МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна інженерія»

Група: 2БКС-29

КВАЛІФІКАЦІЙНА РОБОТА

здобувача освіти денної форми навчання

БКС.29.19.000.КРБ

ПРОХОРОВА МИХАЙЛА МИХАЙЛОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-29**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів**

Проектний матеріал складається з пояснювальної записки на 77 сторінках та графічного (презентаційного) матеріалу на 17 аркушах (слайдах)

Виконавець _____ (Прохоров М.М.)

Керівник проекту _____ (Кривченко Ю.В.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «28» 06 2025 р. Протокол ЕК № 3

Оцінка ЕК 4 (добре) 180

Секретар ЕК _____

АНОТАЦІЯ

У даній кваліфікаційній роботі представлено комплексний аналіз ефективності алгоритмів стискання відео-даних для стрімінгових сервісів, що має важливе значення для оптимізації потокової передачі мультимедійного контенту. Основною метою роботи є дослідження популярних алгоритмів кодування відео (H.264, H.265, VP9), оцінка їх продуктивності та визначення компромісу між якістю зображення та розміром стисненого відеофайлу.

Розроблено інтерактивний застосунок у середовищі на базі мови Python, що дозволяє проводити автоматизоване тестування алгоритмів стискання. В основі розробки лежать бібліотеки OpenCV, NumPy, Matplotlib, ipywidgets та інструмент ffmpeg, що забезпечують ефективну обробку відео, розрахунок метрик якості (MSE, PSNR) та візуалізацію отриманих результатів.

Експериментальна частина роботи включає тестування відеофайлів із різними параметрами стискання, акумулювання отриманих даних у таблицях та побудову трендових графіків залежностей основних показників (PSNR, MSE, розміру файлу, часу кодування) від параметра CRF. Аналіз результатів демонструє, що H.265 забезпечує найкращу компресію при збереженні високої якості зображення, тоді як VP9 дозволяє отримати мінімальний розмір файлу, але за рахунок збільшеного часу кодування.

Отримані у роботі висновки дозволяють визначити оптимальні параметри стиснення для стрімінгових сервісів залежно від вимог до пропускної здатності мережі та рівня деталізації відео. Робота створює основу для впровадження адаптивних алгоритмів відеокодування та їх подальшої оптимізації у сучасних мультимедійних системах.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"28" 05 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Прохорову Михайлу Михайловичу*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів*

затверджена наказом по коледжу від "24" 4 2024 р. № 246

2. Термін здачі студентом кваліфікаційної роботи 20.06.25

3. Вихідні дані до роботи *1. Підтримка основних відеоформатів (mp4, avi, mov, mkv, webm) для завантаження вихідного відео; 2. Забезпечити можливість тестування стискання за допомогою алгоритмів H.264, H.265 та VP9; 3. Реалізувати регулювання CRF в діапазоні від 10 до 40; 4. Забезпечити розрахунок об'єктивних показників – MSE, PSNR, для порівняння якості стисненого відео з еталонним; 5. Реалізувати обчислення розмірів вихідного і стисненого відеофайлів для порівняльного аналізу ефективності алгоритмів*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
Огляд основних понять та класифікація алгоритмів стискування відео-даних; Аналітичний огляд основних відеостандартів; Інтеграція алгоритмів стискування у стрімінгові сервіси; Об'єктивна оцінка якості відеозображень; Розробка застосунку для аналізу ефективності алгоритмів відео-даних для стрімінгових сервісів; Експериментальний аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

5. Перелік графічного матеріалу (слайдів мультимедійної презентації) *Шлях відеосигналу від джерела до глядача стрімінгового сервісу; Порівняння кодеків для стиснення відео-даних (H.264, H.265, VP9); Схеми і стратегії стискування відео-даних; Схеми основних етапів кодування за стандартами MPEG-2, H.264/AVC, H.265/HEVC; Схема адаптивного алгоритму стискування для стрімінгу; Порівняння показників якості результатів стискування відео-даних; Архітектура застосунку для аналізу ефективності алгоритмів відео-даних; Інтерфейс застосунку для аналізу ефективності алгоритмів; Результати тестування ефективності алгоритмів; Порівняння параметрів ефективності алгоритмів при стискуванні відео-даних*

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що їх стосуються

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Кривченко Ю.В.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник роботи *Кривченко Ю.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Вступ. Аналіз технічного завдання	07.06.25	
2.	Огляд основних понять та класифікація алгоритмів	08.06.25	
	стискування відео-даних	09.06.25	
3.	Аналітичний огляд основних відеостандартів	10.06.25	
4.	Інтеграція алгоритмів стискування у стрімінгові сервіси	11.06.25	
5.	Об'єктивна оцінка якості відеозображень	12.06.25	
6.	Розробка застосунку для аналізу ефективності алгоритмів	13.06.25	
7.	Експериментальний аналіз ефективності алгоритмів	14.06.25	
	стискування відео-даних	15.06.25	
8.	Реалізація інтерфейсу застосунку	16.06.25	
9.	Випробування застосунку	17.06.25	
10.	Аналіз результатів тестування	18.06.25	
11.	Розробка питань з охорони праці та техніки безпеки	19.06.25	
12.	Підготовка матеріалів мультимедійної презентації	20.06.25	

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 29. 19 000. 00 КРБ	Випускна робота		
А4			БКС 29. 19 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 29. 19 000. 00 КРБ		
Розробив		Прохоров М.М.		16.06	Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів	Літ.	Аркуш 4 / Аркушів 77
Перевірив		Кривченко Ю.В.		16.06		Н Д П	
Н. Контр.		Петрашова В.І.		20.06.25	ВСП "ОТФК ОНТУ" гр.2БКС-29		
Затверд.		Іванова Л.В.		20.06.25			

ЗМІСТ

ВСТУП

У сучасному світі стрімінгові послуги стали невід'ємною частиною інформаційного простору, що забезпечують миттєву доставку відеоконтенту широкій аудиторії. Широкосмуговий інтернет, мобільні пристрої та зростаючий попит на високоякісне відео стимулюють постійний розвиток технологій стиснення відео-даних. Ефективне стиснення дозволяє зменшити розмір потоків з відеоконтентом, зберігаючи при цьому прийнятну якість зображення, що є особливо важливим для стрімінгових сервісів, де ресурси мережі обмежені, а вимоги до якості стрімінгу високі.

Актуальність дослідження зумовлена необхідністю оптимізації алгоритмів стискання відео для досягнення максимального співвідношення між економністю бітрейту та збереженням візуальної якості. Історично застосовувані стандарти, такі як MPEG-2, поступово витісняються сучаснішими рішеннями, серед яких H.264/AVC набув популярності завдяки своїй здатності забезпечувати значно вищу ефективність стиснення при збереженні високої структурної подібності з оригінальним зображенням. Проте, питання вибору оптимального алгоритму стискання залишається відкритим, оскільки різні методики мають свої переваги та недоліки, що виявляються як у теоретичних, так і в практичних аспектах. Водночас, сучасні об'єктивні методи оцінки якості відеозображень із застосуванням таких метрик, як PSNR, SSIM, MSAD, VQM та ін., дозволяють порівнювати ефективність різних кодеків на основі числових характеристик, що є важливим розрахунковим інструментом для розробників та операторів стрімінгових сервісів.

Мета даної роботи полягає в аналізі ефективності алгоритмів стискання відео-даних з використанням методик стискання з втратами у контексті стрімінгових сервісів. Для досягнення поставленої мети необхідно забезпечити експериментальне дослідження ефективності обраних кодеків на основі реальних відео-даних, із застосуванням як класичних, так і сучасних метрик якості; розробити та інтегрувати web-застосунок, який дозволятиме проводити тестування ефективності алгоритмів стискання відео для стрімінгових сервісів, забезпечуючи візуальне порівняння отриманих результатів.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд основних понять та класифікація алгоритмів стискування відео-даних

Стиснення відео є надзвичайно важливим етапом у цифровій обробці сигналів, оскільки забезпечує зменшення обсягу інформації, що передається чи зберігається, без втрати критично важливих даних, які впливають на сприйняття зображення людиною. Основна ідея стискання полягає у виявленні і усуненні надлишковості у відеоданих. Надлишковість може мати як просторовий, так і часовий характер. Просторова надлишковість виникає завдяки тому, що сусідні пікселі часто мають подібні значення, а часовий аспект пов’язаний із малою зміною вмісту між послідовними кадрами відео. Завдяки йому можливо передавати різницеву інформацію, тобто зберігати лише зміни від одного кадру до наступного.

Алгоритми стискування відео-даних можуть поділятися на два основних типи: стискання без втрат та стискання з втратами. Стиснення без втрат дозволяє повністю відновити вихідні дані під час декодування, що є необхідним у певних сферах, наприклад, архівуванні чи обробці медичних зображень. Проте в багатьох випадках, особливо при передачі потокового відео, прийнятною є невелика втрата інформації, що сприяє суттєвому зменшенню обсягу даних. Саме тому стискання з втратами набуло широкого застосування у сфері цифрового відео.

Основу алгоритмів стиснення становлять методи перетворення сигналу, серед яких виділяється дискретне косинусне перетворення (ДКП). Застосування ДКП дозволяє розкласти зображення на набір ортогональних базисних функцій, де більшість енергії концентрується у низьких частотах, що дозволяє зменшити кількість коефіцієнтів, необхідних для відтворення первинного зображення. Подібним чином працює і двовимірне ДКП, яке застосовується у випадках роботи з блоками пікселів (найчастіше розміром 8×8).

Ще одним важливим компонентом алгоритмів стиснення є квантований етап, на якому амплітуди коефіцієнтів перетворення округлюються до певного значення із застосуванням вагових матриць. Саме цього етапу відноситься найбільший внесок у зниження бітрейту і водночас він визначає рівень втрат у стисканні. Правильне

визначення параметрів квантування є ключовим для досягнення компромісу між ступенем стиснення та якістю відновленого зображення.

Ключову роль у процесі зменшення обсягу даних відіграє і компенсація руху, яка використовується для скорочення часової надлишковості. За допомогою оцінки переміщення об’єктів між кадрами можливо передавати різницеву інформацію між послідовними кадрами, що значно знижує потреби у передачі повного зображення кожного разу. Ефективність методів компенсації руху залежить від точності визначення векторів руху та правильної організації груп кадрів (GOP).

Алгоритми стискування можна класифікувати також за структурою алгоритмічних блоків — від базових, що включають перелічені вище перетворення та квантовані операції, до складних систем, де ці блоки інтегруються з методами адаптивного прогнозування, регулювання бітрейту, обробки потокових даних та специфічних процедур синхронізації. Сучасні стандарти, такі як MPEG-2 і H.264/AVC, демонструють різні підходи до організації процесу стискання: MPEG-2, будучи старішим стандартом, має більш просту архітектуру з обмеженими можливостями адаптивного прогнозування, тоді як H.264/AVC забезпечує більш високий ступінь стиснення і кращу якість відновлення завдяки використанню більш ефективних алгоритмів прогнозування та особливої обробки меж блоків.

Основні поняття, що лежать в основі алгоритмів стискування відео-даних, охоплюють виявлення надлишковості (як просторової, так і часової), застосування перетворення сигналу для розкладу зображення у частотну область, квантовану обробку коефіцієнтів, а також техніки компенсації руху для оптимізації обсягу даних у часовій послідовності. Ця класифікація дозволяє розглядати алгоритми стискування як комплексні системи, що складаються з окремих взаємодіючих блоків, кожен з яких відповідає за певний аспект зниження обсягу даних, забезпечуючи при цьому прийнятний рівень відновлення вихідного зображення.

1.1.1 Різницеве стиснення: просторова та часово-просторова надлишковість

Різницеве стиснення базується на принципі зниження надлишковості сигналу шляхом кодування лише змін (різниць) між значеннями пікселів або кадрів,

що дозволяє значно зменшити обсяг переданої чи збереженої інформації (рис.1.1). У контексті відео-даних надлишковість можна умовно розділити на два основних типи:

1. Просторова надлишковість. Вона виникає через те, що пікселі, розташовані поруч один з одним в одному кадрі, часто мають дуже подібні значення. Це пов'язано з природною гладкістю зображення. Для кодування просторової надлишковості застосовують різницеве стиснення, де замість повної інформації про піксель передається відхилення від певного передбачуваного значення.

Нехай $I(i,j)$ — яскравість пікселя в позиції *(i,j)* кадру, а $\mu_{N(i,j)}$ — середнє значення яскравості для його сусідніх пікселів (наприклад, для блоку розміром 3×3). Тоді різницю (залишковий сигнал) можна обчислити за формулою:

$$d_s(i,j) = I(i,j) - \mu_{N(i,j)} \qquad (1.1)$$

де $\mu_{N(i,j)} = \frac{1}{n}\sum_{(k,l)\in N(i,j)} I(k,l)$,

n — кількість пікселів у сусідньому вікні *N(i,j)*.

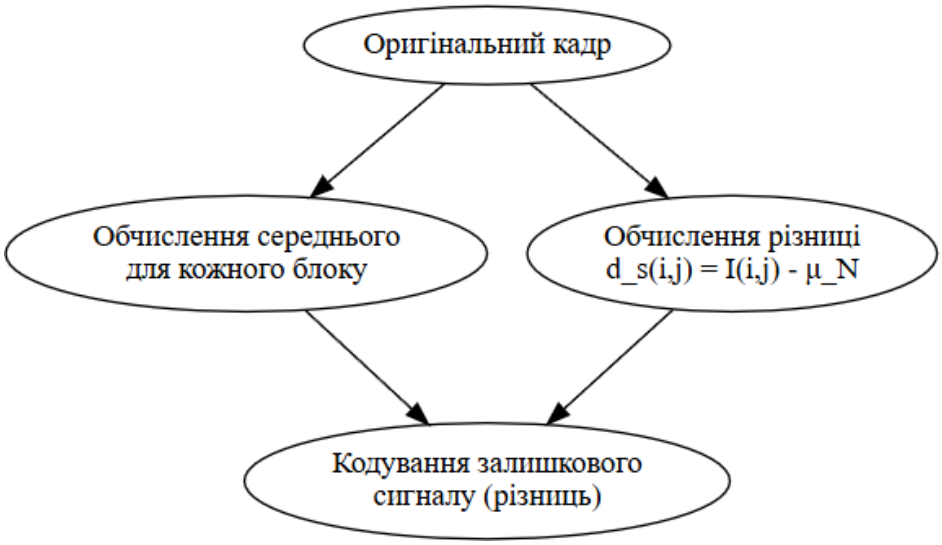


Рисунок 1.1. Схема просторового різницевого стиснення

Часово-просторова надлишковість виникає завдяки тому, що між послідовними кадрами багато спільної інформації (рис.1.2). Сценарії з невеликими змінами (наприклад, статичний фон або повільний рух об'єктів) дозволяють застосовувати методи міжкадрового прогнозування та компенсації руху. Основна ідея полягає у визначенні прогнозного значення поточного кадру на основі попереднього (чи майбутнього), а потім кодуванні різниці між реальним

зображенням і прогнозом.

Нехай $I_n(i,j)$ — яскравість пікселя в поточному кадрі n, а $I_{n-1}(i,j)$ — значення того самого пікселя у попередньому кадрі. Припустимо, що за допомогою алгоритму компенсації руху ми визначили вектор зсуву $(\Delta x, \Delta y)$ для локальної області. Тоді залишковий сигнал після компенсації руху визначається як:

$$d_t(i,j) = I_n(i,j) - I_{n-1}(i-\Delta x, j-\Delta y) \qquad (1.2)$$

Ця формула дозволяє зафіксувати лише змінну частину зображення, котра зумовлена рухом та іншими динамічними змінами, що значно менше за обсяг даних у випадку кодування повного кадру.

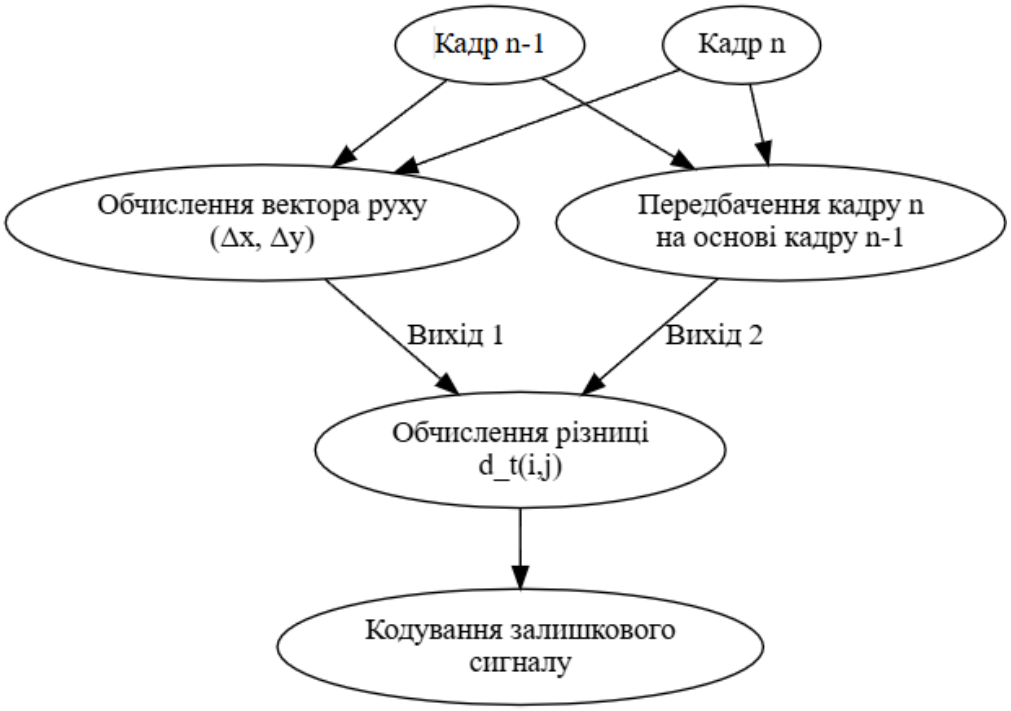


Рисунок 1.2. Схема часово-просторового різницевого стиснення

Таблиця 1.1. Порівняльна характеристика просторової та часово-просторової надлишковості

Тип надлишковості	**Джерело**	**Приклад**	**Метод зменшення обсягу даних**
Просторова	Схожість сусідніх пікселів	Однорідні ділянки	Обчислення відхилень (залишковий сигнал)
Часово-просторова	Схожість послідовних кадрів	Невелика зміна сцени	Міжкадрове прогнозування та компенсація руху $d_t(i,j)$

Застосування різницевого стиснення дозволяє ефективно використовувати як просторову, так і часову надлишковість відео-сигналу. Завдяки обчисленню залишкових сигналів (*residuals*) від першопочаткових значень, що можуть бути прогнозовані за інформацією сусідніх пікселів або попередніх кадрів, досягається суттєве зменшення обсягу даних, що передаються чи зберігаються. Представлені схематичні рисунки та табличне порівняння ілюструють принципи різницевого стиснення, а математичні формули окреслюють основні розрахункові підходи, що використовуються у сучасних відеокодеках, таких як MPEG-2 та H.264/AVC.

Відповідно до схеми, зображеної на рис.1.2 та табл.1.2:

1. Вузли:
   - A: «Кадр n-1» – представляє попередній кадр;
   - B: «Кадр n» – поточний кадр;
   - C: «Обчислення вектора руху (Δx, Δy)» – із кадрів A та B визначається параметр переміщення;
   - D: «Передбачення кадру n на основі кадру n-1» – формуються прогнозні значення для поточного кадру;
   - E: «Обчислення різниці $d_t(i,j)$» – обчислюється різниця між фактичними значеннями та прогнозом;
   - F: «Кодування залишкового сигналу» – залишковий сигнал (різниці) кодується для суттєвого зниження обсягу даних;
2. Зв’язки:
   - Вузли A та B направляють дані як до вузла C (обчислення вектора руху), так і до вузла D (формування прогнозу);
   - Далі, результати вузлів C та D «зливаються», передаючи інформацію до вузла E, де обчислюється різниця $d_t(i,j)$;
   - Нарешті, з вузла E дані передаються до вузла F для кодування залишкового сигналу.
3. Часово-просторова частина цього алгоритму математично представляється наступною формулою:

$$d_t(i,j) = I_n(i,j) - I_{n-1}(i-\Delta x, j-\Delta y) \qquad (1.3)$$

де:

- $I_n(i,j)$ – значення пікселя в позиції ((i,j)) поточного кадру «n»,
- $I_{n-1}(i-\Delta x, j-\Delta y)$ – прогнозоване значення пікселя на основі інформації з попереднього кадру з врахуванням вектора руху $(\Delta x, \Delta y)$.

Таблиця 1.2. Етапи часово-просторового різницевого стиснення

Етап	Вхідні дані	Операція	Вихідні дані
1. Обчислення вектора руху	Кадри n-1 і n	Визначення $(\Delta x, \Delta y)$	Вектор руху
2. Формування прогнозу	Кадр n-1	Прогнозування кадру n	Прогноз кадру
3. Обчислення різниці	Поточний кадр і прогноз	Обчислення $d_t(i,j) = I_n(i,j) - I_{n-1}(i-\Delta x, j-\Delta y)$	Обчислений залишковий сигнал
4. Кодування залишкового сигналу	Залишковий сигнал	Кодування для зниження бітрейту	Стиснений відеопотік

1.1.2 Стратегії стискування: стиснення без втрат та з втратами

Основна мета стискування відео-даних полягає у зменшенні обсягу інформації, що зберігається або передається, без суттєвого впливу на якість відтворення відео. Для цього існують два основних підходи, які відрізняються рівнем збереження вихідної інформації:

1. Стиснення без втрат. При такій стратегії інформація стискається таким чином, що після декодування можна відновити точну копію вихідного сигналу. Це досягається за допомогою статистичних методів кодування, таких як алгоритм Хаффмана, арифметичне кодування або алгоритми Lempel-Ziv (LZ77, LZ78). Формальне представлення: Позначимо I як вихідний відео-сигнал, а C — стиснений код. Для ідеального кодування без втрат маємо: $C = f(I)$, а процес чергового декодування відбувається за інверсною функцією:

$$I = f^{-1}(C) \tag{1.4}$$

де f — оборотна (інвертована) функція кодування.

Схема стискання без втрат наведена на рис.1.3.

