ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ДІБРІВНИЙ ОЛЕСЬ АНДРІЙОВИЧ

УДК 004.932.2

ДИСЕРТАЦІЯ

МЕТОДИКА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПЕРЕДАЧІ ВІДЕОПОТОКУ ПРИ СТИСНЕННІ МЕТОДОМ КОМПЕНСАЦІЇ РУХУ

Спеціальність 123 «Комп'ютерна інженерія»

Подається на здобуття наукового ступеня
доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ О.А.Дібрівний

Науковий керівник:
Бондарчук Андрій Петрович
доктор технічних наук, професор

-Київ 2021-

АНОТАЦІЯ

Дібрівний О.А. Методика підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – «Комп'ютерна інженерія». – Державний університет телекомунікацій. – Київ, 2021.

Дисертаційна робота присвячена дослідженню методики компенсації руху (motion compensation), яка використовується в більшості сучасних технологіях стиснення відеопотоку і базується на передачі кадрів у вигляді скомпенсованої міжкадрової різниці (у вигляді векторів руху окремих блоків на які розбивається зображення), з ціллю зменшення часу затраченого на стиснення відеопотоку.

Проаналізовано поточний стан та перспективи розвитку технологій стиснення відеофайлів, визначено основні алгоритми та методики, що входять до їхнього складу. Виявлено основні проблемні області таких технологій.

Розробка результуючої методики відбувалась за рахунок усунення проблем сучасних методик в двох ключових напрямках:

1. Оцінка схожості блоків зображення
2. Методи блочного пошуку схожих блоків.

На основі аналізу наукової літератури було визначено основні метрики, які використовуються для оцінки ступеня схожості зображень. Було виявлено, що в сучасних системах відеостиснення в якості основної метрики використовується SAD (Sum of absolute differences – сума абсолютних різниць), обчислення якої, при всій її простоті та невеликій обчислювальній

складності може займати від 40 до 80% загального часу кодування відеопотоку.

Для усунення даного недоліку було запропоновано використання шаблонів порівняння за характерними точками, які дозволяють зменшити обрахунку метрики за рахунок зменшення кількості точок порівняння. Такий підхід зумовлений унікальністю розподілу піксельних значень на різних частинах зображення. Аналіз алгоритму оцінки схожості зображень на основі характерних точок, показав, що зменшення кількості контрольних точок призводить до різкого зменшення показника PSNR (peek signal to noise ratio – пікового відношення сигналу до шуму). Це відбулося за рахунок втрати інформації в областях з високою ентропією зображення та появи стійкої невизначеності в областях з низькою ентропією зображення.

Запропоновано для усунення втрати інформації в областях високої ентропії (на границях об’єктів та областях різкого переходу кольору) використання алгоритмів виділення країв зображення і проведення уточнення блоків кандидатів на карті країв зображення. Зручність такого підходу зумовлена тим, що процес пошуку векторів руху відбувається для відеопотоку в форматі YUV, а саме для яскравісної компоненти зображення (Y) та частковим ігноруванням кольорових компонент (людське око більш чутливе до зміни яскравості зображення чим до зміни кольору). При цьому яскравісна компонента Y є зображенням в сірих тонах. Такий підхід дозволив суттєво скоротити втрати PSNR при відносно незначних зменшеннях швидкості обрахунку векторів руху.

Вдосконалено методику існуючих методів блочного пошуку, для усунення основних недоліків використання шаблонного порівняння. Було проведено аналіз сучасних методів блочного пошуку таких як: FS (full search), TSS (three step search), NTSS (New Three Step Search), DS (diamond search), MVFAST (Motion Vector Field Adaptive Search Technique), PMVFAST (Predictive Motion Vector Field Adaptive Search Technique) та виявлено їх

основні недоліки. Формування методу блочного пошуку, для результуючої методики було проведено за рахунок вдосконалення чотирьох основних частин алгоритмів блочного пошуку:

1. Попередня обробка відеокадру, шляхом обрахунку нескомпенсованої різниці з попереднім кадром, та занулення векторів руху блоків, максимальне значення якої не перевищує заданий поріг (2% глибини кольору), для усунення неоднозначності в областях низької ентропії.
2. Формування розширеного набору більш точних предикатів, для скорочення часу уточнення блоків кандидатів та виключення впливу випадкових співпадінь значень пікселів на формування вектору руху.
3. Розширення умов формування порогових значень для умов ранньої зупинки пошуку.
4. Формування набору адаптивних шаблонів для уточнення блоків з яких будуть формуватися вектори руху.

Основним підходом для формування цих чотирьох частин методу було використання суміжних блоків не тільки поточного та попереднього кадрів, а також наступного кадру, за рахунок попередньої обробки частини блоків всіх кадрів відео послідовності (пар кадрів у випадку обробки відеопотоку в реальному часі). Такий підхід обумовлений постійністю руху об’єктів на коротких проміжках часу.

В результаті комбінації шаблонного порівняння для оцінки схожості блоків, алгоритмів виділення країв та вдосконаленого методу блочного пошуку було сформовано результуючу методику знаходження векторів руху - TFEMVFAS (Three by Frame Edge Map Motion Vector Field Adaptive Search – адаптивний алгоритм визначення векторів руху один через три за картою границь), в двох модифікаціях:

1. TFEMVFASS (- TFEMVFA speed search – швидкий пошук) – основною ціллю якої є максимальний виграш в швидкості, за рахунок більш ранньої зупинки уточнень блоків кандидатів, наслідком чого є більш суттєві втрати значення PSNR.
2. TFEMVFAAS (- TFEMVFA accuracy search – точний пошук) – який за рахунок проведення обов'язкових уточнень по блокам на які вказує набір предикатів, що дозволяє помітно скоротити втрати PSNR, за рахунок втрат в швидкості роботи методики.

Протестовано модифікації методики згідно комплексних рекомендацій тестування міжнародної комісії MPEG по оцінці складних відеоінструментів, на прикладі 13 відеопослідовностей, з різними бітрейтами. Досліджено вплив збільшення розмірів блоків на які розбивається зображення на значення PSNR та максимального прискорення. Проведено порівняння результатів роботи запропонованої методики з сучасними методами формування векторів руху: FS, TSS, NTSS, DS, MVFAST, PMVFAST, в результаті визначено, що запропонована методика дозволяє отримати прискорення на рівні 15-17% (рівень прискорення залежить від бітрейту оброблюваної відеопослідовності: вищий бітрейт – вище прискорення) від найближчого аналога при втратах на рівні 0.3-1.2%значень PSNR (втрати PSNR залежать від розміру блоків для яких проводився пошуку, максимальна втрата для блоків 16x16 пікселів, мінімальна для блоків 64x64пікселів) від значень еталонного методу пошуку векторів руху для модифікації A і прискорення на рівні 27-31% при втраті 1.1%-2.1% в залежності розміру блоку та бітрейту відеопослідовності, для модифікації S.

Розроблена методика знаходження векторів руху є цінною для розвитку технологій обробки, передачі та зберігання відеофайлів.

Ключові слова: *стиснення відеопотоку, компенсація руху, оцінка схожості зображень, сума абсолютних різниць, виділення границь об'єктів, LoG, Canny, методи блочного пошуку.*

ABSTRACT

Dibrivny OA The method of increasing the efficiency of video stream transmission during compression by the method of motion compensation. - Qualifying scientific work on the rights of the manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 - "Computer Engineering". - State University of Telecommunications. - Kyiv, 2021.

The dissertation is devoted to the research of motion compensation technique, which is used in most modern technologies of video stream compression and based on frame transmission in the form of compensated interframe difference (in the form of motion vectors of separate blocks into which the image is divided), in order to reduce compression time of video stream.

The current state and prospects of development of video file compression technologies are analyzed, the main algorithms and techniques that are part of them are determined. The main problem areas of such technologies are identified.

The development of the resulting methodology took place by eliminating the problems of modern techniques in two key areas:

1. Similarity estimation of image blocks

2. Methods of block search of similar blocks.

Based on the analysis of the scientific literature, the main metrics used to assess the similarity degree of images were identified. It was found that in modern video compression systems as the main metric used SAD (Sum of absolute differences), the calculation of which, for all its simplicity and low computational complexity can take from 40 to 80% of the total coding time of the video stream.

To eliminate this shortcoming, it was proposed to use comparison templates based on characteristic points, which allow to reduce the calculation of the metric by reducing the number of comparison points. This approach is due to the unique

distribution of pixel values in different parts of the image. When evaluating the performance of the algorithm for estimating the similarity of images based on characteristic points, it was determined that reducing the number of control points leads to a sharp decrease in PSNR (peek signal to noise ratio), due to loss of information in areas with high image entropy and the emergence of persistent uncertainty in areas with low image entropy.

To eliminate the loss of information in areas of high entropy (at the boundaries of objects and areas of sharp color transition), it was proposed to use algorithms for selecting the edges of the image and refining the candidate blocks on the map of the edges of the image. The convenience of this approach is due to the fact that the process of finding motion vectors is for the video stream in YUV format, namely for the luminance component of the image (Y) and partial ignoring of color components (human eye is more sensitive to changes in image brightness than to color change). Y is essentially a grayscale image (image edge selection algorithms work with grayscale images). This approach has significantly reduced the loss of PSNR with relatively small reductions in the speed of calculation of motion vectors.

The next step in the formation of the resulting technique is to improve existing methods of block search, to eliminate the main disadvantages of using a template comparison. An analysis of modern block search methods such as: FS (full search), TSS (three step search), NTSS (New Three Step Search), DS (diamond search), MVFAST (Motion Vector Field Adaptive Search Technique), PMVFAST (Predictive Motion Vector Field Adaptive Search Technique), and identified their main shortcomings was performed. The formation of the block search method for the resulting technique was carried out by improving the four main parts of block search algorithms:

1. Pre-processing of the video frame, by calculating the uncompensated difference with the previous frame, and zeroing the motion vectors of the

blocks, the maximum value of which does not exceed a specified threshold (2% color depth), to eliminate ambiguity in low entropy areas.

2. Formation of an extended set of more accurate predicates, to reduce the time of refinement of candidate blocks and to exclude the influence of random coincidences of pixel values on the formation of the motion vector.

3. Extension of the conditions for the formation of threshold values for the conditions of early search.

4. Forming a set of adaptive templates to specify the blocks from which the motion vectors will be formed.

The main approach for the formation of these four parts of the method was to use adjacent blocks not only of the current and previous frames, but also the next frame, by pre-processing part of the blocks of all frames of the video sequence (pairs of frames in real time video stream processing). This approach is due to the constant movement of objects over short periods of time.

As a result of a combination of template comparison for estimating block similarity, edge selection algorithms and an improved block search method, the resulting method of finding motion vectors - TFEMVFAS (Three by Frame Edge Map Motion Vector Field Adaptive Search) was formed), in two versions:

1. TFEMVFASS (TFEMVFA speed search) - the main purpose of which is the maximum gain in speed, due to the earlier stop of refinements of blocks candidates, resulting in more significant loss of PSNR value.

2. TFEMVFAAS (TFEMVFA accuracy search) - which is due to the mandatory refinement of the blocks indicated by the set of predicates, which can significantly reduce the loss of PSNR, due to losses in the speed of the method.

Both modifications of the method were tested according to the comprehensive testing recommendations of the international commission MPEG for the evaluation of complex video tools, on the example of 13 video sequences, with different bitrates. The effect of increasing the size of the blocks on which the

image is divided on the PSNR value and the maximum acceleration was also investigated. Comparison of the results of the proposed method was carried out with modern methods of motion vectors: FS, TSS, NTSS, DS, MVFAST, PMVFAST, which determined that the proposed method allows to obtain acceleration at 15-17% (the level of acceleration depends on the bitrate of the processed video sequences: higher bitrate - higher acceleration) from the nearest analogue at losses at the level of 0.3-1.2% of PSNR values (PSNR losses depend on the size of the blocks for which the search was performed, maximum loss for 16x16pixel blocks, minimum for 64x64pixel blocks) from the values of the reference vector search method motion for modification A and acceleration at the level of 27-31% with a loss of 1.1% -2.1% depending on the block size and bitrate of the video sequence, for modification S.

The developed method of finding motion vectors is valuable for the development of technologies for processing, transmission and storage of video files.

Keywords: *video stream compression, motion compensation, image similarity estimation, sum of absolute differences, object boundaries selection, LoG, Canny, block search methods.*

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці у яких опубліковані основні наукові результати дисертації:

1. Dibrivniy O.A., Grebenyuk V.V. Usage of edge selection methods to refine candidate blocks in the motion compensation process, based on the SAD algorithm using characteristic points. Sciences of Europe, VOL 1, No 63 (2021) –19-24 p.
2. Dibrivniy O.A., Grebenyuk V.V. Sum of absolute differences algorithm improvement by characteristic points usage. Modern engineering and innovative technologies, Vol 15(2021).
3. Dibrivniy O.A., Usage of characteristic points to reduce time of performance of sad metric calculation during video flow compression by motion compensation method, Телекомунікаційні та інформаційні технології №1 2021.
4. Дібрівний О.А., Гребенюк В.В., Негоденко О.В., Бондарчук А.П. Порівняльний аналіз нереферентних методів оцінки якості відеоматеріалу. Журнал Зв’язок №6(2020).
5. Дібрівний О.А., Крючкова Л.П., Стрельников В.І., Акулінічева М.В., Бортник О.С. Алгоритми виділення контурів зображень об’єктів в інтелектуальних системах відеоспостереження. Журнал Зв’язок №5(2020).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. О.А. Дібрівний, Д.В. Сорокін, О.В. Сеньков, А.П. Бондарчук “Нові сфери застосування сучасних інформаційних мереж”. Науково практична конференція «Проблеми комп’ютерної інженерії». Збірник тез К.:ДУТ 73-34ст. – 2020.
2. О.А. Дібрівний, В.О. Кисельов. “Розробка десктопного додатку для перегляду відеофайлів на мові програмування C#”. XI науково-технічна

конференція студентів та молодих вчених, Збірник тез К.:ДУТ 39-40 – 2020.

3. О.А. Дібрівний, Д.О. Криворученко. “Розробка гри “Layers” в жанрі головоломки за допомогою ігрового двигуна Unity”. Всеукраїнська науково-технічна конференція “Сучасний стан та перспективи розвитку IoT”. Збірник тез. К.: ДУТ 29-30 – 2020.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 14

ВСТУП 15

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВІДЕОПОТОКУ 20

1.1 Поняття кодеків та процесу стиснення відеопотоку..... 20

1.2 Методика компенсації руху 39

1.3 Висновки до розділу 44

2 МЕТОДИКА ВИЗНАЧЕННЯ ПОДІБНОСТІ ЗОБРАЖЕНЬ, ДЛЯ ЗНАХОДЖЕННЯ БЛОКІВ КАНДИДАТІВ В ПРОЦЕСІ КОМПЕНСАЦІЇ РУХУ. 45

2.1 Постановка задачі. 45

2.2 Метрики оцінки подібності зображень..... 46

2.3 Тестування результатів роботи запропонованих шаблонів згідно стандартів MPEG-4 64

Висновки до розділу..... 70

3 ВИКОРИСТАННЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ГРАНИЦЬ ОБ'ЄКТІВ ДЛЯ УСУНЕННЯ НЕОДНОЗНАЧНОСТЕЙ, ВИКЛИКАНИХ ВИКОРИСТАННЯМ ШАБЛОНІВ ПОРІВНЯНЬ..... 73

3.1 Алгоритми розпізнавання границь об'єктів. 73

3.2 Тестування запропонованих алгоритмів згідно стандартів MPEG-4…… 92

Висновки до розділу..... 97

4 МЕТОДИ БЛОЧНОГО ПОШУКУ 99

4.1 Існуючі алгоритми обходу області пошуку векторів кандидатів… 99

4.2 Модернізовані методи та техніки блочного пошуку 106

4.3 Розробка нового методу блочного пошуку векторів руху 115

4.4 Оцінка роботи алгоритму на прикладі тестової послідовності… 126
Висновки до розділу 131
5 ФОРМУВАННЯ МЕТОДИКИ КОМПЕНСАЦІЇ РУХУ НА ОСНОВІ АЛГОРИТМУ TSADL І МЕТОДУ TCMVFAS 134
Висновки до розділу 143
ВИСНОВКИ 145
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 148
ДОДАТОК А 156
ДОДАТОК Б 158

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

SAD – сума абсолютних різниць (Sum of absolute differences).

SSD – сума стандартних відхилень **(**Sum of standard deviation).

PSNR – пікове відношення сигналу до шуму (peek signal to noise ratio).

LoG – Лапласіан Гаусіана (Laplacian of Gaussian).

FS – повний перебір (full search).

TSS – трьох кроковий пошук (three step search).

NTSS – новий трьох кроковий пошук (New Three Step Search).

DS – ромбовий пошук **(**diamond search).

MVFAST – техніка адаптивного пошуку поля векторів руху **(**Motion Vector Field Adaptive Search Technique).

PMVFAST – техніка адаптивного пошуку поля векторів руху з передбаченням (Predictive Motion Vector Field Adaptive Search Technique).

TFMVAS – адаптивна техніка пошуку поля векторів руху три на кадр (Three by Frame Motion Vector Field Adaptive Search).

TFEMVFAS – адаптивна техніка пошуку поля векторів руху три на кадр по краєвій карті (Three by Frame Edge Map Motion Vector Field Adaptive Search).

TFEMVFASS – адаптивна техніка швидкого пошуку поля векторів руху три на кадр по краєвій карті (Three by Frame Edge Map Motion Vector Field Adaptive Speed Search).

TFEMVFAAS – адаптивна техніка точного пошуку поля векторів руху три на кадр по краєвій карті (Three by Frame Edge Map Motion Vector Field Adaptive Accuracy Search).

