезпечують баланс між якістю файлу та його розміром;

3) зменшення розширення: це може сильно зменшити розмір файлу, однак варто оцінити, яке розширення зберігає необхідну та прийнятну якість та підходить для перегляду відео на різних платформах (якщо воно у подальшому буде викладатись);

4) використання змінного бітрейту: такий бітрейт адаптується до змісту самого відео — збільшується у більш важких та складних сценах та зменшується в менш складний моментах;

5) зменшення кількості кадрів в секунду (FPS): це може сильно зменшити розмір вихідного відеофайлу, але майже завжди впливає на якість, тому слід обережно відноситись до цього методу;

6) зменшення або видалення звуку: якщо звук не являється важливим компонентом відео — його можна видалити, або зменшити його бітрейт;

7) використання стиснення з утратами: деякі кодеки використовують стиснення даних з утратами і при цьому зберігають достатньо непогану якість відео.

Слід зауважити, що кожне відео унікальне та ефективність кожної із стратегій буде залежати від його змісту. Тому рекомендується проводити експерименти з різними параметрами та перевіряти результати.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Запис відео з екрану

Перед початком запису відео перш за все необхідно обрати формат у якому буде записано відео. Сучасні програми для запису екрану пропонують доволі багато форматів для запису відео. Найпопулярнішими та найбільш зручними для кодування є формати: mp4, avi, mkv[34-36].

Таблиця 3.1 — Порівняння форматів mp4, avi, mkv

Критерій для порівняння	mp4	avi	mkv
1	2	3	4
Якість відео та аудіо	Підтримує якісне відео та аудіо при низькому розмірі файлу, підтримує майже усі сучасні кодеки	Дозволяє високоякісне відео та аудіо, але доволі часто обмежений у підтримці деяких сучасних кодеків, що впливає на якість	Забезпечує високоякісне відео та аудіо, а також можливість містити декілька аудіо- та субтитрів у одному файлі
Розмір файлу	Зазвичай має менший розмір файлу, в однаковій якості через підтримку багатьох кодеків, що робить їх ідеальним для онлайн відео та стрімінгу	Файли з високою можуть бути доволі великими, що може призвести до обмежень використання в онлайн середовищі	Файли такого формати є досить великими, особливо на відео з високою якістю, але завдяки цьому зберігається висока якість та усі додаткові параметри
Підтримка різноманітних пристроїв та платформ	Має найширшу підтримку більшості пристроїв та платформ	Має добру підтримку, але може виникати несумісність з деякими сучасними платформами	Підтримується значно меншою кількістю платформ, часто виникає несумісність

Кінець таблиці 3.1

1	2	3	4
Додаткові можливості	Доволі простий відео формат, який не має багато додаткових можливостей, порівняно з avi та mkv	Може містити додаткові параметри та властивості, але їх використання може зробити файл більшим та менш підтримуваним	Дозволяє легко включати декілька аудіо та субтитрів в одному файлі, що робить його гнучким для використання у таких цілях

Виходячи з даних про формати відеофайлів — можна зробити висновок, що формат mp4 – є стандартним та найкращим варіантом для запису відео з екрану, як і для запису навчального контенту, важливої інформації або геймплею. Інші формати потрібно використовувати лише у певних випадках.

Наступним важливим параметром для запису відео є вибір FPS (frames per seconds, англ. Кількість кадрів в секунду) для вашого відео. Кількість кадрів в секунду дуже сильно впливає на якість відео та на його розмір, тому до цього потрібно ретельно підійти. Для запису відеолекції доволі часто може бути достатньо усього 10 FPS, у той час як для рухливого відео, наприклад запису геймплею, потрібно 30-60 FPS і навіть більше.

Також важливим параметром, який впливає на якість відеофайлу є вибір відеокодека для запису відео з екрану. Найпопулярнішим кодеком, який підтримується більшістю платформ та пристроїв є програмний відеокодек x264, серед апаратних відеокодеків найпопулярнішими є NVENC (підтримується лише на відеокартах Nvidia) та QSV (підтримується на усіх сучасних процесорах Intel за допомогою технології Quick Sync Video) [16-17].

Таблиця 3.2 — Порівняння кодеків x264 з NVENC та QSV

Критерій для порівняння	x264	NVENC та QSV
1	2	3
Тип кодеку	Програмний	Апаратні

Кінець таблиці 3.2

1	2	3
Якість відео	Може бути доволі висока при високих налаштуваннях	Залежить від параметрів та моделі апаратних ресурсів, зазвичай доволі висока
Продуктивність	Висока, але вимагає значно більших ресурсів від процесора	Дуже ефективна, використовуються апаратні прискорення ресурсів для меншого навантаження
Підтримка обладнання	Підтримується на всіх сучасних системах	Підтримуються тільки з конкретними апаратними ресурсами
Розміри вихідного відеофайлу	Використовуються сучасні технології для зменшення розмірів вихідного відеофайлу, можна встановити бітрейт та керувати розмірами вихідного відеофайлу	Зазвичай відеофайли мають менший розмір, проте використовується змінний бітрейт, який може стати дуже великим під час запису відео з різкою зміною кадрів, наприклад геймплею
Сумісність	Відео можуть бути обмежені підтримкою деяких кодувальників для обробки у майбутньому	Широко підтримуються багатьма сучасними кодувальниками
Навантаження на систему	Високе, оскільки вимагає значних ресурсів процесора та інших компонентів	Низьке, оскільки використовується апаратне прискорення відеокарти або процесора

Можна зробити висновок, що використання апаратних відеокодеків, за можливістю, є більш пріоритетним, ніж програмних.

У середньому відео, однакової якості, записані за допомогою NVENC займають приблизно в 2 рази менше місця, ніж x264 та в 1.2 менше, ніж QSV [16-17, 37].

3.2 Підбір найоптимальніших параметрів

Для реалізації роботи було обрано відеокодек h264, оскільки він підтримується абсолютною більшістю браузерів та програм відеовідтворення, не потребує сильних обчислювальних ресурсів, як h265.

Перший параметр, який необхідно розібрати — це preset. Preset – це набір параметрів, які забезпечують певне співвідношення швидкості кодування та стиснення. Це означає, для більшої якості стиснення відеофайлів потрібно використовувати більш повільніші пресети, для швидкого кодування, але з меншим коефіцієнтом стиснення — потрібно використовувати більш швидкі.

Для аналізу усіх варіантів пресету візьмемо відео розміром 85.8 мб та довжиною 4 хвилини 11 секунд.

Таблиця 3.3 — Порівняння пресетів

Preset	Вихідний розмір відеофайлу	Час кодування
veryslow	18.8 мб	5:35
slower	22.6 мб	2:43
slow	31.2 мб	1:40
medium	32.8 мб	1:31
fast	33.2 мб	1:28
faster	33.7 мб	1:20
veryfast	48.9 мб	0:58
superfast	72.8 мб	0:47
ultrafast	83.2 мб	0:31

Виходячи з таблиці, можна зробити висновок, що найкращий коефіцієнт стиснення мають пресети veryslow та slower, проте вони потребують набагато більше обчислювальних ресурсів. Пресет veryslow стискає відео приблизно на 20% краще, ніж slower, але його робота займає в 2 рази більше часу, тому його варто використовувати лише у випадку, коли необхідний максимальний ступінь

стиснення відеофайлу і потрібно враховувати, що для великих відеофайлів таке кодування може зайняти доволі багато часу та потребувати обчислювальних ресурсів.

Найоптимальнішими варіантами, з точки зору стиснення та складності обробки будуть: slow, medium та fast.

Faster, veryfast та superfast мають порівняно невеликий коефіцієнт стиснення та можуть використовуватися лише у випадках, коли необхідно стиснути відеофайл у найкоротший термін.