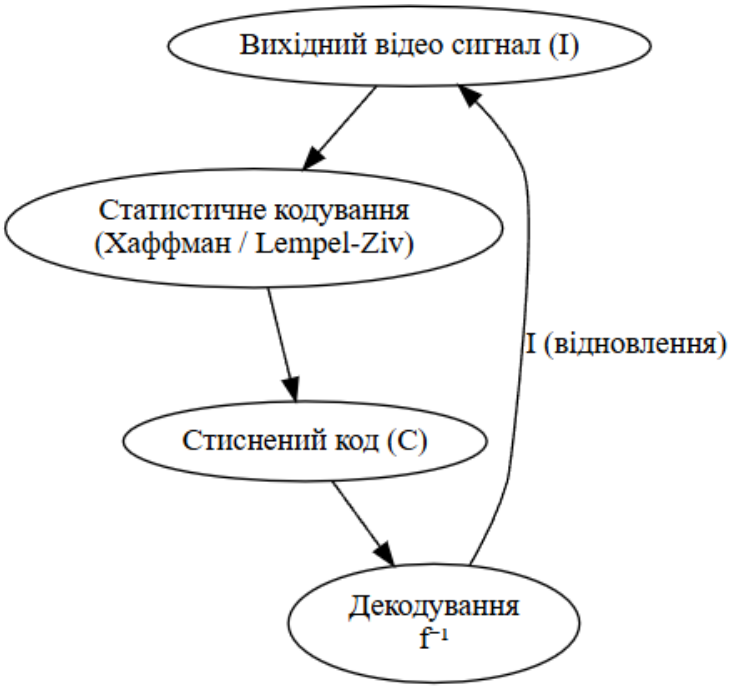


Рисунок 1.3. Схема стискання без втрат

2. Стиснення з втратами. У стратегії стиснення з втратами деякі дані відкидаються. Це обґрунтовано тим, що людське око менш чутливе до деталей високої частоти, що дозволяє застосувати перетворення сигналу для ізоляції менш важливої інформації. Основними етапами такого стиснення є застосування дискретного косинусного перетворення (ДКП) для розкладу зображення на базисні функції, квантоване округлення отриманих коефіцієнтів із застосуванням вагових матриць і подальше кодування залишкового сигналу. Формальне представлення: Нехай I — вихідний сигнал. Спочатку здійснюється перетворення: $X = DCT(I)$ Потім коефіцієнти піддаються квантуванню:

$$C' = Q(X) \qquad (1.5)$$

де $Q(\cdot)$ — оператор квантованого округлення. При декодуванні використовується обернений процес: $I' = DCT^{-1}\left(Q^{-1}(C')\right)$. Тут I' — відновлений сигнал, який відрізняється від I через втрати інформації в процесі квантованого округлення. Схема стискання з втратами наведена на рис. 1.4.

Обидві стратегії стискування відео-даних знаходять своє місце залежно від вимог до збереження точності і ефективності передачі інформації (табл.1.3). Стиснення без втрат гарантує відновлення оригінального відео, але забезпечує обмежене зменшення розміру даних. Натомість стискнення з втратами ж дозволяє

досягнути значно вищих коефіцієнтів стиснення за рахунок відсікання менш важливої для сприйняття інформації, що робить його оптимальним для стрімінгових сервісів і сучасних мультимедійних застосувань.

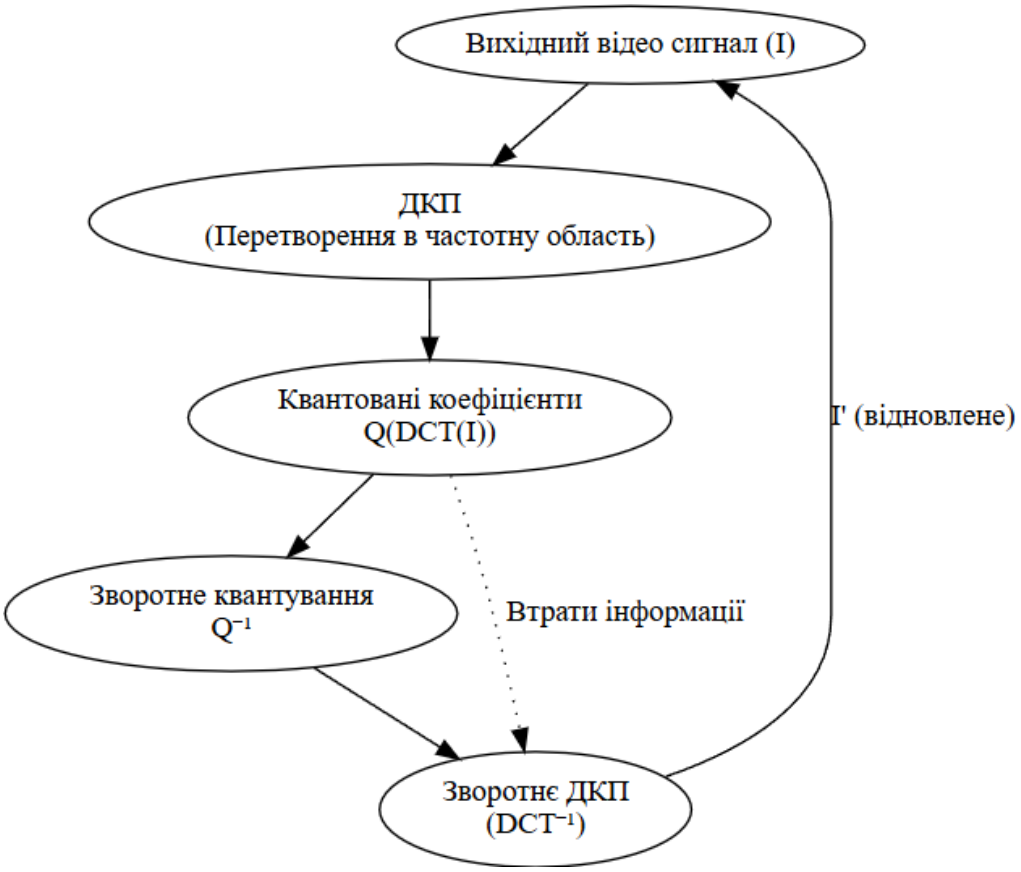


Рисунок 1.4. Схема стискання з втратами

Таблиця 1.3. Порівняльна характеристика стратегій стискування

Параметр	**Стиснення без втрат**	**Стиснення з втратами**
Відновлення даних	Повне (оригінальні дані відновлюються)	Часткове (відновлення із втратами, відмінне від оригіналу)
Коефіцієнт стискання	Нижчий, зазвичай 2–3:1	Вищий (може сягати десятків або більше:1)
Методи кодування	Статистичні методи (Хаффман, арифметичне, LZ алгоритми)	Перетворення (ДКП), квантоване округлення, прогнозування
Застосування	Архівування, професійне редагування, медичні зображення	Онлайн-стрімінг, мобільні відеосервіси, споживчі системи
Обчислювальна складність	Зазвичай нижча	Вища (через додаткові обрахунки перетворення та квантування)

1.1.3 Дискретне косинусне перетворення та його двовимірна версія

Дискретне косинусне перетворення (ДКП, Discrete Cosine Transform, DCT) є одним із ключових інструментів у сфері стиснення зображень і відео. Воно перетворює сигнал (або блок пікселів) з просторової області у частотну, що дозволяє ефективно представити інформацію за допомогою набору коефіцієнтів, більшість яких при роботі із натуральними зображеннями мають малі (або близькі до нуля) значення. Це дає можливість відсіяти менш значущі деталі за допомогою квантування, що є основою стиснення з втратами.

Для послідовності чисел *x(n)*, де $n = 0, 1, 2, \dots, N-1$, одновимірне ДКП визначається за правилом:

$$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right], \quad k = 0,1, \dots, N-1, \quad (1.6)$$

Це перетворення дозволяє представити вихідний сигнал як суму косинусних базисних функцій з певними амплітудами *X(k)*.

Для обробки зображень, які зазвичай представлені як двовимірні сітки пікселів, застосовують двовимірне ДКП. Нехай *I(i,j)* – яскравість пікселя блоку розміром $N \times M$, де $i = 0, 1, \dots, N-1$ та $j = 0, 1, \dots, M-1$. Двовимірне ДКП визначається наступною формулою:

$$X(u,v) = \alpha(u), \alpha(v) \sum_{i=0} N-1 \sum_{j=0} M-1 \, I(i,j) \cos\left[\frac{\pi(2i+1)u}{2N}\right] \cos\left[\frac{\pi(2j+1)v}{2M}\right] \quad (1.7)$$

де $u = 0, 1, \dots, N-1$ і $v = 0, 1, \dots, M-1$, а коефіцієнти нормалізації $\alpha(u)$ і $\alpha(v)$ мають вид, аналогічний до одновимірного перетворення.

Це дозволяє отримати матрицю коефіцієнтів, де головний елемент *X(0,0)* (DC-компонента) відображає середню яскравість блоку, а інші (AC-компоненти) – інформацію про різні частотні складові.

На рис.1.5 наведено схематичне представлення процесу обчислення двовимірного ДКП із застосуванням послідовного одновимірного перетворення по рядках і потім по стовпцях. Цей схематичний план демонструє послідовну обробку:

- Крок 1: Вхідний блок зображення розбивається на окремі ряди;
- Крок 2: Для кожного ряду застосовується одновимірне ДКП, що дає проміжну

матрицю;

– Крок 3: Проміжна матриця транспонується, і для кожного стовпця знову обчислюється одновимірне ДКП, в результаті чого формується остаточна матриця коефіцієнтів, що представляє двовимірне ДКП.

Рисунок 1.5. Схема розрахунку двовимірного ДКП

Таблиця 1.4. Основні характеристики 1D та 2D ДКП

Параметр	**Одновимірне ДКП**	**Двовимірне ДКП**
Вхідний сигнал	Послідовність *x(n)*	Матриця *I(i,j)* (зображення або блок зображення)
Формула	$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right]$	$X(u,v) = \alpha(u)\alpha(v) i = 0 N-1$ $\sum_{j=0} M-1\, I(i,j) \cos\left[\frac{\pi(2i+1)u}{2N}\right]$ $\cos\left[\frac{\pi(2j+1)v}{2M}\right]$
Нормалізація	$\alpha(0) = \sqrt{\frac{1}{N}},; \alpha(k) = \sqrt{\frac{2}{N}}; (k > 0)$	Аналогічні коефіцієнти по кожній осі
Застосування	Стиснення одновимірних сигналів	Стиснення зображень та відео, обробка блоків (наприклад, 8×8)

Дискретне косинусне перетворення є фундаментальним інструментом для представлення відео-даних у частотній області. Завдяки перетворенню сигналів в набір косинусних базисних функцій, яке здійснюється з використанням

нормалізаційних коефіцієнтів, можливо виділити основну енергетичну компоненту зображення (DC-компоненту) та окремі частотні складові (AC-компоненти). Двовимірна версія цього перетворення, що застосовується до блоків пікселів, є основою для багатьох сучасних алгоритмів стиснення, зокрема, у стандартах MPEG та H.264/AVC.

1.2.4 Квантування і зигзагоподібне сканування

Після обчислення двовимірного ДКП блоку зображення (найчастіше 8×8 пікселів) вихідна матриця коефіцієнтів має більшість значень, що характеризують високочастотну інформацію, набагато менші за значення основної (DC) компоненти. Оскільки високочастотні коефіцієнти (AC-компоненти) часто стають дуже малими або рівними нулю при зменшенні точності, доцільним є їхнє округлення (квантування).

Квантування є процесом приведення значень коефіцієнтів ДКП до меншої кількості можливих дискретних значень із застосуванням квантуючої матриці. Формально, для кожного коефіцієнта *C(i,j)* блоку його квантоване значення *Y(i,j)* обчислюється як

$$Y(i,j) = (\cdot)\left(\frac{C(i,j)}{Q(i,j)}\right), \qquad (1.7)$$

де:

- *C(i,j)* — відповідний коефіцієнт ДКП,
- *Q(i,j)* — елемент квантуючої матриці, який обирається залежно від бажаного ступеня стиснення та рівня втрат,
- $(\cdot)$ — функція округлення до найближчого цілого числа.

За допомогою квантуючої матриці можна регулювати компроміс між ступенем стиснення (розміром файлу) та якістю відновленого зображення. Чим більші значення *Q(i,j)*, тим грубіше відбувається квантування, що може призвести до втрати деяких деталей, але водночас зменшує обсяг даних для кодування. Наприклад, для деяких стандартних форматів використовується так звана стандартна квантуюча матриця, яку можна представити відповідно табл.1.5. Значення у цій таблиці можуть змінюватися залежно від налаштувань кодеку.

Таблиця 1.5. Приклад стандартної квантуючої матриці (8×8)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Після квантування матриці коефіцієнтів зображення залишається багато малих (часто нульових) значень, що зазвичай розташовані у високочастотних компонентах (на периферії блоку). Щоб максимально ефективно використати ці властивості, застосовують зигзагоподібне сканування.

Метою зигзагоподібного сканування є перетворення матриці квантованих коефіцієнтів у одномірний вектор, де найважливіші (низькочастотні) компоненти розміщено на початку, а менш важливі (високочастотні) — в кінці. Такий порядок сприяє наступним етапам кодування за допомогою кодування довжин серій, оскільки наприкінці вектора часто з’являється довга послідовність нулів. У таблиці зигзагоподібного порядку для блоку 8×8 кожне число позначає порядковий номер елемента у вихідному векторі після сканування. Наприклад, елемент, що знаходиться у позиції (0,0) матриці, отримує індекс 0, елемент (0,1) – індекс 1, елемент (1,0) – індекс 2, і так далі.

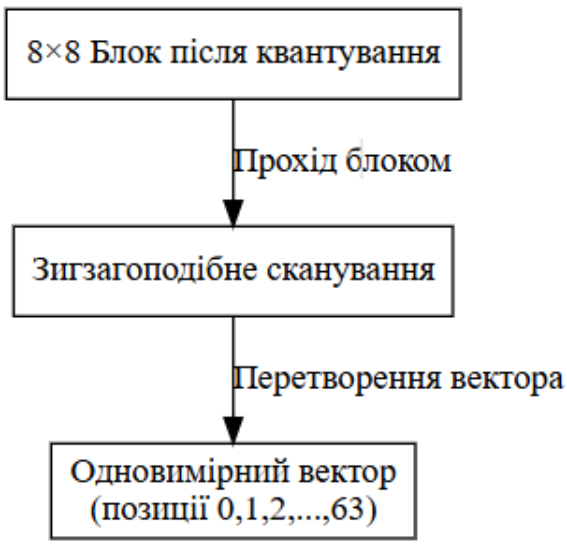


Рисунок 1.6. Схема зигзагоподібного сканування блоку 8×8

Схема на рис.1.6 демонструє, як квантовані коефіцієнти з блоку 8×8 шляхом зигзагоподібного сканування формують одномірний вектор, в якому низькочастотні коефіцієнти (мають найбільшу значущість) з'являються на початку.

Квантування дозволяє аппроксимувати коефіцієнти ДКП для зменшення точності зберігання даних, що сприяє суттєвому зниженню обсягу інформації. Застосування квантуючої матриці забезпечує можливість налаштування компромісу між ступенем стиснення та якістю відновленого зображення. Далі використання зигзагоподібного сканування дозволяє упорядкувати квантувані коефіцієнти так, що найбільш важливі компоненти (низькі частоти) розміщуються на початку вектора, а численні нульові чи малозначущі значення – в кінці. Це створює оптимальні умови для застосування алгоритмів кодування довжин серій, що забезпечує додаткове зниження бітрейту без істотної втрати якості. Разом ці етапи складають ключовий компонент сучасних алгоритмів стиснення відео, що використовуються у відео-кодеках типу MPEG та H.264/AVC.

1.2 Аналітичний огляд основних відеостандартів

Сучасні відеокодеки зазвичай спираються на два ключові підходи:

1. Кодеки, що застосовують базові методи компресії з використанням дискретного косинусного перетворення (ДКП) та квантування (наприклад, MPEG-2), які забезпечують базове стискання даних завдяки просторовому та часовому прогнозуванню, але мають обмежену ефективність при зниженні бітрейту;

2. Алгоритми нового покоління (H.264/AVC, H.265/HEVC), що впроваджують більш ефективні методи міжкадрового прогнозування, адаптивного блокового розбиття, підвищену точність компенсації руху, а також вдосконалені схеми квантування і кодування залишкових сигналів, що дозволяє досягати значно кращого співвідношення стиснення та якості.

Крім того, останнім часом з'являються сучасні алгоритми, орієнтовані спеціально на потреби стрімінгових сервісів. Вони враховують вимоги до мінімізації затримок, адаптивної зміни якості в залежності від пропускної здатності мережі та враховують інші специфічні параметри, що стосуються режимів доставки відео через мережеві канали.

1.2.1 Алгоритм стискування MPEG-2

Стандарт MPEG-2 був розроблений для забезпечення ефективного стискання відео-даних із застосуванням методів, що дозволяють знизити бітрейт при збереженні прийнятного рівня якості зображення. Основні принципи алгоритму стискування MPEG-2 ґрунтуються на комбінованому використанні внутрішньокадрового (intra-frame) та міжкадрового (inter-frame) кодування, а також на організації відеопотоку у вигляді груп кадрів (GOP). Основні етапи алгоритму:

1. Передобробка та перетворення у кольоровий простір: Для збереження сумісності з системами телевізійного мовлення сигнал камери RGB перетворюється у систему YCbCr. Сигнал Y представляє яскравість (luma), а сигнали Cb і Cr – колірні різниці (chroma). При цьому можлива субдискретизація кольорових компонент (наприклад, 4:2:0 або 4:2:2) з метою зменшення обсягу даних без істотного впливу на сприйняття зображення;

2. Блокове розбиття: Зображення розбивається на блоки/макроблоки (зазвичай 16×16 пікселів) із поділом на менші блоки — 8×8 для коефіцієнтів ДКП, що використовується для подальшого перетворення. Схематичне представлення блокового розбиття показане на рис.1.7;

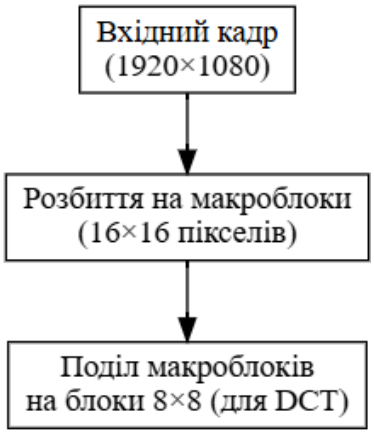


Рисунок 1.7. Схематичне представлення блокового розбиття

3. Внутрішньокадрове кодування (I-кадри): Внутрішньокадрове кодування виконується шляхом застосування Дискретного Косинусного Перетворення (ДКП) до блоків 8×8, після якого отримані коефіцієнти проходять етап квантової апроксимації. Формула для 2D ДКП:

$$X(u,v) = \alpha(u)\alpha(v) \sum_{i=0} N-1 \sum_{j=0} M-1 I(i,j) \cos\left[\frac{\pi(2i+1)u}{2N}\right] \cos\left[\frac{\pi(2j+1)v}{2M}\right], \quad (1.8)$$

де *I(i,j)* – значення пікселя блоку, *N* та *M* (зазвичай 8) – розміри блоку;

4. Міжкадрове кодування (P- та B-кадри): MPEG-2 використовує міжкадрове прогнозування для зниження надлишковості за рахунок використання відмінностей між послідовними кадрами. При цьому застосовується корекція руху (motion compensation). Прогнозування та компенсація руху: За допомогою аналізу сусідніх кадрів визначається вектор руху $(\Delta x, \Delta y)$для кожного макроблоку. Результуюча різниця $d_t(i,j)$ обчислюється за формулою:

$$d_t(i,j) = I_n(i,j) - I_{n-1}(i - \Delta x, , j - \Delta y), \qquad (1.9)$$

де $I_n(i,j)$ – значення пікселя в поточному кадрі, а $I_{n-1}(i - \Delta x, j - \Delta y)$ – прогнозоване значення з попереднього кадру.

5. Квантований етап та кодування залишкового сигналу: Після застосування перетворень отримані коефіцієнти піддаються квантуванню із застосуванням квантуючих матриць, після чого залишковий сигнал (як у випадку внутрішньокадрового, так і міжкадрового кодування) піддається ентропійному кодуванню (наприклад, за допомогою кодів Хаффмана або змінної довжини);

6. Організація потоку даних: Стиснені дані організовуються у вигляді пакетованих потоків (elementary streams) та мультиплексуються у транспортні потоки. Стандарт MPEG-2 визначає структуру груп кадрів (GOP), яка включає послідовність I-, P- і B-кадрів. Така організація потоків дозволяє здійснювати синхронізацію, управління буферами та ефективну передачу даних через канал.

Таблиця 1.6. Основні параметри алгоритму MPEG-2

Параметр	**Значення/Опис**
Розмір макроблоку	Зазвичай 16×16 пікселів
Розмір блоку для ДКП	8×8 пікселів
Система кольорового перетворення	RGB → YCbCr (субдискретизація, зазвичай 4:2:0 або 4:2:2)
Методи кодування	Внутрішньокадрове (I-кадри) та міжкадрове (P-/B-кадри)
Алгоритм компенсації руху	Визначення вектора руху $(\Delta x, \Delta y)$ та обчислення різниці

На рис.1.8 наведено спрощену схему, що ілюструє основні етапи кодування відео за стандартом MPEG-2.

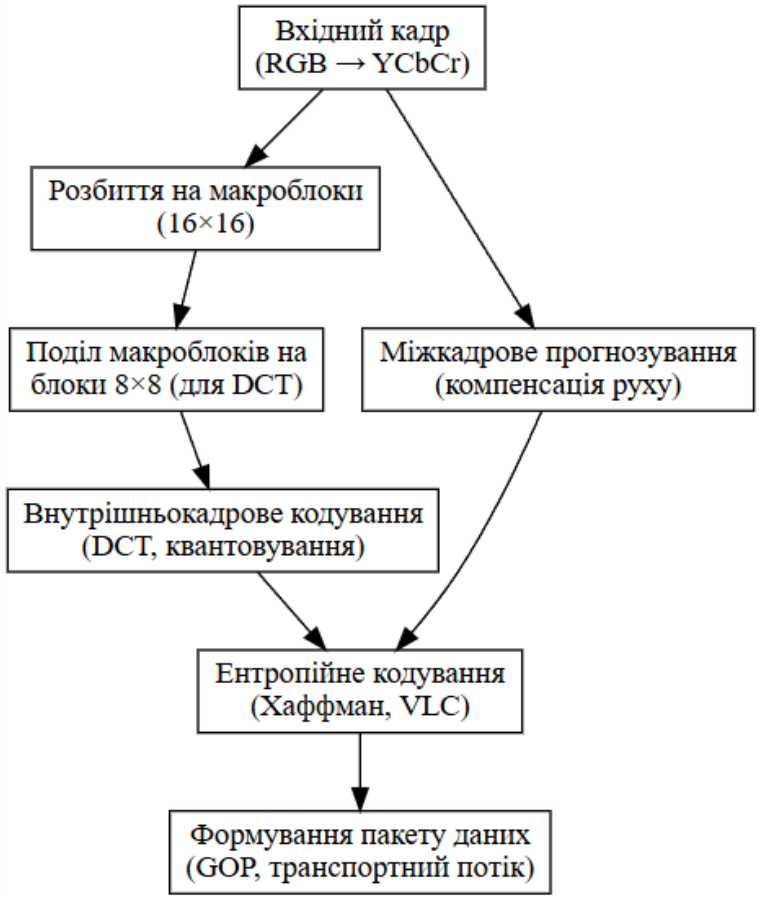


Рисунок 1.8. Схема основних етапів кодування за стандартом MPEG-2

Ця схема демонструє, як вхідний кадр (після перетворення у систему YCbCr) розбивається на макроблоки, які потім поділяються на менші блоки, до яких застосовується ДКП із подальшим квантуванням. Паралельно здійснюється міжкадрове прогнозування для компенсації руху, після чого залишковий сигнал проходить ентропійне кодування і формування пакету даних для передачі.