ВСТУП

Актуальність теми. В сучасному світі в день відбувається мільйони годин перегляду та обробки відеофайлів, жоден з яких не проходить з матеріалом в чистому вигляді, оскільки одна хвилина нестиснутого відеопотоку може займати кілька десятків гігабайт, що є неприпустимим для використання і передачі через мережу інтернет. Тому важливе місце в сучасних системах роботи з відеофайлами займає процес стиснення відеопотоку, який дозволяє зменшити обсяг важливої інформації в кілька десятків разів.

Під час аналізу сучасних технологій стиснення відеопотоку, було визначено що в основі більшості таких систем лежить методика компенсації руху, яка передає відеопотік у вигляді скомпенсованої міжкадрової різниці [1,2]. Проблематикою реалізації даної методики займалися науковці, серед яких: Zhang K., Chen Y.W., Zhang L., Chien W.J., Karczewicz M. [3,4,5,6]. У цих роботах розглядаються основні шляхи вирішення головних проблем стиснення відеопотоку, серед яких: надлишковий час стиснення і помітні втрати якості зображення.

При аналізі останніх публікацій, було визначено два проблемних напрямки вдосконалення існуючих методик компенсації руху, на основі яких було вирішено провести формування результуючої методики, яка була сформована на основі вдосконалення алгоритмів оцінки подібності зображень та методів блочного пошуку.

Дослідженню алгоритмів оцінки схожості зображень присвячені наукові роботи Hang H.M., Guevorkian D., Launiainen A., Tourapis A.M., Oscar C., Au L., Liou M.L., Wang M., Zhang X [7,8,9]. Так було визначено, що в якості основної метрики оцінки схожості зображень найчастіше використовується сума абсолютних різниць (SAD), обрахунок якої при всій

своїй простоті може займати віж 40 до 80% часу кодування відеопослідовності.

Методи блочного пошуку, дослідженням яких займалися науковці, зокрема: Cheung T.C-H., Po L.M., Wu X., Xu W., Yang Z., Ahmed Z., Hussain A. J., Al-Jumeily D., Ghanbari M., Zhu C., Lin X., Chau L-P., Parker P., Stark T [10,11,12]. Основним недоліком підходів, що описані в вищезгаданих роботах було, ігнорування часової постійності руху, а саме використання інформації з наступних кадрів відеопослідовності і, як наслідок, зменшення рівню PSNR.

У зв’язку з цим, постає актуальне наукове завдання, сутність якого полягає в формуванні методики компенсації руху, за рахунок вдосконалення в алгоритмів оцінки схожості зображень та методів блочного пошуку.

Зв’язок роботи з науковими програмами, планами, темами. Тематика дисертаційної роботи і результати отримані при її розробці, пов’язані із пріоритетними напрямками розвитку інформаційних та комунікаційних технологій до 2025р.

Дисертаційна робота виконана відповідно до планів наукової і науково-технічної діяльності Державного університету телекомунікацій і є частиною досліджень в рамках науково-дослідної роботи: ««Контроль та прогнозування перевантажень в комп'ютерних мережах» №0120U105655.

Мета і задачі дослідження. Мета дослідження – зменшення затримки передачі відеопотоку при стисненні методикою компенсації руху, за рахунок зменшення часу кодування.

У відповідності до поставленої мети, було визначено основні завдання дослідження:

- проаналізувати поточний стан та перспективи розвитку технологій стиснення відеопотоку;
- дослідити сучасні алгоритми оцінки схожості зображень;

- удосконалити алгоритм оцінки схожості зображень на основі метрики SAD, за рахунок використання порівняння по характерних точках;
- зменшити втрати значень PSNR стиснутих відеопослідовностей за рахунок використання алгоритмів виділення країв;
- вдосконалити метод блочного пошуку, шляхом використання інформації з наступного кадру відеопослідовності;
- сформувати результуючу методику пошуку векторів руху, яка дозволить отримати виграш у швидкості при кодуванні відеопотоку;
- виконати експериментальну оцінку роботи отриманої методики.

Об'єктом дослідження є процес кодування відеопотоку методом компенсації руху.

Предметом дослідження є методи і моделі обробки відеоданих.

Методи дослідження. В дисертаційній роботі нові наукові результати були отримані на основі математичного аналізу (метрики оцінки схожості зображень, алгоритми виділення країв, метрики оцінки якості зображень), й синтезу складних технічних систем (для дослідження якості стиснення відеопотоку). Також використовувались методи теорії наближень (використання характерних точок), торії людського сприйняття (сприйняття змін в окремих значеннях кадрів відеопотоку) та методи обробки відеофайлів (для дослідження процесу стиснення відеопотоку).

Наукова новизна одержаних результатів полягає у наступному:

1. Удосконалено алгоритм оцінки схожості зображень на основі метрики SAD за рахунок використання порівняння за характерними точками, що дозволяє скоротити кількість результуючих пікселів по яких проводиться розрахунок метрики. Використання такого підходу дозволяє суттєво скоротити час обрахунку вищезгаданої метрики, ціною втрати значення PSNR стисненого відеопотоку.

2. Вперше використано алгоритми виділення країв для проведення пошуку схожих блоків в областях з високою ентропією зображення (на границях об’єктів та в областях різкого переходу кольору), що надає змогу зменшити втрати PSNR спричинені використанням шаблонів порівняння на основі характерних точок.
3. Вдосконалено методи блочного пошуку векторів руху, за рахунок використання інформації з наступних кадрів відеопослідовності, шляхом попередньої обробки відеопослідовності, на чотирьох основних етапах методів блочного пошуку:
   3.1 Попередньої оцінки відеокадрів
   3.2 Формування предикатів векторів руху
   3.3 Формування умов ранньої зупинки
   3.4 Використання адаптивних шаблонів уточнення пошуку.
4. Вперше розроблено результуючу методику пошуку векторів руху у двох модифікаціях, яка за рахунок використання вдосконаленого алгоритму оцінки схожості зображень, алгоритмів виділення країв і вдосконаленого методу блочного пошуку, дозволяє отримати суттєвий виграш у швидкості стиснення відеопотоку при мінімальних втратах в значеннях PSNR.

Практичне значення одержаних результатів полягає в тому, що в дослідженні розроблено та доведено до практичної реалізації запропоновану методику пошуку векторів руху для стиснення відеопотоку. Розроблені методики відкривають можливості суттєвого зниження швидкості стиснення відеопотоку, при мінімальних втратах якості відеопотоку для онлайн та офлайн систем обробки відеофайлів.

Результати досліджень прийняті до впровадження в ТОВ «Vega» в ТОВ «НТЦ Енергозв'язок», в ТОВ «СУРВАЙВАЛ, ЕКСПЛОРЕР ЕНД КРАФТ ГЕЙМЗ» а також використовуються в навчальному процесі кафедри інженерії програмного забезпечення Державного університету

телекомунікацій при викладанні дисциплін «Розробка комп'ютерних ігор» та «Програмування мобільних пристроїв» для студентів спеціальності 121 «програмна інженерія» денної форми навчання.

Особистий внесок здобувача. Основні наукові та прикладні результати дисертаційної роботи, що опубліковано в [13], отримані здобувачем особисто. У роботах, що виконані у співавторстві [14] проведено аналіз впливу характерних точок на якість стисненого відеопотоку, [15] – проаналізовано вплив використання алгоритмів виділення країв на швидкість стиснення відеопотоку, [16] – порівняно результати використання різних шаблонів порівняння для оцінки схожості зображення, [17] – оцінено вплив алгоритмів виділення країв на результуючу якість відеопотоку.

Апробація результатів дисертації. Теоретичні та практичні результати були представлені та обговорені в ході конференцій:

- Науково-практична конференція «Проблеми комп'ютерної інженерії». – К.: ДУТ, – 2грудня 2020р.
- XI Науково-технічна конференція студентів та молодих вчених «Сучасні інфокомунікаційні технології». – К.: ДУТ, – 11 грудня 2020р.
- Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT». – К.: ДУТ, 3 квітня 2020.

Публікації. За результатами дисертаційного дослідження було опубліковано 5 наукових праць. Основні наукові положення викладено у 5 наукових статтях [13-17]. За матеріалами виступів на науково технічних конференціях опубліковано 3 тез доповідей [18,19,20].

Структура та обсяг дисертації. Робота складається зі вступу, 5 розділів, 2 додатків та списку використаних джерел з 70 найменувань, загальний обсяг роботи складає 161 сторінка, з них 125 сторінок основного тексту.

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВІДЕОПОТОКУ

1.1 Поняття кодеків та процесу стиснення відеопотоку

1.1.1 Загальне поняття стиснення відеопотоку

Стиснення відео (англ. Video compression) - технологія цифрової компресії телевізійного сигналу, що дозволяє скоротити кількість даних, які використовуються для подання відео-потоку. Стиснення відео дозволяє ефективно зменшувати потік, необхідний для передачі відео по каналах передачі, зменшувати простір, необхідний для зберігання даних на носії. Недоліки: при використанні стиснення з втратами з'являються характерні, іноді чітко видні артефакти - наприклад, блочність (розбиття зображення на блоки 8x8, 16x16 пікселів), замилювання (втрата дрібних деталей зображення) і т.д. Існують і способи стиснення відео без втрат, але на сьогоднішній день, рівень стиснення який дозволяють отримати ці методи, недостатній для обробки та передачі відео в сучасних інтернет мережах.

Відео по суті своїй є тривимірним масивом кольорових пікселів [21]. Два вимірювання означають вертикальне та горизонтальне розширення кадру, а третій вимір - час. Кадр - масив всіх пікселів, видимих камерою в даний момент часу, або просто зображення. У відео можливі також так звані напівкадри. Стиснення було б неможливо, якби кожен кадр був унікальний і розташування пікселів було повністю випадковим, але це не так. Тому можна стискати, по-перше, саму картинку - наприклад, фотографія блакитного неба без сонця фактично зводиться до опису граничних точок і градієнта заливки. По-друге, можна стискати схожі сусідні кадри. В остаточному підсумку, алгоритми стиснення картинок і відео схожі, якщо розглядати відео як тривимірне зображення з часом, в якості третьої координати.

Процеси стиснення зображення називають кодуванням [22], тоді як розтиснення – декодуванням. Програми які кодують зображення називають кодеками.

1.1.2 Стандарт кодеків MPEG4

Стандарт MPEG4 базується на більш ранньому стандарті MPEG2, який, в свою чергу, є модифікацією стандарту MPEG1. Як наслідок, багато з методів, що використовуються в стандарті MPEG4, вже були реалізовані в більш ранніх стандартах.

Передбачається, що початковий відеопотік представлений в форматі YUV 4: 2: 0 і має горизонтальні і вертикальні розміри зображення, кратні 16 (найбільш часто використовувані розміри - 352x288 (так званий формат CIF)). При такому поданні, зображення зберігається у вигляді трьох колірних компонентів: одного компонента яскравості (Y) і двох компонентів колірної насиченості (U, V). Розміри компоненти яскравості збігаються з розмірами початкового зображення, тоді як розміри компонентів колірної насиченості в два рази менше за кожним з вимірів. Зазначене подання дуже зручне, так як дозволяє істотно полегшити процес обробки інформації за рахунок концентрації більшої частини оброблюваної інформації в одній з компонент (Y). Можливість такої концентрації обумовлюється особливостями зорового сприйняття людини - людське око більш чутливе до зміни яскравості і менш чутливе до зміни кольору [23].

При обробці відео-потоку кожне зображення (фрейм) може бути представлено у вигляді суперпозиції об'єктних відео-площин (video object plane - VOP). Максимальна кількість таких площин - 256. Кожна площина також є зображенням у форматі YUV 4: 2: 0. Вона містить певну частину вихідного (повного) зображення і, як правило, формально пов'язана з якоюсь візуально помітною його частиною. Площини кодуються незалежно. При декодуванні зображення «збирається» з окремих декодованих площин з

використанням інформації про форму об'єктів (в більшості випадків вона зберігається окремо), що містяться в тій чи іншій площині (інформація про форму утворює так звані альфа-площини (alpha planes)). В окремому випадку маємо всього одну площину, яка містить все вихідне зображення. Внаслідок відсутності ефективних методів поділу зображення на окремі об'єкти, саме цей підхід зустрічається найчастіше.

При обробці, відео-площини поділяються на чотири типи: I-площини, P-площини, B-площини і S-площини. I-площини кодуються без використання будь-якої інформації про інші площини. I-площину можна декодувати, не декодуючи жодні інші площини, що дозволяє забезпечити швидкий доступ до довільних частин закодованого відео потоку. Кожна I-площина розбивається на макроблоки розміру 16x16 (тут і далі вказується розмір для компоненти Y; відповідний розмір для компонент U і V - 8x8). Такі макроблоки називаються intra-макроблок. Intra-макроблок в свою чергу розбивається на 4 intra-блоки розміру 8x8, кожен з яких бере участь у подальшому блоковому кодуванні/декодуванні (все сказане стосується лише компоненти Y; для компонентів U і V в якості кодованого блоку виступає блок 8x8, відповідний макроблоку компоненти Y).

P-площини кодуються з використанням інформації про попередні I- і P площини. P-площина також розбивається на макроблоки 16x16 і блоки 8x8, однак, крім intra-макроблоків і intra-блоків, тут присутні inter-макроблоки і inter-блоки. Перед кодуванням над inter-макроблоками/блоками здійснюється процедура компенсації руху (motion compensation): в попередніх площинах знаходиться макроблок/блок (він може знаходитися на довільній позиції, не обов'язково кратній 16 або 8), що максимально відповідає поточному inter-макроблоку/блоку (т.зв. пророкування (prediction)), після чого макроблоки/блоки почергово віднімаються один з одного, в результаті цієї операції визначають різницевий макроблок/блок, або ж помилку передбачення (prediction error), який в подальшому використовується при кодуванні.

Важливо відзначити, що площини, на основі яких побудована процедура компенсації руху (опорні площини (reference planes)), повинні збігатися з площинами, відновлюваними в процесі декодування (при обчисленні помилки передбачення, ні в якому разі не повинна використовуватися інформація, що доступна тільки на етапі кодування). Результатом дії описаної процедури крім блоків, що містять помилку передбачення, є поле двокомпонентних векторів руху (motion vectors)[24].

Пошук схожих блоків проводиться тільки для компонента Y, вектори для компонентів U і V виходять шляхом ділення навпіл векторів для першої компоненти. Відповідно до стандарту, вектори руху мають в загальному випадку довжину, кратну 1/2 або 1/4. Для отримання нецілих векторів, опорні площини збільшуються (масштабуються) відповідно в 2 і в 4 рази по кожному з вимірів (для збільшення використовується білінійна інтерполяція). Крім збільшення, стандарт приписує застосування так званого педдінга (padding). Процедура полягає в розширенні площини з усіх боків на величину, рівну 16 точкам для компонента Y і 8 - для компонентів U і V (і те й інше в цілих одиницях). Новим точкам присвоюються кольори найближчих до них граничних точок початкової площини. Педдінг (в даному випадку, можливо, має сенс використовувати термін розширення) сприяє підвищенню ефективності компенсації руху поблизу кордонів відео-площин.

Компенсація руху може здійснюватися як макроблоками, так і блоками, тому для кожного макроблоку можуть бути знайдені або 1, або 4 вектори руху. Вибір здійснюється на основі порівняння обсягів, того скільки буде займати закодований макроблок в тому і в іншому випадку з урахуванням вкладу вектору руху.

Як було відзначено, P-площини можуть містити як inter-, так і intra-макроблоки. Завдяки компенсації руху, inter-макроблоки, як правило, кодуються з більшою ефективністю в порівнянні з intra-макроблоками, проте використання intra-макроблоків не пов'язане зі зберіганням додаткової

інформації про вектори руху, що знову ж таки може виявитися більш вигідним. Дилема долається так само, як і в попередньому випадку: тип використовуваного макроблоку вибирається виходячи з обсягу його сумарного розміру.

Стандарт MPEG4 передбачає застосування особливого типу компенсації - компенсація з перекриттям (Overlapped motion compensation). Компенсація з перекриттям використовується тільки для блоків компоненти Y. При використанні цього методу різниця береться не між поточним блоком і деяким схожим блоком, що належить попереднім площинам, а між поточним блоком і зваженою суперпозицією трьох схожих блоків. В якості відповідних трьох векторів руху вибираються наступні вектори: вектор для даного блоку і два вектори для блоків, сусідніх до даного блоку в оброблюваному макроблоці.

B-площина є розширенням поняття P-площини. Відмінність B-площини від P-площини полягає в тому, що при кодуванні B-площині для компенсації руху можуть використовуватися як попередні, так і наступні I- і P-площини (зі зрозумілих причин, самі B-площини для цього використовуватися не можуть). Відповідно вводяться поняття прямого і зворотного пророкування (forward and backward prediction). Кожен макроблок в площині може бути передбачений макроблоком на попередній площині, макроблоком на наступній площині і суперпозицією цих макроблоків. Не вдаючись в подробиці процедури компенсації руху для B-площин, відзначу, що використання двонаправленого передбачення (bidirectional prediction) дозволяє помітно підвищити ефективність передбачення, а отже, і ефективність кодування.

Останній тип закодованої відео-площини - S-площина - має безпосереднє відношення до спрайтів (sprite). Спрайтом в стандарті MPEG4 називається частина зображення [25], видима на протязі певного інтервалу в відео-потоці. В ролі спрайту найчастіше виступає фон (задній план). При

декодуванні з використанням спрайтів, фрейми частково відновлюються з окремих областей спрайту шляхом відображення цих областей на ту чи іншу область декодованого фрейму з використанням перспективного перетворення (крім відображення, застосовуються також і інші спеціальні процедури). Спрайт зазвичай зберігається і кодується окремо, при цьому спосіб кодування ідентичний способу кодування I-площин.

Окремим випадком використання спрайтів є S (GMC) площини. У ролі спрайту тут виступає одна з попередніх I- або P-площин. Цей метод майже нічим не відрізняється від методу обробки P-площин, за тим винятком, що процедурі компенсації руху передує процедура глобальної (застосовується до всієї площині) компенсації руху з використанням перспективного перетворення. Технологія GMC (Global motion compensation), найчастіше застосовується для нівелювання руху і зміни оптичних параметрів камери.