Після обробки відеофайлу кожним із пресетів, окрім ultrafast, зменшується бітрейт відео відповідно до ступеня стиснення, проте це не відчувається на якості відео. Ultrafast — пресет, який має найгірший коефіцієнт стиснення та іноді навіть може збільшити розмір вихідного файлу для певних вхідних відеофайлів (особливо великих за розміром). Проте обробка завдяки ultrafast збільшує бітрейт відеофайлу, що робить його корисним у певних випадках [19, 28].

Якщо для відеофайлу потрібно вказати певний бітрейт для вашого відеофайлу — можна використати команду -b:v, яка встановлює заданий бітрейт для відео.

Наприклад, якщо ви хочете, щоб бітрейт вихідного файлу був 2000 кбіт/с, необхідно додати параметр -b:v із значенням 2000k: -b:v 2000k. У такому випадку для вихідного відеофайлу буде обраний саме ваше значення бітрейту, але це може вплинути на розмір відеофайлу.

Також варто зауважити, що у випадку використання заданого бітрейту, параметр необхідно прописати саме після команди використання -preset, оскільки його використання перезапише встановлений вами бітрейт.

Наприклад: -preset slow -b:v 2000k.

Загалом час кодування вихідного файлу кожним із пресетів залежить від безпосередньо вибраного вами пресету, бітрейту вихідного відеофайлу та обладнання на якому приходить обробка. Чим вищий бітрейт тим вищим буде час кодування.

На рисунку 3.1 подано діаграму, яка показує приблизний час кодування відеофайлу з роздільною здатністю 1080p (full hd) для кожного пресету, в залежності від бітрейту вихідного файлу.

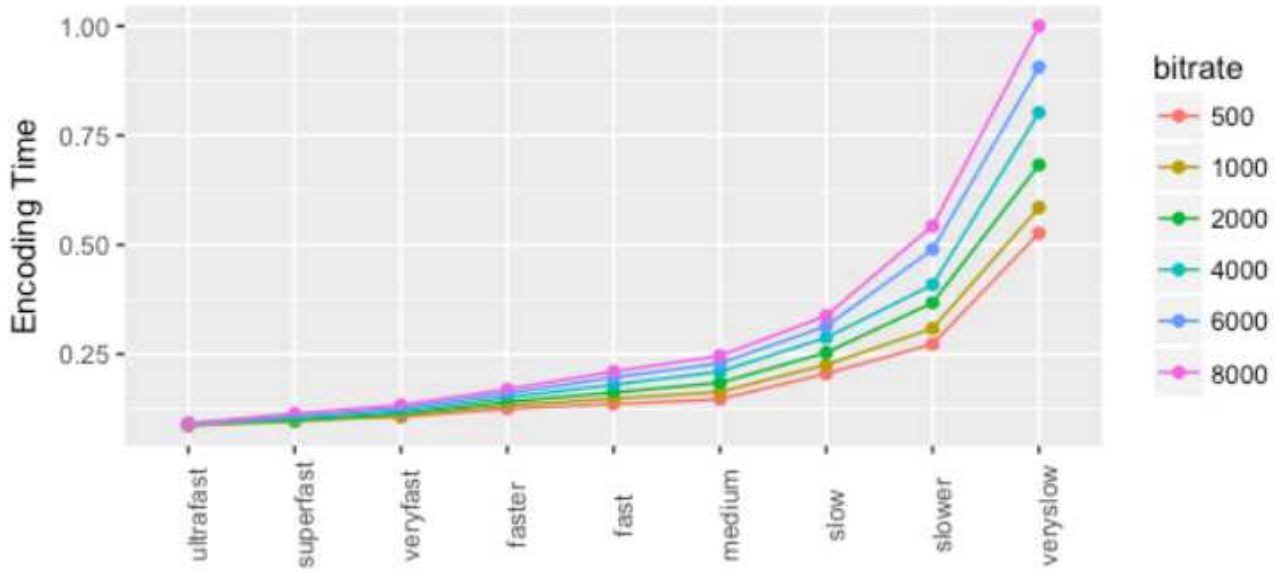


Рисунок 3.1 — Час кодування

При переході на більш повільну швидкість кодування у середньому час кодування збільшується на 20-40%. При переході на найдовшу швидкість veryslow – час кодування збільшується більше, ніж в 2 рази.

Наступним важливим параметром є Constant Rate Factor (CRF) (англ. Постійний коефіцієнт швидкості) — цей параметр дозволяє кодеку досягти певної якості для всього вихідного відеофайлу. CRF налаштовує так званий квантувальник для кожного кадру, він отримує бітрейт, необхідний для підтримання необхідного рівня якості і це забезпечує найбільшу ефективність стиснення за один прохід.

Параметру CRF не можна вказати бітрейт, щоб отримати певний розмір відеофайлу — це означає, що його не можна використовувати разом з -b:v, оскільки CRF автоматично регулює бітрейт в залежності від обраного значення.

Діапазон значень параметра CRF становить від 0 до 51, де 0 — це початкова якість без втрат, 51 — видає найгіршу якість з можливих. За замовчуванням встановлюється значення 23. Найчастіше використовуються значення в діапазоні 17-28.

Вибір між параметрами CRF і -b:v залежить від ваших конкретних потреб і ситуації.

Якщо ваша головна мета — це стабільна якість відео, і ви не хочете перейматися точним розміром файлу або бітрейтом, тоді CRF буде гарним вибором. Він автоматично регулює бітрейт, щоб забезпечити певний рівень якості, незалежно від складності контенту.

З іншого боку, якщо вам важливій конкретний розмір файлу або бітрейт, наприклад, якщо ви плануєте передавати відео через обмежений канал інтернету або зберігати на пристрої з обмеженим обсягом пам'яті, тоді ви можете скористатися параметром -b:v. Він дозволяє вам точно встановити бітрейт відео в заданому діапазоні.

Отже, якщо вам важлива якість, використовуйте CRF. Якщо вам важливій конкретний розмір файлу або бітрейт, використовуйте -b:v.

Параметр, використовуючи який, можна дуже сильно зменшити розмір відеофайлу: -framerate або -r. Завдяки framerate можна налаштувати кількість кадрів в секунду у вихідному відеофайлі. Наприклад, щоб встановити кількість кадрів 30, необхідно додати параметр: -r 30. Проте з цим параметром потрібно бути дуже обережним, оскільки зміна кількості кадрів в секунду у відеофайлі може сильно зменшити його якість [28].

Варто зауважити, що для кожного відео потрібно підбирати окремі параметри, щоб досягти найбільшого стиснення відеофайлу при збережені максимальної якості.

3.3 Розробка програми

Перед початком розробки потрібно обрати платформу, на якій буде працювати програми та засоби, завдяки якими вона буде розроблена.

Найпопулярнішою та найзручнішою операційною системою для роботи — є Windows. Windows Forms, які працюють на платформі .NET є зручним засобом для розробки додатків з графічним інтерфейсом на мові програмування C# в ОС Windows. Саме тому програма буде розроблятися завдяки цим засобам.

Найкращим варіантом для розробки проектів з Windows Forms є середовище розробки Microsoft Visual Studio, саме тому це середовище розробки було обрано для роботи над проектом.

Наступним кроком буде вибір бібліотеки, яка допомагає працювати з FFmpeg, оскільки FFmpeg є найкращим засобом для будь-якої роботи з відеофайлами.

На платформі .NET існує 3 основних бібліотеки для роботи з FFmpeg: FFmpeg.NET, Xabe.FFmpeg та FFmpeg.AutoGen.