Алгоритм стискування MPEG-2 є базовим прикладом комбінованого алгоритму, який використовує як внутрішньокадрове, так і міжкадрове кодування для досягнення ефективного стиснення відео-даних. Основні операції включають перетворення кольорового простору, блокове розбиття, застосування ДКП на блоках 8×8, квантування коефіцієнтів із використанням квантуючих матриць, компенсацію руху та ентропійне кодування залишкового сигналу. Організація даних у вигляді груп кадрів (GOP) забезпечує можливість синхронізації та ефективної передачі потоків даних. Цей стандарт заклав основу для подальших удосконалених методів стискування, таких як H.264/AVC та H.265/HEVC, які використовують додаткові алгоритмічні оптимізації для підвищення ефективності стиснення.

1.2.2 Алгоритм стискування H.264/AVC

Стандарт H.264/AVC (Advanced Video Coding) є одним із найпоширеніших сучасних методів стискування відео-даних, що забезпечує значно вищу ефективність порівняно з попередніми стандартами (наприклад, MPEG-2). Цей алгоритм відзначається більш гнучким підходом до розбиття кадру, адаптивним прогнозуванням як всередині кадру (intra-prediction), так і між кадрами (inter-prediction), застосуванням вдосконалених методів компенсації руху, а також використанням потужних алгоритмів ентропійного кодування (CAVLC — контекстно-адаптивне кодування змінної довжини або CABAC — контекстно-адаптивне бінарне арифметичне кодування). Основні етапи алгоритму H.264/AVC:

1. Блокове розбиття та адаптивне прогнозування:

- Адаптивне розбиття: Вхідний кадр розбивається на макроблоки, типово розміром 16×16 пікселів. Проте з метою більш точного прогнозування та компенсації руху макроблок може поділятися на субблоки різного розміру (від 16×16 до 4×4) залежно від характеру зображення. Це дозволяє оптимізувати кодек за рахунок адаптації до локальних характеристик відеосцени;
- Intra-prediction (внутрішньокадрове прогнозування): Для блоків, що кодуються без посилання на інші кадри (I-кадри), застосовують кілька режимів прогнозування із використанням вже оброблених сусідніх блоків. Наприклад, для блоків 4×4 визначено дев’ять режимів прогнозування (горизонтальний, вертикальний, діагональний, тощо);

Формально, прогнозоване значення *P(i,j)* для пікселя блоку може бути визначене як функція від його сусідніх значень:

$$P(i,j) = f(I(i-1,j),,I(i,j-1),,I(i-1,j-1),,\dots) \quad (1.10)$$

де $f(\cdot)$ представляє адаптивний алгоритм прогнозування, що враховує конкретний режим;

- Inter-prediction (міжкадрове прогнозування): Для блоків, що кодуються за допомогою інформації з інших кадрів (P- та B-кадри), виконується оцінка руху. Алгоритм шукає оптимальні вектори руху $(\Delta x, \Delta y)$ для кожного субблоку, що мінімізують різницю між поточним блоком і відповідною областю в опорному кадрі

чи кадрах. Різницевий сигнал обчислюється за формулою:

$$d(i,j) = I_n(i,j) - I_{ref}(i - \Delta x, j - \Delta y), \qquad (1.11)$$

де $I_n(i,j)$ — значення пікселя в поточному кадрі, а I_{ref} — значення пікселя з опорного кадру;

2. Ціле числове перетворення та квантування:

– Ціле перетворення: H.264/AVC застосовує ціле перетворення для блоків розміру 4×4 (для внутрішньокадрових і міжкадрових залишкових блоків). Цей алгоритм є адаптацією ДКП, але розроблений таким чином, щоб мінімізувати обчислювальні витрати та спрощувати апаратну реалізацію. Формально, перетворення для блоку 4×4 може бути подано аналогічно звичайному ДКП із певними змінами у коефіцієнтах;

– Квантування: Отримані коефіцієнти після перетворення квантуються за допомогою адаптивних квантуючих матриць;

3. Ентропійне кодування. Для кодування відновленого останнього залишкового сигналу H.264/AVC використовує два основних методи:

– CAVLC (Context-Adaptive Variable Length Coding): Використовується для профілів з нижчою обчислювальною складністю;

– CABAC (Context-Adaptive Binary Arithmetic Coding): Забезпечує більш високий ступінь стиснення за рахунок арифметичного кодування і використовується у більш потужних профілях (Main та High Profiles);

4. In-loop deblocking фільтр. Одна з особливостей H.264/AVC — застосування вбудованого фільтра де-блокування (in-loop deblocking filter), який зменшує артефакти блоковості, характерні для блочного кодування, покращуючи сприйняття відновленого зображення. Фільтр застосовується після декодування окремих блоків та перед наступними етапами прогнозування.

Схема на рис.1.9 ілюструє послідовне виконання наступних операцій: вихідний кадр спочатку розбивається на макроблоки, далі здійснюється адаптивне прогнозування (як intra-, так і inter-предикція), після чого обчислюється залишковий сигнал за допомогою цілого перетворення, квантування отриманих коефіцієнтів і їхнє ентропійне кодування. Окрім цього, застосовується in-loop deblocking фільтр

для покращення якості відновленого зображення.

Таблиця 1.7. Основні характеристики алгоритму H.264/AVC

Параметр	**Опис**
Розбиття кадру	Макроблоки 16×16, що можуть бути розбиті на субблоки (16×16, 16×8, 8×16, 8×8, 4×4)
Intra-prediction	9 режимів для блоків 4×4; прогнозування на базі сусідніх пікселів
Inter-prediction	Різні методики прогнозування з компенсацією руху; підтримка B-кадрів
Ціле перетворення	Ціле перетворення для блоків 4×4
Квантування	Адаптивне квантування із використанням змінних квантуючих матриць
Ентропійне кодування	CAVLC для базових профілів, CABAC для підвищення ефективності
In-loop deblocking фільтр	Забезпечує згладження меж блоків і зменшення артефактів

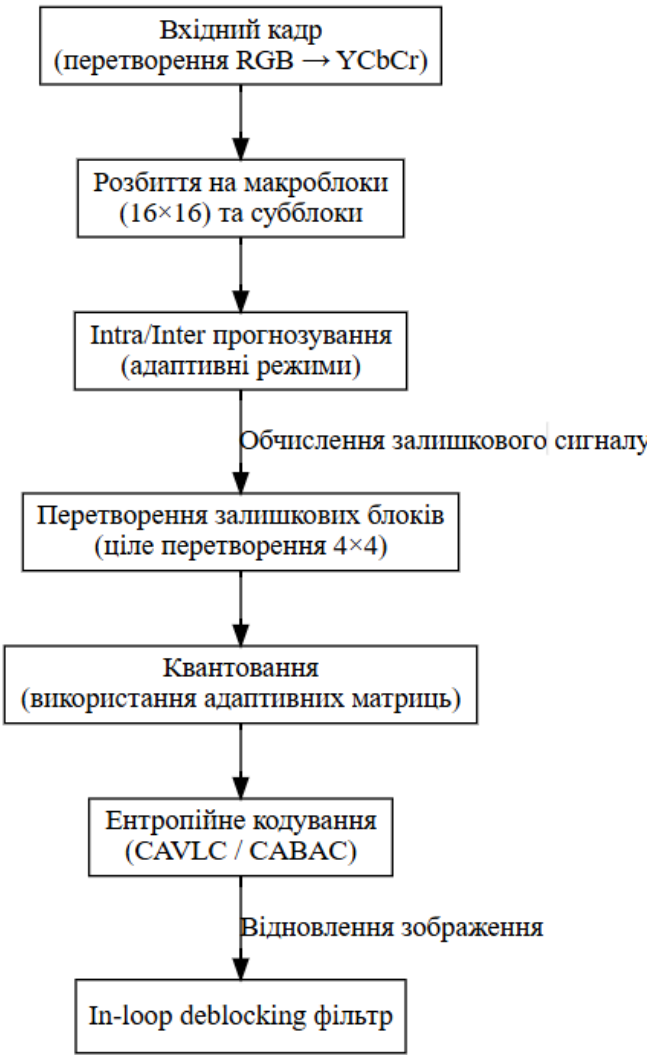


Рисунок 1.9. Спрощена блок-схема алгоритму стискування H.264/AVC

Алгоритм стискування H.264/AVC використовує комплексний підхід для досягнення високого ступеня стиснення з мінімальними втратах. Завдяки адаптивному поділу кадру, багатоканальному прогнозуванню, цілим перетворенням, гнучким режимам квантування та ефективним методам ентропійного кодування, стандарт H.264/AVC забезпечує значно кращу компресію за нижчого бітрейту порівняно з попередніми методами. Окрім цього, застосування in-loop deblocking фільтру допомагає значно знизити видимість блокових артефактів, що підвищує суб'єктивну якість відновленого зображення, що є особливо важливим для сучасних стрімінгових застосунків.

1.2.3 Алгоритм стискування H.265/HEVC

Стандарт H.265/HEVC (High Efficiency Video Coding) розроблено з метою значного підвищення ефективності стиснення відео-даних порівняно з попередніми стандартами (наприклад, H.264/AVC). Завдяки ряду алгоритмічних удосконалень H.265/HEVC дозволяє досягати приблизно 50 % зниження бітрейту при збереженні подібної якості зображення. Основною його особливістю є гнучка блокова структура, яка забезпечує адаптацію до локальних характеристик зображення та більш точне прогнозування. Основні особливості та етапи алгоритму HEVC:

1. Блокова структура за допомогою CTU. На відміну від H.264, який використовує макроблоки (зазвичай 16×16 пікселів), HEVC вводить концепцію Coding Tree Unit (CTU), розмір якого може сягати до 64×64 пікселів. CTU розбивається на менші одиниці звані Coding Units (CU), які можуть мати різний розмір (від 64×64 до 8×8 пікселів). Додатково кожен CU ділиться на Prediction Units (PU) та Transform Units (TU). Така ієрархічна структура дозволяє адаптивно вибирати оптимальний розмір блоку для прогнозування та трансформування, що сприяє кращій компресії;

2. Прогнозування (Prediction). HEVC підтримує два типи прогнозування:

– Intra-prediction: Використовується для кодування кадрів, які кодуються незалежно від інших (I-кадри). Тут застосовують до 33 режимів прогнозування для блоків різного розміру, що дозволяє максимально точно передбачати значення пікселів на основі інформації з уже декодованих сусідніх блоків;

– Inter-prediction: Використовується для кодування відмінностей між послідовними кадрами (P- і B-кадри). За допомогою вдосконалених алгоритмів компенсації руху знаходяться оптимальні вектори руху для кожного PU, що мінімізують залишкову різницю між поточним блоком і прогнозованою областю з опорного кадру. Формально, залишкова інформація (signal difference) для міжкадрового прогнозування обчислюється як:

$$d(i,j) = I_n(i,j) - I_{\text{ref}}(i - \Delta x,, j - \Delta y), \qquad (1.12)$$

де:

– $I_n(i,j)$ — значення пікселя в поточному кадрі,

– $I_{\text{ref}}(i - \Delta x,, j - \Delta y)$ — прогнозоване значення пікселя з опорного кадру із врахуванням вектора руху $(\Delta x,, \Delta y)$.

3. Цілі трансформації та квантування. HEVC застосовує цілі трансформації для перетворення залишкових блоків, переважно для блоків TU розміру 4×4, 8×8, 16×16 та навіть 32×32. Основна ідея трансформації полягає у перетворенні просторових залишкових даних у частотну область, де більшість енергії концентрується в низькочастотних компонентах. Як і в попередніх стандартах, для цього використовують алгоритми, що є похожими до ДКП, проте з деякими модифікаціями для цілочислової реалізації. Формула для 2D цілого перетворення виглядає схоже:

$$X(u,v) = \alpha(u)\alpha(v) \sum_{i=0} N-1 \sum_{j=0} M-1 I(i,j) \cos\left[\frac{\pi(2i+1)u}{2N}\right] \cos\left[\frac{\pi(2j+1)v}{2M}\right], \quad (1.13)$$

де *I(i,j)* — залишкові значення пікселів, *N, M* — розміри блоку (наприклад, 4 чи 8), а коефіцієнти нормалізації $\alpha(u)$ визначаються за правилами, подібними до тих, що використовуються в H.264. Після перетворення коефіцієнти піддаються адаптивному квантуванню за допомогою квантуючої матриці, що забезпечує можливість регулювання ступеня стиснення;

4. Ентропійне кодування. Для підвищення ефективності кодування залишкових даних HEVC використовує удосконалений алгоритм контекстно-адаптивного бінарного арифметичного кодування (CABAC). CABAC забезпечує більш складне, але ефективне кодування в порівнянні з класичним кодуванням змінної довжини (VLC) і є однією з основних причин підвищеної ефективності

HEVC;

5. In-loop deblocking фільтр. На відміну від попередніх стандартів, HEVC застосовує вдосконалений in-loop deblocking фільтр, який зменшує блоковість та інші артефакти, виникаючі внаслідок агресивного квантування. Фільтр застосовується до меж трансформних блоків і допомагає зберегти суб'єктивну якість відео на високому рівні.

Таблиця 1.8. Основні характеристики стандарту H.265/HEVC

Параметр	**Опис**
Розмір CTU	До 64×64 пікселів
Розбиття CTU	Розбиття на Coding Units (CU); CU можуть бути розміром 64×64, 32×32, …, 8×8
Субблоки (PU і TU)	PU для прогнозування; TU для трансформування (4×4, 8×8, 16×16, 32×32)
Прогнозування	Різноманітні режими intra-prediction (до 33 режимів) та міжкадрове (inter) prediction із точним визначенням векторів руху
Квантування	Адаптивне квантування з можливістю регулювання параметрів квантовуючої матриці
Ентропійне кодування	CABAC – контекстно-адаптивне бінарне арифметичне кодування
Deblocking фільтр	Вдосконалений in-loop deblocking фільтр для зменшення артефактів

На рис. 1.10. наведено приклад коду, який створює спрощену блок-схему алгоритму H.265/HEVC. Ця схема демонструє основні етапи: вхідний кадр перетворюється у формат YCbCr, далі виконується розбиття на CTU, які поділяються на CU для прогнозування (як intra-, так і inter-) із визначенням PU, обчислюється залишковий сигнал за допомогою трансформування (TU), коефіцієнти квантовуються з використанням адаптивних квантуючих матриць, далі застосовується CABAC для ентропійного кодування залишків, і на останньому етапі – in-loop deblocking фільтр для зменшення артефактів.

Алгоритм стискування H.265/HEVC є суттєвим кроком уперед у порівнянні з попередніми стандартами. Використання гнучкої блокової структури CTU з можливістю адаптивного розбиття на CU, PU та TU дозволяє краще пристосовувати кодування до локальних особливостей відеоконтенту. Покращені алгоритми

прогнозування, ефективне ціле трансформування, квантування із адаптивними квантуючими матрицями та високоефективне ентропійне кодування CABAC в сукупності забезпечують високу ефективність стиснення. Крім того, in-loop deblocking фільтр значно покращує суб'єктивну якість відновленого зображення, зменшуючи блоковість та інші характерні для блочного кодування артефакти. Таким чином, HEVC дозволяє зберігати високоякісне зображення при значному скороченні обсягу даних, що є важливим для потокових застосувань та стрімінгових сервісів у сучасних умовах обмежених мережевих ресурсів.

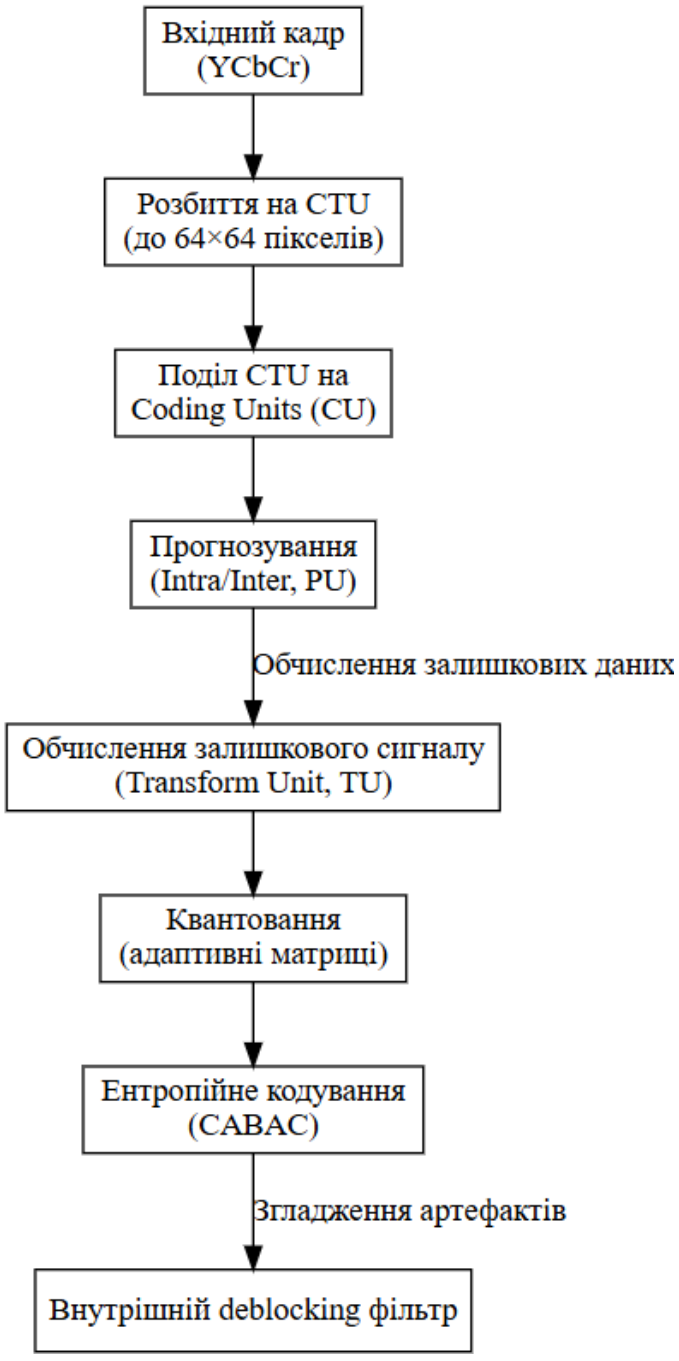


Рисунок 1.10. Спрощена блок-схема алгоритму стискування H.265/HEVC

1.2.4 Сучасні алгоритм стискування для стрімінгу

Сучасні методи стискування для стрімінгу орієнтуються не лише на максимальне зменшення обсягу даних, але й на адаптивність до змін пропускної здатності, мінімальну затримку та високий рівень відновлення якості у випадку пакетних втрат чи змін умов мережі. Основні принципи сучасних алгоритмів для стрімінгу:

1. Адаптивне стиснення та масштабованість. Сучасні алгоритми стискування для стрімінгу часто базуються на принципі масштабованого відеокодування (Scalable Video Coding, SVC), яке розширює основний стандарт (наприклад, H.264 або H.265) додатковим шаром масштабованості. Це дозволяє формувати кілька рівнів якості та роздільної здатності в одному потокові, з можливістю динамічно адаптуватися до доступної пропускної здатності мережі. Наприклад, масштабована структура може бути представлена як набір базового шару та одного або кількох додаткових шарів, завдяки чому кінцевий пристрій може вибирати оптимальну якість відтворення;

2. Адаптивне управління параметрами кодування. Сучасні алгоритми походять від потреби реального часу адаптувати параметри стискання залежно від умов мережі. Наприклад, параметр квантування може динамічно коригуватися залежно від змін у пропускній здатності, що дозволяє підтримувати низьку затримку та задовільну якість зображення. Одним із прикладів є адаптивне квантування, яке може бути описане наступною умовною формулою:

$$Q_{\text{adaptive}} = Q_{\text{base}} + \beta \cdot \Delta B, \qquad (1.14)$$

де

- Q_{base} — базовий коефіцієнт квантування,
- ΔB — відхилення доступної смуги пропускання від цільового значення,
- β — коефіцієнт адаптації, що визначає, наскільки агресивно слід коригувати параметр;

3. Використання методів машинного навчання. Новітні напрямки досліджень включають інтеграцію алгоритмів машинного навчання для прогнозування оптимальних параметрів кодування, оцінки якості зображення на льоту та

адаптивного управління затримками. Такі підходи дозволяють враховувати суб'єктивні особливості сприйняття якості зображення користувачами, що є критично важливим для стрімінгових сервісів.

4. Модульна архітектура та реальна адаптація. У стрімінгових системах стиснення часто організовується за принципом модульності: окремі блоки відповідають за кодування відео, управління буферами, аналіз інтервалів затримок і даних, що надходять. Таку архітектуру можна зобразити наступною схемою.