Площини різних типів, що йдуть підряд, діляться на групи що кодуються окремо. На початку групи повинна знаходитися I-площина, так як тільки вона може бути декодована першою. Далі розташовані площини інших типів. B-площини зазвичай чергуються з P- або S (GMC) - площинами (хоча це і не обов'язково, наприклад, нерідко одна P- або S (GMC) - площина чергується з двома і більше B-площинами, що йдуть поспіль). Розподіл площин на групи полегшує процес доступу до інформації, а також забезпечує велику стійкість. Останнє обумовлюється тим, що в разі помилки пошкодженим виявляється не весь закодований потік, а тільки окрема його частина.

Розглянемо коротко процедури кодування intra/inter-блоків і векторів руху. Intra і inter-блоки кодуються практично одним і тим же методом за невеликими винятками: спочатку проводиться блочне дискретне косинусе перетворення (discrete cosine transform - DCT), потім отримані в результаті перетворення коефіцієнти (DCT-коефіцієнти) квантуються, після чого квантовані коефіцієнти кодуються з використанням заздалегідь фіксованої

системи кодів змінної довжини. Основна відмінність методу кодування intra-блоків від методу кодування inter-блоків полягає в тому, що в intra-блоках по-особливому обробляється старший коефіцієнт перетворення, що описує середній по блоку рівень сигналу (DC-коефіцієнт): кодується не сам коефіцієнт, а різниця між ним і DC-коефіцієнтом деякого сусіднього блоку. схема кодування всіх інших коефіцієнтів виглядає наступним чином: здійснюється обхід блоку в зигзагоподібному порядку - від старшого коефіцієнта, що знаходиться в лівому верхньому кутку блоку, до молодшого, що знаходиться в правому нижньому кутку (передбачені три варіанти зигзагоподібного обходу). У процесі обходу послідовно кодуються групи коефіцієнтів наступного типу: послідовність нульових коефіцієнтів і перший та наступний за ними ненульовий коефіцієнт. Групи нумеруються, причому один номер резервується для специфічного випадку - ситуації, коли всі, коефіцієнти що залишились - нульові. Груповим номерами зіставляються коди змінної цілої довжини. Довжина вибирається виходячи з наступного критерію: номерам груп, які трапляються частіше, ставляться в відповідності коротші коди. Система кодів є префіксною: коди, що належать системі, не є початком інших кодів з цієї системи. Префіксність робить можливим однозначне декодування [26].

Спосіб кодування коефіцієнтів блоків, отриманих в результаті застосування дискретного косинусного перетворення, обумовлений наступною особливістю: при обробці реальних зображень, як правило, DCT-коефіцієнти зменшуються за абсолютною величиною, при русі по блоку зліва направо і зверху вниз; відповідно, під час зиґзаґоподібного обходу коефіцієнти зменшуються за абсолютною величиною, причому, будучи квантовими, дуже багато з них(особливо коефіцієнти, що опиняються під час обходу останніми) мають нульове значення.

Крім наведеного способу кодування, існує кілька модифікованих способів, при яких з коефіцієнтів верхнього рядка і лівого стовпчика блоку

попередньо віднімаються коефіцієнти, що стоять на тих же позиціях в сусідніх блоках.

Стандартом передбачається дві схеми квантування коефіцієнтів, що отримуються в результаті дискретного косинусного перетворення. Одна з них повторює схему, використовувану в стандарті H.263, і полягає в розподілі всіх коефіцієнтів, крім DC коефіцієнта в intra-блоках, на одне і те ж число - подвоєний параметр квантування. при цьому в випадку inter-блоків перед квантуванням з абсолютного значення коефіцієнта віднімається значення, що дорівнює половині параметра квантування. DC-коефіцієнт в intra-блоці завжди ділиться на 8. В альтернативній схемі більш ретельно враховуються особливості зорового сприйняття інформації. Для DC-коефіцієнта використовується нерівномірне квантування, при якому дільник залежить не тільки від параметра квантування, але і від типу кольорової компоненти. Всі інші коефіцієнти кодуються в два етапи. спочатку кожен з них ділиться на відповідне йому число зі стандартної для даного типу блоку (intra або inter) матриці квантування (розмір матриці збігається з розміром блоку - 8x8). Матриця влаштована таким чином, щоб при русі зліва направо і згори донизу подільники збільшувалися. (Справа в тому, що зір сильніше сприймає зміни, що описуються низькими частотами гармонійного спектру, тому високими частотами можна практично безболісно знехтувати). На другому етапі частково квантовані коефіцієнти повторно діляться на число, що дорівнює подвоєному параметру квантування.

Вектори руху кодуються покоординатно. Як і у випадку з DCT-коефіцієнтами, при генерації коду використовуються системи префіксних кодів змінної довжини. Кодується не координата вектору, а різниця між нею і однією з відповідних координат вектору деякого сусіднього блоку (у виборі беруть участь координати векторів трьох сусідніх блоків; вибирається середня з трьох координат). Різним значенням різниці координат ставляться у

відповідність різні коди змінної довжини, при цьому маленьким за абсолютним значенням різницям відповідають коротші коди.

Описані методи кодування коефіцієнтів і векторів руху використовуються не завжди: виникають ситуації, коли дані процедури взагалі не потрібні. Однією з таких ситуацій є випадок рівності нулю всіх DCT-коефіцієнтів блоку. Про те виконується ця умова чи ні, сигналізує спеціальний прапорець. Крім цього прапора, існує також і інший прапорець, наявність якого дозволяє спростити процедуру кодування. Він сигналізує про те, чи кодується даний inter-макроблок взагалі. Якщо прапор має одиничне значення, ніяка додаткова інформація, яка описувала коефіцієнти і вектори руху не передається: в якості векторів руху беруться нульові вектори, також нульовою покладається помилка передбачення.

У стандарті MPEG4 обумовлено застосування двох методів пост-обробки відеопослідовності: деблокінгу (deblocking) і дерінгінг (deringing). Перший застосовується для зменшення блочного ефекту - особливого типу помилки відновлення закодованого зображення, обумовленого кодуванням з високим рівнем втрат; другий являє собою техніку адаптивного згладжування сигналу, що дозволяє усувати окремі високочастотні спотворення.

Серед іншого стандарт MPEG4 описує формат представлення деяких дуже специфічних візуальних об'єктів: нерухомого зображення (still image), анімації лиця (Face animation) і сітчастого об'єкту (mesh object).

Фотознімок (як правило, звичайна картинка) кодується із застосуванням вейвлет-перетворення [27]. Здійснюється стандартна двомірна частотна декомпозиція, при цьому в якості базису може бути використаний класичний в вейвлет-технологіях біортогональний базис, або якийсь специфічний базис, параметри якого заздалегідь невідомі і передаються разом з кодом. Коефіцієнти, що отримуються в наслідок вейвлет-перетворення, квантуються з використанням рівномірного скалярного

квантувача, після чого кодуються за стандартною схемою, в якій застосовуються так звані нуль-дерева (zero-tree). Код генерується на основі високоефективного арифметичного кодування.

У стандарті описується досить оригінальна технологія, призначена для опису міміки людського обличчя. Особа відновлюється на основі його параметричної моделі (анімація лиця). Уніфікований опис дозволяє представляти обличчя в досить економній формі, в порівнянні зі звичайними методами кодування (наприклад, описаними вище). Найбільш ймовірна область застосування такого рішення - кодування відео-конференцій.

Спеціальне подання об'єктів на площині у вигляді трикутної сітчастої структури з накладеною на неї текстурою носить назву mesh-технологія. опис топології сітки здійснюється шляхом кодування координат вершин трикутників, а також векторів руху, що визначають зміну положення цих вершин у часі. паралельно з трансформацією топології сітки проводиться трансформація текстури що накладається. Сітчасте уявлення об'єктів виявляється ефективно в тих випадках, коли ці об'єкти спочатку задаються в векторному вигляді.

Вище коротко викладено методики обробки інформації, що застосовуються в стандарті MPEG4.

1.1.3 Стандарт H.264

При розробці стандарту вперше були об'єднані зусилля фахівців Міжнародного телекомунікаційного Союзу (International Telecommunication Union - ITU) в особі їх стандартизаційного сектору (ITU-T), Міжнародної Організації по Стандартизації (International Organization for Standardization - ISO) і Міжнародної Комісії з Електротехніки (International Electrotechnical Commission - IEC). До об’єднання перша організація займалась розробкою телекомунікаційних стандартів, тоді як останні дві - стандартів загального

плану. Раніше ними було запропоновано стандарти кодування відеоінформації H.261, H.263 (ITU-T); JPEG, JPEG2000, MPEG (ISO / IEC).

На відміну від стандарту MPEG4, технології, описувані в стандарті H.264[28], призначені для обробки звичайних відео-послідовностей, заздалегідь не розділених на відео-площини. Природно, це не є обмеженням, так як всі методи кодування можуть бути з тим же успіхом застосовані і до випадку багато-площинного кодування.

Описані в новому стандарті методи в цілому не сильно відрізняються від методів, передбачених стандартом MPEG4. Проте, новий стандарт включає в себе кілька рішень, які заслуговують на окрему увагу. В якості базових одиниць кодування в даному випадку виступають блоки розміру 4x4, 8x8, 8x4 і 4x8. При цьому розмір макроблоку залишається тим же - 16x16, що дає можливість здійснювати компенсацію руху з великою кількістю різних варіантів розбиття макроблоку на блоки. Розбиття здійснюється в два етапи. Спочатку макроблок 16x16 розбивається на одну або кілька прямокутних областей, кожна з яких має свій, відмінний від інших вектор руху (можливі наступні варіанти розбиття: 16x16, 16x8 + 16x8, 8x16 + 8x16 і 8x8 + 8x8 + 8x8 + 8x8). На другому етапі кожна область розміру 8x8 в свою чергу розбивається на області одного щ розмірів: 8x8, 8x4 + 8x4, 4x8 + 4x8 і 4x4 + 4x4 + 4x4 + 4x4. Для блоків 4x4 застосовується досить специфічне перетворення, відмінне від дискретного-косинусного перетворення. Дане перетворення має схожу з дискретно-косинусним перетворенням ефективність, але меншу обчислювальну складність (використовуються тільки операції додавання і зсуву). Вельми оригінальною ідеєю в даному випадку є повторне кодування старших коефіцієнтів перетворення. Кодування кожного блоку це повторне застосування 24 процедур кодування блоків (16 для компонента Y і по 4 для компонентів U і V). Отримані 16 старших коефіцієнтів перетворення для компонента Y і, відповідно, по 4 старших коефіцієнта для компонентів U і V піддаються повторному

перетворенню. Для блоку старших коефіцієнтів використовуються компоненти Y дещо відмінні від попереднього перетворення 4x4, а для блоку старших коефіцієнтів компонентів U і V - спеціальне перетворення 2x2.

Для блоків 8x8, 8x4 і 4x8 також передбачені свої перетворення, які, однак, не є обов'язковими. Замість кодування на основі перетворення може застосовуватися екстраполяція. При використанні такого рішення значення яскравості / кольору всередині блоку виходять шляхом наближення, в якому враховуються значення яскравості / кольору граничних точок сусідніх блоків.

Як видно, компенсація руху для стандарту H.264 зазнала досить суттєвих змін порівняно зі стандартом MPEG4. Слід виділити також ще дві його особливості: велика точність представлення векторів руху і інший спосіб збільшення масштабу опорних фреймів які використовуються для отримання передбачення, допускається точність рівна 1/4, або 1/8 (в MPEG4 максимальна точність, як відомо, становить 1/4). При збільшенні або зменшенні опорних фреймів застосовується досить складна багато точкова інтерполяція. У сукупності ці рішення дозволяють отримати досить вагомий приріст в ефективності.

Великий інтерес викликає метод кодування квантованих коефіцієнтів перетворення. Для intra-макроблоків/блоків застосовується так зване intra-передбачення зі значень коефіцієнтів intra-макроблоків/блоків віднімаються числа, одержувані з сукупностей значень деяких коефіцієнтів сусідніх макроблоків/блоків з використанням однієї з декількох схем передбачення (для макроблоків передбачено 4 схеми, а для блоків - 9 схем). При кодуванні отриманих різниць для випадку intra-блоків і помилок передбачення векторів руху, для випадків inter-блоків використовується метод, практично ідентичний методу що описується в стандарті MPEG4 [29].

Дієвим прийомом збільшення ефективності представлення відеоінформації виявляється використання для генерації коду арифметичного кодування замість префіксного. Особливістю пропонованої реалізації

арифметичного кодування є те, що кодування приймає на вхід тільки бінарні символи. З одного боку, це дещо ускладнює застосування даного способу генерації коду (доводиться використовувати бінарну декомпозицію для багатосимвольного алфавіту), з іншого боку, подібний підхід робить можливими досить швидкі реалізації і спрощує процес побудови імовірнісних моделей. У стандарті описується високопродуктивна таблична реалізація методу.

Стандарт H.264 з самого початку був орієнтований на застосування, перш за все, в комунікаційних системах, тому особливу увагу тут приділено підвищенню завадостійкості і забезпечення високого ступеня гнучкості інформаційного представлення, необхідної для забезпечення зручності передачі закодованої відеоінформації по мережах різної природи. Передбачаються особливі алгоритми поділу закодованого потоку на блоки, що забезпечують потрібні характеристики інформаційного представлення для кожного конкретного випадку.

В заключення можна відзначити, що описаний стандарт є суттєвим покращенням стандарту MPEG4, і дозволяє отримати ефективність, на 30% більшу ніж в форматі MPEG4 [29].

1.1.4 Технологія MVC

Подібно стандарту H.264, технологія MVC орієнтована на застосування до послідовностей фреймів, а не відео-площин, що, ні в якій мірі не обмежує її загальності.

Загальна схема алгоритму обробки інформації, так само як і у випадку зі стандартом H.264, практично ідентична схемі, описаній в стандарті MPEG4. Відмінності, знову ж таки, полягають в деталях реалізації.

Розбиття макроблоку на блоки дуже нагадує метод, який використовується в MPEG4. Основна відмінність полягає в тому, що, якщо

макроблок в компоненті Y розбивається на 4 блоки 8x8, відповідний йому макроблок 8x8 в компонентах U і V розбивається на 4 блоки 4x4.

Для отримання базисного розкладання використовується кілька просторово-частотних перетворень. Крім стандартного дискретного-косинусного перетворення 8x8, передбачається дискретне-косинусне перетворення для блоків 8x4, 4x8 і 4x4, а також альтернативне перетворення для блоків 8x8 і 4x4, іменоване перетворенням Кархунена-Лоєва (Karhunen-Loeve transform). Альтернативне перетворення дозволяє добре описувати колірні збурення, що поширюються фронтально від деякої похилої лінії, що проходить через центр блоку. Замість базисного розкладання може застосовуватися екстраполяція аналогічно тому, як це робиться в стандарті H.264.

Так само як і в стандарті H.264, для intra-блоків використовується intra-передбачення. Існує в цілому 13 методів передбачення, вибір яких здійснюється в залежності від особливостей оброблюваного відеопотоку. Основний інтерес в технології MVC привертає методика компенсації руху, для отримання передбачення блоку використовується 6-параметричне афінне перетворення. При обчисленні помилки з блоку віднімається прямокутна область, що переводиться в деяку область фрейму афінним перетворенням, що використовується для передбачення. Для обчислення точних параметрів перетворення використовуються бікубічні сплайни для компоненти Y і білінійна інтерполяція для компонентів U і V. При кодуванні параметри перетворення квантуються.

На відміну від стандарту H.264, в даній технології для генерації коду не використовується арифметичне кодування, що слід розглядати як явний недолік. Однак підбір системи префіксних кодів виглядає цілком виправданим, так що не виключено, що виграш від застосування арифметичного кодування є все-таки не таким істотним.

При розгляді технології MVC слід звернути особливу увагу на алгоритм деблокінгу. Тут використовується схема нелінійної фільтрації, що забезпечує високі результати як по візуальній якості, так і за кількісним відношенням сигнал/шум.

1.1.5 Стандарт AVC/H.264

Стандарт стиснення відео AVC [30] (Advanced Video Coding) був запропонований групою експертів JVT (Joint Video Team) в травні 2003 року. У той час він був революційний прорив в технологіях стиснення відео. Новий стандарт повністю перевершив стандарти MPEG-2 і MPEG-4 Part 2 (SP, ASP). За деякими оцінками, для зберігання відео, стисненого за стандартом AVC, потрібно в 2 рази менший обсяг пам'яті, ніж для відео, стисненого за стандартом MPEG-2 при однаковій якості [30].

Новий стандарт дозволив отримати відео стандартної чіткості телевізійної якості при потоці в 1,5 Мбіт/с. Такий ступінь стиснення дозволяє передати приблизно 12 стислих телевізійних каналів в смузі частот, до того зайнятою одним аналоговим ТВ-каналом. Також, впровадження AVC дозволило телеоператорам надавати нові послуги для відео в місцях, де вони до цього були не доступні і відкрило можливість "упаковувати" більшу кількість каналів відео в вузький і дорогий діапазон частот при передачі. Переваги в ефективності кодування, такі як гарна якість відео при низькому бітрейті(бітова швидкість – кількість бітів інформації на секунду відеопотоку), зробили AVC безумовним лідером в системах інтернет телебачення, і вивели цю індустрію на якісно новий рівень. Також AVC дозволив істотно збільшити якість цифрового телебачення і зробив широкодоступним телебачення високої чіткості HDTV.

Низька плата за ліцензію від MPEG-LA також сприяла стрімкому впровадженню стандарту, і, до теперішнього часу, H.264/AVC успішно закріпився на ринку. У 2010 році кількість рішень на базі AVC перевершила

кількість аналогічних рішень на базі застарілого стандарту MPEG-2 і збільшувалася з кожним роком аж до прийняття наступного стандарту стиснення відео H.265 / HEVC.