Найкращим та найпотужнішим з цих інструментів є Xabe.FFmpeg. Основі переваги xabe.FFmpeg перед своїми конкурентами:

1) повний функціонал — Xabe.FFmpeg дозволяє використовувати всі можливості FFmpeg безпосередньо у C#, можна використовувати використовувати всі функції та опції FFmpeg;

2) асинхронна підтримка — Xabe.FFmpeg надає асинхронні методи для виконання операцій з відео та аудіо, завдяки цьому можна уникнути блокування основного потоку програми під час обробки великих файлів або операцій, що займають багато часу;

3) простота використання — Xabe.FFmpeg надає простий та зрозумілий API для взаємодії з функціями FFmpeg у C#;

4) кросплатформеність — Xabe.FFmpeg підтримується на різних операційних системах, що робить його ідеальним вибором для розробки кросплатформених додатків, або масштабування у майбутньому;

5) активна підтримка та оновлення — Xabe.FFmpeg сьогодні активно розвивається та підтримується спільнотою, часті оновлення виправляють помилки та додають нові функції, що робить дану бібліотеки актуальною та надійною на сьогоднішній момент;

6) додаткові функції та зручні інструменти — Xabe.FFmpeg надає додаткові функції та інструменти, які полегшують роботу з відео та аудіо у C#, наприклад, зчитування та запис потоків у реальному часі, можливість отримання метаданих файлу та інші.

Саме тому Xabe.FFmpeg було обрано для реалізації задачі.

Для роботи Xabe.FFmpeg на пристрої кожного кінцевого користувача має бути встановлений FFmpeg та шлях до нього має бути встановлений як системний [38].

Додати Xabe.FFmpeg до свого додатку у Microsoft Visual Studio можна за допомогою пакетного менеджера NuGet ввівши в пошук: Xabe.FFmpeg. Або за допомогою команди:

PM> Install-Package Xabe.FFmpeg [39]

Вигляд Xabe.FFmpeg у пакетному менеджері NuGet подано на рисунку 3.2.

Рисунок 3.2 — Вигляд Xabe.FFmpeg у пакетному менеджері NuGet

Найкраще обирати найновіші версії бібліотеки, проте варто брати до уваги, чи буде обрана версія бібліотеки коректно працювати з вашою версією .NET.

Базовою структурою в Xabe.FFmpeg є потоки, тому корисно знати їхню структуру. Найважливіші методи, які використовуються в Xabe.FFmpeg поділяються на 3 потоки: робота з відео (Interface IvideoStream), робота з аудіо (Interface IAudioStream) та робота з субтитрами (Interface ISubtitleStream).

Під час роботи з відео можна встановлювати параметри: висоти та ширини відео, поворот відео, кількості кадрів в секунду, бітрейту, співвідношення сторін, формат пікселів.

Під час роботи з аудіо можна встановлювати значення: часу відтворення аудіо, бітрейту аудіо, частоти дискретизації звуку, зміни звукових доріжок.

Під час роботи з субтитрами можна встановлювати значення: мови субтитрів та самі субтитри.

Для роботи з усіма цими значеннями використовуються різноманітні методи, які встановлюють задані значення та проводять дії над ними.

Усі ці потоки працюють на базі основного IStream,[40].

Структура потоків, які використовуються в Xabe.FFmpeg показані на рисунку 3.3.

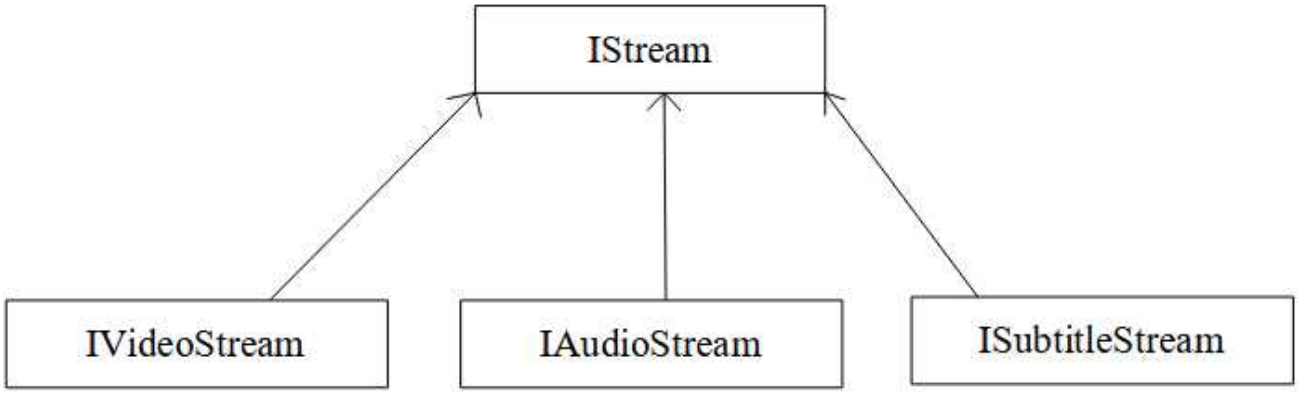


Рисунок 3.3 — Структура потоків в Xabe.FFmpeg

Після аналізу основної структури Xabe.FFmpeg можна починати роботу над безпосередньою розробкою програми.

На першому кроці розробки програми необхідно визначити її структуру. Було визначено 3 найоптимальніші параметри, які використовуються для стиснення відеофайлів: CRF (Constant Rate Factor), Preset та бітрейт. Для введення цих параметрів можна використовувати форму з полями для вводу. Для параметру Preset буде використовуватися поле з випадаючим списком, тому що кількість варіантів вибору там обмежена та їх потрібно ввести точно.

Наступним кроком буде реалізація вибору файлу, для цього на форму додається кнопка “Select File”, при натисненні на яку буде відкриватися діалогове вікно, та можна буде обрати файл. Для збереження шляху до файлу використовується ще одне поле для вводу (також туди можна ввести шлях до файлу вручну).

Останнім кроком буде додавання форми, у якій потрібно буде вказати назву вихідного файлу та кнопки для початку конвертації “Convert File”.

Загальний вигляд готової форми зображено на рисунку 3.4.

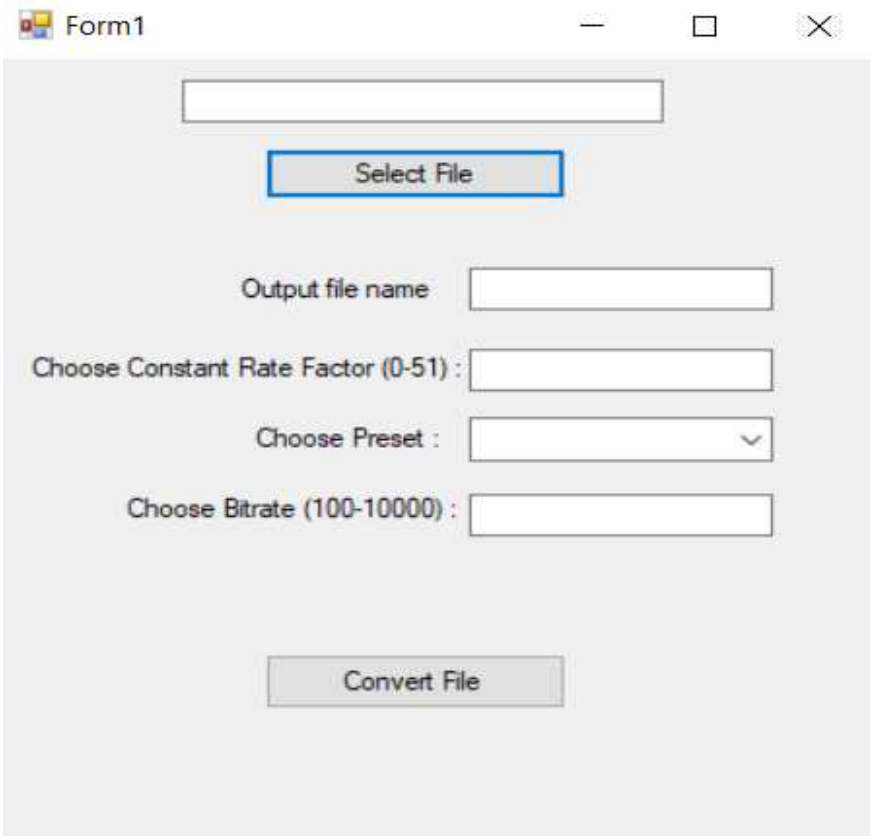


Рисунок 3.4 — Загальний вигляд готової форми

Можна побачити, що біля поля вводу Constant Rate Factor встановлена підказка (0-51), тому що діапазон значень для CRF становить саме від 0 до 51.