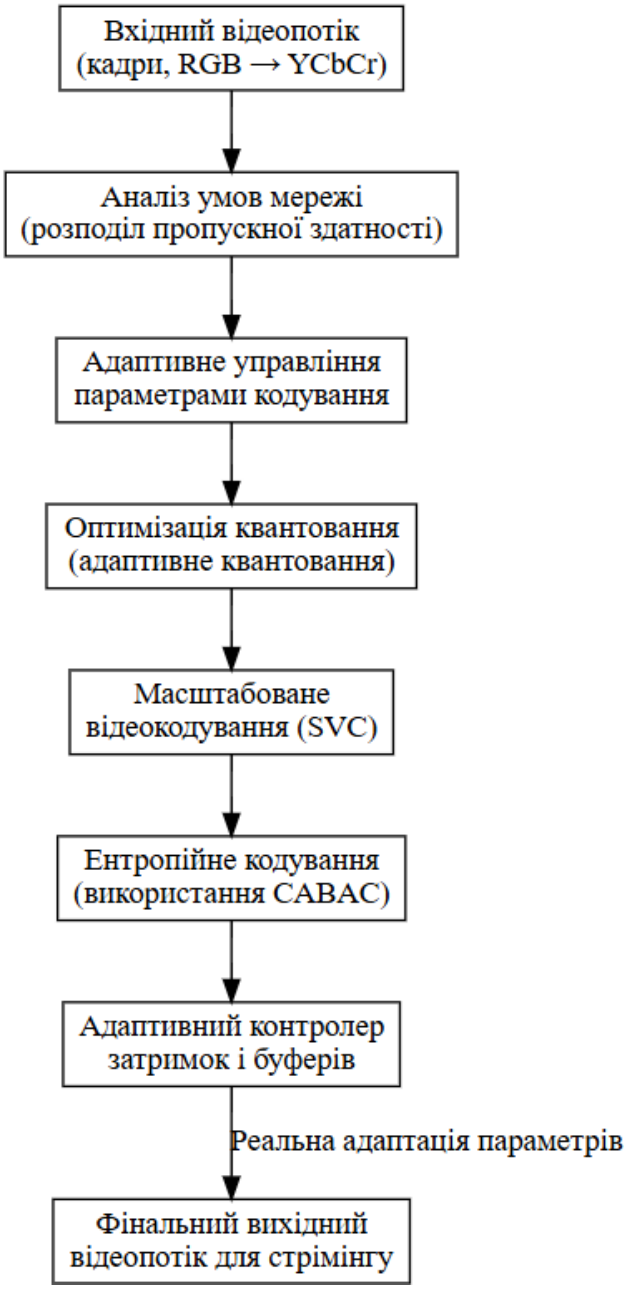


Рисунок 1.11. Схема адаптивного алгоритму стискування для стрімінгу

Схема сучасного алгоритму стискування для стрімінгу на рис. 1.11. демонструє, як із вхідного відеопотоку (після попереднього перетворення)

здійснюється аналіз мережевих умов, адаптивне управління параметрами кодування, оптимізація квантування та масштабоване відеокодування. Далі частина стискання виконується за допомогою ефективного ентропійного кодування, після чого адаптивний контролер затримок і управління буферами забезпечує максимально сталі характеристики потоку для кінцевого відтворення.

Таблиця 1.9. Основні характеристики алгоритмів стискування для стрімінгу

Параметр	**Опис**
Адаптивність	Динамічна зміна параметрів кодування (наприклад, квантування) залежно від мережевих умов
Масштабоване кодування (SVC)	Формування декількох рівнів якості в одному потокові, які можуть адаптуватися до доступної смуги
Низька затримка	Алгоритми розраховані на реальну роботу в режимі онлайн із забезпеченням мінімальних затримок
Інтеграція з машинним навчанням	Використання AI для прогнозування оптимальних параметрів кодування та оцінки якості у реальному часі
Контроль буферів і синхронізація	Адаптивне управління буфером для стабільної роботи стрімінгової системи

Сучасні алгоритми стискування для стрімінгу розроблені з урахуванням особливостей мережевих умов і вимог до реального часу. Вони забезпечують адаптивну зміну параметрів кодування, використання масштабованого відеокодування, інтеграцію методів машинного навчання для оптимізації якості і адаптивний контроль затримок. Підхід до побудови таких алгоритмів спрямований на забезпечення максимальної якості відео при мінімальному бітрейті, що є надзвичайно важливим для потокових сервісів, які працюють у динамічно змінному середовищі з обмеженими ресурсами мережі.

1.3 Інтеграція алгоритмів стискування у стрімінгові сервіси

У сучасному цифровому середовищі стрімінгові сервіси стають головним каналом доставки відеоконтенту до кінцевих користувачів. Ефективність таких систем у великій мірі залежить від правильного впровадження алгоритмів стискування, що дозволяють зменшити обсяг передаваних даних при збереженні високої якості зображення. Інтеграція алгоритмів стискування (таких як MPEG-2, H.264/AVC, H.265/HEVC, а також сучасних адаптивних рішень для стрімінгу) у

стрімінгові сервіси охоплює кілька ключових аспектів. Основні компоненти інтеграції:

1. Підготовка відеоконтенту та первинне кодування. На початковому етапі, відеоматеріал (як правило, у форматі RAW або високоякісному з оригінальними даними) передається через відповідний модуль, де застосовуються алгоритми стискування. У цьому процесі можуть бути застосовані різні стандарти кодування залежно від наступного способу доставки (наприклад, H.264/AVC або H.265/HEVC). Результатом є декілька версій відео з різними характеристиками (бітрейт, роздільну здатність, частота кадрів);

2. Адаптивне трансдування та мультироздільність (Adaptive Transcoding та Scalable Video Coding). Багато сучасних платформ використовують адаптивні алгоритми, що дозволяють формувати різні варіанти (renditions) одного відео. Це базується як на масштабованому відеокодуванні (SVC), так і на динамічній зміні параметрів кодування (наприклад, адаптивне квантування). Завдяки цьому клієнтський пристрій отримує оптимальну версію відео, виходячи з поточних умов мережі та характеристик пристрою;

3. Пакетування, мультиплексування та протоколи адаптивного стрімінгу. Після кодування відео проходить етап пакетування, де відео- та аудіо-дані організовуються згідно з вимогами протоколів адаптивного стрімінгу, таких як HTTP Live Streaming (HLS) або MPEG-DASH. У цьому процесі відбувається сегментація відео на невеликі частини (наприклад, сегменти по 2–10 секунд) із доданням метаданих та інформації про часові мітки. Це дозволяє клієнтському додатку швидко перемикатися між варіантами якості в залежності від пропускної здатності мережі;

4. Адаптивний контроль якості та управління затримками. Інтеграція алгоритмів стискування не завершується на стадії пакетування. Сучасні стрімінгові сервіси використовують адаптивні контролери, що забезпечують зворотний зв’язок з клієнтами і модулем кодування. Наприклад, параметри кодування, такі як коефіцієнт квантування, можуть коригуватися в режимі реального часу за допомогою адаптивних алгоритмів;

5. Інтеграція з CDN та оптимізація мережевої передачі. Для забезпечення широкої доступності відеоконтенту, стиснені потоки надсилаються до мереж контент-доставки (CDN), що розподіляють потоки по географічно розподілених серверах. Це дозволяє мінімізувати затримку, знизити завантаження основних серверів та забезпечити стабільну якість стрімінгу для кінцевого користувача.

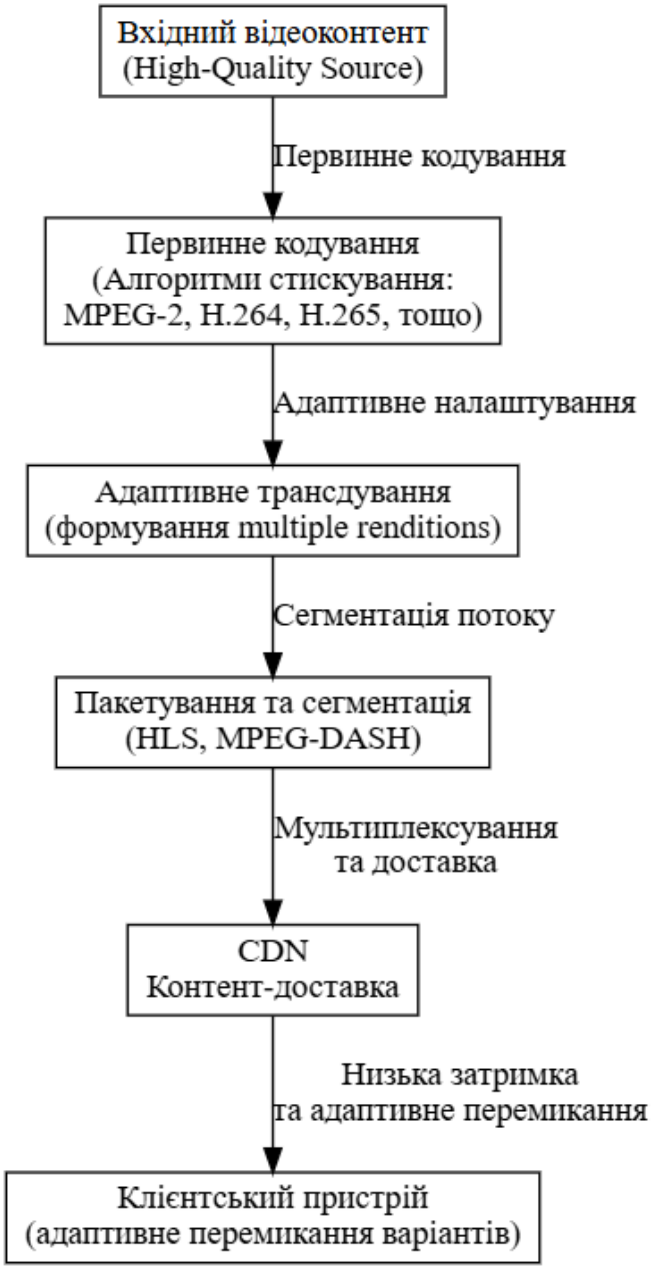


Рисунок 1.12. Інтеграція алгоритмів стискування у стрімінгову систему

Рис.1.12 ілюструє основні етапи інтеграції стисканих відеопотоків: від первинного формування високоякісного відеоматеріалу та його кодування, через адаптивне транскодування й поділ на сегменти, до розподілу через CDN і кінцевої динамічної адаптації якості на клієнтському пристрої.

Таблиця 1.10. Основні переваги інтеграції сучасних алгоритмів стискування

Параметр	Опис
Ефективність стискання	Забезпечує високий ступінь стиснення за рахунок сучасних алгоритмів (H.264, H.265, SVC)
Адаптивність	Динамічна адаптація параметрів кодування згідно з умовами мережі та ресурсами користувача
Низька затримка	Реальний час обробки та мінімальна затримка у потоковій передачі, що є критично важливими для стрімінгу
Мультиформатність	Можливість створення різних варіантів якості в одному відеопотоці
Оптимізація доставки	Інтеграція з CDN забезпечує високу доступність відеоконтенту та зменшення затримок

Інтеграція алгоритмів стискування у стрімінгові сервіси є складним багаторівневим процесом, який охоплює не лише ефективне кодування відео, а й подальше адаптивне управління параметрами для забезпечення оптимальної якості відтворення незалежно від умов мережі. Сучасні технології, такі як масштабоване відеокодування (SVC), динамічна адаптація квантування та інтеграція з протоколами адаптивного стрімінгу (HLS, MPEG-DASH), дозволяють забезпечити ефективну передачу даних при мінімальних затримках та високій стабільності. Завдяки комплексному підходу, що включає використання алгоритмів стискування, адаптивного управління та інтелектуального моніторингу мережевих умов, стрімінгові сервіси можуть забезпечувати якісне та стійке відтворення відеоконтенту для широкої аудиторії.

1.4 Об’єктивна оцінка якості відеозображень

Об’єктивна оцінка якості відеозображень здійснюється за допомогою різноманітних математичних метрик, які дозволяють кількісно визначити ступінь відхилення обробленого відео від еталонного. Різні групи метрик можуть бути класифіковані залежно від того, чи враховують вони суто числову різницю між пікселями ("класичні"), чи властивості структурної схожості зображення, або ж орієнтовані на більш комплексну оцінку якості з урахуванням суб’єктивних аспектів. Далі наведено детальний опис кожної групи.

1.4.1 Класичні метрики якості PSNR та APSNR

PSNR (Peak Signal-to-Noise Ratio) – одна з найпоширеніших метрик для оцінки якості відеозображення. Вона базується на понятті середньоквадратичної помилки (MSE) між оригінальним та відновленим зображенням. Формула PSNR записується як:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{\text{MSE}}\right) \qquad (1.15)$$

де:

- MAX_I – максимальне можливе значення пікселя (наприклад, 255 для 8-бітного зображення),
- MSE – середньоквадратична помилка, яку розраховують як

$$\text{MSE} = \frac{1}{wh}\sum_{i=1} w \sum_{j=1} h[I(i,j) - I'(i,j)]^2 \qquad (1.16)$$

де w і h – ширина та висота зображення, $I(i,j)$ – значення пікселя оригінального зображення, $I'(i,j)$ – значення пікселя відновленого зображення.

APSNR (Average PSNR) – середнє значення PSNR, яке розраховується за кількома кадрами або блоками зображення. Якщо маємо N кадрів, то APSNR визначається як:

$$\text{APSNR} = \frac{1}{N}\sum_{k=1}^{N} \text{PSNR}_k \qquad (1.17)$$

де PSNR_k – PSNR для (k)-го кадру.

Ці метрики широко застосовуються завдяки простоті обчислень, але їхня головна недолік полягає у тому, що вони не завжди адекватно відображають суб'єктивне сприйняття якості зображення.

1.4.2 Класичні метрики якості MSE та MSAD

MSE (Mean Squared Error) вже було описано вище при розгляді PSNR. Це середнє квадратичне відхилення між пікселями оригінального та обробленого зображення:

$$\text{MSE} = \frac{1}{wh}\sum_{i=1} w \sum_{j=1} h[I(i,j) - I'(i,j)]^2 \qquad (1.18)$$

Чим менше значення MSE, тим більш схожими є зображення. MSAD (Mean Square Absolute Difference), або середня квадратична абсолютна різниця, вимірює усереднену абсолютну різницю між пікселями. Формально вона визначається як:

$$\text{MSAD} = \frac{1}{wh}\sum_{i=1} w \sum_{j=1} h|I(i,j) - I'(i,j)| \qquad (1.19)$$

Попри свою простоту, MSAD може бути чутливішою до невеликих змін, що іноді робить її показником, який краще відображає різниці у сприйнятті дрібних деталей, хоча обидві метрики є базовими та використовуються для порівняння алгоритмів стискання.

1.4.3 Метрики з урахуванням структури зображення: SSIM та MS-SSIM

SSIM (Structural Similarity Index Measure) – метрика, що оцінює якість зображення, враховуючи не лише значення пікселів, але й їхню структурну схожість. SSIM порівнює два зображення за трьома основними компонентами: яскравість, контраст і структуру. Формально SSIM визначається як:

$$\text{SSIM}(x,y) = [l(x,y)]\alpha \cdot [c(x,y)]\beta \cdot [s(x,y)]^{\gamma}, \quad (1.20)$$

де

- $l(x,y) = \frac{2\mu_x\mu_y+C_1}{\mu_x 2+\mu_y 2+C_1}$ – коефіцієнт порівняння яскравості,
- $c(x,y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x 2+\sigma_y 2+C_2}$ – коефіцієнт порівняння контрасту,
- $s(x,y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$ – коефіцієнт порівняння структури.

За умов часто використовують $\alpha = \beta = \gamma = 1$ і $C_3 = C_2/2$. Значення SSIM змінюється від 0 до 1, де 1 відповідає ідеальній схожості між зображеннями.

MS-SSIM (Multi-Scale SSIM) – розширена версія SSIM, яка враховує схожість зображень на декількох масштабах. MS-SSIM розбиває зображення на різні рівні (через ітеративну фільтрацію і зменшення роздільної здатності) і обчислює SSIM для кожного рівня. Остаточне значення MS-SSIM визначається як добуток значень SSIM, зважених певними коефіцієнтами:

$$\text{MS-SSIM}(x,y) = \prod_{j=1} M\big[c_j(x,y)\big]\beta_j \cdot \big[s_j(x,y)\big]\gamma_j \cdot [l_M(x,y)]\alpha, \qquad (1.21)$$

де M – кількість масштабів, а β_j, γ_j та α– параметри вагової оцінки. MS-SSIM дозволяє більш точно відобразити сприйняття якості людиною, оскільки враховує візуальні особливості зображень на різних масштабах.

1.4.4 Показники якості: VQM, NQI, Delta

Окрім класичних метрик, для оцінки якості відео застосовують такі показники:

– VQM (Video Quality Metric): VQM – комплексна метрика, яка враховує не тільки просторові, але й часові аспекти спотворень, такі як мерехтіння, артефакти руху та інші неточності, що виникають при стисканні. Зазвичай VQM базується на аналізі локального контрасту, структури та змін між кадрами. Алгоритм VQM може включати аналіз залишкових відхилень, побудову локальних карт контрастності та їх зважене об'єднання для отримання числового показника якості;

– NQI (Universal Image Quality Index): NQI – метрика, яка враховує як статистичні відхилення між зображеннями, так і їхню кореляцію. Вона розраховується за допомогою наступного виразу:

$$\mathrm{NQI}(x,y) = \frac{4\sigma_{xy}\mu_x\mu_y}{\left(\mu_x 2 + \mu_y 2\right)\left(\sigma_x 2 + \sigma_y 2\right)} \qquad (1.22)$$

де μ_x, μ_y – середні значення оригінального та спотвореного зображення, $\left(\sigma_x, \sigma_y\right)$ – їх стандартні відхилення, а σ_{xy} – коваріація. Значення NQI також коливається від 0 до 1, де близьке до 1 значення свідчить про високу якість;

– Delta: Delta (або показник різниці) визначає абсолютну або відносну різницю між кольоровими компонентами пікселів оригінального та обробленого зображення. Цей показник дозволяє простими засобами оцінити, наскільки точно були відтворені відмінності в колірному просторі. Формула може бути записана наступним чином:

$$\Delta = \frac{1}{wh}\sum_{i=1} w \sum_{j=1} h|I(i,j) - I'(i,j)| \qquad (1.23)$$

де (w) і (h) – ширина та висота зображення.

Таблиця 1.11. Порівняльна таблиця показників якості

Метод	Основні характеристики	Діапазон значень
PSNR/APSNR	Оцінка на основі MSE; висока PSNR показує менший рівень спотворень	0 – ∞ (де 1 – ідеал)
MSE/MSAD	Простий числовий показник середньої помилки; чутливий до змін	0 – ∞
SSIM/MS-SSIM	Оцінка структурної схожості, контрасту та яскравості; ближче відображає суб'єктивне сприйняття	0 – 1
VQM	Комплексний показник, що враховує просторові та часові спотворення	Нижчі значення – краща якість
NQI	Універсальний індекс, що базується на середніх, стандартних відхиленнях та кореляції	0 – 1
Delta	Абсолютна різниця між відповідними пікселями	0 – ∞

Об'єктивна оцінка якості відеозображень здійснюється за допомогою ряду метрик, кожна з яких має свої переваги та обмеження. Класичні показники, такі як PSNR, APSNR, MSE та MSAD, забезпечують простий числовий аналіз, але не враховують структурних особливостей зображення. Метрики, такі як SSIM та MS-SSIM, дозволяють оцінювати якість з урахуванням властивостей «структурної» схожості, що більше відповідає сприйняттю людиною. Додаткові показники VQM, NQI та Delta забезпечують більш комплексний аналіз якості відео, враховуючи як просторові, так і часові аспекти. Використання комплексного набору метрик дозволяє розробникам краще оцінити якість відновленого зображення, вибрати оптимальні алгоритми стискання та налаштувати параметри відеопотоків для забезпечення високої якості відтворення у сучасних стрімінгових сервісах.

1.5 Розробка застосунку для аналізу ефективності алгоритмів відео-даних для стрімінгових сервісів

Зі зростанням популярності онлайн-відео та стрімінгових сервісів зростає й потреба в ефективному стисненні відео-даних, яке забезпечує оптимальне співвідношення між розміром потоку та якістю зображення. Для прийняття обґрунтованих рішень щодо вибору та оптимізації алгоритмів стискання необхідно мати засіб, який дозволяє проводити об'єктивну оцінку отриманої якості відео. Саме

тому розробка спеціалізованого застосунку для аналізу ефективності алгоритмів відео-даних є актуальною задачею.

Основною метою розроблюваного застосунку є:

- Збір даних: Забезпечити можливість завантаження вихідних (еталонних) відео та відео, стиснутих різними алгоритмами (MPEG-2, H.264/AVC, H.265/HEVC, сучасні алгоритми для стрімінгу);
- Обчислення метрик: Реалізувати модуль, який автоматизовано обчислює об'єктивні метрики якості, такі як PSNR, APSNR, MSE, MSAD, SSIM, MS-SSIM, VQM, NQI та Delta;
- Візуалізація результатів: Надати інтерфейс для перегляду графіків покадрових змін, побудови порівняльних таблиць і структурних схем, що відображають ефективність алгоритмів стискання;
- Адаптивне коригування: Забезпечити можливість тестування адаптивних режимів кодування, наприклад, динамічної корекції параметрів (як, наприклад, адаптивне квантування), із врахуванням відхилень пропускної здатності мережі (формула $Q_{\text{adaptive}} = Q_{\text{base}} + \beta \cdot \Delta B$);
- Інтеграція з потоковими системами: Продемонструвати, як алгоритмічна частина застосунку інтегрується з протоколами адаптивного стрімінгу (HLS, MPEG-DASH) та CDN (мережами доставки контенту).

1.5.1 Розробка архітектури та схеми роботи застосунку

Безпосередньо інтегроване середовище застосунку складається з декількох модулів:

- Інтерфейс користувача: Веб-інтерфейс із зручною панеллю завантаження відеофайлів, вибору параметрів тестування та перегляду результатів обчислень;
- Модуль попередньої обробки: Цей блок відповідає за конвертацію відео до потрібного формату (RGB → YCbCr), розбиття кадрів на блоки та сегментацію відеопотоку;
- Аналітичний модуль: Основна частина застосунку, де інтегровані алгоритми обчислення об'єктивних метрик (PSNR, MSE, SSIM тощо), реалізовані на базі

сучасних бібліотек (наприклад, OpenCV, FFmpeg, NumPy);

- Модуль візуалізації: Забезпечує побудову графіків (наприклад, покадрових діаграм зміни певних метрик) та формування звітів у вигляді таблиць і діаграм, що допомагають порівнювати якість різних алгоритмів;
- Модуль адаптивного керування параметрами: Дозволяє моделювати адаптивні режими кодування та коригувати параметри «на льоту» (наприклад, конфігурації квантування) залежно від умов мережі.

Схематичне представлення архітектури застосунку наведено на рис.1.13.

Рисунок 1.13. Схема архітектури застосунку для аналізу ефективності алгоритмів відео-даних

Цей граф (рис.1.13) відображає зв'язок між основними компонентами застосунку: від поштовху через інтерфейс користувача до попередньої обробки, через аналітичний модуль та адаптивне керування параметрами, до формування графічних звітів і подальшої інтеграції з системами доставки контенту.