Стандарт H.264 забезпечує вдосконалену технологію кодування за методами, схожим з технологією кодування попередніх стандартів MPEG і ITU-T. Більш висока продуктивність і якість забезпечуються новими інструментальними засобами, які включають в себе наступні:

- покращена оцінка руху. Оцінка руху дозволяє шукати суб-макроблоки різного розміру від 16x16 до 4x4 пікселів. Вектори руху тепер допускають точність до 1/4 пікселя для сигналу яскравості, і до 1/8 пікселя для сигналу кольоровості. Крім цього, суттєво покращено кодування векторів руху; використовується їх передбачення;
- просторове передбачення. H.264 виконує внутрішньо-кадрове передбачення для intra-кодованих блоків, дозволяючи застосувати до 9 різних способів такого передбачення в залежності від напрямку;
- оптимізація параметрів кодування. Класичний метод кодування передбачає прийняття локально оптимальних рішень на кожному етапі. Очевидно, що в такому випадку результуюче рішення може виявитися не оптимальним. Стандарт AVC пропонує новий алгоритм оптимізації параметрів кодування RDO(Rate distortion optimization), суть якого полягає у виборі таких параметрів, використання яких найкращим чином позначиться на результаті;
- модифіковане ДКП. Для перетворення залишкової інформації, використовується модифіковане ціле дискретне косинусне перетворення (МДКП), яке запобігає помилці округлення. Важливою відмінністю від попередніх стандартів є розміри блоків для ДКП. AVC дозволяє здійснювати перетворення над блоками розміру 8x8 і 4x4 пікселя;

- фільтрація границь блоків. Ще одним нововведенням стандарту AVC стало використання деблокуючого фільтру, основним завданням якого є згладжування блокових артефактів на кордонах макроблоків в зображенні. Таким чином, покращуючи візуальне сприйняття кожного кадру і всього відеоряду в цілому.
- покращення кодування при плавних рухах. В AVC доданий ряд нових умов для кодування макроблоків в режимі «skip». Фактично, в цьому випадку, макроблок не кодують, а використовується інший макроблок в тій же позиції, але з іншого кадру. Таким чином, досягається суттєвий виграш на малих бітрейтах або при плавних рухах камери, коли вся картинка рухається однаково;
- ентропійне кодування. Стандарт забезпечує два більш продуктивних процеси ентропійного кодування. Context-adaptive variable length coding (CAVLC - контекстно-адаптоване кодування з різною довжиною кодового слова) - ентропійний кодер, принцип дії якого близький до алгоритму стиснення Хаффмана. CAVLC дозволяє швидко стискати інформацію, при цьому забезпечує прийнятну ступінь стиснення. Context-adaptive binary arithmetic coding (CABAC - контекстно-адаптоване двійкове арифметичне кодування) являє собою арифметичний кодер. Використання CABACa дозволяє домогтися ефективності стиснення близькою до максимально можливої, однак він вимагає значно більше ресурсів, ніж CAVLC.

1.1.6 Стандарт HEVC/H.265

Замість макроблоків що застосовуються в H.264 в HEVC використовуються блоки з деревовидною структурою кодування. Виграш кодера HEVC - в застосуванні блоків більшого розміру[31]. Це можна

побачити в тестах PSNR(peak signal-to-noise ratio – пікове відношення сигнал-шум) з моделлю кодеру HM-8.0, зрівнюючи результати кодування з різними розмірами блоків. В результаті тестів було показано, що відносно кодування блоками 64×64 пікселя бітрейт збільшується на 2,2% - при використанні блоків розміром 32×32 – і на 11,0% при використанні блоків 16x16. У тестах кодування відео з роздільною здатністю 2560×1600 пікселів при використанні блоків розміром 32×32 пікселів, бітрейт збільшується на 5,7%, а при використанні блоків розміром 16x16 пікселів - на 28,2% в порівнянні з відео, де використані блоки розміром 64×64, при однаковому піковому відношенні сигнал-шум. Також було показано, що застосування блоків більшого розміру більш ефективно при кодуванні відео з високою роздільною здатністю. Ще одним результатом тестів було те, що для декодування відео, закодованого з блоками розміром 16×16, потрібно на 60% більше часу, ніж при використанні блоків 64×64. Тобто застосування блоків великих розмірів підвищує ефективність кодування при одночасному скороченні часу декодування [32]

У ході дослідження [32] було проведено порівняння ефективності кодування основного профілю H.265 з кодеками H.264/MPEG-4 AVC High Profile(HP), MPEG-4 Advanced Simple Profile(ASP), H.263 High Profile Latency(HLP) і H.262/MPEG-2 Main Profile (MP). Були закодовані відео розважальних програм і дев'ять тестових відео-послідовностей з 12 різними бітрейтами з використанням даної моделі HEVC HM-8.0, п'ять з них були з HD-дозволом, а чотири - з роздільною здатністю WVGA (800×480). Зменшення бітрейту визначалося на основі PSNR (таблиця 1.1).

Таблиця 1.1 – Порівняння стандартів кодування при рівному PSNR

Стандарт відеокодування	Середнє скорочення бітрейту			
	H.264/MPEG-4 AVC HP	MPEG-4 ASP	H.263 HLP	H.262/MPEG-2 MP
HEVC MP	35.4%	63.7%	65.1%	70.8%
H.264/MPEG-4 AVC HP		44.5%	46.6%	55.5%
MPEG-4 A SP			3.9%	19.7%
H.263 HLP				16.2%

При кодуванні відео в HEVC застосовується такий же «гібридний» підхід, що і у всіх сучасних кодеках, починаючи з H.261. Він полягає в застосуванні внутрішнього і міжкадрового (Intra-/Inter-) передбачення і двовимірного кодування з перетворенням.

У кодері HEVC кожен відеокадр ділиться на блоки. Перший кадр відеопослідовності кодується з використанням тільки внутрішньо кадрового передбачення, тобто застосовується просторове передбачення очікуваного рівня відліку всередині кадру по сусіднім відліках, при цьому відсутня залежність від інших кадрів. Для більшості блоків всіх інших кадрів послідовності, як правило, використовується режим міжкадрового часового передбачення. У режимі міжкадрового передбачення на підставі даних про величину відліків опорного кадру і вектору руху оцінюються поточні відліки кожного блоку. Кодер і декодер створюють ідентичні міжкадрові передбачення шляхом застосування [33] алгоритму компенсації руху за допомогою векторів руху і даних зворотного режиму, які передаються в якості додаткової інформації.

Різницевий сигнал передбачення, який представляє собою різницю між опорним блоком кадру і його пророкуванням, піддається лінійному просторовому перетворенню. Потім коефіцієнти перетворення

масштабуються, квантуються, застосовується ентропійне кодування, і потім передаються разом з інформацією передбачення.

Кодер в точності повторює цикл обробки декодером так, що в обох випадках будуть генеруватися ідентичні передбачення наступних даних. Таким чином, перетворені квантовані коефіцієнти піддаються зворотному масштабуванню і потім зворотному перетворенню, щоб повторити декодоване значення різницевого сигналу. Різниця потім додається до передбачення, і отриманий результат фільтрується для згладжування артефактів, отриманих поділом на блоки і при квантуванні. Остаточне уявлення кадру (ідентичне кадру на виході декодера) зберігається в буфері декодованих кадрів, і буде використовуватися для прогнозування подальших кадрів. В багатьох випадках, порядок кодування і декодування обробки кадрів часто відрізняється від порядку, в якому вони надходять з джерела.

Передбачається, що відеоматеріал на вході кодера HEVC має прогресивну розгортку. У HEVC не представлено явних функцій кодування чергуванням розгортки, так як вона не використовується в сучасних дисплеях і її поширення знижується. Проте, в HEVC були представлені метадані, що дозволяють вказати кодеру, що було закодовано відео з чергуванням розгортку в одному з двох режимів: у вигляді окремих зображень, як два поля (парні або непарні рядки кадру), або весь кадр цілком. Цей ефективний метод забезпечує кодування відеосигналу з чергуванням рядків, минаючи необхідність навантажувати декодери підтримкою спеціального процесу декодування.

1.2 Методика компенсації руху

Компенсація руху (англ. Motion Compensation) - один з основних алгоритмів, що застосовуються при обробці і стисненні відеоданих[34]. Алгоритм використовує подібність сусідніх кадрів в відео послідовності і

знаходить вектори руху окремих частин зображення (зазвичай блоків 16×16 і 8×8, в сучасних кодерах також використовуються блоки розміром 32x32 та 64x64). Задача оцінки руху є ключовою при роботі з цифровим відео. Причиною цього є виняткова важливість інформації про рух при аналізі часової складової відеосигналу, яка, в свою чергу, є найважливішою характеристикою відеосигналів і багато в чому визначає специфіку завдань цифрової обробки відео.

У перших системах цифрового стиснення відеокадри оброблялися незалежно один від одного. Кожен кадр кодувався як зображення, а не як частина відео-потоку. Потім з'явилися алгоритми, що використовують віднімання сусідніх кадрів. У них кодувалися не кадри, а різниця кожного кадру з попереднім. Цей прийом зумовив значне зростання ефективності алгоритмів стиснення завдяки тому факту, що сусідні кадри відео-потоку, як правило, дуже схожі, і їх різниця часто близька до нуля. Винятком з цього правила були випадки наявності руху між сусідніми кадрами, або повна зміна вигляду кадру(найчастіше зустрічається в кінематографі). Наступним кроком стала поява алгоритмів, що використовують компенсацію руху. Компенсацією руху називається перетворення одного кадру з пари послідовних кадрів, що використовує інформацію про рух між цими кадрами, і здійснене таким чином, щоб положення і розміри всіх видимих об'єктів в кадрі опинилися найбільш близькими до їх положення і розміру на другому кадрі пари. Іншими словами, компенсація руху робить один з кадрів пари максимально схожим на інший, використовуючи інформацію про рух між ними. Таким чином, компенсація руху дозволяє використовувати при стисненні надмірність відео-потоку в часі навіть при наявності руху між кадрами, чого не могли робити алгоритми стиснення відео попереднього покоління.

Зауважимо, що існують дві основні області застосування алгоритмів оцінки руху (ОР): стиснення і аналіз відео. У цих областях до алгоритмів

оцінки руху висуваються різні вимоги. При стисненні відео критичне значення має об'єм інформації про рух і компенсація міжкадрової різниці (різниці між поточним і опорними кадрами). При цьому неважливо, чи відповідає напрямок знайдених векторів руху реальному руху об'єктів в відео-потоці: головне - мінімізація компенсованої міжкадрової різниці. В алгоритмах відео-аналізу, навпаки, обсяг інформації про рух не має ніякого значення. Основне значення тут має точність знайденої інформації про рух, відповідність знайдених векторів реального руху в відео-послідовності (true motion estimation). Через особливу важливість завдання ОР, на цей момент було розроблено безліч різних алгоритмів, які можна розділити на наступні основні групи: блокові методи, методи оптичного потоку, фазової кореляції, глобальної оцінки руху, стеження за особливостями, багато-кадрової ОР, а також алгоритми, що комбінують прийоми методів зазначених категорій. Однією з найбільш популярних є група блочних методів. Це обумовлено універсальністю, невисокою обчислювальною складністю і порівняно високою ефективністю алгоритмів цієї категорії. Не останню роль зіграла також простота їх апаратної реалізації. Саме з цих причин в задачах компресії і передачі відео традиційно використовуються в основному блокові алгоритми ОР

1.2.1 Модель руху на зображенні

Цифрове відео являє собою упорядкований набір кадрів (відеопослідовність). Відеопослідовність будемо позначати як $I(t)$, де t - порядковий номер кадру. Кожен кадр - це матриця пікселів, розмір цієї матриці позначимо $m \times n$.

Будемо використовувати такі позначення: $p = \langle x, y \rangle \in [0, m-1] \times [0, n-1]$ - вектор координат пікселя в кадрі, $I(p, t)$ - яскравість пікселя з координатами $p = \langle x , y \rangle$ в кадрі $I(t)$. Взагалі кажучи, яскравість - це тільки один з колірних параметрів пікселя. Існує безліч колірних моделей (RGB, YUV, Lab і т. д.), В

кожній з яких колір визначається декількома компонентами. Однак людське око найбільш чутливе до компоненти яскравості. З цієї причини, далі будемо припускати наявність тільки компоненти яскравості (півтонове відео).

Інформацією про рух в обробці відео називають двовимірний масив векторів руху, розмір якого дорівнює розміру кадру $m \times n$. При цьому під вектором руху в заданій точці розуміється вектор зміни координат цієї точки між двома заданими кадрами.

Розглянемо тривимірну сцену без руху і два кадри $I_1 = I(t_1)$, $I_2 = I(t_2)$, отриманих за допомогою камери з різних точок. Точці P тривимірної сцени в кадрі I_1 відповідає піксель з координатами p_1. Будемо його називати образом точки P на кадрі I_1. Тоді образом точки P на кадрі I_2 буде піксель з координатами p_2. Відразу зауважимо, що під образом тут розуміється не саме зображення точки P на кадрі, а вектор координат її проекції на матрицю камери. Це означає, що образ точки P визначено навіть тоді, коли її зображення на кадрі відсутнє, наприклад, внаслідок наявності будь-якого об'єкта на передньому плані. Вектор руху v в точці p2 для пари кадрів I_1 і I_2 визначається як:

$$V_{t_1}^{t_2}(p_2) = \langle u, v \rangle = p_1 - p_2 \qquad (1.1)$$

В даному прикладі розглянуто випадок нерухомої сцени і камери, що рухається. У загальному випадку, зміна координат точок тривимірної сцени на кадрах може бути обумовлена як рухом камери, так і рухом об'єктів сцени[35].

1.2.2 Загальна схема блочних алгоритмів

Кожен кадр відеопослідовності розбивається на безліч незалежних блоків B_{xy}, x, y - координати блоку. Розбиття проводиться так, що блоки покривають весь кадр, тобто їх сумарна площа дорівнює площі кадру. При цьому в останніх реалізаціях кодеків, блоки можуть бути різного розміру.

Для зручності подальшого викладу кадр, для блоків якого проводиться пошук відповідників, будемо називати поточним, а кадр, в якому проводиться пошук - опорним.

Завдання оцінки руху зводиться до задачі пошуку вектору руху v_{xy} для кожного блоку B_{xy}. При цьому вектори v_{xy} визначаються співвідношенням (1.2):

$$v_{xy} = \arg\min_{v_{xy} \in O} F(t, x, y, v_{xy}) \tag{1.2}$$

$$O = \{\langle x, y \rangle \mid x \in [-v_{\max}, v_{\max}], y \in [-u_{\max}, u_{\max}]\}; \tag{1.3}$$

де O - область пошуку векторів руху, $v_{\max}$, $u_{\max}$ - цілі числа, які обираються в залежності від обраного методу обходу області пошуку, часто використовувані значення 16, 32, 64; $F(t, x, y, v_{xy})$ - функція відповідності блоків, це міра близькості блоків поточного і попереднього кадрів. Прикладами такої функції можуть бути SAD (Sum of Absolute Differences), SSD(Sum of standard deviation) та RDO про яку згадувалось раніше.

Суть роботи алгоритмів даної групи полягає в наступному. Для кожного блоку поточного кадру проводиться мінімізація функції відповідності блоків по 4-му аргументу, при цьому область мінімізації може бути будь-якою, єдиним обмеженням є те, що вона повинна бути підмножиною області пошуку O. Як вектор руху для кожного блоку вибирається аргумент мінімуму функції відповідності, обчислений в цьому блоці. Фактично при обчисленні функції відповідності проводиться визначення "схожості" двох блоків: блоку поточного кадру і блоку опорного кадру, зміщеного на вектор v_{xy}. Таким чином, процес мінімізації функції відповідності є пошуком блоку попереднього кадру, найбільш "схожого" на поточний блок[36].

Важливо зауважити, що розмір області пошуку визначає максимальний модуль векторів рухів. На практиці нерідкі випадки, коли алгоритм оцінки

руху не в змозі знайти вірні вектори руху тільки тому, що амплітуда руху в відео занадто велика.

1.3 Висновки до розділу

В даному розділі було проаналізовано техніки та алгоритми, що лежать в основі сучасних технологічних рішень стиснення відеопотоку. Так в більшості сучасних кодеків використовується методика компенсації руху, яка дозволяє представляти кожен кадр відеопослідовності як різницю в значеннях піксельних компонент відносно попереднього кадру, з використанням векторів руху, які дозволяють замінити передачу окремих значень піксельних компонент на вектор руху $\vec{V}_{x,y}$ який показує зміщення частини зображення на наступному кадрі. Знаходження векторів відбувається за рахунок порівняння блоків зображень один з одним, та оцінки їх подібності спеціальними метриками.

2 МЕТОДИКА ВИЗНАЧЕННЯ ПОДІБНОСТІ ЗОБРАЖЕНЬ, ДЛЯ ЗНАХОДЖЕННЯ БЛОКІВ КАНДИДАТІВ В ПРОЦЕСІ КОМПЕНСАЦІЇ РУХУ.

2.1 Постановка задачі.

Для побудови оптимального методу визначення подібності зображень спочатку потрібно визначитись з характеристиками вхідних даних, на яких будуть проводитись розрахунки обраною методикою чи її частинами.

Під час розрахунку результатів роботи запропонованих нижче алгоритмів будемо використовувати 2 набори даних, з розміром кадру 1920x1080 пікселів. Першим набором даних будуть слугувати дві матриці розміру 1920x1080, з випадково згенерованими значеннями комірок в межах від 0 до 255, що відповідає розмірності кольорової компоненти для 8бітного формату. Другим набором, буде слугувати записаний за допомогою OBS-studio відео потік «Hearthstone» довжиною 1.25секунди, з частотою кадру 29.97кадрів/секунда, в форматі MPEG4(H264), кольоровим форматом YUV 4:2:0 та бітрейтом 24822 кбіт/с.