Для значення бітрейту встановлено 100-10000, хоча для нього не існує певних меж, все тому, що дуже при дуже великих значеннях бітрейту виходять дуже великі файли за розміром та їх конвертація може зайняти дуже багато часу. Значення 10000 і так дуже велике і його достатньо для абсолютної кількості файлів. При маленьких значеннях бітрейту — якість файлу, зазвичай, стає дуже, тому встановлене обмеження 100.

У подальшому будуть реалізовані перевірки, щоб користувач не міг ввести інші значення.

Для введення пресетів використовується випадаючий список зображений на рисунку 3.5.

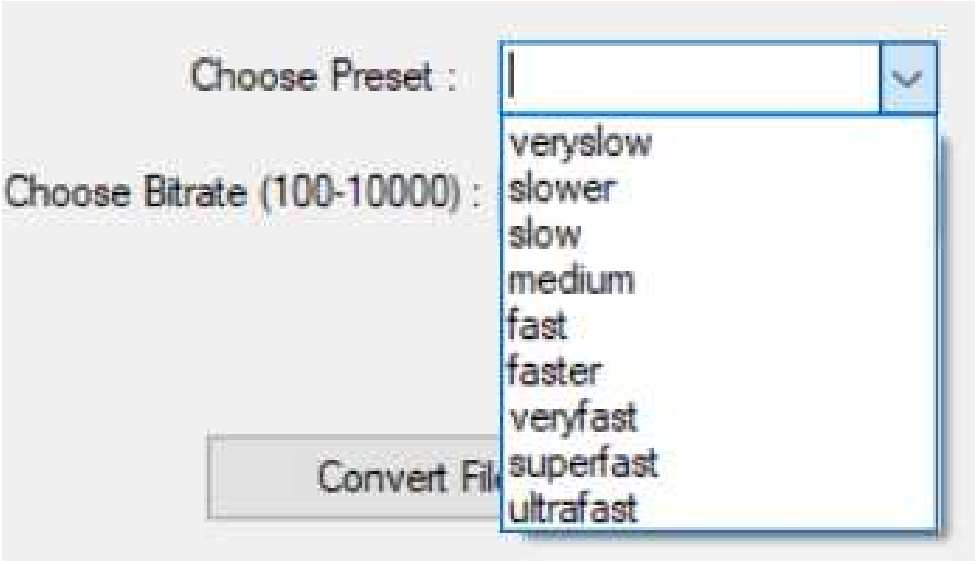


Рисунок 3.5 — Перелік пресетів

Після налаштувань форми можна приступати до безпосередньої розробки програми.

Кодування відеофайлу може зайняти повний проміжок часу, тому щоб уникнути “замороження” додатку та блокування основного потоку, метод, який відповідає за кодування відеофайлу повинен бути асинхронним. Для визначення

асинхронних методів використовується слово “async”. Також буде використовуватися “Task”, оскільки такі методи можуть виконувати паралельно декілька завдань і чекати на їх завершення. Метод, який повертає “Task”, виконується у фоновому режимі і можна використовувати “await”, щоб чекати на його завершення. “Await” – ключове слово, яке використовується для асинхронного очікування завершення операції, воно призупиняє виконання методу, доки асинхронна операція не завершиться.

Асинхронні методи можуть генерувати винятки, які слід обробляти. Необхідно використовувати блоки “try-catch” для обробки таких винятків в асинхронному коді.

У самому коді асинхронного методу перш за все необхідно підключитися до бібліотеки FFmpeg, вказавши шлях до неї на пристрої та створити нову змінну на базі класу FFmpeg з бібліотеки Xabe.FFmpeg.

До змінної вже можна додавати параметри, які будуть працювати з бібліотекою FFmpeg напряму.

Параметри вказуються у подвійних лапках за допомогою методу “AddParameter”. Якщо необхідно вказати значення певної змінної використовується базова конструкція C#: ($“{назва_змінної}”).

Код на C# для введення назви файлу або шляху до нього (змінна inputPath відповідає за шлях до файлу):

```
ffmpeg.AddParameter($“-i {inputPath}”)
```

Код на C# для використання кодеку H264 для кодування відео:

```
ffmpeg.AddParameter($“-c:v libx264”)
```

Код на C# для встановлення заданого постійного коефіцієнту швидкості (змінна crf):

```
ffmpeg.AddParameter($“-crf {crf}”)
```

Код на C# для встановлення заданого пресету (змінна preset):

```
ffmpeg.AddParameter($“-preset {preset}”)
```

Код на C# для встановлення заданого бітрейту в кілобітах/секунду (змінна bitrate):

ffmpeg.AddParameter($“-b:v {bitrate}k”)

Код на C# для визначення вихідного відеофайлу (змінна outputPath відповідає за назву та шлях до вихідного відеофайлу):

ffmpeg.AddParameter($“{outputPath}”) [13, 27-28, 40]

Виконання усіх цих команд по черзі в одному коді призведе виконання команди FFmpeg, подібні до якої команди вже розглядалися раніше та які відповідають за конвертацію відеофайлу за певними параметрами.

Введення CRF та бітрейту не є обов'язковим для роботи програми, вони використовуються лише для більш точного керування розміром та якості відеофайлу після його конвертації. Можна вказати обидва параметри або один із них. При вказанні обидвох параметрів бітрейт перекриє CRF. Вказання Preset є обов’язковим для роботи програми.

Наступним важливим кроком у роботі є встановлення фільтру, завдяки якому можна буде обирати лише відеофайли. Після натиснення на кнопку “Select Files” відкривається діалогове вікно у якому можна обрати файли. Вибір лише файлів із розширенням mp4 відбувається за допомогою методу “Filter”, який знаходиться у класі “OpenFileDialog”, у нього передаються параметри “MP4 files (*.mp4)|*.mp4|All files (*.*)|*.*”. Після цього не можна буде обрати файли з іншом розширенням. У подальшому можна додати інші розширення відеофайлів.

Після вибору потрібного відеофайлу та натиснення на кнопку “Ок” діалогове вікно закривається, а шлях до файлу з його назвою записується у текстове поле над кнопкою.

При натиснені на кнопку “Convert File” відбувається зчитування даних з полів: шлях до файлу вхідного файлу, шлях до вихідного файлу, CRF (вводити необов’язково, Preset та бітрейт (вводити не обов’язково).

Алгоритм роботи програми при встановлення усіх параметрів зображено на рисунку 3.6.