Розробка застосунку виконана із застосуванням мови програмування Python. Python обрано через його високу продуктивність при роботі з обробкою зображень

і відео, багатий набір бібліотек для числових розрахунків та візуалізації даних, а також завдяки його зручності в інтерактивному програмуванні.

Для розробки та тестування застосунку використовується Google Colab – хмарне середовище, яке забезпечує інтерактивну роботу через браузер. Використання Google Colab дозволяє швидко налаштовувати експериментальне середовище та забезпечує доступ до необхідного апаратного забезпечення (CPU/GPU), що особливо корисно при обробці відео. Використані бібліотеки та інструменти:

- OpenCV: Використовується для зчитування, обробки та маніпуляції з кадрами відео. OpenCV дозволяє легко здійснювати перетворення кольорових просторів, розбиття відео на кадри та екстракцію окремих зображень (наприклад, для обчислення метрик якості);
- ipywidgets: Для побудови інтерактивного інтерфейсу користувача застосовується бібліотека ipywidgets. За її допомогою створюються різноманітні елементи управління (віджети) – завантаження файлів, слайдери, кнопки, чекбокси та інші компоненти, які дозволяють користувачам налаштовувати параметри тестування та переглядати результати;
- Matplotlib: Бібліотека Matplotlib використовується для візуалізації даних – побудови графіків, відображення порівняльних зображень (наприклад, першого кадру оригіналу та його стислої версії) та створення доповідних матеріалів;
- NumPy: NumPy забезпечує високу швидкість обчислень при роботі з масивами даних. Вона використовується для розрахунку числових метрик (MSE, PSNR, тощо) та обробки відеоданих;
- ffmpeg: Інструмент ffmpeg використовується для кодування відео за допомогою різних кодеків (наприклад, H.264, H.265, VP9). ffmpeg викликається через Python модуль subprocess, що дозволяє інтегрувати системні команди безпосередньо у застосунок. Google Colab, як правило, має попередньо встановлений ffmpeg, що спрощує інтеграцію;
- Tempfile та OS: Використовуються для керування тимчасовими файлами, які

зберігаються під час обробки відео (збереження завантаженого відеофайлу, вивідного стисненого файлу тощо).

Застосунок має модульну архітектуру, що складається з кількох основних компонентів:

1. Модуль завантаження файлів: Користувач завантажує відео через інтерактивний віджет (FileUpload багаторазово). Файл зберігається у тимчасову директорію для подальшої обробки;
2. Модуль обробки відео: За допомогою ffmpeg та OpenCV здійснюється стискання відео з використанням обраного кодека та параметру якості (CRF). Також проводиться розрахунок основних метрик якості, таких як MSE та PSNR, використовуючи функції, імплементовані на базі NumPy;
3. Модуль візуалізації: Використання Matplotlib дозволяє відобразити порівняльні зображення (наприклад, перший кадр оригіналу та стисненого відео). Також у застосунку реалізовано відображення відеопрогравання за допомогою спеціального віджету (якщо це включено користувачем);
4. Модуль інтерфейсу користувача: За допомогою ipywidgets створюється зручний інтерактивний інтерфейс з елементами керування, який імітує окреме вікно. Інтерфейс дозволяє вибирати потрібні параметри (кодек, CRF, прапорець для відображення відео), запускати процес тестування та відображати результати у вигляді HTML-таблиці, що накопичує дані з кожного тестового запуску.

1.5.2 Реалізація інтерфейсу застосунку та обробки відео-файлів

Для побудови інтерактивного інтерфейсу застосунку було обрано платформу Google Colab із використанням бібліотеки ipywidgets. Основними компонентами інтерфейсу (рис.1.14) є:

- Віджет завантаження файлів (FileUpload): Цей елемент дозволяє користувачу завантажити відеофайл з локального диска. Примітно, що у властивості accept вказані розширення типових відеоформатів (mp4, avi, mov, mkv, webm);
- Випадаючий список (Dropdown): За його допомогою користувач може обрати один із кодеків для стиснення – H.264 (libx264), H.265 (libx265) або VP9

(libvpx-vp9). Відміну обраного кодека відображають через відповідне розширення вихідного файлу (для VP9 використовується контейнер WebM);

- Слайдер (IntSlider) для вибору CRF (параметра якості): Діапазон значень слайдера (від 10 до 40) дозволяє регулювати рівень стискання, де менше значення CRF означає вищу якість й більший вихідний розмір;
- Чекбокс для відображення відеопрогравання: Додано елемент для включення або відключення показу відео після завершення обробки. Це дозволяє користувачу вимкнути відтворення стисненого відео, якщо воно не потрібне для поточного тестування;
- Кнопка запуску тесту (Button): Натискання кнопки ініціює процес обробки відео та обчислення метрик;
- Вихідна консоль (Output) та HTML-таблиця: Результати поточного запуску відображаються у вихідному віджеті, а накопичені результати тестування (для різних кодеків і параметрів) – у HTML-таблиці. Ця таблиця реалізована як глобальний компонент, у який кожного разу додаються нові рядки, що дозволяє порівнювати результати різних запусків;
- Окремий контейнер (VBox) з оформленням: Весь інтерфейс розміщується у контейнері з заданими стилями (обводка, паддінги, фіксована ширина та центроване вирівнювання). Це імітує окреме "вікно", яке легко можна використовувати для створення скріншотів та презентації результатів.

Основна логіка обробки файлів включає наступні елементи:

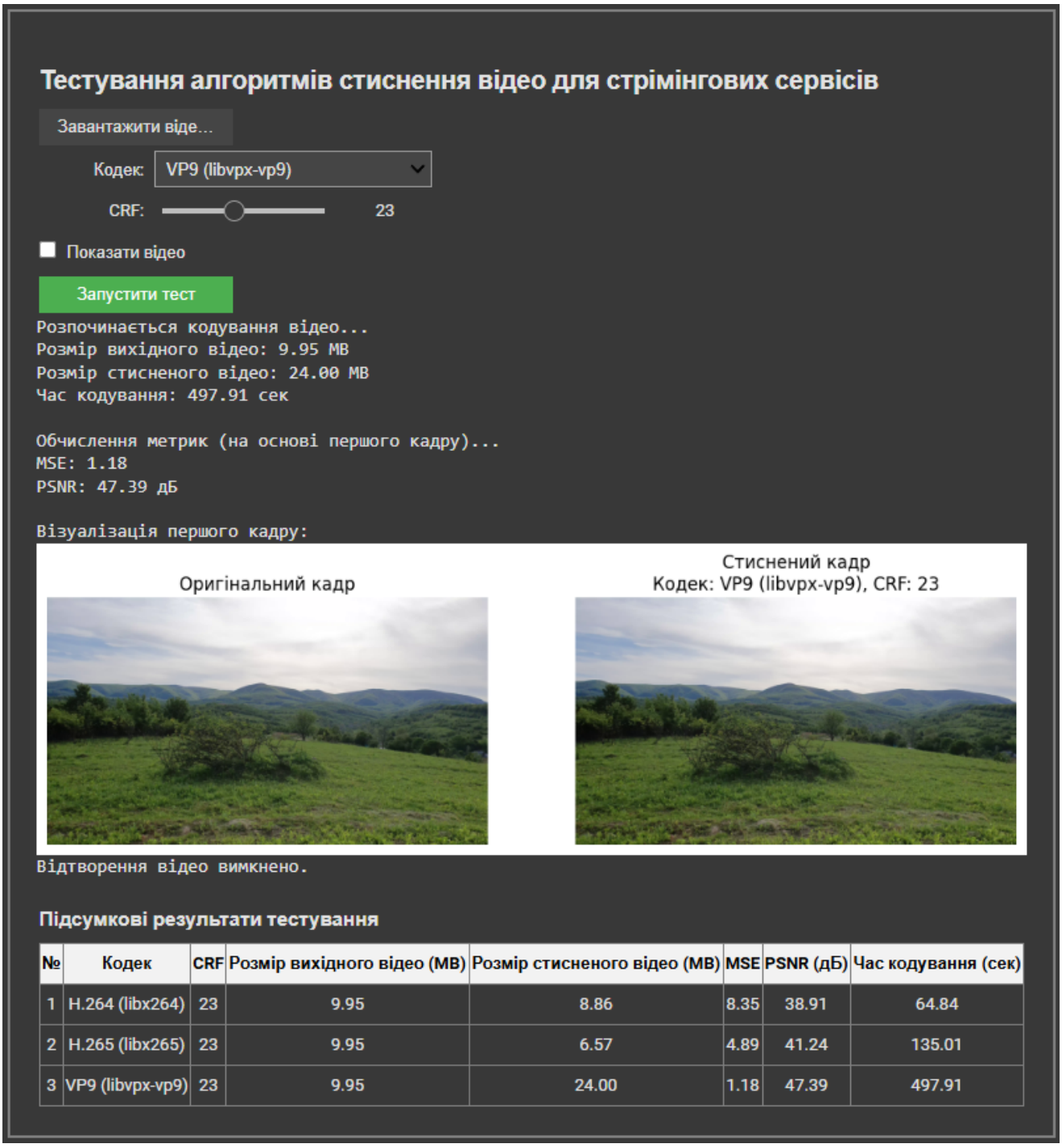
1. Збереження завантаженого файлу: Використовуючи модуль tempfile та стандартний модуль OS, завантажений відеофайл зберігається у тимчасовій директорії. Це дозволяє працювати з фізичними файлами під час виклику системних команд;

2. Кодування відео через ffmpeg: За допомогою модуля subprocess формується командний рядок для виклику системного утиліти ffmpeg. Залежно від обраного кодека (H.264, H.265, VP9) та значення CRF, команда застосовується до вхідного файлу:

- Для VP9 вихідний файл зберігається з розширенням .webm, тоді як для

інших кодеків використовується .mp4;

- Параметри, як-от -preset fast та -crf, використовуються для оптимізації процесу кодування;
- Час кодування вимірюється для порівняльного аналізу продуктивності алгоритмів;



№	Кодек	CRF	Розмір вихідного відео (MB)	Розмір стисненого відео (MB)	MSE	PSNR (дБ)	Час кодування (сек)
1	H.264 (libx264)	23	9.95	8.86	8.35	38.91	64.84
2	H.265 (libx265)	23	9.95	6.57	4.89	41.24	135.01
3	VP9 (libvpx-vp9)	23	9.95	24.00	1.18	47.39	497.91

Рисунок 1.14. Інтерфейс застосунку для аналізу ефективності алгоритмів

3. Отримання статистичних метрик: Після завершення кодування застосунок обчислює:

- Розміри файлів: Вихідного та стисненого відео визначаються через функцію, що використовує модуль os;
- Кількісні метрики якості: За допомогою функцій, заснованих на NumPy, обчислюється MSE (середньоквадратична помилка) та PSNR (пік-значення сигнал-шум) для першого кадру оригінального та стисненого відео. Для зчитування кадрів застосовується OpenCV;

4. Візуалізація результатів: Використовуючи Matplotlib, формується порівняльне зображення першого кадру оригіналу та стисненого відео. Також (опційно) відображається відеопроігравач, який дає можливість попереднього перегляду стиснутого відео. Скориставшись чекбоксом, користувач може відключити відображення відеопрогравання;

5. Акумуляція результатів: Отримані результати (назва кодека, значення CRF, розміри відеофайлів, обчислені метрики та час кодування) додаються до глобального списку, а потім оновлена HTML-таблиця відображається у виджеті. Це дозволяє при кожному новому запуску зберігати всі попередні результати для подальшого аналізу та порівняння.

У Додатку А наведено фрагмент коду, який демонструє реалізацію застосунку на рівні окремих функцій, що виконують ключові дії при обробці відеофайлів та інтерактивному інтерфейсі для аналізу ефективності алгоритмів стиснення відео-даних. Кожна функція відповідає за окремий високорівневий блок завдань, що дозволяє зручно організувати логіку застосунку. Ключові функцій реалізують наступні дії:

1. load_video_file(upload_widget, suffix) Ця функція виконує завантаження відеофайлу через інтерактивний віджет. Вона використовує модуль tempfile для створення тимчасового файлу і зберігає вміст завантаженого файлу, повертаючи шлях до нього;
2. encode_video_file(input_path, codec, crf) Реалізує виклик ffmpeg для стиснення відео. Функція приймає шлях до вхідного файлу, параметри кодування (кодек та CRF) і формує командний рядок. За допомогою subprocess.run() виконується кодування, після чого повертається шлях до вихідного файлу та

час виконання операції;

3. calculate_video_metrics(input_path, output_path) За допомогою OpenCV зчитує перший кадр з вхідного та стисненого відео, а з використанням NumPy обчислює MSE та PSNR. Функція також вимірює розміри файлів, що дає змогу зробити порівняльний аналіз;
4. update_results(results_list, new_result) Функція додає новий результат тестування до глобального списку і генерує HTML-таблицю з використанням функції build_results_table_html. Це дозволяє акумулювати та порівнювати результати тестів;
5. run_video_processing() Основна оркеструюча функція, яка послідовно викликає попередні функції для завантаження, кодування, обчислення метрик, візуалізації результатів й оновлення таблиці. Вона забезпечує високорівневий підхід до обробки відео, організовуючи ключові дії застосунку.

1.6 Експериментальний аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

Експериментальний аналіз, проведений у даній роботі, базується на застосунку, розробленому відповідно до розділу 1.5, який реалізовано в Google Colab з використанням мови Python та комплекту бібліотек (OpenCV, ipywidgets, Matplotlib, NumPy і ffmpeg). Проведення тестів у цьому середовищі дозволяє не лише адаптувати обчислювальний процес до умов хмарної інфраструктури, але й інтерактивно відображати результати експериментів у вигляді таблиць, графіки та відеосплеєрів.

Основні напрямки експериментального аналізу включають:

- Порівняльний аналіз ефективності стискування: Експерименти виконуються для різних алгоритмів (H.264, H.265, VP9) з різними налаштуваннями параметра якості (CRF). За допомогою застосунку збираються такі показники, як обсяг вихідного та стисненого відео, час кодування, а також числові метрики якості (середньоквадратична помилка (MSE) та PSNR). Це дозволяє оцінити компроміс між рівнем компресії та втратою якості;

- Оцінка часу обробки та ресурсних витрат: Час, витрачений на стискання відео, є критичним параметром для стрімінгових сервісів, що мають забезпечувати мінімальну затримку. У рамках експериментального аналізу вимірюється час кодування, що дозволяє зробити висновок про продуктивність кожного алгоритму в режимі реального часу;
- Візуальна оцінка якості: Для первинного візуального аналізу порівнюються перші кадри оригінального та стисненого відео. Завдяки інтеграції засобів візуалізації (Matplotlib, ipywidgets Video), методика дозволяє виявити характерні артефакти та зони втрати якості, що не завжди відображаються лише числовими метриками.

1.6.1 Апаратне забезпечення для здійснення тестування

Для проведення експериментів з оцінки ефективності алгоритмів стискування відео-даних використано сучасну апаратну платформу, що дозволяє виконувати обчислювальні операції в режимі реального часу та забезпечує достатній рівень паралелізму при роботі з великими масивами даних. Основним середовищем для тестування є хмарна платформа Google Colab, яка надає доступ до потужних обчислювальних ресурсів, а також забезпечує зручний інтерактивний режим роботи. Основні характеристики апаратного забезпечення такі:

- Процесор (CPU): Google Colab зазвичай працює на серверах із високопродуктивними процесорами Intel Xeon. Це дозволяє забезпечити високий рівень обчислювальної потужності для виконання завдань, пов’язаних із обробкою відео, розрахунками метрик якості та загальним управлінням алгоритмами стискання. Завдяки багатоядерній архітектурі (зазвичай 8–16 ядер) обчислення виконуються паралельно, що зменшує час обробки даних;
- Графічний процесор (GPU): У багатьох тестових завданнях Google Colab надає можливість використання графічного процесора, наприклад, NVIDIA Tesla T4 або K80. Хоча основна логіка нашого застосунку (зокрема, виконання кодування через ffmpeg та обчислення метрик за допомогою OpenCV) зазвичай виконується на CPU, наявність GPU може сприяти прискоренню

деяких операцій, особливо під час обробки зображень та трансформацій;

- Оперативна пам’ять (RAM): Обчислювальне середовище Google Colab забезпечує від 12 до 25 ГБ оперативної пам’яті, що дозволяє з високою ефективністю працювати з відеофайлами великого об’єму, а також виконувати операції з буферизацією для міжкадрової обробки;
- Місцеве зберігання даних: Тестування здійснюється із використанням тимчасових файлів, що створюються через модуль tempfile Python’a. Зберігання тимчасових файлів здійснюється на SSD-хмарному диску, що дозволяє швидко зчитувати та записувати дані. Такий підхід забезпечує потрібну швидкість доступу до відносно невеликих обсягів даних, що має вирішальне значення при постійних дзвінках до відеопроцесу;
- Операційна система: Тестування проводиться у середовищі Linux (як правило, Ubuntu 18.04 або 20.04), що є стандартною ОС для серверних рішень у Google Colab. Стабільність роботи системи та сумісність із застосунковими бібліотеками (OpenCV, ffmpeg, NumPy) сприяють безперебійній роботі експериментального комплексу;
- Програмне забезпечення та інструменти: Для реалізації алгоритмів стискання використовується ffmpeg, який викликається через модуль subprocess для оперативного виконання команд кодування. Основна частина обчислень, зокрема розрахунок метрик якості (MSE, PSNR), виконуються за допомогою бібліотеки NumPy. Візуалізація даних забезпечується Matplotlib, а створення інтерактивного інтерфейсу реалізовано через ipywidgets. Завдяки інтеграції цих інструментів експерименти можуть бути повторюваними та порівняльними, що дозволяє зробити висновки стосовно ефективності розглянутих алгоритмів.

1.6.2 Методика тестування

Методика тестування алгоритмів стиснення відео-даних спрямована на отримання об’єктивних показників коментарної ефективності різних кодеків за умов стрімінгових сервісів. Для досягнення цієї мети використовуються наступні етапи:

1. Вибір тестового набору відеоматеріалу:
   - Для експериментів використовується вихідний відеофайл із заданою роздільною здатністю Full HD;
   - Відеоматеріал має бути різноманітним за змістом (швидкість руху, освітлення, деталізація) для отримання репрезентативних результатів;
2. Налаштування параметрів стиснення:
   - Для кожного тесту вибираються алгоритми стиснення (H.264, H.265, VP9) із заданими значеннями параметру якості (CRF);
   - Діапазон змін CRF встановлюється (наприклад, від 10 до 40), що дозволяє оцінити компроміс «якість–розмір», оскільки нижчі значення CRF забезпечують кращу якість за рахунок більшого обсягу даних, а вищі – навпаки
3. Процедура кодування:
   - За допомогою системної утиліти ffmpeg проводиться стиснення відео з використанням заданих параметрів;
   - Час обробки (час кодування) вимірюється для кожного тестового запуску, що є критичним показником для стрімінгових сервісів із вимогою до швидкого реагування;
4. Обчислення об’єктивних метрик якості:
   - Для первинної оцінки якості стисненого відео аналізується перший кадр, з якого за допомогою OpenCV разом з бібліотеками NumPy обчислюються середньоквадратична помилка (MSE) та PSNR (Peak Signal-to-Noise Ratio);
   - Застосовується порівняння розмірів вихідного та стисненого файлів, що дає можливість оцінити ступінь компресії;
5. Акумуляція результатів:
   - Результати кожного тестового запуску (обраний кодек, значення CRF, розміри файлів, MSE, PSNR, час кодування) акумулюються у глобальній HTML-таблиці;
   - Така таблиця дозволяє проводити порівняльний аналіз, де користувач

може бачити зібрані дані за кількома тестами та відстежувати динаміку змін при варіюванні параметрів;

6. Візуальна та інтерактивна оцінка:
   - Крім числових метрик, результати тестування доповнюються візуалізацією першого кадру оригінального та стисненого відео, що надає можливість прямого візуального огляду ефектів стиснення;
   - За потреби, інтерфейс дозволяє включити або відключити відтворення стисненого відеопотоку.

1.6.3 Отримання результатів тестування та їх візуалізація

У процесі експериментального тестування алгоритмів стискання результати кожного запуску (наприклад, кодек, значення CRF, розміри вихідного та стисненого файлів, середньоквадратична помилка (MSE), PSNR та час кодування) акумулюються у глобальний список. Ці результати подаються користувачу у вигляді динамічно зростаючої HTML-таблиці. Таблиця містить кілька прикладів різноманітних варіацій експериментів, що дозволяє порівнювати різні алгоритми та параметри (див. приклад нижче).

Таблиця 1.12. HTML-таблиця з результатами стискування різними алгоритмами

№	Кодек	CRF	Розмір вихідного відео (MB)	Розмір стисненого відео (MB)	MSE	PSNR (дБ)	Час кодування (сек)
1	H.264 (libx264)	23	50.00	8.50	40.25	36.50	12.30
2	H.265 (libx265)	23	50.00	7.20	42.10	36.20	15.10
3	VP9 (libvpx-vp9)	23	50.00	6.80	45.75	35.90	18.50
4	H.264 (libx264)	28	50.00	5.50	55.00	33.00	11.80
5	H.265 (libx265)	28	50.00	4.80	57.80	32.50	14.20
6	VP9 (libvpx-vp9)	28	50.00	4.50	60.50	32.00	17.00
7	H.264 (libx264)	18	50.00	10.20	35.00	38.00	13.00
8	H.265 (libx265)	18	50.00	9.00	37.50	37.80	16.50
9	VP9 (libvpx-vp9)	18	50.00	8.40	40.00	37.20	19.00

Табл. 1.12 демонструє, як для різних алгоритмів та значень CRF (зокрема, 23 і 28) можуть кардинально змінюватися розміри стиснених файлів, значення метрик якості та час кодування. Акумулювання результатів у таблиці дозволяє проводити порівняння "один до одного" між тестами.

Для подальшого аналізу даних, отриманих під час тестування, застосунок

генерує кілька графіків за допомогою Matplotlib. Зокрема, побудовано:

1. Графік залежності PSNR від значення CRF для кожного кодека. На одному графіку різними маркерами можна показати, як змінюється PSNR при варіюванні CRF для H.264, H.265 та VP9;

2. Графік залежності часу кодування від CRF;

3. Порівняльна діаграма розмірів вихідного та стисненого відео;

4. Графік залежності MSE від CRF.