Як згадувалось раніше, компенсація руху відбувається шляхом розбиття кадру зображення на макроблоки розмірністю 16x16, 32x32, 64x64 пікселів. Оцінка схожості відбувається шляхом порівняння певних характеристик кадру відео-послідовності з відповідними макроблоками в деякому околі положення вхідного макроблоку, на наступному кадрі відео-послідовності. Для ілюстративної оцінки, яка буде давати найкращий результат для кожної з метрик будемо використовувати повний перебір в деякій області навколо оброблюваного блоку. Стандартною областю пошуку, при повному переборі для блоку 16x16 пікселів береться область ±32x±32 пікселя, при цьому нам треба буде зробити 4096 розрахунків для кожної з обраних метрик, для одного блоку, при цьому в нас буде більше 8 тисяч

блоків для кожного кадру(для блоків 16x16), що призведе до 10^6 розрахунків метрики для 2х кадрів.

Метою даного розділу є формування методики оцінки схожості зображень, яка буде оптимізувати швидкість виконання алгоритмічних розрахунків при мінімальній втраті в якості/істинності інформації оброблюваних зображень.

Розроблена методика буде складатися з 2х алгоритмів. Перший алгоритм початкової оцінки схожості блоків, який буде знаходити в області пошуку n – блоків кандидатів, значення обраної метрики схожості яких буде мінімальною. Другий алгоритм уточнення пошуку, який з обраних блоків кандидатів буде обирати однин, опираючись на особливості сприйняття людського ока та бажаний результат в зміні якості зображення.

2.2 Метрики оцінки подібності зображень

2.2.1 Використання середньоквадратичного відхилення як метрики подібності зображень [37]

Середньоквадратичне відхилення SSD (середнє відхилення, середньоквадратичне відхилення, квадратичне відхилення, стандартне відхилення, стандартний розкид) - в теорії ймовірностей і статистиці найбільш поширений показник розсіювання значень випадкової величини щодо її математичного очікування. Зазвичай зазначені терміни означають квадратний корінь з дисперсії випадкової величини, але іноді можуть означати той чи інший варіант оцінки цього значення [38].

Середньоквадратичне відхилення позначається грецькою буквою σ і визначається як:

$$\sigma = \sqrt{D|X|} \qquad (2.1)$$

де $D|X|$ - дисперсія випадкової величини.

Середньоквадратичне відхилення вимірюється в одиницях виміру самої випадкової величини і використовується при розрахунку стандартної помилки середнього арифметичного, при побудові довірчих інтервалів, при статистичній перевірці гіпотез, при вимірюванні лінійного взаємозв'язку між випадковими величинами.

На практиці, коли замість точного розподілу випадкової величини в розпорядженні є лише вибірка, стандартне відхилення, як і математичне очікування, оцінюють[39] (вибіркова дисперсія), і робити це можна різними способами. Терміни «стандартне відхилення» і «середньоквадратичне відхилення» зазвичай застосовують до квадратного кореню з дисперсії випадкової величини (певного через її справжній розподіл), але іноді і до різних варіантів оцінки цієї величини на підставі вибірки [40].

У випадку аналізу схожості зображень, випадковою величиною для розрахунку дисперсії можна використати яскравісну компоненту Y та кольорову компоненту U (формули будемо представляти для розрахунку яскравісної компоненти). Нехай A - набір значень яскравісної компоненти вхідного кадру відеопотоку, а B - відповідний набір значень наступного кадру відеопотоку. Тоді y_i - значення яскравості для і-го пікселя, k – об’єм вибірки (в нашому випадку 16х16 – 64пікселя), тоді середнє арифметичне значення (вибіркове середнє) – оцінку математичного очікування величини яскравості по блоку можна розрахувати наступним чином:

$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k}|a_i| \qquad (2.2)$$

Після цього можемо розрахувати стандартне відхилення одним з двох основних способів:

1. Оцінка стандартного відхилення на підставі зміщеної оцінки дисперсії (іноді просто вибіркова дисперсія):

$$SDD = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(b_i - \bar{x})^2} \quad (2.3)$$

2. Оцінка стандартного відхилення на основі незміщеної оцінки дисперсії (підправлена вибіркова дисперсія):

$$SDD_0 = \sqrt{\frac{\mathrm{k}}{k-1}\sum_{i=1}^{k}(b_i - \bar{x})^2} \quad (2.4)$$

2.2.2 Використання метрики PSNR для покращення оцінки схожості зображень за допомогою SSD[41]

PSNR(peak signal-to-noise ratio – пікове співвідношення сигналу до шуму) це інженерний термін який означає відношення між максимальним значенням характеристики та потужністю шуму, що спотворює значення характеристики. Як наслідок широкого динамічного діапазону більшості зображень PSNR вимірюється в децибелах за логарифмічною шкалою.

Найчастіше дана метрика використовується для вимірювання рівню спотворень при стисненні зображень і визначається через середньоквадратичне відхилення для двох монохроматичних зображень в нашому випадку *A* і *B*, розміру *m*x*n* в нашому випадку це $m=n=\sqrt{k}$, одне з яких вважається зашумленним відображенням іншого. Взявши середньоквадратичне відхилення з формули 2.5, можемо визначити значення PSNR:

$$P = 10\log_{10}\left(\frac{Y^2{}_{\max}}{SDD_0}\right) = 20\log_{10}\left(\frac{Y_{\max}}{\sqrt{SDD_0}}\right) \quad (2.5)$$

де $Y^2{}_{\max}$ - максимальне значення яскравісної компоненти пікселя (для 8бітного зображення – 255).

Таким чином порівняння схожості двох зображень зводиться до порівняння їх PSNR. Варто відзначити що для пари блоків алгоритм SSD з використанням PSNR дає доволі високі результати в плані якості, але

потребує значних витрат ресурсів (множення - операція повільна, навіть таблиця квадратів не сильно прискорює процес) і при цьому є доволі чутливою до зміни яскравості [42]. Також, через взаємозалежність пікселів даний алгоритм важко піддається розпаралелюванню. В сучасних системах SSD використовують в якості уточнюючого алгоритму по невеликій кількості блоків кандидатів, оскільки ж для нас одним з основних пріоритетів є прискорення процесу кодування, SSD в якості початкового алгоритму пошуку блоків кандидатів розглядатися не буде.

2.2.3 Сума абсолютних різниць

SAD - надзвичайно швидкий показник завдяки своїй простоті. По суті, це найпростіша з можливих метрик і обчислюється шляхом взяття абсолютної різниці між кожним пікселем у вхідному макроблоці та відповідним пікселем у блоці, що використовується для порівняння. Ці відмінності підсумовуються для створення простої метрики подібності блоків. Крім того, це дуже ефективно для широкого пошуку векторів руху по багатьох блоках. Ще однією перевагою SAD є відносно легке розпаралелювання, оскільки він аналізує кожен піксель окремо, що дає можливість реалізувати алгоритм з інструкціями, такими як ARM NEON або x86 SSE2. Наприклад, SSE має інструкцію з внутрішньої абсолютної різниці (PSADBW) спеціально для цієї мети. Однак при всій своїй простоті, кількість операцій може досягати 10^5 для кожного блоку і зазвичай займає від 40до80% від загального часу кодування відео-потоку. Точність алгоритму САД, як правило, недостатня для формування правильних векторів руху, і в більшості випадків після знаходження блоків-кандидатів остаточний вибір блоку виконується з використанням інших, більш повільних, але більш точних показників, які краще враховують сприйняття людиною; такими

методами можуть бути сума абсолютних перетворених різниць (SATD), сума квадратів різниць (SSD) і оптимізація викривлення швидкості (RDO).

Для розрахунку SAD для двох кадрів *A* і *B*, можна записати наступну формулу:

$$SAD(A,B) = \sum_{i=1}^{n} |a_i - b_i| \qquad (2.6)$$

де $A = \{a_1, ... a_n\}$ **і** $B = \{b_1, ... b_n\}$ - два набори піксельних даних, відповідних кадрів. При стисненні відео потоку, сума абсолютних різниць найчастіше розраховується для макроблоків розміром 16x16 пікселів. Так, для порівняння двох блоків буде проведено 256 операцій віднімання і скільки ж операцій додавання. Після чого з отриманих SAD, в межах деякого відхилення, формується набір векторів кандидатів, подальша обробка яких відбувається одним з більш повільних методів, наприклад, описаним вище SDD [43].

Одним зі способів зменшення кількості операцій, і як наслідок часу виконання алгоритму, є використання шаблонів порівняння, або так званого порівняння по характерним точкам. Під час цього процесу, замість того, що б брати різницю між кожним пікселем поточного макроблоку та відповідним пікселем з макроблоку опорного кадру, обирається шаблон, згідно з яким частина пікселів буде проігнорована. З врахуванням, що зображення це по суті матриця, розмір якої збігається з розширенням зображення а вхідний макроблок – матриця розмірністю 16x16, можна прив’язати шаблонізацію до візуального зображення матриці. Шаблони використані при розрахунку алгоритму SAD, представлені на рис 2.1. де чорним кольором позначені пікселі макроблоку, що будуть використовуватися для пошуку.

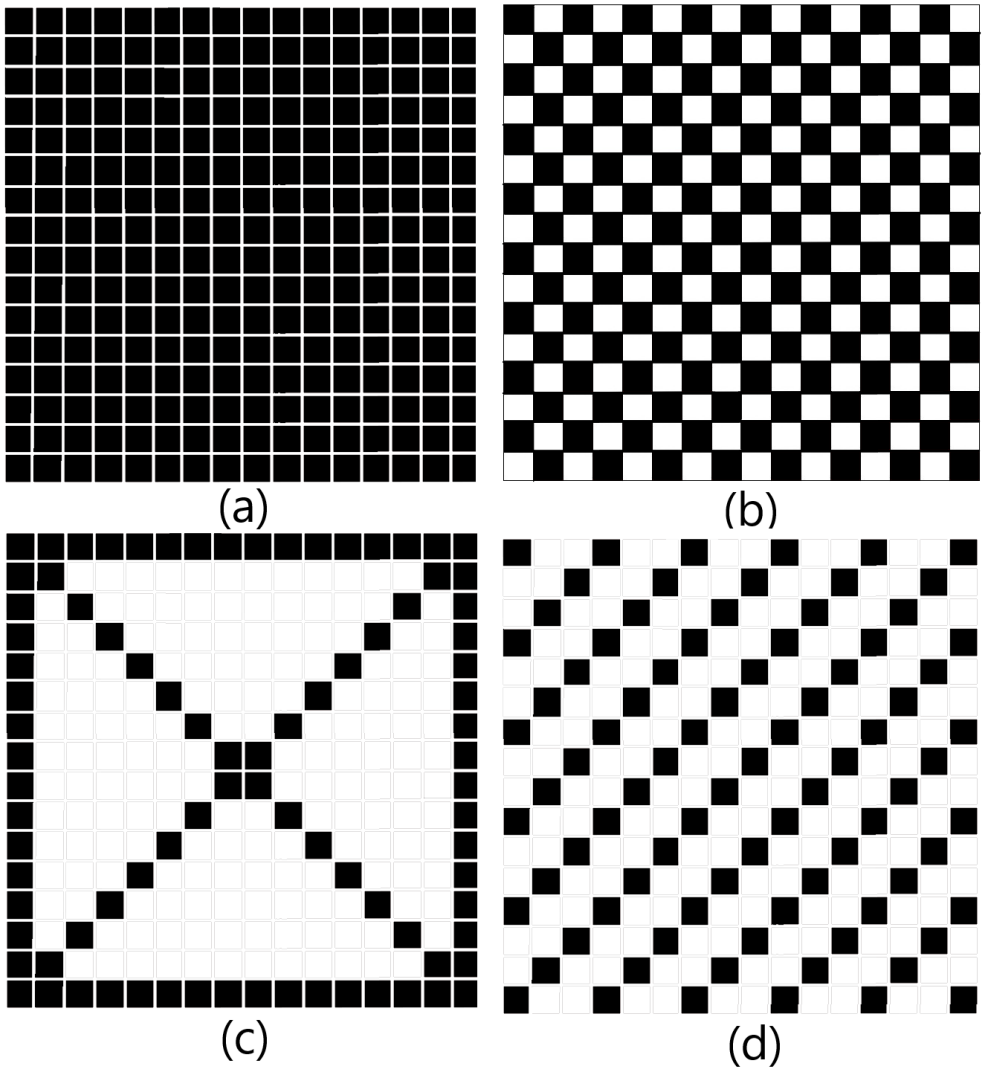


Рисунок 2.1 – Шаблони виконання SAD, а) повний перебір, б) «шаховий» перебір(перебір кожного другого пікселя) с)перебір по головним діагоналям та краєвим точкам, в) перебір кожного третього пікселя.

Для зручності опису обраних шаблонів позначимо їх наступним чином. повний перебір – full SAD (FSAD), «шаховий» перебір – half SAD (HSAD), діагональний – diagonal SAD (DSAD), перебір кожного третього пікселя – third SAD (TSAD).

Зменшення кількості операцій і, як наслідок, зменшення часу обрахунку для обраних шаблонів представлено в таблиці 2.1.

Таблиця 2.1 – Зменшення кількості операцій для обраних шаблонів пошуку.

	HSAD	DSAD	TSAD
К-сть операцій	128	88	86
Різниця в кількості операцій від FSAD	128	168	170
% зменшення кількості операцій відносно FSAD	50	65.6	66.4

Як видно з таблиці, за рахунок використання шаблонів порівняння було отримано суттєве зменшення кількості операцій і, як наслідок, зменшення часу виконання алгоритму, що навіть при розпаралеленні може сягати аналогічного виграшу в часі. При цьому з'являється питання, наскільки сильно зменшення кількості результуючих точок вплине на точність підбору блоків-кандидатів. Для початку проведемо оцінку на випадково згенерованих даних (варто відзначити, що випадково генеровані дані не піддаються ефективному стисненню, але для нас головною метою є показати ефективність пошуку схожих блоків). Для цього запустимо алгоритм підбору векторів кандидатів на матриці розмірністю 1920x1080 з областю пошуку 64 пікселя, для всіх чотирьох шаблонів пошуку, після чого зрівняємо результати, отримані за допомогою FSAD, з кожним з 3х шаблонів, та визначимо відносну різницю значень суми абсолютних різниць. Результати обрахунків для двох матриць з областю пошуку ±32x±32 пікселів можна побачити на рис. 2.2.

Як бачимо з графіку, HSAD дає найкращі результати в плані відхилення від результатів, отриманих класичним SAD, як по середньому значенню, так і по точних співпадіннях блоків кандидатів. Середнє та максимальне відхилення для обраних шаблонів представлені в таблиці 2.2.

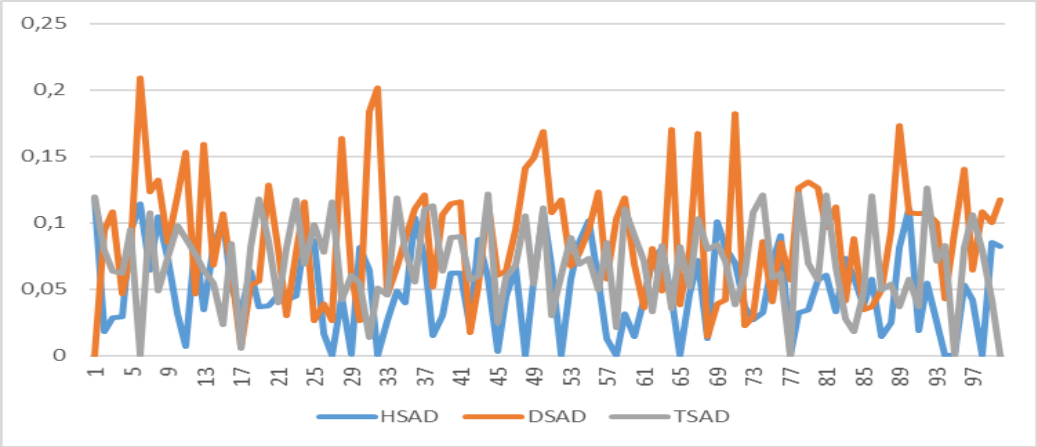


Рисунок 2.2 – Розподіл значень відносної різниці між FSAD та запропонованими шаблонами порівняння.

З даних таблиці, можемо зробити висновок, що шаблон DSAD при більшій кількості контрольних точок, ніж TSAD, показує гірший результат, що зумовлено відносно низькою густиною розподілу контрольних точок в середині матриці, та відносною простотою унікальності рисунку шаблону. TSAD при меншій кількості показує кращі результати оскільки, діагональне розташування дозволяє більш точно підлаштовуватись під випадкові співпадіння значень пікселів.

Таблиця 2.2 – Значення середніх та максимальних відносних відхилень для обраних шаблонів порівняння, для розрахунку випадкових даних.

	HSAD	DSAD	TSAD
Максимальне відносне відхилення	0,107	0,21	0,126
Середнє відносне відхилення	0,05	0,113	0,07

Ще одним важливим спостереженням є те, що при відносній різниці в 5% для 8бітного формату зображення, різниця в яскравісній компоненті (людське око більш сприятливе до зміни яскравості ніж до кольору) буде складати 13 тонів, для 9% - 23. Варто відзначити, що такі різниці в значеннях абсолютних сум обумовленні випадковим заповненням значень комірок, що

призводить до високої ентропії значень. При цьому в реальних наборах даних, для цифрового відео різниця в значеннях кольоровості і яскравості сусідніх пікселів буде значною в переважній більшості на гранях об’єкту та в місцях різкого переходу кольору.

Також ще одним важливим спостереженням, при роботі з обраними шаблонами пошуку, є помітне зниження відхилення результату пошуку від класичного SAD при збільшенні розміру вхідного блоку до 32х32 та 64х64 пікселів. При цьому, як можна побачити в таблиці 2.3., при рості розміру блоку, відносні відхилення значень для алгоритму TSAD максимально наближаються до значень отриманих за допомогою HSAD, оскільки при збільшенні розміру макроблоків збільшується вплив унікальності рисунку розподілу значень пікселів, в той час як відхилення для шаблону DSAD залишаються відносно високими, через збільшення пустої області в середині шаблону.