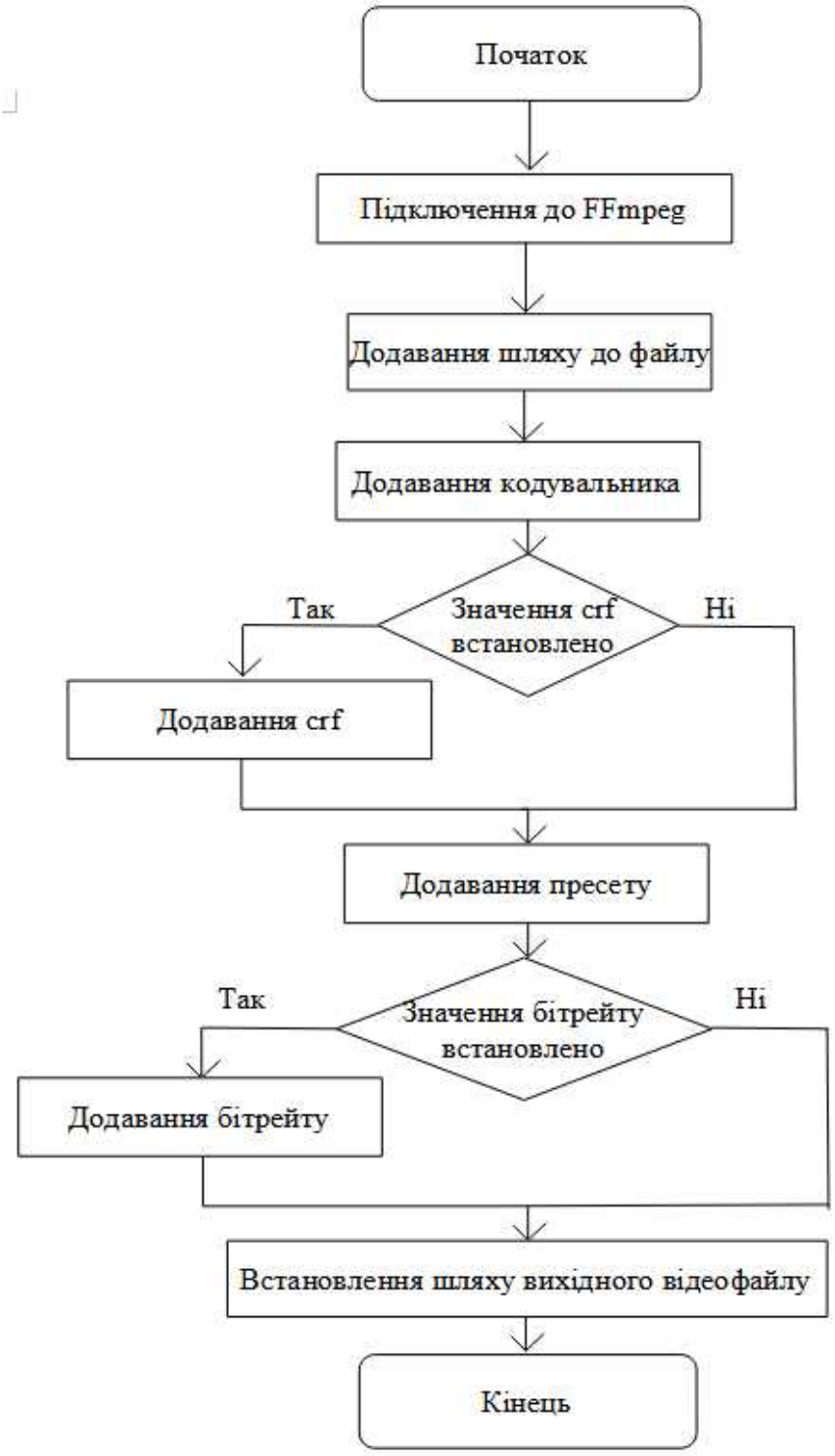


Рисунок 3.6 — Встановлення параметрів

Вигляд форми з введеними даними зображено на рисунку 3.7.

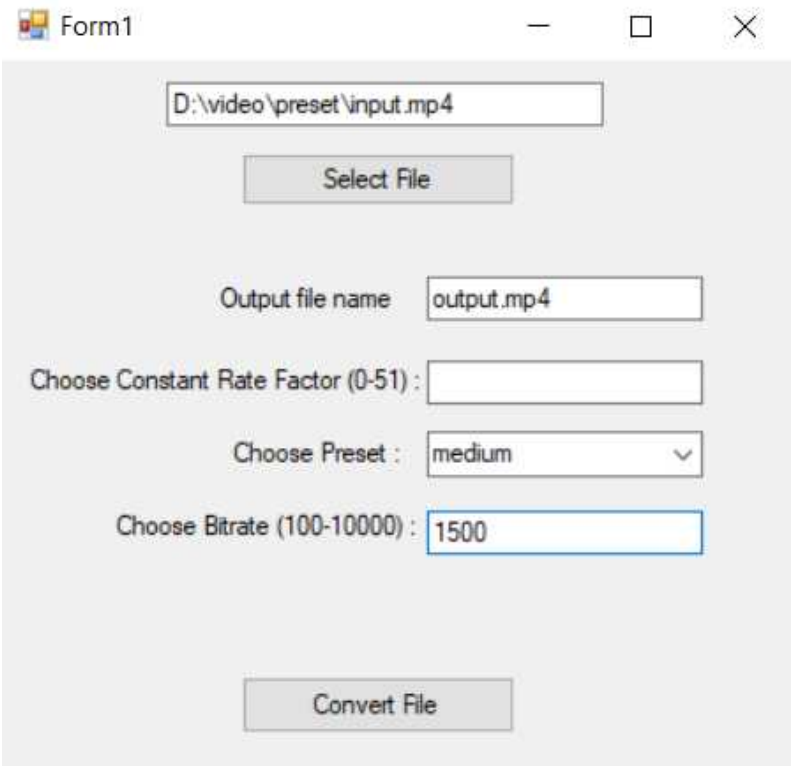


Рисунок 3.7 — Форма з введеними даними

Після початку кодування, якщо не відбулось ніяких помилок з’являється повідомлення про те, що робота почалась та вихідний файл з’явиться за декілька хвилин. Після закінчення роботи з’являється повідомлення про успішне кодування.

Приклад повідомлення про успішний початок кодування зображено на рисунку 3.8.

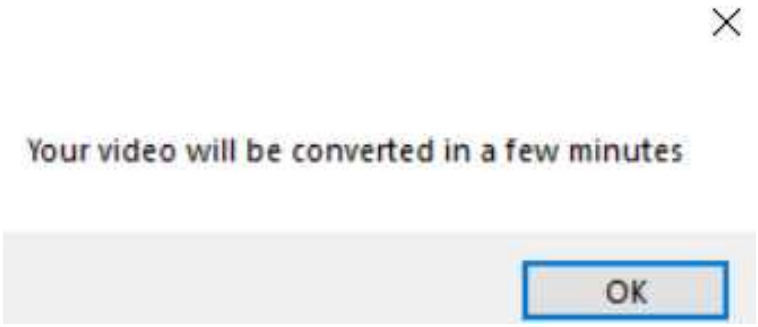


Рисунок 3.8 — Приклад повідомлення про успішний початок кодування

У асинхронному методі, який відповідає за конвертацію файлів реалізована конструкція try-catch, яка відповідає за виявлення помилок. При виявленні помилки з’являється спеціальне повідомлення зі змістом помилки, щоб можна була зрозуміти, що відбулось не так.

Приклад повідомлення з помилкою зображено на рисунку 3.9.

Рисунок 3.9 — Приклад повідомлення з помилкою

У цьому випадку програма на може знайти пошкоджений файл.

Після цього роботу над програмою можна вважати завершеною та можна приступати до аналізу відеофайлів до та після стиснення, щоб підібрати найкращі параметри для стиснення.

3.4 Висновки

Розроблена програма дозволяє кодувати відеофайли формату mp4 за допомогою кодеку h264, використовуючи різні параметри. Найпростішим параметром, який відповідає за якість відео є CRF та Preset, який відповідає за швидкість кодування та навантаження на процесор. Потрібно перевірити як відповідає вибір параметру CRF на вихідний розмір відеофайлу та його бітрейт. Для перевірки був обраний файл mp4 довжиною 4 хвилини 22 секунди, 60 кадрів в секунду, розміром 108 мб, бітрейт 3451 кбіт/с.

Таблиця 3.4 — Розміру відеофайлу з різними значеннями CRF

CRF	Preset	Вихідний бітрейт	Вихідний розмір (мб)
25	Fast	1136	36.1
25	Medium	1097	34.8
25	Slow	1044	33.2
26	Slow	953	30.3
27	Slow	871	27.7
30	Medium	720	22.1
35	Medium	380	12.3

При значеннях CRF більше 26-27 у відео вже присутні невеликі ознаки втрати якості. Між значеннями CRF 26 і 27 значення бітрейту становлять 871 та 720 кбіт/с, тому якщо стоїть мета максимально стиснути відеофайл — замість CRF можна використовувати бітрейт та встановити значення, наприклад 900. При

бітрейті 900 розмір відеофайлу становить 28.5 мб та не так помітні ознаки втрати якості.