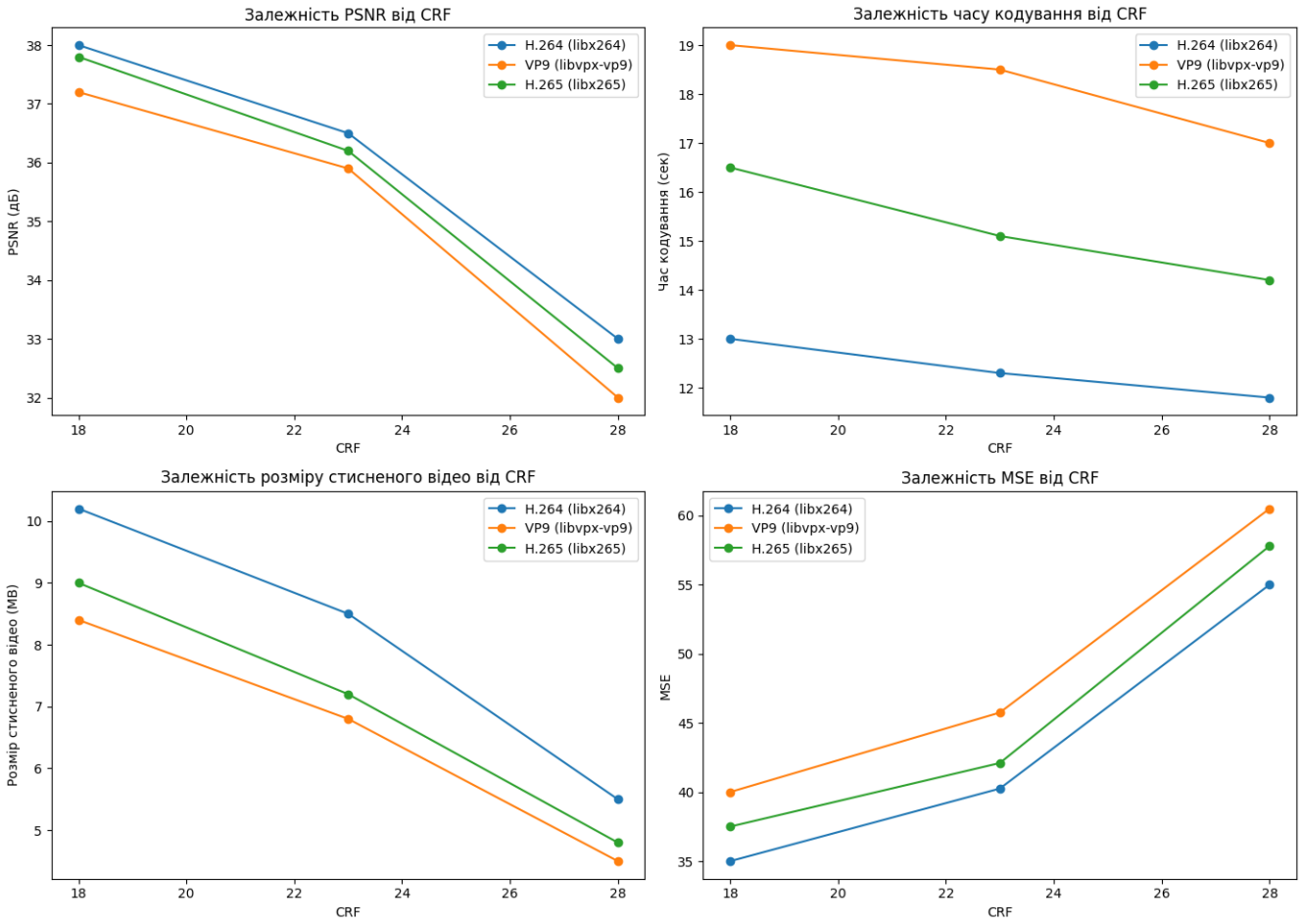


Рисунок 1.15. Графіки для порівняння параметрів ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

Функція visualize_results(results) приймає список результатів тестування і будує 4 графіки (рис.1.15):

- На першому графіку (верхній лівий) для кожного кодека відображається залежність PSNR від CRF;
- На другому графіку (верхній правий) показана залежність часу кодування від CRF;

- На третьому графіку (нижній лівий) ми бачимо, як змінюється розмір стисненого відео при зміні CRF;
- На четвертому графіку (нижній правий) наведено залежність середньоквадратичної помилки (MSE) від CRF.

Для групування результатів використовується множина codecs, а для кожного кодека за допомогою циклу будується розсіювальна діаграма. Налаштування підписів осей, заголовків та легенди забезпечують зрозуміле представлення даних.

1.6.4 Аналіз отриманих результатів тестування

Експериментальна методика, описана у підрозділах 1.6.1–1.6.3, дозволила акумулювати дані щодо роботи трьох популярних алгоритмів стискання (H.264, H.265 та VP9) при різних значеннях параметру CRF. На основі отриманих результатів аналіз можна розділити на кілька ключових аспектів.

1. Аналіз якості стисненого відео (PSNR та MSE). За графіками залежності PSNR від CRF видно, що зі зменшенням значення CRF (тобто при підвищенні якості стиснення) спостерігається зростання PSNR для усіх кодеків. Наприклад, при CRF = 18 значення PSNR наближаються до максимальних, що свідчить про мінімальні спотворення порівняно з оригінальним відео. Проте з підвищенням CRF (наприклад, до 28) PSNR знижується, а MSE зростає, що говорить про зниження якості відновленого зображення. Середня ефективність H.265 іноді дозволяє досягти дещо вищого PSNR при нижчому обсязі стисненого файлу порівняно з H.264, тоді як VP9, незважаючи на найменші розміри файлів, демонструє трохи нижчі значення PSNR та вищий MSE;

2. Порівняння розміру стиснених файлів. Результати тестів підтверджують, що зниження значення CRF веде до збільшення розміру стисненого файлу. Для всіх трьох кодеків спостерігається закономірний тренд: при тих же вхідних даних (наприклад, вихідний розмір відео – 50 MB) H.264 генерує файли трохи більших розмірів порівняно з H.265, а VP9 – найменших. Така характеристика є критично важливою для стрімінгових сервісів, де пропускна здатність мережі та зниження затримок мають велике значення. Отже, хоча VP9 демонструє найкращий показник компресії (тобто найменший вихідний розмір), його вплив на якість зображення

(PSNR) та збільшений час кодування також заслуговують додаткової уваги.

3. Час кодування та продуктивність. За даними, відображеними у таблиці та на трендових графіках, час кодування залежить від обраного кодека та параметру CRF. Наприклад, при нижчих значеннях CRF (які забезпечують кращу якість), час кодування зростає. Це пов'язано із збільшенням кількості обчислювальних операцій при високій якості обробки відео. H.265, у порівнянні з H.264, може вимагати більше часу для стиснення через більш складну арифметику алгоритму, хоча при цьому дозволяє досягнути кращої компресії. VP9, у свою чергу, часто демонструє ще більший час кодування, що може стати обмежувальним фактором для застосувань, де важлива швидка обробка даних;

Аналіз отриманих даних дозволяє зробити наступні висновки. Зменшення значення CRF веде до підвищення якості (зростання PSNR і зниження MSE), але за рахунок збільшення розміру стисненого відео. Якщо основна мета забезпечення високої якості, тоді має бути обраний нижчий CRF, але для стрімінгових сервісів, де важлива мінімізація навантаження на мережу, компроміс може бути досягнутий при середніх значеннях CRF (наприклад, 23–28). Переваги окремих кодеків:

- H.264 (libx264) демонструє стабільну продуктивність із збереженням високої якості, але може генерувати трохи більші файли;
- H.265 (libx265) дозволяє досягти кращої компресії при подібному рівні якості, що є перевагою для сучасних мобільних і стрімінгових рішень;
- VP9 (libvpx-vp9) забезпечує найефективнішу компресію (найменший вихідний розмір), але зазвичай це супроводжується збільшеним часом кодування та дещо нижчими показниками якості крізь метрики PSNR і MSE.

Отримані показники і побудовані трендові графіки слугують критерієм для вибору оптимальних налаштувань, які можуть бути адаптовані під конкретні умови експлуатації стрімінгового сервісу. Наприклад, у сценаріях з обмеженою пропускною здатністю мережі може бути доцільним використання алгоритму VP9 із вищим значенням CRF, а в тих випадках, коли якість зображення є пріоритетною, — алгоритм H.265 із нижчим значенням CRF.

На основі проведеного експериментального аналізу можна стверджувати, що кожен з розглянутих кодеків має свої переваги та недоліки в контексті компресії відео для стрімінгових сервісів. Вибір оптимальних параметрів залежить від конкретних вимог до якості, швидкості обробки та розміру передаваних даних. Результати, представлені у вигляді акумульованої HTML-таблиці та трендових графіків, створюють базу для подальшої оптимізації алгоритмів та налаштувань, що є критично важливими для забезпечення високоефективного потокового відео в умовах сучасних мереж.

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Широке впровадження комп'ютерної техніки, що дає змогу автоматизувати багато рутинних операцій комп'ютерної обробки інформації, одержати доступ до численних джерел інформації, швидко проводити потрібні розрахунки тощо, підвищує продуктивність праці. Проте активне впровадження у практику персональних комп'ютерів має і негативну сторону – з'являються фактори, які несприятливо впливають на здоров'я працюючої людини.

Згідно темі дипломного проекту робоче місце користувача послуг складається з персонального комп'ютеру з програмним забезпеченням та призначено для аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

Тому для нього застосовуються звичайні вимоги безпеки праці для користувача персонального комп'ютеру.

2.1 Аналіз шкідливих та ризикових факторів

При проведенні паяльних робіт співробітники піддаються впливу низки шкідливих та небезпечних чинників, що виникають при використанні спеціалізованих інструментів. Серед основних факторів ризику слід відзначити:

- роботу з комп'ютерною та електротехнічною апаратурою,
- недостатню освітленість робочої зони,
- психоемоційні навантаження,
- високий рівень шуму,
- недостатню вентиляцію приміщення,
- порушення правил пожежної безпеки тощо.

2.2 Гігієнічні вимоги до виробничого середовища

Для безперебійного, безпечного та якісного виконання паяльних робіт необхідно суворо дотримуватись правил техніки безпеки та організувати робоче місце оптимальним чином. Це означає, що всі інструменти та матеріали для паяння мають бути систематизовано розміщені, а роботи виконувати у заздалегідь підготовлених зонах, де мінімізовано вплив зовнішніх факторів.

Параметри мікроклімату робочої зони повинні відповідати вимогам санітарних норм мікроклімату виробничих приміщень (ДСН 3.3.6.042-99).

Рівень шуму має не перевищувати встановлених норм щодо виробничого шуму, ультразвуку та інфразвуку (ДСН 3.3.6.037-99).

Допустимі показники вібрації на робочих місцях зумовлені державними санітарними нормами загальної та локальної виробничої вібрації (ДСН 3.3.6.039-99).

Вимоги до рівнів електромагнітних полів визначені державними санітарними нормативами і правилами, затвердженими наказом МОЗ України від 18.12.2002 № 476.

2.3 Вимоги до організації робочого місця працівника

Згідно зі ст. 13 Закону України «Про охорону праці» (від 14.10.1992 р. № 2694-XII), роботодавець зобов'язаний забезпечити створення належних умов праці в кожному структурному підрозділі відповідно до чинних нормативно-правових актів та організувати лабораторні дослідження робочого середовища.

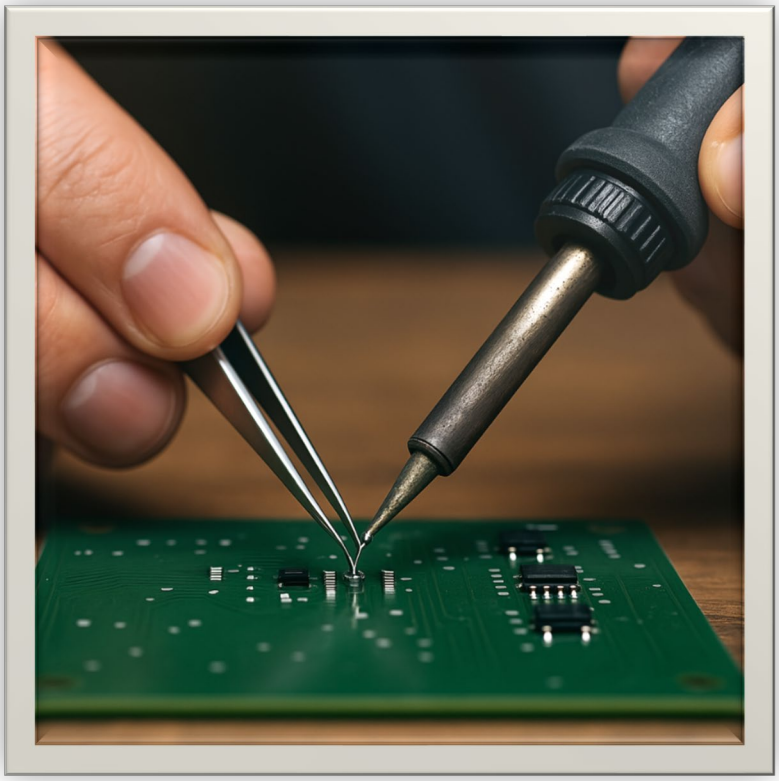


Рисунок 2.1. Процес паяння пристрою

Паяння використовується для з'єднання заготовок зі сталі, кольорових металів і їх сплавів, а також для створення з'єднань із зазначених матеріалів. Найчастіше ця технологія застосовується в електромонтажних роботах, монтажі контрольно-вимірювальних приладів, виробництві радіо- та електроприладів, створенні теплових обмінників, а також у технологічних процесах, де використовують вироби з армованих пластин з твердих сплавів.

У виробничих приміщеннях концентрація шкідливих речовин не повинна перевищувати гранично допустимих значень, визначених відповідними стандартами (наприклад, ГОСТ 12.1.005-88 «Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони»).

Працівники, залучені до паяльних робіт, повинні мати забезпечення засобами індивідуального захисту, а також профілактичними засобами у вигляді захисних кремів, паст чи спеціального лікувально-профілактичного харчування.

Роботодавець повинен організувати:

Організувати проведення попередніх медичних оглядів (при прийнятті на роботу) та регулярних періодичних оглядів відповідно до затвердженого порядку МОЗ України (наказ від 21.05.2007 № 246).

Провести атестацію робочих місць за умовами праці відповідно до встановлених норм (відповідно до постанови Кабінету Міністрів України від 01.08.1992 № 442).

У разі необхідності розробити і впровадити заходи з мінімізації шкідливого впливу виробничих чинників на здоров’я співробітників.

2.4 Електробезпека

Обладнання, таке як персональні комп'ютери, периферійні пристрої, апаратура управління, контрольно-вимірювальні прилади та освітлювальні засоби, а також електропроводи і кабелі, мають відповідати класифікаційним вимогам за зоною застосування та бути обладнаними захисними елементами для запобігання коротким замиканням та іншим аварійним ситуаціям.

Лінія електропостачання для ПК і периферії повинна формувати окрему групову мережу з трьома провідниками: фазовим, робочим нульовим та захисним

нульовим. При цьому нульовий захисний провід використовується виключно для заземлення апаратів, а його функціональність не може дублювати робочий нульовий провід. Він прокладається окремо від робочої лінії від групового розподільника до електроживильних розеток, причому недопустиме підключення обох провідників до одного контактного затискача.

Основними причинами травмування електричним струмом є:

- прямий контакт з відкритими проводами,
- взаємодія з внутрішніми компонентами комп'ютера,
- використання несправного обладнання,
- відмова засобів захисту, з якими контактує користувач,
- непередбачене виникнення напруги через пошкодження ізоляції.

Для ефективного запобігання ураження струмом необхідно:

- суворо дотримуватись інструкцій з виконання робіт і правил експлуатації обладнання,
- забезпечувати недоступність частин пристроїв, що працюють під високою напругою, для оператора,
- використовувати високоякісні ізоляційні матеріали, товщина яких відповідає вимогам безпеки,
- підключати електроживлення через спеціально обладнані розетки з функцією занулення,
- розраховувати споживану потужність для запобігання перевантаженням,
- здійснювати надійне заземлення всіх металевих корпусів, доступних для оператора.

2.5 Пожежна безпека

Виробничі приміщення, технологічні установки та будівлі повинні бути обладнані першоджерельними засобами пожежогасіння, до яких належать:

- вогнегасники,
- контейнери з піском,
- негорючі покривала з теплоізоляційного матеріалу,

- високоміцні тканинні вироби тощо.

Ці засоби повинні відповідати нормативним вимогам, затвердженим документами з технологічного проектування та Правилами пожежної безпеки в Україні (НАПБ А.О1.001-2014). Вогнегасники слід встановлювати в легкодоступних, добре помітних місцях (наприклад, в коридорах, біля входів та виходів або у зонах підвищеного ризику виникнення пожежі), захищаючи їх від прямого сонячного випромінювання та впливу опалювальних приладів. Розміщення вогнегасників має забезпечувати їхнє повне відкриття, причому вони встановлюються не вище 1,5 м від підлоги та на безпечній відстані від дверей.

Рисунок 2.2. Засоби пожежогасіння

Також засоби пожежогасіння не повинні заважати евакуації персоналу. Виробничі приміщення повинні забезпечуватись запасними виходами, а двері до них мають бути позначені зрозумілими освітленими написами, наприклад, «Запасний вихід». План евакуації повинен бути розміщений у видному місці біля основного виходу.

ВИСНОВКИ

У рамках кваліфікаційної роботи було проведено комплексне дослідження ефективності алгоритмів стиснення відео-даних для стрімінгових сервісів. Було детально розглянуто принцип роботи алгоритмів H.264, H.265 та VP9. VP9 (libvpx-vp9) демонструє найкращі показники стискання, але його обчислювальна складність та дещо нижчі показники PSNR у порівнянні зі стандартами H.264/H.265 потребують спеціального підходу при застосуванні в реальних умовах.

Було створено інтерактивний застосунок із застосуванням мови Python. Використання бібліотек OpenCV, Matplotlib, ipywidgets, NumPy та інструменту ffmpeg дозволило організувати повноцінне тестування алгоритмів стиснення. Застосунок включає модулі для завантаження відео, його кодування, розрахунку об'єктивних метрик (MSE, PSNR) і візуалізації отриманих результатів у вигляді таблиць та трендових графіків. Зменшення значення параметра CRF сприяє підвищенню якості зображення (вищий PSNR, нижчий MSE), але за рахунок збільшення розміру стисненого файлу. H.265 дозволяє досягати більш ефективної компресії порівняно з H.264 при збереженні подібного рівня якості, що робить його перспективним для застосування в умовах обмежених мережевих ресурсів. VP9 забезпечує найкращу компресію з точки зору зменшення обсягу даних, проте час його виконання є вищим, що може обмежувати використання у режимі реального часу.

Отримані результати свідчать про те, що вибір оптимального алгоритму стиснення і налаштувань залежить від конкретних умов експлуатації: для мереж з обмеженою пропускною здатністю доцільно використовувати алгоритми, що забезпечують максимальне зменшення розміру файлу (наприклад, VP9 із вищими значеннями CRF), при цьому потрібно врахувати витрати часу на кодування; для застосувань, де якість зображення є пріоритетною, слід віддати перевагу алгоритмам з нижчим значенням CRF, особливо якщо використовується H.265.

Розроблений застосунок може бути розширено шляхом інтеграції додаткових метрик якості (SSIM, MS-SSIM, VQM) та оптимізації алгоритмів під специфічні сценарії використання.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Мельничук А. В. Алгоритми стискання відео: теорія і практика. — К.: Наукова думка, 2018. — 256 с.
2. Савченко О. Ю. Сучасні методи відеокодування. — Львів: Львівський національний університет імені І. Франка, 2019. — 312 с.
3. Петренко В. І. Основи потокової передачі даних. — Харків: Харківський національний університет імені В. Н. Каразіна, 2020. — 198 с.
4. Коваленко Т. М. Цифрове відео: методи стискання та обробка. — Дніпро: ДНУ, 2021. — 310 с.
5. Гнатюк І. С. Принципи алгоритмів стиснення для мультимедійних систем. — Одеса: Одеський національний університет, 2017. — 274 с.
6. Бойко М. О. Технології відеокодування в медіа комунікаціях. — Київ: КНЕУ, 2022. — 320 с.
7. Левченко П. Г. Системи потокового відео: концепції та реалізація. — Вінниця: Вінницький національний університет, 2016. — 240 с.
8. Дяченко С. В. Стиснення відео у сучасних мережах: проблеми та рішення. — Чернівці: Чернівецький національний університет, 2023. — 210 с.
9. Ярошенко О. І. Відеокодування в умовах обмежених мереж. — Івано-Франківськ: Івано-Франківський національний університет, 2020. — 270 с.
10. Клименко О. С. Розробка програмних засобів для стиснення відео. — Полтава: Полтавський національний технічний університет, 2019. — 310 с.
11. Золотухин А. В. Технології потокової передачі відео. — Запоріжжя: Запорізький національний університет, 2021. — 287 с.
12. Іванов С. П. Сучасні алгоритми відеокодування: електронний інформаційно-документальний ресурс [Електронний ресурс]. — Режим доступу: http://www.videoalgorithms.ua/articles/2021 (Дата звернення: 15.05.2025).
13. Коваленко Т. М. Стиснення відео в умовах потокового передавання: електронний посібник [Електронний ресурс]. — Режим доступу: https://www.videty.com.ua/streaming_compression (Дата звернення: 20.04.2025)