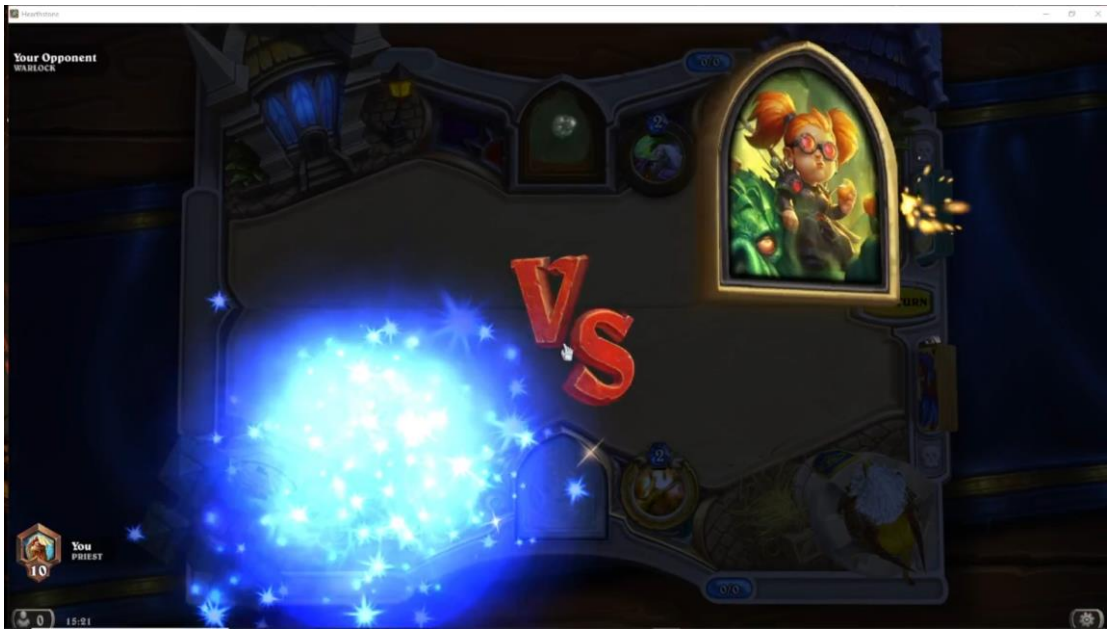
Таблиця 2.3 – Значення середніх та максимальних відносних відхилень для обраних шаблонів порівняння, для розрахунку випадкових даних, при збільшенні розміру пошукових блоків до 32х32 та 64х64 пікселі.

	розмір блоку	HSAD	DSAD	TSAD
Максимальне відносне відхилення	32х32	0,091	0,201	0,99
	64х64	0,073	0,196	0,78
Середнє відносне відхилення	32х32	0,043	0,09	0,05
	64х64	0,037	0,078	0,041

При цьому, на виконання розрахунків SAD (в усіх обраних шаблонах) для блоків 64х64 витрачається на 3.8%, для блоків 32х32 на 2.3%, менше часу ніж для алгоритму з використанням блоків розміром 16х16 блоків (час на обрахунок одного блоку збільшується, але зменшується загальна кількість блоків).

Тепер розглянемо реальний набір даних відео-потоку з характеристиками описаними вище. На рис 2.3 представлено перші два кадри оброблюваного відео-потоку.

(a)



(b)

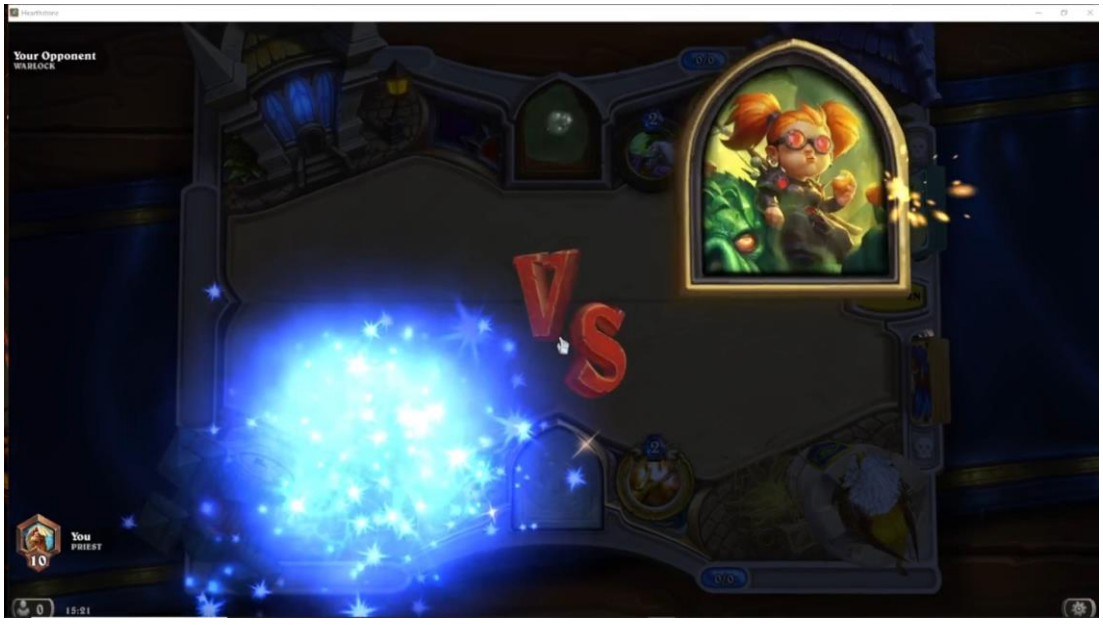


Рисунок 2.3 – Перші кадри тестової відео-послідовності

Розбивши перший кадр відео-послідовності на макроблоки розміром 16x16 пікселів (рис 2.4) можемо побачити, що ентропія в більшості макроблоків є відносно невеликою.

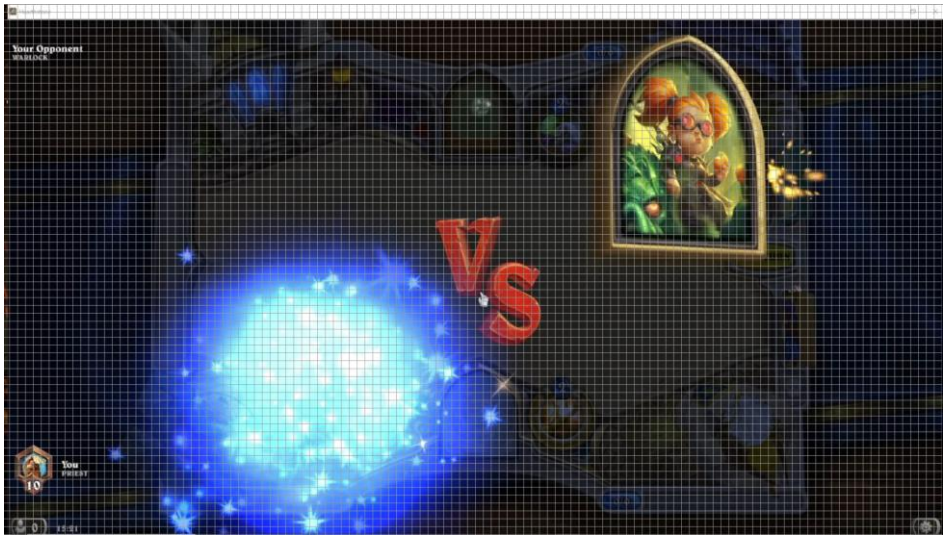


Рисунок 2.4 – Розбиття кадру на макроблоки розміром 16x16пікселів

Як результат, було отримано зовсім іншу картину значень відносного відхилення суми абсолютних різниць (таблиця 2.4).

Таблиця 2.4 – Значення середніх та максимальних відносних відхилень для обраних шаблонів порівняння, для розрахунку реальних даних.

	HSAD	DSAD	TSAD
Максимальне відносне відхилення	0,048	0,112	0,052
Середнє відносне відхилення	0,025	0,063	0,028

В подальшому шаблону DSAD використовуватися не буде, як наслідок його відносно високого відхилення в сумі абсолютних різниць, при однаковому зменшенні кількості операцій порівняно з TSAD, що пов’язано з великими областями пустот (області, які не беруться для розрахунку пікселів)

При розрахунку реальних даних, ще одним важливим показником ефективності обраного алгоритму оцінки схожості блоків є кількість блоків кандидатів для кожного окремого блоку. При чому, ця кількість зазвичай є

найбільшою в однорідних областях кадру, які залишаються незмінними при зміні кадру. Для обраної відеопослідовності кількість блоків кандидатів для незмінних областей при використанні алгоритму SAD може сягати кілька десятків (для тестової відео-послідовності максимальна кількість таких блоків сягала 34). Тим не менш, для розрахунку таких областей на практиці можна брати будь-який з блоків кандидатів, оскільки їхній SAD буде рівний 0, або доволі близьким до нього. Тобто нас цікавлять ті блоки, для яких SAD буде більшим за деяке значення, при нехтуванні яким, в нас не буде виникати візуальної зміни зображення, що може бути вловлена людським оком.

Що ж до блоків, які цікавлять нас найбільше, то це блоки, в яких з’являється міжкадрова різниця. Абсолютна нескомпенсована різниця перших двох кадрів відео-послідовності (рис 2.5).

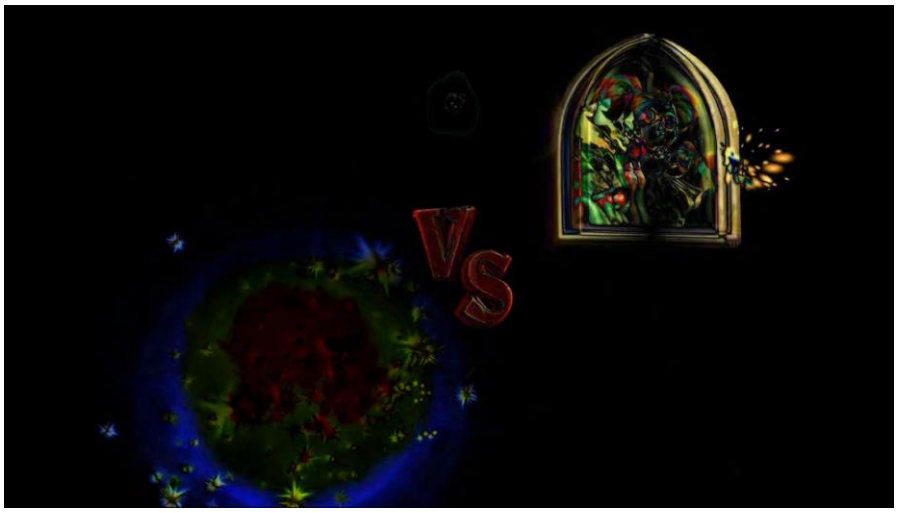


Рисунок 2.5 – Абсолютна нескомпенсована різниця в перших двох кадрах тестової відео-послідовності.

Якщо ж при цьому розбити абсолютну нескомпенсовану різницю на макроблоки, аналогічно до повного зображення, можемо виділити області, в яких ентропія піксельної інформації для нас буде найбільш вагомою (рис 2.6). Також це області, в яких нас буде найбільше цікавити кількість векторів кандидатів, отриманих в результаті роботи SAD, HSAD I TSAD.

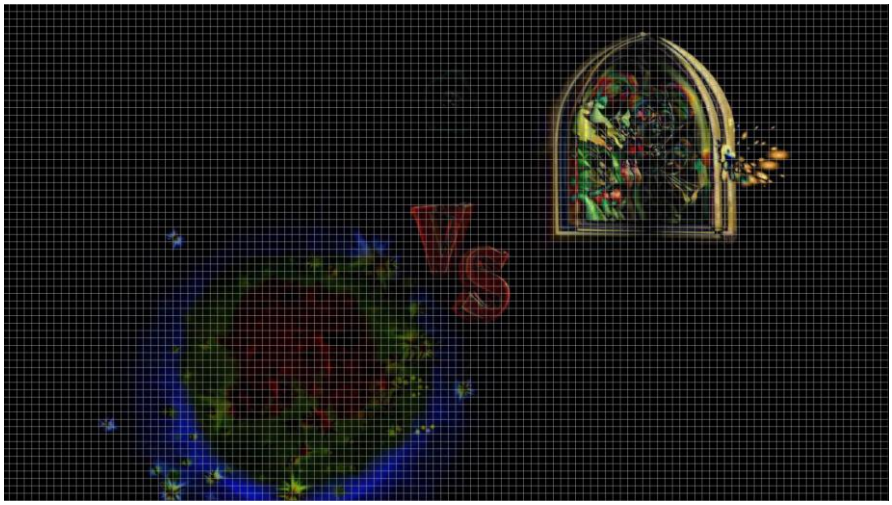


Рисунок 2.6 – Абсолютна нескомпенсована різниця в перших двох кадрах тестової відео-послідовності.

Виділимо область розміром 10x10блоків (160x160пікселів) на другому кадрі нашої відео-послідовності(початок області в точці *T(1232, 352)*, результати обрахунку FSAD, HSAD і TSAD якої буде оцінено для отримання інформації по кількості векторів кандидатів (рис.2.7).

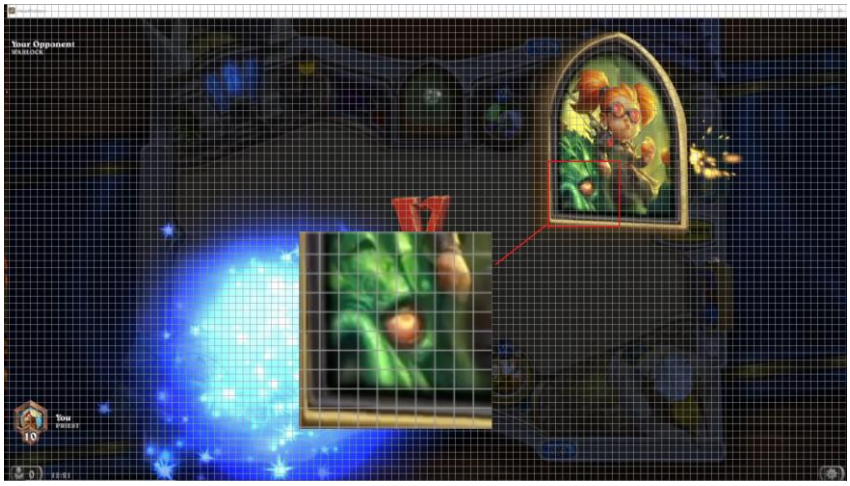


Рисунок 2.7 – Область для оцінки кількості векторів кандидатів першого кадру відеопослідовності

Так розподіл кількості блоків кандидатів розміром 16x16, для обраних алгоритмів SAD представлені на рис 2.8.

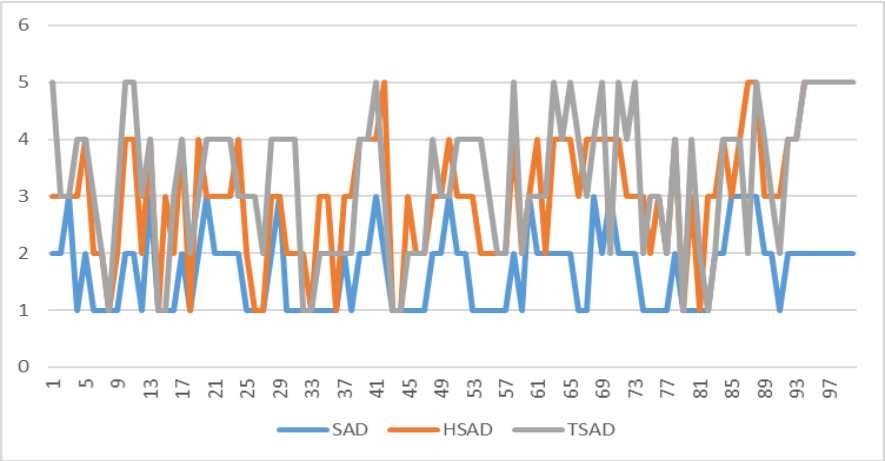


Рисунок 2.8 – Кількість векторів кандидатів для 100блоків в середині області обрахунків.

Як бачимо з рис. 2.8. Використання HSAD і TSAD призвело до збільшення кількості блоків кандидатів для шуканого блоку, за рахунок чого час уточнення блоків кандидатів може вирости. Також ще важливими для нас є точки, в яких SAD дає більшу кількість векторів кандидатів, ніж це робить TSAD. Такий результат може мати два наслідки:

1. Зменшення кількості блоків кандидатів за рахунок унікальності рисунку шаблона (підбір найкращого з можливих варіантів за рахунок точного співпадіння пікселів).
2. Зменшення кількості блоків кандидатів за рахунок пропускання блоків в яких значення SAD підходить під визначене і є найкращим для конкретного блоку.

Як наслідок зменшення кількості блоків кандидатів, доцільно порахувати перекриття отриманих блоків кандидатів, отриманих з використанням TSAD і HSAD, відносно тих що отримує SAD, а також

середню кількість блоків кандидатів для кожного з методу в обраній області, та по всьому зображенні (таблиця 2.5):

Таблиця 2.5 – Загальні характеристики кількості та перекриття блоків кандидатів.

	SAD	HSAD	TSAD
Середня кількість блоків кандидатів для області обрахунків	1,7	3,05	3,28
Середня кількість блоків кандидатів для всього зображення	1,96	2,409	2,479
Перекриття блоків отриманих SAD та обраним алгоритмом для області обрахунків	_____	98,73%	97,51%

Як видно з таблиці, для області обрахунків кількість векторів кандидатів виросла на 44% для HSAD і 51% для TSAD для обраної області, або 18,9% і 21% для всієї області пошуку (зменшення середньої кількості блоків на всій області пояснюється тим, що в деяких випадках запропоновані алгоритми показують кращий вибірковий пошук за рахунок унікальності рисунку шаблону, для областей де росте кількість блоків кандидатів), що може призвести до відповідного приросту часу на стадії уточнення блоків, яка займає 5-15% часу кодування відео-потоку, за умови використання складних метрик уточнення. Тим не менш, виграш в часі все рівно залишається доволі помітним, оцінки часу виконання повного алгоритму визначення блоків кандидатів будуть приведені нижче.