Для різних відеофайлів значення можуть бути різними. Наприклад, для лекції, записаної у 10-20 кадрів в секунду можна встановлювати значення CRF до 30-35. Для відео зі швидкозмінюваними картинками такі значення становлять близько 26-28.

Після кодування відеофайл такий відеофайл можна зберігати на пристрої та відкривати будь-якою програмою або завантажувати на YouTube, чи інший відеохостинг, оскільки кодек H264 чудово підтримується усіма пристроями та платформами.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було проаналізовано основні методи стиснення відеофайлів та вибрано найкращі з них для виконання роботи, обрано найоптимальніші параметри для стиснення відеофайлів.

Основним кодеком було обрано H264, тому що відеофайлі, закодовані за допомогою H264, легко читаються усіма відеоплеєрами та підтримуються абсолютною більшістю платформ. Також H264 забезпечує найоптимальніший коефіцієнт стиснення при не дуже сильному навантажені на процесор під час кодування.

У першому розділі проведено аналіз платформ на яких можна зберігати відеофайли для довготривалого використання. Були розглянуті основні підходи до стиснення відеофайлів на яких базуються усі сучасні алгоритми. Була розглянута бібліотека FFmpeg, яка надає зручний та якісний доступ до конвертації та кодування відеофайлів для їх стиснення. В процесі роботи розглянуто основні програми для запису відеофайлів та засоби, які вони використовують під час роботи .

У другому розділі проведено аналіз існуючих рішень для стиснення відеофайлів. Проведено аналіз найпопулярніших відеокодеків для стиснення та відеокодеків, якими користуються сучасні відеохостинги.

Для стиснення відеофайлів було використано базові можливості FFmpeg. Також проведений аналіз популярних програм для стиснення та конвертації відеофайлів: Video to Video та Handbrake. У результаті стиснення було порівняно розміри вхідних та вихідних відеофайлів та проаналізовано параметри, які використовувалися для цього стиснення.

У третьому розділі був зроблений детальний аналіз форматів відеофайлів та кодувальників для запису відео, щоб обрати найоптимальніший варіант для

стиснення. Була розроблена програма на мові програмування C# з використанням бібліотеки Xabe.FFmpeg, яка надає доступ до FFmpeg.

Програма допомагає стиснути відеофайл за допомогою кодека H264 та використовує 3 основні параметри:

1) CRF – відповідає за якість відео після кодування;

2) Preset – відповідає за час кодування та навантаження на процесор під час кодування;

3) Bitrate – відповідає за якість відео (вказує на кількість біт, використовуваних для передачі даних в одиницю часу), може використовуватись замість CRF, для більш точного керування розміром відеофайлу.

Розміри та якість відеофайлів після обробки було проаналізовано для визначення найоптимальніших параметрів стиснення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hardrik S. Comparative Performance of Jm and FFmpeg codecs of H264 AVC video compression standard. San Diego State University, 2012. 82 p.

2. Uitto M. Energy consumption evaluation of H264 and HEVC video encoders in high-resolution live streaming. *VTT Technical Research Centre of Finland Ltd*., Oulu, Finland, 2013. P. 700-746.

3. Alvarez M., Salami E., Ramirez A., Valero M. HD-VideoBench. A Benchmark for Evaluating High Definition Digital Video Applications. *IEEE 10th International Symposium on Workload Characterization*, Boston, MA, USA. 2007. P. 188-201.

4. Uma Nair. Vimeo vs YouTube: Choosing the right video platform. 2023. URL: https://www.gumlet.com/learn/vimeo-vs-youtube/. (дата звернення: 16.02.2024).

5. Bhaskaran V., Konstantinedes K. Image and video compression standards: algorithms and architectures, 1997. 180 p.

6. Difference between inter and intra frame compression. URL: https://www.tutorialspoint.com/difference-between-inter-and-intra-frame-compression. (дата звернення: 12.03.2024).

7. Rahul Nanwani. Video encoding 101: A comprehensive guide. 2023. URL: https://imagekit.io/blog/video-encoding/. (дата звернення: 15.04.2024).

8. Akhtar A., Li Z. Inter-Frame compression for dynamic point cloud geometry coding. *IEEE Transactions on image processing*. January 2024. Vol. 33. P. 584-594.

9. Bingyao L. Sematics-Guided and Saliency-Focused Learning of Perceptual Video Compression. *IEEE Access 2024.* Vol. 12. P. 68611-68623.

10. Zeng H., Fang Y. Implementation of video transcoding client based on FFmpeg. *Advanced Materials Research 2013*. September 2013. Vol. 756-759. P. 1748-1752.

11. Auwera G., Prasanth T. D., Reisslein M. Traffic characteristics of H.264/AVC variable bit rate video. *2008 IEEE Communications Magazine*. November 2008. P. 164-174.

12. Merritt L., Vanam R. X264: A high performance H.264/AVC Encoder. University of Washington, Seattle. 2006. 13 p.

13. Документація FFmpeg. URL: https://ffmpeg.org/ffmpeg.html. (дата звернення: 05.05.2024).

14. OBS Studios's документація. URL: https://docs.obsproject.com/. (дата звернення: 16.02.2024).

15. HyperCam vs OBS Comparison Chart. URL: https://sourceforge.net/software/compare/HyperCam-vs-OBS/. (дата звернення: 18.02.2024).

16. Intel Quick Sync Video (QSV). URL: https://en.wikipedia.org/wiki/Intel_Quick_Sync_Video. (дата звернення: 11.04.2024).

17. Nvidia NVENC. URL: https://en.wikipedia.org/wiki/Nvidia_NVENC. (дата звернення: 11.04.2024).

18. Dumić E., Grgić S., Frank D., Šakić K. Subjective quality assessment of H.265 versus H.264 Video Coding for High-Definition Video Systems. *2015 13th International Conference on Telecommunications (conTEL)*. Graz, Austria. July 2015. 338 p.

19. Відеокодек H.264 (AVC). URL: https://uk.wikipedia.org/wiki/H.264. (дата звернення: 03.03.2024).

20. Yash Patadia. The state of video codecs in 2024. 2024. URL: https://www.gumlet.com/learn/video-codec/. (дата звернення: 01.04.2024).

21. Відеокодек H.265 (HEVC). URL: https://uk.wikipedia.org/wiki/H.265. (дата звернення: 03.03.2024).

22. Shashikanth Manjunatha. A comprehensive guide on H.265 codec. 2024. URL: https://www.gumlet.com/learn/what-is-h265/. (дата звернення: 12.03.2024).

23. Akshay Bhonde. H.266 – A guide to Versatile video Coding. 2023. URL: https://www.gumlet.com/learn/h266-codec/. (дата звернення: 12.03.2024).

24. Akshay Bhonde. VP9 codec – What is VP9. 2024. URL: https://www.gumlet.com/learn/vp9-codec/. (дата звернення: 15.03.2024).

25. Алгоритм роботи VC-1. URL: https://wiki.multimedia.cx/index.php/VC-1.

26. Ashik TS. AV1 vs VP9: The battle of the Codecs. 2024. URL: https://www.gumlet.com/learn/av1-vs-vp9/. (дата звернення: 18.03.2024).

27. Інструменти та компоненти FFmpeg. URL: https://ffmpeg.org/ffmpeg-all.html. (дата звернення: 02.03.2024).

28. H.264 video encoding guide. URL: https://trac.ffmpeg.org/wiki/Encode/H.264. (дата звернення: 26.02.2024).

29. H.265/HEVC video encoding guide. URL: https://trac.ffmpeg.org/wiki/Encode/H.265. (дата звернення: 26.02.2024).

30. FFmpeg and VP9 Encoding Guide. URL: https://trac.ffmpeg.org/wiki/Encode/VP9. (дата звернення: 26.02.2024).

31. Video to Video Converter документація. URL: https://www.videotovideo.org/help/. (дата звернення: 07.03.2024).

32. Handbrake документація https://handbrake.fr/docs/en/.

33. Video compression picture types. URL: https://en.wikipedia.org/wiki/Video_compression_picture_types. (дата звернення: 11.03.2024).