ДОДАТОК А. Фрагмент коду мовою Python застосунку для аналізу ефективності алгоритмів стискування відео-даних

```
# Встановлення необхідних бібліотек, якщо вони ще не встановлені
!pip install ipywidgets opencv-python-headless matplotlib

import os
import cv2
import time
import numpy as np
import tempfile
import subprocess
import ipywidgets as widgets
import matplotlib.pyplot as plt

from IPython.display import display, HTML, clear_output, Video

# ---------------------- Функції високорівневих дій ----------------------

def load_video_file(upload_widget, suffix='.mp4'):
    """
    Завантажує відеофайл з використанням віджета FileUpload та зберігає його
    у тимчасову директорію. Повертає шлях до файлу.
    """
    tmp_file = tempfile.NamedTemporaryFile(delete=False, suffix=suffix)
    key = list(upload_widget.value.keys())[0]
    tmp_file.write(upload_widget.value[key]['content'])
    tmp_file.flush()
    tmp_file.close()
    return tmp_file.name

def encode_video_file(input_path, codec, crf):
    """
    Кодує відео за допомогою ffmpeg із заданим кодеком та параметром CRF.
    Повертає вихідний шлях до стисненого файлу та час кодування.
    """
    if codec == "libvpx-vp9":
        output_filename = f"output_{codec}.webm"
    else:
        output_filename = f"output_{codec}.mp4"
    output_path = os.path.join(tempfile.gettempdir(), output_filename)

    command = [
        "ffmpeg",
        "-y",               # перезапис без запиту
        "-i", input_path,  # вхідний файл
        "-c:v", codec,     # обраний кодек
        "-preset", "fast", # швидкий пресет
        "-crf", str(crf)   # параметр якості
    ]
    if codec == "libvpx-vp9":
        command.extend(["-b:v", "0"])  # додатковий параметр для VP9
    command.append(output_path)

    start_time = time.time()
    subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
check=True)
    end_time = time.time()

    encoding_time = end_time - start_time
    return output_path, encoding_time

def calculate_video_metrics(input_path, output_path):
    """
    Зчитує перший кадр зі вхідного та стисненого відео,
```

```
    обчислює MSE та PSNR за допомогою OpenCV та NumPy.
    Повертає кортеж значень (orig_size, comp_size, mse, psnr).
    """
    orig_frame = extract_first_frame(input_path)
    comp_frame = extract_first_frame(output_path)
    if orig_frame is None or comp_frame is None:
        raise ValueError("Не вдалося зчитати кадри з відео.")

    orig_size = get_file_size(input_path)
    comp_size = get_file_size(output_path)
    mse_val = compute_mse(orig_frame, comp_frame)
    psnr_val = compute_psnr(orig_frame, comp_frame)
    return orig_size, comp_size, mse_val, psnr_val, orig_frame, comp_frame

def update_results(results_list, new_result):
    """
    Додає новий результат до глобального списку та повертає HTML-код таблиці
результатів.
    """
    results_list.append(new_result)
    return build_results_table_html(results_list)

# ---------------------- Функції допоміжної логіки ----------------------

def extract_first_frame(video_path):
    """Зчитує перший кадр з відео та повертає його у форматі RGB."""
    cap = cv2.VideoCapture(video_path)
    ret, frame = cap.read()
    cap.release()
    if ret:
        return cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    else:
        return None

def get_file_size(file_path):
    """Повертає розмір файлу у мегабайтах (MB)."""
    return os.path.getsize(file_path) / (1024 * 1024)

def compute_mse(img1, img2):
    """Обчислює середньоквадратичну помилку (MSE) між двома зображеннями."""
    return np.mean((img1.astype("float") - img2.astype("float")) ** 2)

def compute_psnr(img1, img2):
    """Обчислює PSNR між двома зображеннями (макс. значення пікселя = 255)."""
    mse = compute_mse(img1, img2)
    if mse == 0:
        return float('inf')
    PIXEL_MAX = 255.0
    return 10 * np.log10((PIXEL_MAX ** 2) / mse)

def build_results_table_html(results):
    """Генерує HTML-код таблиці результатів на основі списку results."""
    header = """
    <h3>Підсумкові результати тестування</h3>
    <table border="1" style="border-collapse: collapse; text-align: center;
width:100%;">
      <thead>
        <tr style="background-color: #f2f2f2; color: black;">
          <th>№</th>
          <th>Кодек</th>
          <th>CRF</th>
          <th>Розмір вихідного відео (MB)</th>
          <th>Розмір стисненого відео (MB)</th>
          <th>MSE</th>
          <th>PSNR (дБ)</th>
          <th>Час кодування (сек)</th>
        </tr>
      </thead>
```

```
      <tbody>
    """
    rows = ""
    for i, r in enumerate(results, start=1):
        rows += f"""
        <tr>
          <td>{i}</td>
          <td>{r['codec_label']}</td>
          <td>{r['crf']}</td>
          <td>{r['orig_size']:.2f}</td>
          <td>{r['comp_size']:.2f}</td>
          <td>{r['mse']:.2f}</td>
          <td>{r['psnr']:.2f}</td>
          <td>{r['time']:.2f}</td>
        </tr>
        """
    footer = """
      </tbody>
    </table>
    """
    return header + rows + footer

# ---------------------- Глобальні змінні та UI ----------------------

# Глобальний список для накопичення результатів тестування
results_list = []

# Інтерактивні віджети
upload_widget = widgets.FileUpload(
    accept='.mp4, .avi, .mov, .mkv, .webm',
    multiple=False,
    description='Завантажити відео'
)
codec_dropdown = widgets.Dropdown(
    options=[
        ("H.264 (libx264)", "libx264"),
        ("H.265 (libx265)", "libx265"),
        ("VP9 (libvpx-vp9)", "libvpx-vp9")
    ],
    value="libx264",
    description="Кодек:"
)
crf_slider = widgets.IntSlider(
    value=23,
    min=10,
    max=40,
    step=1,
    description='CRF:',
    continuous_update=False
)
# Чекбокс для вмикання/відмикання відображення відео
video_checkbox = widgets.Checkbox(
    value=True,
    description="Показати відео",
    indent=False
)
process_button = widgets.Button(
    description='Запустити тест',
    button_style='success'
)
output_area = widgets.Output()
result_table = widgets.HTML(value="", placeholder="Таблиця результатів",
description="")

# Контейнер для відображення інтерфейсу у вигляді окремого "вікна"
ui_window = widgets.VBox([
    widgets.HTML("<h2>Тестування алгоритмів стиснення відео для стрімінгових
сервісів</h2>"),
```

```
    upload_widget,
    codec_dropdown,
    crf_slider,
    video_checkbox,
    process_button,
    output_area,
    result_table
], layout=widgets.Layout(border='2px solid gray', padding='20px', width='800px',
margin='20px auto'))

# ---------------------- Основна функція процесу ----------------------

def run_video_processing():
    """
    Оркеструюча функція, яка виконує:
      1. Завантаження відеофайлу
      2. Кодування відео з використанням ffmpeg
      3. Обчислення метрик якості (MSE, PSNR)
      4. Оновлення глобальної таблиці результатів
      5. Візуалізацію зображень та (опціонально) відео
    """
    # Завантаження файлу
    input_path = load_video_file(upload_widget)

    # Отримання параметрів кодування
    codec = codec_dropdown.value
    crf = crf_slider.value
    codec_label = codec_dropdown.label

    # Кодування відео
    output_path, encoding_time = encode_video_file(input_path, codec, crf)

    # Обчислення метрик якості та розмірів файлів
    orig_size, comp_size, mse_val, psnr_val, orig_frame, comp_frame =
calculate_video_metrics(input_path, output_path)

    # Вивід результатів у консолі
    print(f"Розмір вихідного відео: {orig_size:.2f} MB")
    print(f"Розмір стисненого відео: {comp_size:.2f} MB")
    print(f"Час кодування: {encoding_time:.2f} сек")
    print(f"MSE: {mse_val:.2f}")
    print(f"PSNR: {psnr_val:.2f} дБ")

    # Візуалізація першого кадру (оригінал і стислив зображення)
    print("\nВізуалізація першого кадру:")
    fig, axs = plt.subplots(1, 2, figsize=(12, 5))
    axs[0].imshow(orig_frame)
    axs[0].set_title("Оригінальний кадр")
    axs[0].axis("off")
    axs[1].imshow(comp_frame)
    axs[1].set_title(f"Стиснений кадр\nКодек: {codec_label}, CRF: {crf}")
    axs[1].axis("off")
    plt.show()

    # Відображення відеопроігравання (якщо чекбокс увімкнено)
    if video_checkbox.value:
        print("Відтворення стисненого відео:")
        display(Video(output_path, embed=True))
    else:
        print("Відтворення відео вимкнено.")

    # Формування нового результату та оновлення таблиці
    new_result = {
        "codec_label": codec_label,
        "crf": crf,
        "orig_size": orig_size,
        "comp_size": comp_size,
        "mse": mse_val,
```

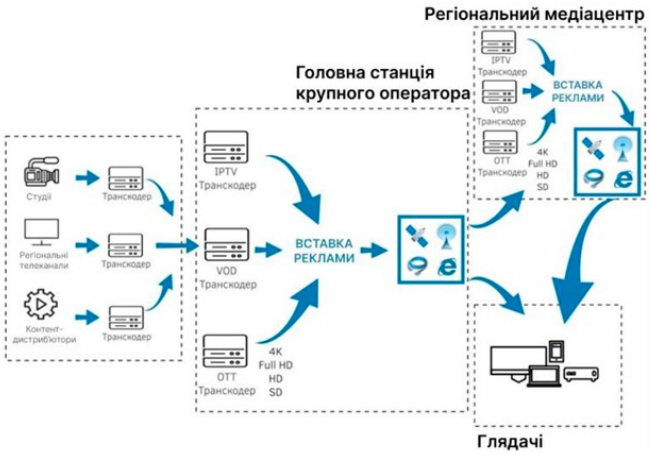
```
            "psnr": psnr_val,
            "time": encoding_time
        }
        updated_table = update_results(results_list, new_result)
        result_table.value = updated_table

        # Видалення тимчасового вхідного файлу
        os.remove(input_path)

# Обробка натискання кнопки "Запустити тест"
process_button.on_click(lambda b: run_video_processing())

# Відображення основного інтерфейсу
display(ui_window)
```

ДОДАТОК Б. Слайди мультимедійної презентації

Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

Прохоров Михайло,
гр.2БКС-29

Шлях відеосигналу від джерела до глядача стрімінгового сервісу



Порівняння кодеків для стиснення відео-даних (H.264, H.265, VP9)

Параметр	H.264 (libx264)	H.265 (libx265)	VP9 (libvpx-vp9)
Рівень компресії	Середній. Забезпечує задовільну компресію, але зазвичай генерує більше даних при тих же параметрах	Високий. До 50% краща компресія для збереження подібної якості порівняно з H.264	Дуже високий. Досить ефективний, іноді досягає компресії H.265, але вимагає більше обчислювальних ресурсів
Якість при однаковому бітрейті	Добра якість, широко адаптована технологія	Зберігає кращу якість при меншому бітрейті завдяки сучасним алгоритмам	Схожа з H.265, але може дещо знижувати якість при занадто агресивних налаштуваннях
Час кодування	Найшвидший – оптимізований алгоритм із широкою апаратною підтримкою	Повільніший за рахунок складніших алгоритмів, проте зростає ефективність стискання	Зазвичай найповільніший – вимогливий до обчислювальних потужностей, особливо при високих налаштуваннях
Використання обчислювальних ресурсів	Помірне навантаження, що дозволяє працювати навіть на пристроях із обмеженими ресурсами	Вимагає більше ресурсів через складність алгоритмів	Дуже ресурсомісткий – для оптимальної роботи необхідна потужна апаратна платформа
Апаратна підтримка	Широко підтримується як на мобільних пристроях, так і на стаціонарних платформах	Підтримка поступово зростає, але поки більш нове обладнання орієнтоване на H.265	Обмежена апаратна підтримка, проте набуває популярності у великих онлайн-платформах (YouTube тощо)
Ліцензійні витрати та права	Ліцензійований стандарт; використання може бути обмеженим ліцензійними умовами	Також ліцензійований; використання вимагає сплати роялті, що може впливати на масштабування	Відкритий стандарт – безкоштовний для використання, що є перевагою для масштабованих онлайн-сервісів
Сумісність	Висока сумісність із більшістю пристроїв, браузерів та медіаплеєрів	Сумісність з деякими пристроями обмежена; потребує сучасного програмного забезпечення	Сумісність покращується, проте її рівень може бути нижчим, особливо на старих пристроях
Основні сфери застосування	Широко використовується у стрімінгу, телебаченні, відеоконференціях та мобільних пристроях	Ідеально підходить для відео високої роздільності (HD, 4K), де важлива компресія і економія даних	Використовується переважно в інтернет-відео (наприклад, на YouTube) та для спеціалізованих потокових сервісах

Порівняльна характеристика стратегій стискування

Параметр	Стиснення без втрат	Стиснення з втратами
Відновлення даних	Повне (оригінальні дані відновлюються)	Часткове (відновлення із втратами, відмінне від оригіналу)
Коефіцієнт стискання	Нижчий, зазвичай 2–3:1	Вищий (може сягати десятків або більше:1)
Методи кодування	Статистичні методи (Хаффман, арифметичне, LZ алгоритми)	Перетворення (ДКП), квантоване округлення, прогнозування
Застосування	Архівування, професійне редагування, медичні зображення	Онлайн-стрімінг, мобільні відеосервіси, споживчі системи
Обчислювальна складність	Зазвичай нижча	Вища (через додаткові обрахунки перетворення та квантування)

Схема стискання без втрат

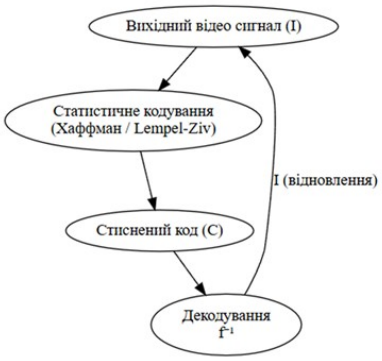


Схема стискання з втратами

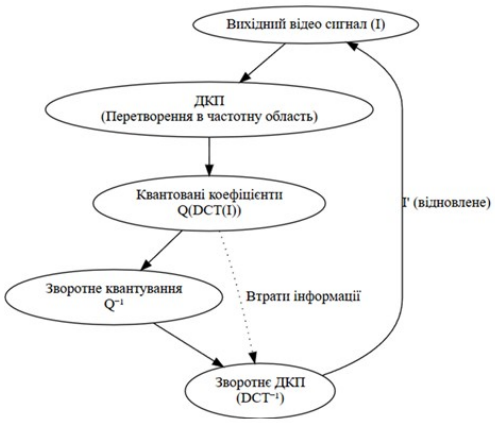


Схема просторового різницевого стиснення

Схема часово-просторового різницевого стиснення

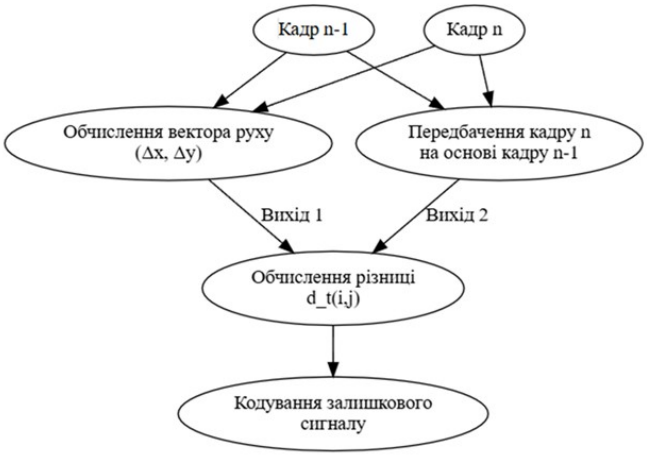


Схема просторово–часового оброблення під час стиснення відеоданих

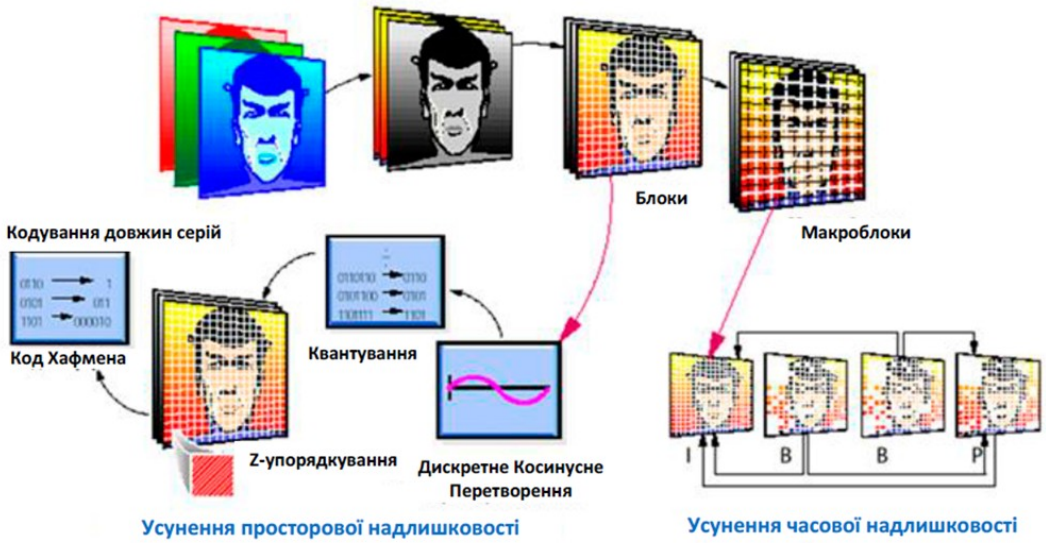


Схема розрахунку двовимірного ДКП

Основні характеристики одновимірного та двовимірного ДКП

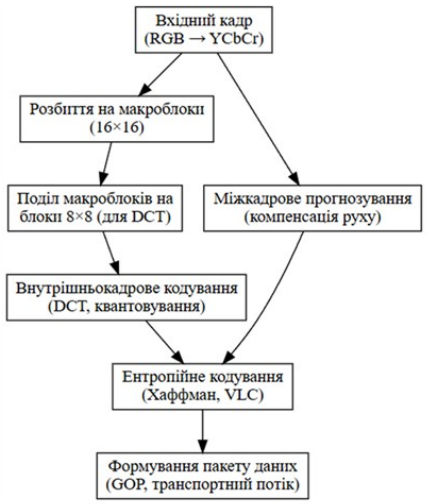
Параметр	Одновимірне ДКП	Двовимірне ДКП
Вхідний сигнал	Послідовність x(n)	Матриця I(i,j) (зображення або блок зображення)
Формула	$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right]$	$X(u,v) = \alpha(u)\alpha(v)i = 0^{N-1}$ $\sum_{j}^{M-1} I(i,j) \cos\left[\frac{\pi(2i+1)u}{2N}\right]$ $\cos\left[\frac{\pi(2j+1)v}{2M}\right]$
Нормалізація	$\alpha(0) = \sqrt{\frac{1}{N}}; \alpha(k) = \sqrt{\frac{2}{N}}; (k > 0)$	Аналогічні коефіцієнти по кожній осі
Застосування	Стиснення одновимірних сигналів	Стиснення зображень та відео, обробка блоків (наприклад, 8×8)

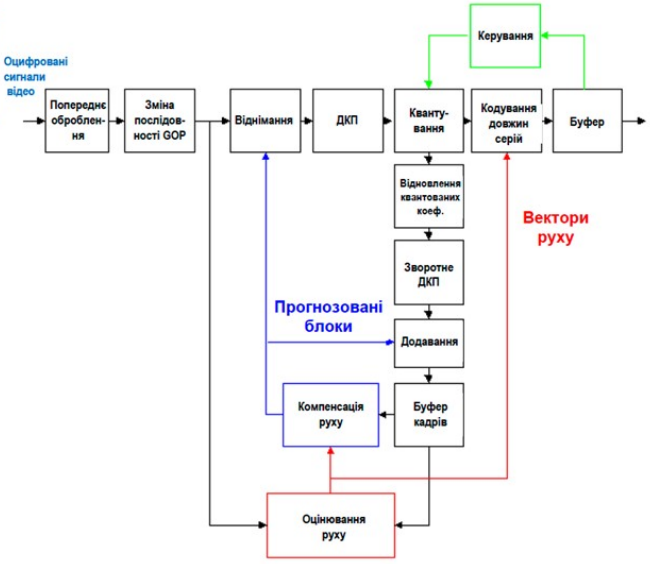
Схематичне представлення блокового розбиття

Основні параметри алгоритму MPEG-2

Параметр	Значення/Опис
Розмір макроблоку	Зазвичай 16×16 пікселів
Розмір блоку для ДКП	8×8 пікселів
Система кольорового перетворення	RGB → YCbCr (субдискретизація, зазвичай 4:2:0 або 4:2:2)
Методи кодування	Внутрішньокадрове (I-кадри) та міжкадрове (P-/B-кадри)
Алгоритм компенсації руху	Визначення вектора руху ($\Delta x, \Delta y$) та обчислення різниці

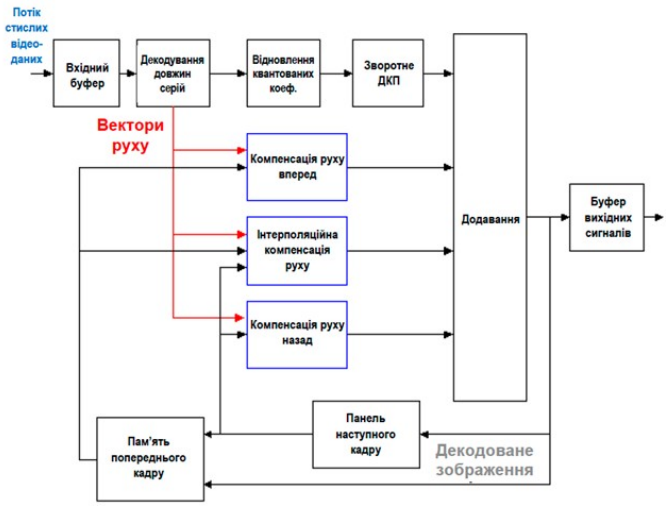
Схема основних етапів кодування за стандартом MPEG-2





Блок–схема типового кодера стиснення відеоданих

Типова блок–схема декодування стислих відеоданих



Спрощена блок-схема алгоритму стискування H.264/AVC

Основні характеристики алгоритму H.264/AVC

Параметр	Опис
Розбиття кадру	Макроблоки 16×16, що можуть бути розбиті на субблоки (16×16, 16×8, 8×16, 8×8, 4×4)
Intra-prediction	9 режимів для блоків 4×4; прогнозування на базі сусідніх пікселів
Inter-prediction	Різні методики прогнозування з компенсацією руху; підтримка B-кадрів
Ціле перетворення	Ціле перетворення для блоків 4×4
Квантування	Адаптивне квантування із використанням змінних квантуючих матриць
Ентропійне кодування	CAVLC для базових профілів, CABAC для підвищення ефективності
In-loop deblocking фільтр	Забезпечує згладження меж блоків і зменшення артефактів