Ще одним наслідком використання алгоритмів HSAD і TSAD є втрата 1.17% і 2.49% блоків кандидатів, отриманих за рахунок використання SAD, щоправда оскільки й сам SAD не дає ідеальне співпадіння блоків, помилкою такого рівня можна знехтувати. Також варто відзначити, що найбільша втрата блоків кандидатів буде проходити в областях з максимальною

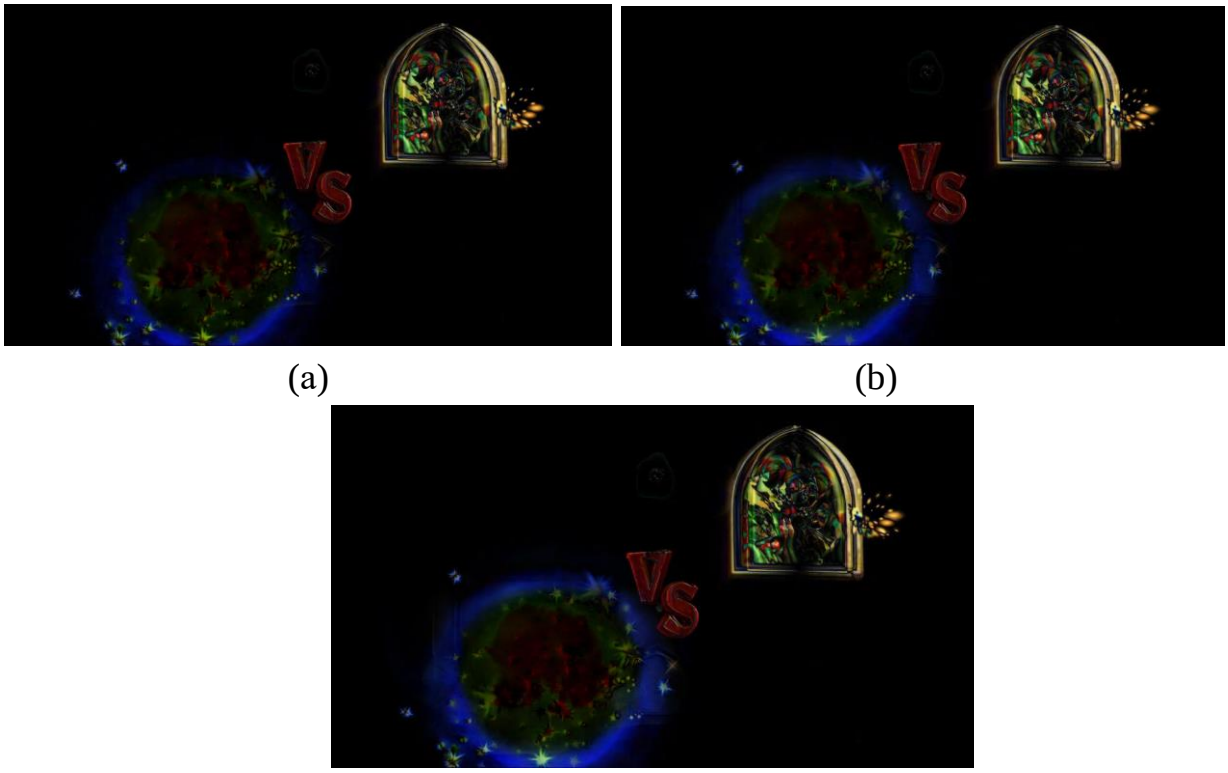
ентропією – на межі об’єктів та різкого переходу кольору, а зростання кількості блоків кандидатів, в областях з низькою ентропією (однорідні області, наприклад задній фон) для яких в більшості випадків вектори руху будуть нульовими. Це може спричинити появу артефактів на границях об’єктів таких, як розмиття, або викривлення границь, а також розмиття однорідних ділянок зображення. Ще одним наслідком використання обраних шаблонів порівняння може бути зміна яскравості зображення. При цьому варто відзначити, що збільшення розміру макроблоків призведе до зменшення середньої кількості блоків кандидатів та збільшення їхнього перекриття (таблиця 2.6).

Таблиця 2.6 – Загальні характеристики кількості та перекриття блоків кандидатів для розмірів 32х32 та 64х64пікселя.

	SAD	HSAD	TSAD
Середня кількість блоків кандидатів для області обрахунків для блоків 32х32	1,34	1,58	1,51
Середня кількість блоків кандидатів для всього зображення 32х32	1,22	1,295	1,289
Перекриття блоків для області обрахунків 32х32	_____	99,12%	98,66%
Середня кількість блоків кандидатів для області обрахунків для блоків 64х64	1,139	1,191	1,175
Середня кількість блоків кандидатів для всього зображення 64х64	1,037	1,053	1,051
Перекриття блоків для області обрахунків 64х64		99,56%	99,58%

Як бачимо, збільшення розміру макроблоків призводить до зменшення кількості блоків кандидатів, як для звичайного SAD, так і для запропонованих шаблонів.

Для того, що б перевірити висунуті твердження на практиці візьмемо нашу відео-послідовність, закодуємо та декодуємо її за допомогою обраних шаблонів та класичним SAD, доповненими алгоритмом SSD на основі PSNR, та методиці, описаній в розділі 1. для технології HEVC з розміром блоку 16x16, та порівняємо отримані результати, визначивши абсолютну нескомпенсовану різницю для перших двох кадрів відео-послідовності. Після того як було отримано декодоване зображення, отримаємо нескомпенсовану абсолютну різницю між оригінальним кадром та кадрами, отриманими через SAD, HSAD і TSAD. При цьому варто розуміти, що навіть при кодуванні перший кадр відео-послідовності залишається майже не змінним відносно оригінального зображення. Результати кодування представлені на рис.2.9.



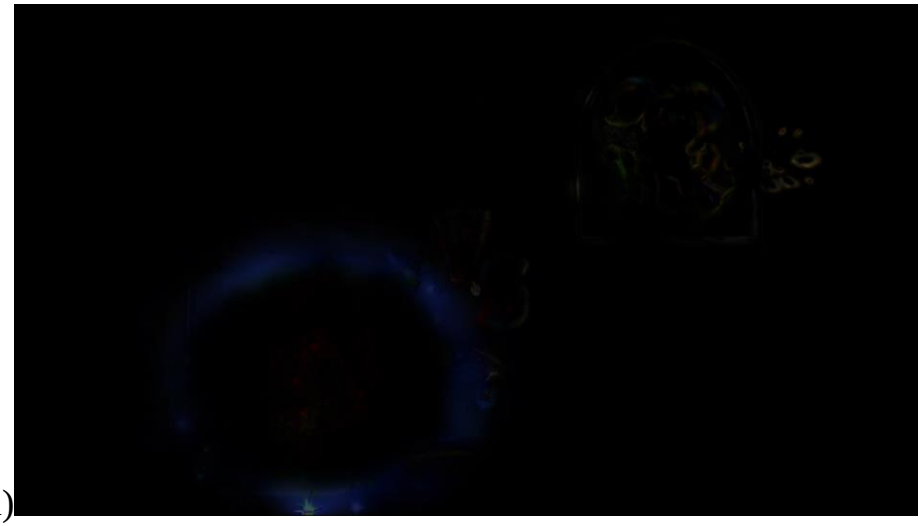
(a) (b)

(c)

Рисунок 2.9 – Абсолютна нескомпенсована різниця між першим та другим кадром відео-послідовності та закодованим за допомогою (a) SAD (b) HSAD (c) TSAD

Як можна помітити з рис. 2.8 на краях, буде отримано значні артефакти, що проявляються в зміні кольорової гами, розмитості країв, викривлення границь об’єктів. Що б краще проілюструвати відмінність кадрів отриманих класичним SAD та обраними шаблонами пошуку, візьмемо абсолютну різницю між кадровою різницею отриманою SAD і різницею отриманою HSAD і TSAD, таким чином виділивши лише артефакти спричинені використанням шаблонів порівняння.

(a)



(b)

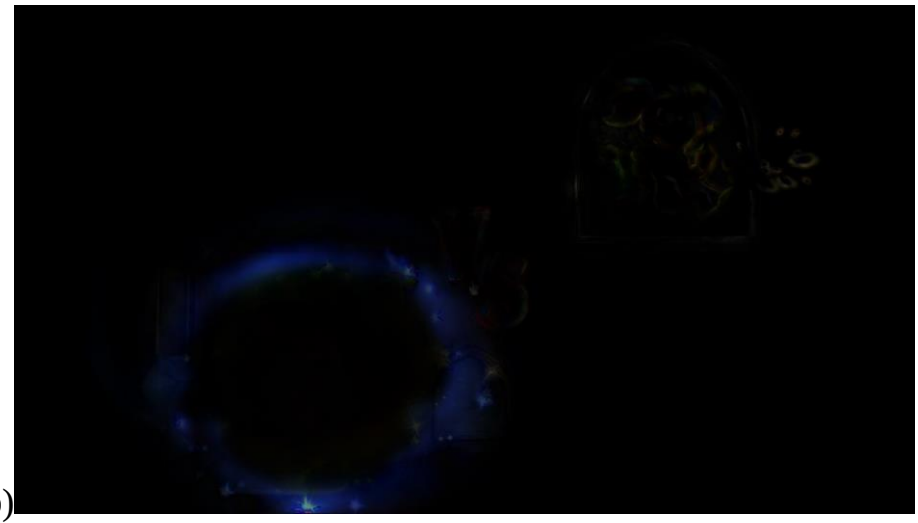


Рисунок 2.10– Абсолютна нескомпенсована різниця між кадровою різницею зображення кодованого за допомогою SAD і (a) HSAD (b) TSAD

Як видно з рис. 2.10, спотворення інформації було отримане в основному на краях об’єктів. Обрахувавши зміну інформаційної складової в

отриманих блоках можемо оцінити загальну зміну кольорової та яскравісної компонент для проблемних областей зображення.

2.3 Тестування результатів роботи запропонованих шаблонів згідно стандартів MPEG-4

Для обрахунку загальних параметрів стиснення використаємо програмне забезпечення кодера MPEG-4 і проведемо обробку 12 відеопослідовностей згідно з комплексними умовами тестування запропонованих спеціальною групою MPEG з оцінки складних відеоінструментів [44]. Також в підрозділі буде наведено результати обробки тестової відеопослідовності «Hearthstone» описаної в розділі 2.1

Тестування буде проводитись за двома параметрами:

1. PSNR (пікове відношення сигналу до шуму - формула 2.5), яке дозволить оцінити загальне спотворення інформації для стисненої відеопослідовності.
2. Пришвидшення роботи алгоритму, яке буде розраховуватися шляхом нормування відносно еталонного алгоритму – метрики SAD.

В даному розділі в якості методу блочного перебору буде виступати алгоритм повного перебору (Full search – FS), а вектор руху буде розраховуватись для значення мінімальної метрики (SAD/HSAD/TSAD) для блоків в середині області пошуку, без використання умов ранньої зупинки пошуку а також використання предикатів (значень векторів руху сусідніх блоків). Такий підхід дозволить оцінити вплив метрики для крайнього випадку, перебору і обрахунку всіх блоків.

Для початку розглянемо результати стиснення відеопотоків з малими бітрейтами (10-112 кбіт/с), для блоків розміром 16х16 пікселів та областю пошуку 16х16 пікселів.

Таблиця 2.7 – Середнє значення PSNR для відеопослідовностей з малим бітрейтом для обробки блоками 16х16пікселів

Відеопослідовність(кбіт/с)	SAD	HSAD	TSAD
Hall-Monitor(10).	30.36	28.83	28.25
Mother & Daughter(24)	34.81	33.32	32.73
Coastguard(48)	28.89	27.38	26.79
Coastguard (112)	27.04	25.57	24.96
Foreman (112)	30.04	28.56	27.95

Як можна побачити з даних таблиці, використання шаблонів порівняння (зменшення кількості результуючих точок) призводить до помітного зниження PSNR. Так за зменшення кількості точок порівняння і як наслідок часу розрахунку втричі, максимальна втрата становить 2.11dB, що складає 6,95% від результатів отриманих еталонною метрикою SAD, для шаблону HSAD (зменшення кількості і точок на 50%) – 1.53dB, або 5% від результату еталонної метрики для відеопослідовності Hall-Monitor(10) з мінімальним бітрейтом серед розглянутих відеопослідовностей. Також як можна побачити на прикладі відеопослідовностей Coastguard(48) і Coastguard(112) збільшення бітрейту призводить до незначних зменшень втрат PSNR 0.05dB для HSAD і 0.03dB для TSAD.

Розглянемо результати роботи запропонованих шаблонів на прикладі відепослідовностей з середнім (512-1024кбіт/с) бітрейтом, що дозволить нам підтвердити припущення про вплив бітрейту на величину втрати PSNR. При цьому все так же будемо використовувати блоки розміром 16х16 пікселів та відповідну область пошуку.

Таблиця 2.8 – Середнє значення PSNR для відеопослідовностей з середнім бітрейтом для обробки блоками 16x16пікселів

Відеопослідовність(кбіт/с)	SAD	HSAD	TSAD
Foreman(512)	34.51	33.05	32.45
Foreman(1024)	35.47	34.08	33.46
Table tennis(1024)	34.98	33.57	32.95

Як бачимо на прикладі відеопослідовності Foreman, ріст бітрейту призводить до незначного зменшення в втраті PSNR відносно еталонної метрики: Foreman(112) – 1.48dB, Foreman(512) – 1.45dB, Foreman(1024) – 1.39dB для шаблону HSAD і Foreman(112) – 2.09dB, Foreman(512) – 2.06dB, Foreman(1024) – 2.01dB для TSAD. Можна спрогнозувати, що подальше збільшення бітрейту, не дозволить зменшити втрати PSNR до того рівню, що б вони стали прийнятними для стиснення відеопослідовності, натомість є сенс розглянути збільшення розмірів блоків та області пошуку до 32x32 пікселя, що дозволить зробити вплив рисунку шаблону більш помітним (таблиця 2.9).

Таблиця 2.9 – Середнє значення PSNR для відеопослідовностей з малим та середнім бітрейтом для обробки блоками 32x32пікселі.

Відеопослідовність(кбіт/с)	SAD	HSAD	TSAD
Coastguard (112)	27.07	25.81	25.35
Foreman (112)	30.38	29.1	28.65
Foreman(512)	34.85	33.62	33.16
Foreman(1024)	35.56	34.39	33.91
Table tennis(1024)	35.01	33.78	33.34

Як бачимо з даних таблиці, збільшення розмірів блоку призвело до помітного зменшення в різниці PSNR запропонованих шаблонів та еталонної метрики. Так, для відеопослідовності Foreman(1024) різниця PSNR зменшилась з 1.39dB до 1.23dB (на 11.5%) для HSAD і з 2.01dB до 1.65dB(17,9%) для TSAD. Як бачимо, збільшення розмірів блоку до 32x32 пікселів, більш помітно вплинуло на PSNR шаблону TSAD що пояснюється більшим впливом унікальності розташування пікселів. Таким чином є сенс в подальшому збільшенні розмірів блоку та області пошуку до 64x64 пікселів(табл. 2.10).

Таблиця 2.10 – Середнє значення PSNR для відеопослідовностей з малим та середнім бітрейтом для обробки блоками 64x64пікселі.

Відеопослідовність(кбіт/с	SAD	HSAD	TSAD
Coastguard (112)	27.13	26.09	25.91
Foreman (112)	30.69	29.64	29.45
Foreman(512)	35.21	34.2	34.03
Foreman(1024)	35.62	34.65	34.47
Table tennis(1024)	35.15	34.16	33.98

З результатів наведених в таблиці 2.10 можна побачити, що подальше збільшення розмірів блоку дозволяє отримати помітний приріст PNSR. Варто відзначити, що при збільшенні розміру блоків зростає як середній PNSR еталонної метрики так і різниця цього значення відносно значень PNSR, отриманих на основі використання запропонованих шаблонів. Ще одним спостереженням є доволі сильне наближення значень PNSR шаблону TSAD до значень отриманих на основі шаблону HSAD. Для більш наочного порівняння помістимо ці значення у таблицю 2.11

Таблиця 2.11 – Різниця PSNR еталонної метрики і запропонованих шаблонів для різних розмірів блоку

Відеопослідовність(кбіт/с)	Шаблон порівняння	16x16	32x32	64x64
Coastguard (112)	HSAD	1.47	1.26	1.04
	TSAD	2.08	1.72	1.22
Foreman (112)	HSAD	1.48	1.28	1.05
	TSAD	2.09	1.73	1.24
Foreman(512)	HSAD	1.45	1.23	1.01
	TSAD	2.06	1.65	1.18
Foreman(1024)	HSAD	1.39	1.21	0.97
	TSAD	2.01	1.65	1.15
Table tennis(1024)	HSAD	1.41	1.23	0.99
	TSAD	2.03	1.67	1.17

Таким чином, якщо для блоків розміром 16x16 блоків для відеопослідовності Foreman, середня різниця в PSNR по всім бітрейтах для шаблонів TSAD і HSAD становила 0.61dB для блоків розміром 64x64 пікселі ця різниця зменшилась до 0.18dB(на 70,5% відносно початкової різниці). При цьому подальше збільшення блоків до 128x128 пікселів може призвести до значного впливу артефактів блочності, що може помітно зменшити значення PSNR(наприклад для відеопослідовності Foreman(1024) при компенсації блоками 128x128 середнє PSNR впало до 30.53dB). Тому в подальшому, для зменшення втрати PSNR слід розглянути використання додаткових алгоритмів.