34. MP4 file format. URL: https://en.wikipedia.org/wiki/MP4_file_format. (дата звернення: 10.04.2024).

35. Avi file format. URL: https://en.wikipedia.org/wiki/Audio_Video_Interleave. (дата звернення: 10.04.2024).

36. MKV file format. URL: https://en.wikipedia.org/wiki/Matroska. (дата звернення: 10.04.2024).

37. Joey Davis. The Shift from hardware to software codecs: five ways it`s revolutionizing the digital media landscape. 2023. URL: https://www.avixa.org/pro-av-trends/articles/the-shift-from-hardware-to-software-codecs-five-ways-it-s-revolutionizing-the-digital-media-landscape. (дата звернення: 11.04.2024).

38. Xabe.FFmpeg tutorial. URL: https://ffmpeg.xabe.net/tutorial.html. (дата звернення 03.05.2024).

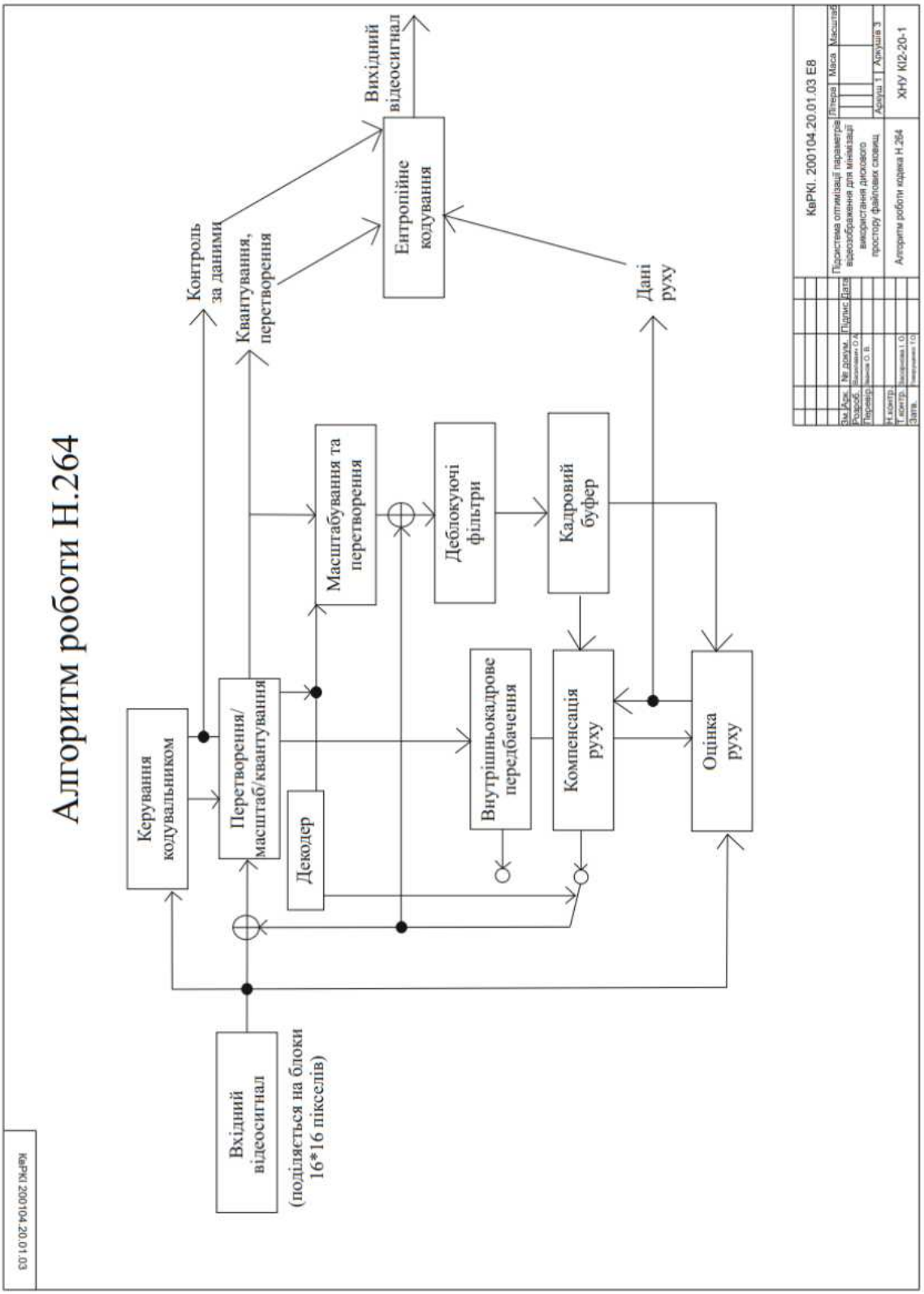
39 Xabe.FFmpeg on nuget packages. URL: https://www.nuget.org/packages/Xabe.FFmpeg. (дата звернення: 03.05.2024).

40. Xabe.FFmpeg офіційна документація. URL: https://ffmpeg.xabe.net/docs.html. (дата звернення: 03.05.2024).

Додаток А

(обов’язковий)

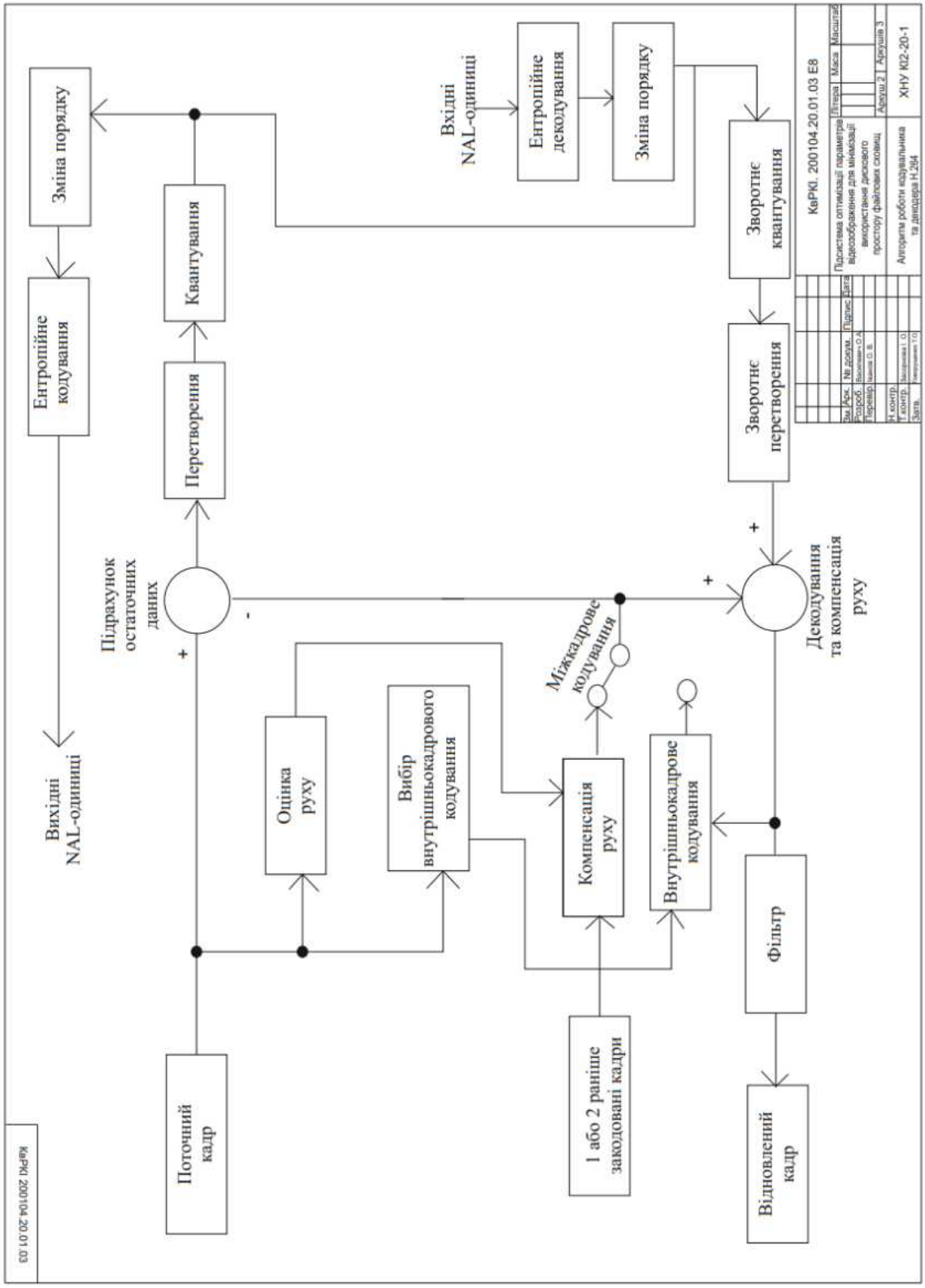
Копія креслення «Алгоритм роботи Н.264»