Схема процесу кодування відеопотоку

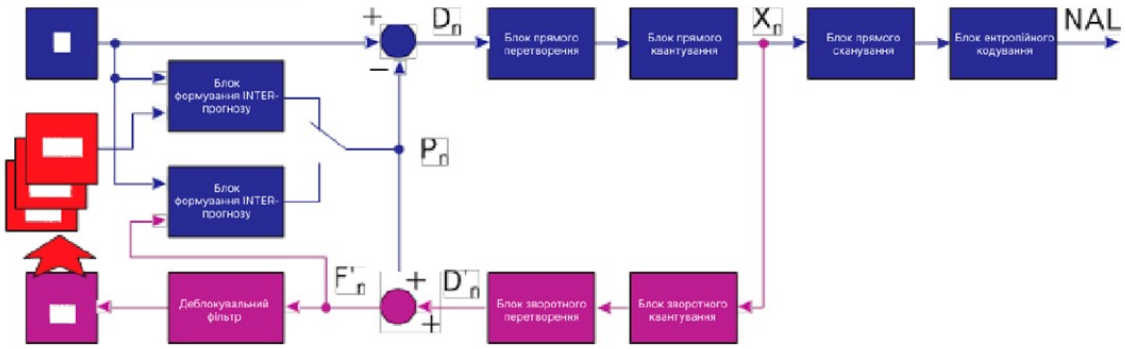
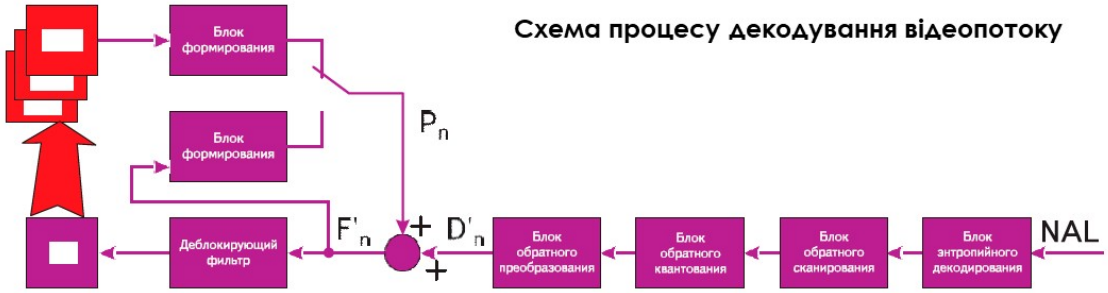
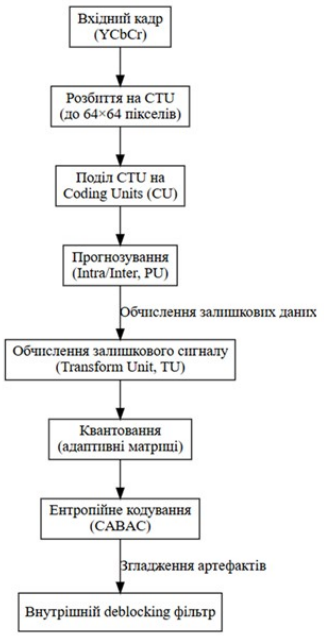


Схема процесу декодування відеопотоку



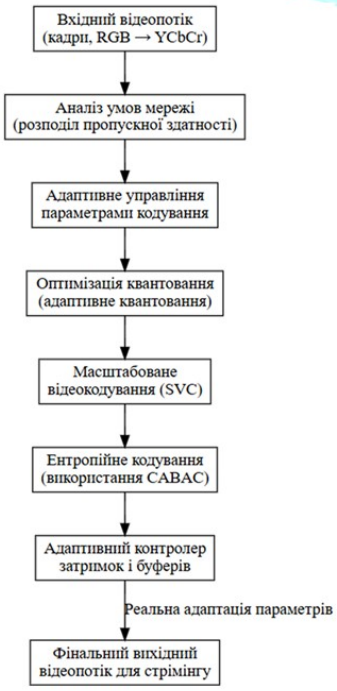
Спрощена блок-схема алгоритму стискування H.265/HEVC



Основні характеристики стандарту H.265/HEVC

Параметр	Опис
Розмір CTU	До 64×64 пікселів
Розбиття CTU	Розбиття на Coding Units (CU); CU можуть бути розміром 64×64, 32×32, ..., 8×8
Субблоки (PU і TU)	PU для прогнозування; TU для трансформування (4×4, 8×8, 16×16, 32×32)
Прогнозування	Різноманітні режими intra-prediction (до 33 режимів) та міжкадрове (inter) prediction із точним визначенням векторів руху
Квантування	Адаптивне квантування з можливістю регулювання параметрів квантовуючої матриці
Ентропійне кодування	CABAC – контекстно-адаптивне бінарне арифметичне кодування
Deblocking фільтр	Вдосконалений in-loop deblocking фільтр для зменшення артефактів

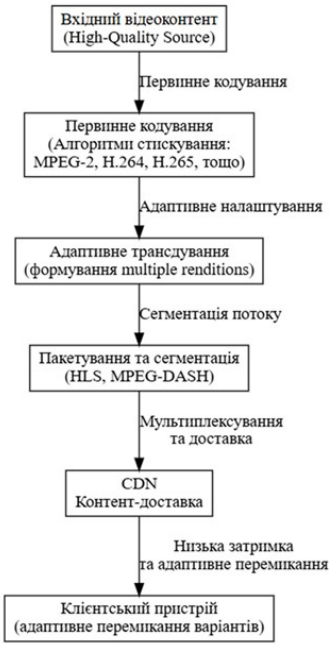
Схема адаптивного алгоритму стискування для стрімінгу



Основні характеристики алгоритмів стискування для стрімінгу

Параметр	Опис
Адаптивність	Динамічна зміна параметрів кодування (наприклад, квантування) залежно від мережевих умов
Масштабоване кодування (SVC)	Формування декількох рівнів якості в одному потокові, які можуть адаптуватися до доступної смуги
Низька затримка	Алгоритми розраховані на реальну роботу в режимі онлайн із забезпеченням мінімальних затримок
Інтеграція з машинним навчанням	Використання AI для прогнозування оптимальних параметрів кодування та оцінки якості у реальному часі
Контроль буферів і синхронізація	Адаптивне управління буфером для стабільної роботи стрімінгової системи

Інтеграція алгоритмів стискування у стрімінгову систему



Основні переваги інтеграції сучасних алгоритмів стискування

Параметр	Опис
Ефективність стискання	Забезпечує високий ступінь стиснення за рахунок сучасних алгоритмів (H.264, H.265, SVC)
Адаптивність	Динамічна адаптація параметрів кодування згідно з умовами мережі та ресурсами користувача
Низька затримка	Реальний час обробки та мінімальна затримка у потоковій передачі, що є критично важливими для стрімінгу
Мультиформатність	Можливість створення різних варіантів якості в одному відеопотоці
Оптимізація доставки	Інтеграція з CDN забезпечує високу доступність відеоконтенту та зменшення затримок

Порівняльна таблиця показників якості результатів стискування відео-даних

Метод	Основні характеристики	Діапазон значень
PSNR/APSNR	Оцінка на основі MSE; висока PSNR показує менший рівень спотворень	0 – ∞ (де 1 – ідеал)
MSE/MSAD	Простий числовий показник середньої помилки; чутливий до змін	0 – ∞
SSIM/MS-SSIM	Оцінка структурної схожості, контрасту та яскравості; ближче відображає суб'єктивне сприйняття	0 – 1
VQM	Комплексний показник, що враховує просторові та часові спотворення	Нижчі значення – краща якість
NQI	Універсальний індекс, що базується на середніх, стандартних відхиленнях та кореляції	0 – 1
Delta	Абсолютна різниця між відповідними пікселями	0 – ∞

Схема алгоритму обчислення індексу SSIM

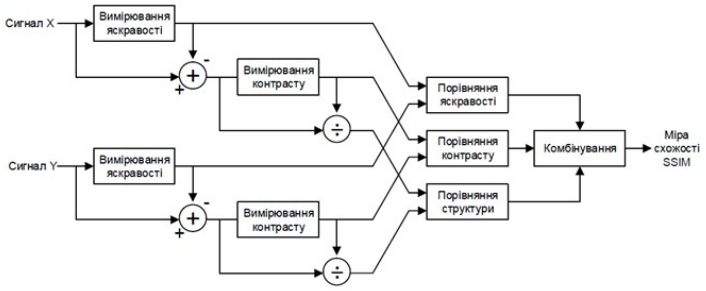


Схема алгоритму обчислення міри VQM

Схема алгоритму обчислення індексу MS–SSIM

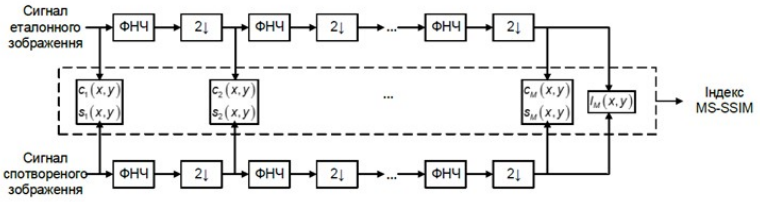
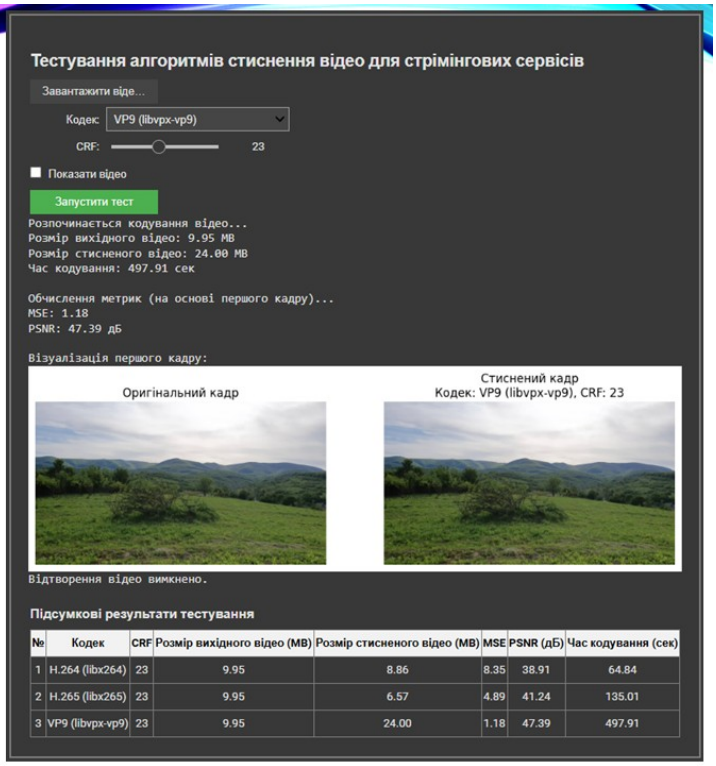


Схема архітектури застосунку для аналізу ефективності алгоритмів відео-даних

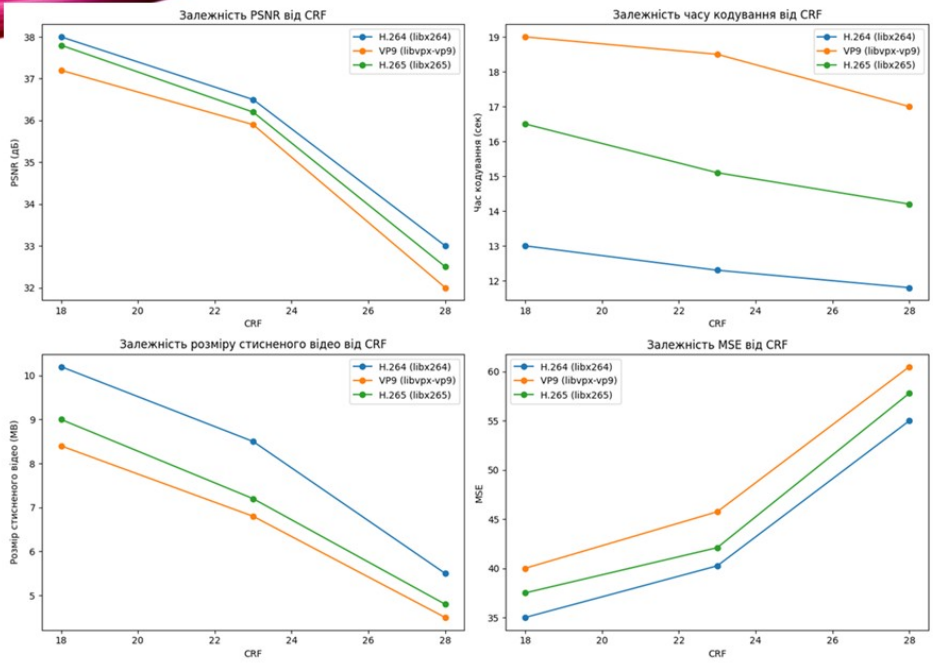
Інтерфейс застосунку для аналізу ефективності алгоритмів стискування відео-даних



№	Кодек	CRF	Розмір вихідного відео (MB)	Розмір стисненого відео (MB)	MSE	PSNR (дБ)	Час кодування (сек)
1	H.264 (libx264)	23	50.00	8.50	40.25	36.50	12.30
2	H.265 (libx265)	23	50.00	7.20	42.10	36.20	15.10
3	VP9 (libvpx-vp9)	23	50.00	6.80	45.75	35.90	18.50
4	H.264 (libx264)	28	50.00	5.50	55.00	33.00	11.80
5	H.265 (libx265)	28	50.00	4.80	57.80	32.50	14.20
6	VP9 (libvpx-vp9)	28	50.00	4.50	60.50	32.00	17.00
7	H.264 (libx264)	18	50.00	10.20	35.00	38.00	13.00
8	H.265 (libx265)	18	50.00	9.00	37.50	37.80	16.50
9	VP9 (libvpx-vp9)	18	50.00	8.40	40.00	37.20	19.00

Результати тестування ефективності алгоритмів стискування відео-даних

Порівняння параметрів ефективності алгоритмів при стискуванні відео-даних



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Прохорова Михайла Михайловича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Кривченко Юрій Вікторович*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів*

Обсяг розрахунково-пояснювальної записки *77* сторінок

Обсяг графічної (презентаційної) частини *17* аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті випускної роботи та технічному завданню. Тематика кваліфікаційної роботи є актуальною для своєї галузі та присвячена аналізу ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел. У основному розділі виконано огляд основних понять та класифікація алгоритмів стискування відео-даних; аналітичний огляд основних відеостандартів; інтеграція алгоритмів стискування у стрімінгові сервіси; об'єктивна оцінка якості відеозображень; розробка застосунку для аналізу ефективності алгоритмів; Експериментальний аналіз ефективності алгоритмів. Розглянуто питання охорони праці та техніки безпеки

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату ідей у роботі не виявлено

г) перелік позитивних якостей кваліфікаційної роботи

Чітко розкрито принципи ДКП, квантування, компенсації руху та сучасні стандарти MPEG-2, H.264, H.265, VP9 і SVC. Обрано різні CRF, вимірюється обсяг файлу, MSE, PSNR, час кодування; результати акумулюються в інтерактивній таблиці й графіках.

д) основні недоліки кваліфікаційної роботи

Відсутні повтори й статистичний аналіз (середнє ± стандартне відхилення, довірчі інтервали) для оцінки варіабельності результатів.

Час кодування залежить від однопоточних викликів ffmpeg; не досліджено паралельність чи GPU-прискорення.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

« 23 » 06 2025 р.

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про кваліфікаційну роботу бакалавра

Прохорова Михайла Михайловича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів*

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) Обсяг і якість виконання роботи (графічного матеріалу і розрахунково-пояснювальної записки) *Випускна робота виконана відповідно технічному завданню. Пояснювальна записка до випускної роботи містить 77 сторінок. У пояснювальній записці розглянуто проблему забезпечення ефективності та вибору доречного алгоритму стискування відео-даних для стрімінгових сервісів. Графічна частина складається з 17 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра, роботу виконано у повному обсязі.*

б) Самостійність роботи *Протягом виконання випускної бакалаврської роботи Прохоров Михайло поступово та послідовно виконував всі етапи, проявив ініціативу у створенні загальної концепції та реалізації випускної роботи. Всі роботи Прохоров Михайло виконував самостійно, з оглядом на рекомендації керівника.*

в) Теоретична підготовка здобувача освіти

Прохоров Михайло під час роботи над випускною бакалаврською роботою вивчив достатню кількість літературних джерел за даною тематикою.

Вважаю, що теоретична підготовка здобувача освіти добра і він готовий до захисту роботи.

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

Під час виконання роботи Прохоров Михайло мав змогу самостійно приймати окремі рішення з виконання програмної частини роботи та показав вміння організовано працювати над поставленою задачею, складати та оформлювати презентацію проекту, користуючись сучасними комп'ютерними програмними засобами, такими як Python, HTML, Google Colab, ipywidgets

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім`я, по батькові *Кривченко Юрій Вікторович*

Місце роботи і посада керівника роботи *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії, голова циклової комісії комп'ютерних технологій та програмної інженерії*

Підпис

«20» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Прохоров М.М.,
здобувач освіти гр. 2БКС-29, та

Кривченко Ю.В.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів» (автор роботи – Прохоров М.М., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Прохоров М.М. /

Керівник _____ / Кривченко Ю.В./

«16» червня 2025 р.

ДОВІДКА

кафедри комп'ютерної інженерії
про допуск до захисту кваліфікаційної роботи
здобувача (здобувачки) освіти II курсу
відділення комп'ютерних систем групи 2БКС-29

Прохорова Михайла Михайловича

на тему *Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до кваліфікаційної роботи виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 03.06.2025 р. значення коефіцієнту подібності в роботі становить 3,76%, коефіцієнт цитування – 3,14%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) кваліфікаційної роботи

здобувача (здобувачки) освіти *Прохорова М.М.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до кваліфікаційної роботи виконана у повному обсязі. Випускна кваліфікаційна робота відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Зав. кафедри КІ (підпис) *Іванова Л.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/3/2025
Дата редагування 6/3/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів

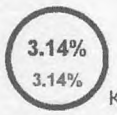
Автор
Прохоров Михайло Михайлович
Науковий керівник / Експерт
Кривченко Юрій Вікторович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

15017
Кількість слів

122071
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	62
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	454
Парафрази (SmartMarks)	a	39

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content	110 0.73 %
2	https://www.mathworks.com/matlabcentral/answers/106474-image-compression-using-dct-zigzag-scan-and-runlength-coding	66 0.44 %
3	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	44 0.29 %
4	https://studfile.net/preview/7305433/	36 0.24 %

5	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	32 0.21 %
6	https://henry-yip.github.io/ImageConvolution/	26 0.17 %
7	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	23 0.15 %
8	https://card-file.ontu.edu.ua/bitstreams/7f031b5d-95bb-43c2-8b3f-0fa2499416c4/download	16 0.11 %
9	http://uadoc.zavantag.com/text/18266/index-1.html	15 0.10 %
10	YFCNU/2015/iftc/iftc_2015_089.pdf [redacted] 10/29/2019 Yuriy Fedkovych Chernivtsi National University(CNU) course papers (Deanery)	15 0.10 %

з домашньої бази даних (0.04 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.04 %

з програми обміну базами даних (0.41 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	YFCNU/2015/iftc/iftc_2015_089.pdf [redacted] 10/29/2019 Yuriy Fedkovych Chernivtsi National University(CNU) course papers (Deanery)	20 (2) 0.13 %
2	Таразанов_М_О_ВТ-23-М [redacted] 12/5/2024 Pryazovskyi State Technical University (PSTU)	15 (2) 0.10 %
3	Програмні засоби збільшення роздільної здатності теплових зображень на основі зображення більшої роздільної здатності у видимому спектрі [redacted] 6/9/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	11 (1) 0.07 %
4	Кваснюк_Шпарик.docx [redacted] 12/18/2020 Vasyl Stefanyk Precarpathian National University (VSPNU) (Факультет природничих наук)	10 (1) 0.07 %
5	bitstream_2d15661e-0bfe-459a-98a0-9ba3f330c0d9 [redacted] 12/8/2024 National Technical University "Kharkiv Polytechnic Institute" students papers (National Technical University "Kharkiv Polytechnic Institute" students papers)	5 (1) 0.03 %

з Інтернету (3.32 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content**	**110 (1) 0.73 %**
2	**https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download**	**96 (5) 0.64 %**
3	**https://www.mathworks.com/matlabcentral/answers/106474-image-compression-using-dct-zigzag-scan-and-runlength-coding**	**66 (1) 0.44 %**
4	**https://studfile.net/preview/7305433/**	**48 (2) 0.32 %**

5	https://henry-yip.github.io/ImageConvolution/	**32 (2) 0.21 %**
6	https://card-file.ontu.edu.ua/bitstreams/d42aac6d-ab01-4a74-b9cb-ced2a9eff719/download	31 (4) 0.21 %
7	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	23 (1) 0.15 %
8	https://card-file.ontu.edu.ua/bitstreams/7f031b5d-95bb-43c2-8b3f-0fa2499416c4/download	16 (1) 0.11 %
9	https://pol.dcz.gov.ua/novyna/atestaciya-robochyh-misc-0	16 (2) 0.11 %
10	http://uadoc.zavantag.com/text/18266/index-1.html	15 (1) 0.10 %
11	http://repository.lnup.edu.ua:8080/jspui/bitstream/123456789/2401/1/%D0%9A%D0%A0_%D0%94%D1%80%D0%B0%D0%B3%D1%83%D0%BB%D0%B0.pdf	10 (1) 0.07 %
12	https://french.chnu.edu.ua/pro-kafedru/kolektyv-kafedry/dranenko-halyna-florivna/	8 (1) 0.05 %
13	https://lektsii.org/3-116117.html	8 (1) 0.05 %
14	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	7 (1) 0.05 %
15	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	7 (1) 0.05 %
16	https://test.ksau.kherson.ua/files/_sylabus/194%20%D0%A1%D0%B8%D0%BB%D0%B0%D0%B1%D1%83%D1%81%D0%B8/%D0%9E%D0%9A38-%D0%9C%D0%A0%20%D0%B4%D0%BB%D1%8F%20%D0%BF%D1%96%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B8%20%D0%BA%D0%B2%D0%B0%D0%BB%D1%96%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%BE%D1%97%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B8.pdf	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія» Група:** 2БКС- 29

КВАЛІФІКАЦІЙНА
РОБОТА

здобувача **освіти денної форми навчання** БКС. 29.19.000. КРБ

ПРОХОРОВА
МИХАЙЛА МИХАЙЛОВИЧА

м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія»**
Група: 2 БКС-29

ПОЯСНЮВАЛЬНА **ЗАПИСКА До кваліфікаційної роботи бакалавра на тему:**_____
_____
_____ **Проектний матеріал складається з пояснювальної записки на** _____ **сторінках та графічного (презентаційного) матеріалу на** _____ **аркушах (слайдах)** Виконавець _____ (Прохоров М.М.)
Керівник проекту _____ (Кривченко Ю. В.) **Консультанти: з розділу охорони праці та техніки безпеки** _____ (Чорновол Н.І.) з нормоконтролю _____ (Петрашова **В.І.) старший** консультант _____ (Кривченко Ю. В.) **До захисту** допущений