Додаток Б

(обов’язковий)

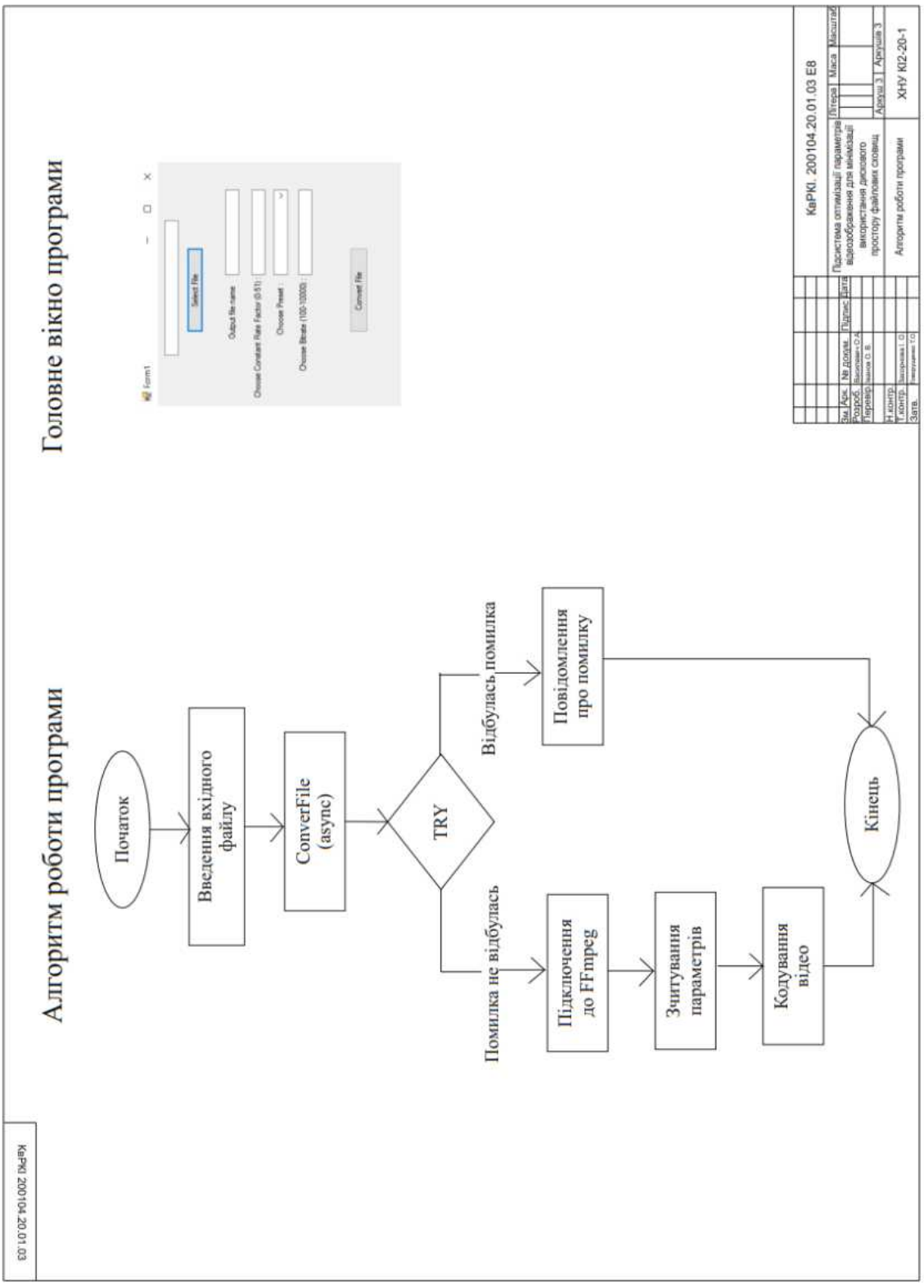
Копія креслення «Алгоритм роботи кодувальника та декодера H.264»



Додаток В

(обов’язковий)

Копія креслення «Алгоритм роботи програми»



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016323998

Дата перевірки:
05.06.2024 16:00:00 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
05.06.2024 18:14:54 EEST

ID користувача:
100005591

Назва документа: **Василевич_Підсистема оптимізації параметрів відеозображення для мінімізації використа...**

Кількість сторінок: 71 **Кількість слів:** 11987 **Кількість символів:** 88982 **Розмір файлу:** 769.05 KB **ID файлу:** 1016122619

2.17%
Схожість

Найбільша схожість: 0.73% з Інтернет-джерелом (http://elar.khmnu.edu.ua/jspui/bitstream/123456789/13884/1/%d0%9f..

2.06% Джерела з Інтернету	145	Сторінка 73
1.17% Джерела з Бібліотеки	161	Сторінка 73

0.04% Цитат

Цитати	1	Сторінка 74
Посилання	1	Сторінка 74

0%
Вилучень

Немає вилучених джерел

Wed Jun 05 14:40:47 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

ID: 128571 Назва: БКР Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ Додано в БД: 2024-06-05 Автора: О. А. Василевич Керівники: О. В. Іванов Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	71087	1053	883 (1%)	12 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Василевич Олексій Андрійович

Тема: Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___64___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено аналіз предметної області, а саме: проаналізовано принципи роботи основних підходів до стиснення відеофайлів, розглянуто бібліотека FFmpeg та принципи роботи з нею, проведено аналіз основного програмного забезпечення для запису відеофайлів, також виконано постановку задачі. В другому розділі кваліфікаційної роботи було проведено аналіз найпопулярніших відеокодеків для стиснення та відеокодеків, якими користуються сучасні відеохостинги, для стиснення відеофайлів було використано базові можливості FFmpeg, також проведений аналіз популярних програм для стиснення та конвертації відеофайлів, а саме: Video to Video та Handbrake, результаті стиснення було порівняно розміри вхідних та вихідних відеофайлів. В третьому розділі кваліфікаційної роботи була розроблена програма оптимізації параметрів відеофайлів на мові програмування C# з використанням бібліотеки Xabe.FFmpeg, розміри та якість вихідних відеофайлів було проаналізовано для визначення найбільш оптимальних параметрів.

4. Позитивні сторони роботи: Виконана робота має високу практичну цінність.

5. Негативні сторони роботи: недостатня увага реалізації спроєктованої підсистеми.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (4.0/C)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Яцельчук Тамара Іванівна, канд. техн. наук, доцент кафедри інженерії програмного забезпечення

"13" червня 2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Василевича Олексія Андрійовича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Підсистема оптимізації параметрів відеозображення для мінімізації використання дискового простору файлових сховищ

Автор: Василевич Олексій Андрійович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Іванов Олексій Валентинович, к. т. н., доц. каф. КІІС

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості складає 2.17% і адресується до 306 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи О. В. Іванов

Гарант ОП С. М. Лисенко

Завідувач кафедри КІІС Т. О. Говорущенко