Для повноти картини варто розглянути вплив використання шаблонів порівняння на відепослідовності з високим бітрейтом (4000-9000кбіт/с). Оскільки використання блоків розміром 64x64 дають найкращі результати, оцінку результатів будемо проводити на основі використання саме цих блоків (табл.2.12)

Таблиця 2.12 – Середнє значення PSNR для відеопослідовностей з високим бітрейтом (4000-9000 кбіт/с) для обробки блоками 64x64пікселів

Відеопослідовність(кбіт/с)	SAD	HSAD	TSAD
Basket (4000)	26.72	25.61	25.43
Basket (9000)	30.73	29.64	29.47
Bus(4000)	29.93	28.79	28.58
Bus(9000)	34.04	32.93	32.74

Як бачимо з таблиці, втрати PSNR для тестової відеопослідовності залишаються на досить високому рівні і відповідають втратам, що були отримані для нижчих бітрейтів(може бути наслідком більшої складності відеопослідовності), максимальна втрата PSNR складає 1.35dB для відеопослідовності Bus(4000). Так можна зробити висновок, що зменшення втрати PSNR як наслідок збільшеного бітрейту буте також сильно залежати від відеопослідовності і не буде сильно впливати на результати стиснення.

Останній тест проведемо на відеопослідовності «Hearthstone», яка по суті є ідеальним випадком відеопослідовності, оскільки записана з мінімальним впливом шумів. Результати обробки відеопослідовності «Hearthstone» алгоритмами на основі блоків розміром 16x16, 32x32 та 64x64 наведені в таблиці 2.13

Таблиця 2.13 – Середнє значення PSNR для відеопослідовності «Hearthstone» з бітрейтом 24822 кбіт/с.

Розмір блоку	Показник	SAD	HSAD	TSAD
16x16	PSNR	37.43	35.91	35.26
32x32	PSNR	37.97	36.66	36.17
64x64	PSNR	38.31	37.2	37.02

Як бачимо з даних наведених в таблиці 2.13, для відеопослідовності «Hearthstone» з бітрейтом 24822кбіт/с різниця в PSNR для еталонного алгоритму та запропонованих шаблонів зросла до максимальних 2.17dB, що є доволі дивною поведінкою з урахуванням зменшення кількості шумів і пов’язана з двома особливостями роботи шаблонів порівняння:

1. Для кадрів з великими областями низької ентропії, сильнішим стає вплив випадкових співпадінь в значеннях пікселів.
2. Для кадрів з великою кількістю областей великої ентропії(великою кількістю об’єктів), можуть втрачатися вектори кандидати, за рахунок чого відбуватися втрата «істинних» векторів руху.

Висновки до розділу

В даному розділі було проведено аналіз основних метрик оцінки схожості зображень, серед яких SAD (сума абсолютних різниць) – використовується найчастіше, завдяки своїй простоті та невеликій вичислювальній складності. Не дивлячись на це, процес оцінки схожості зображень може займати від 40 до 80% часу стиснення відеопотоку. Для зменшення часу на обрахунок метрики SAD було запропоновано використання трьох шаблонів порівняння на основі використання характерних точок HSAD (кожен другий), TSAD (кожен третій) і DSAD(діагональний), які дозволили зменшити кількість точок порівняння та часу обрахунку на 50, 66.4 і 65.6% відповідно.

Використання запропонованих шаблонів було протестоване трьома способами:

1. Знаходження однакових блоків для матриці з випадково згенерованими даними, що покриває випадок найгіршого розподілу значень пікселів. Такий підхід дозволив виключити з подальшого розгляду шаблон DSAD, який при майже однаковому зменшенні кількості точок порівняння відносно TSAD показав значно гірші

результати в плані співпадіння векторів руху зі знайденими еталонною метрикою SAD.

2. Знаходження векторів руху для тестової відеопослідовності «Hearthstone» з низьким рівнем шуму, що дозволило провести також візуальну оцінку результатів використання шаблонів порівняння.
3. Тестування за рекомендаціями міжнародної групи MPEG для оцінки складних відеоінструментів. Тестування було проведено на 13 різних відеопослідовностях для різних бітрейтів (10 - 24822 кбіт/с). Обробка відеопослідовностей проводилась блоками трьох розмірів(16x16, 32x32, 64x64) за допомогою алгоритму повного перебору в областях що відповідають розмірам блоку. Так для блоків розміром 16x16 пікселів максимальна втрата PSNR для шаблонів HSAD склала – 1.49dB, TSAD – 2.11dB, для блоків розміром 32x32, HSAD склала – 1.26dB, TSAD – 1.71dB, і для блоків 64x64 - HSAD склала – 1.03dB, TSAD – 1.27dB, що є доволі високими втратами.

Такі високі втрати PSNR викликані наступними особливостями використання шаблонів порівняння.

1. Для областей з низькою ентропією зображення, використання шаблонів порівняння без умов ранньої зупинки пошуку призводить до зростання кількості блоків кандидатів (при формуванні набору з блоків кандидатів для яких значення метрики нижче порогової), а також знаходження хибних метрик за рахунок співпадіння піксельних значень.
2. Для областей з високою ентропією, використання шаблонів порівняння призводить до втрати блоків кандидатів і як результат артефактів на границях об’єктів та областях різкого переходу кольору.

Для усунення описаних вище недоліків пропонується використати дві допоміжні технології:

1. Алгоритм виділення границь об’єктів для усунення неоднозначностей та уникнення втрат в областях з великою ентропією.
2. Метод блочного пошуку, з умовами ранньої зупинки та використанням наборів предикатів, що дозволить виключити з розгляду блоки з нульовими векторами руху, а також уникнути втрат в областях з високою ентропією.

3 ВИКОРИСТАННЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ГРАНИЦЬ ОБ'ЄКТІВ ДЛЯ УСУНЕННЯ НЕОДНОЗНАЧНОСТЕЙ, ВИКЛИКАНИХ ВИКОРИСТАННЯМ ШАБЛОНІВ ПОРІВНЯНЬ

3.1 Алгоритми розпізнавання границь об'єктів.

За рахунок того, що зміна освітлення і кольору не сильно впливають на краї зображення, пошук країв проводиться на зображенні в відтінках сірого, що в варіанті подання відео-потоку в форматі YUV є яскравісною компонентою Y.

Ідея застосування алгоритмів пошуку країв зображення полягає в наступному: після створення масиву блоків кандидатів за допомогою TSAD відкидаються ті, в яких сума абсолютних різниць нижче ніж показник ΔS який обиратиметься виходячи з бажаних параметрів якості зображення, відразу вважається правильними кандидатами і уточнення для них проводиться одним з стандартних методів наприклад SDD або RDO. Далі скомпенсовані області видаляються з набору даних для подальшого розрахунку і проводимо виділення країв за допомогою одного з алгоритмів які будуть описані далі в розділі. Після цього, для блоків, різниця яких є більшою за обраний поріг валідності, вираховуємо SAD між блоками кандидатами та блоком над яким проводиться пошук виключно по отриманій карті країв об'єктів, обираючи той, для якого різниця SAD буде найменшою. Розглянемо методи виділення границь об'єктів.

3.1.1 Оператор Собеля

Оператор Собеля - це один кращих алгоритмів виділення кордонів, який використовується в області обробки зображень[45]. Це дискретний диференціальний оператор, який обчислює наближене значення градієнта яскравості зображення. Результатом застосування оператора Собеля в кожній

точці зображення є або вектор градієнта яскравості в цій точці, або його норма. Іншими словами, оператор обчислює градієнт яскравості зображення в кожній точці. Так знаходиться напрямок найбільшої зміни яскравості і величина її зміни в цьому напрямку. Результат показує, наскільки «різко» або «плавно» змінюється яскравість зображення в кожній точці, а значить, ймовірність знаходження точки на межі, а також орієнтацію кордону. На практиці обчислення величини зміни яскравості (ймовірності приналежності до межі) надійніше і простіше в інтерпретації, ніж розрахунок напрямку [46].

Математично градієнт функції двох змінних для кожної точки зображення (якою і є функція яскравості) - двовимірний вектор, компонентами якого є похідні яскравості зображення по горизонталі і вертикалі. У кожній точці зображення градієнтний вектор орієнтований в напрямку найбільшого збільшення яскравості, а його довжина відповідає величині зміни яскравості. Це означає, що результатом оператора Собеля в точці області постійної яскравості буде нульовий вектор, а в точці, що лежить на кордоні областей різної яскравості - вектор, що перетинає кордон в напрямку збільшення яскравості. Процес сегментації з використанням оператора Собеля заснований на простому переміщенні маски фільтра від точки до точки зображення: в кожній точці *(x, y)* відгук фільтра обчислюється з використанням попередньо заданих зв'язків. В разі лінійної просторової фільтрації відгук задається сумою добутку коефіцієнтів фільтра на відповідні значення пікселів в області, покритої маскою фільтра. Для маски 3x3 елемента, показаної на рис. 3.1, результат (відгук) *R* лінійної фільтрації в точці *(x, y)* зображення складе:

$$R = (w-1,-1)\,f(x-1,y-1) + \ldots + w(0,0)\,f(x,y) + w(1,1)\,f(x+1,y+1) \qquad (3.1)$$

що, по суті є сумою добутків коефіцієнтів маски на значення пікселів безпосередньо під маскою.

Варто зауважити, що коефіцієнт *w(0,0)* буде відповідати точці з значенням *f(x,y)*, вказуючи тим самим, що маска центрована в точці *(x,y)*. При

виявленні перепадів яскравості використовуються дискретні аналоги похідних першого і другого порядку.

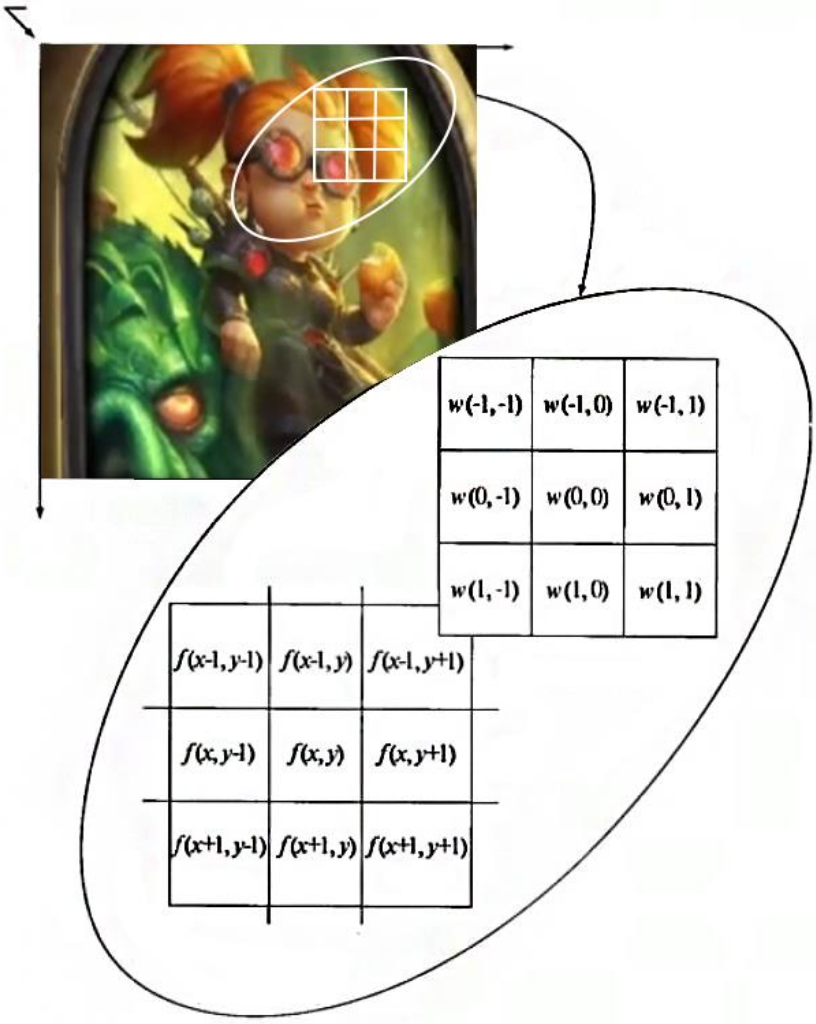


Рисунок 3.1 – Схема просторової фільтрації

Перша похідна одновимірної функції *f(x)* визначається як різниця значень сусідніх елементів:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \qquad (3.2)$$

Тут використаний запис у вигляді похідної для того, щоб зберегти ті ж позначення в разі двох змінних *f(x,y)*, де доведеться мати справу з частковими

похідними по двом просторовим осям. Аналогічно, друга похідна визначається як різниця сусідніх значень першої похідної:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{3.3}$$

Обчислення першої похідної цифрового зображення засновано на різних дискретних наближеннях двомірного порядку.

По визначенню, градієнт зображення *f(x,y)*, в точці *(x,y)* – це вектор:

$$f(x,y) = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.4}$$

Напрямок вектору градієнта збігається з напрямком максимальної швидкості зміни функції *f* в точці *(x, y)*[47]. Важливу роль при виявленні контурів відіграє модуль цього вектору, який позначається ∇f і дорівнює:

$$\nabla f = |\nabla f| = \sqrt{G_x^2 + G_y^2} \tag{3.5}$$

Ця величина дорівнює значенню максимальної швидкості зміни функції *f* в точці *(x,y)*. Напрямок вектору градієнта також є важливою характеристикою. позначимо *α(x,y)* кут між напрямком вектору ∇f в точці *(x,y)* і віссю x:

$$\alpha(x,y) = artcg(\frac{Gy}{Gx}) \tag{3.6}$$

Звідси легко знайти напрям контуру в точці *(x,y)*, який перпендикулярний напрямку вектору градієнта в цій точці. А обчислити градієнт зображення можна, обчисливши величини часткових похідних *∂f/∂x* і *∂f/∂y* для кожної точки.

Gx і *Gy* - це дві матриці, де кожна точка містить наближені похідні по *x* і *y*. Вони обчислюються шляхом множення матриці *Gx* і *Gy* і підсумовуванням обох матриць, в результаті отриманий результат записується в поточні координати *x* і *y* в нове зображення:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.7}$$

Матриці *Gx* і *Gy:*

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \times A \text{ і } Gx = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \times A \qquad (3.8)$$

Результатом застосування оператора Собеля є двовимірна карта градієнта для кожної точки [48]. Її можна обробити і показати як зображення, на якій дільниці з великою величиною градієнта (в основному, межі) буде видно як білі лінії. Приклад роботи оператору на першому кадрі оброблюваної відео-послідовності представлено на рис. 3.2

Рисунок 3.2 – Приклад роботи оператора Собеля на першому кадрі відеопослідовності

3.1.2 Метод виділення границь Canny

Одним з найбільш часто реалізованих програмно методом є Canny[49]. Фактично це набір послідовно застосовуваних алгоритмів. Такий підхід стійкий до шуму і дає, як правило, кращі результати в порівнянні з іншими

методами. Але так як це набір алгоритмів, то і швидкодія даного методу поступається більш простим операторам.

Алгоритм Canny складається з чотирьох етапів:

1. Розмиття зображення (зменшується дисперсія адитивного шуму на зображенні).
2. Диференціювання розмитого зображення і обчислення значень градієнта в напрямку *x* і напрямку *y*.
3. Принцип подавлення «НЕ максимум».
4. Порогова обробка.

На першому етапі алгоритму Canny відбувається згладжування зображення за допомогою маски фільтром Гауса.

Рівняння розподілу Гауса в N вимірах має вигляд:

$$G(r) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-r^2/(2\sigma^2)} \qquad (3.9)$$

або в часному випадку для двух вимірів:

$$G(r) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-(u^2-v^2)/(2\sigma^2)} \qquad (3.10)$$

де r – радіус розмиття, $r^2 = u^2+v^2$, σ - стандартне відхилення розподілу Гаусса.

У разі двох вимірювань ця формула задає поверхню, що має вигляд концентричних кіл з розподілом Гаусса від центральної точки. Пікселі, де розподіл відмінний від нуля, використовуються для побудови матриці згортки, яка застосовується до вихідного зображення. При цьому значення кожного пікселя стає середньозваженими для околиці. Початкове значення пікселя приймає найбільшу вагу (має найвище Гаусове значення), і сусідні пікселі приймають менші ваги в залежності від відстані до них. В теорії, розподілення у кожній точці зображення буде ненульовим, що вимагало б

обчислення вагових коефіцієнтів для кожного пікселя зображення. Але, на практиці, коли розраховується дискретне наближення функції Гаусса, не враховують пікселі на відстані понад 3σ, тому що вони досить малі. Таким чином, програмі, що фільтрує зображення, досить розрахувати матрицю [6σ] × [6σ], щоб гарантувати достатню точність наближення розподілу Гаусса[50].

Далі відбувається пошук градієнта області за допомогою згортки згладженого зображення з похідною від функції Гауса як у вертикальному, так і в горизонтальному напрямках. Використовуємо оператор Собеля для вирішення цього завдання. Контроль здійснюється шляхом простого переміщенні маски фільтра від точки до точки на зображенні. У кожній точці *(x,y)* відгук фільтра обчислюється з використанням попередньо заданих зв'язків.

Для даного кроку використовуємо наступні матриці:

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \qquad K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}; \tag{3.11}$$

Gx і *Gy* - дві матриці, де кожна точка містить наближені похідні по x і по y. Вони обчислюються шляхом множення матриці Gx і Gy і підсумовуванням обох матриць, в результаті отриманий результат записується в поточні координати x і y в нове зображення:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.12}$$

Використовуючи цю інформацію, також можна вирахувати напрямок градієнту:

$$\theta = artcg(\frac{|Gy|}{|Gx|}) \tag{3.13}$$

В результаті відбувається початкове виділення границь об’єктів.

Наступним кроком відбувається порівняння кожного пікселя з сусідніми вздовж напрямку градієнта і обчислюється локальний максимум. Інформація про направлення градієнта необхідна для того, щоб видаляти

пікселі поруч з кордоном, не розриваючи сам кордон поблизу локальних максимумів градієнта: пікселями кордонів оголошуються точки, в яких досягається локальний максимум градієнта в напрямку вектору градієнта. Значенням напрямку повинен бути кратний 45° (рис 2.9).

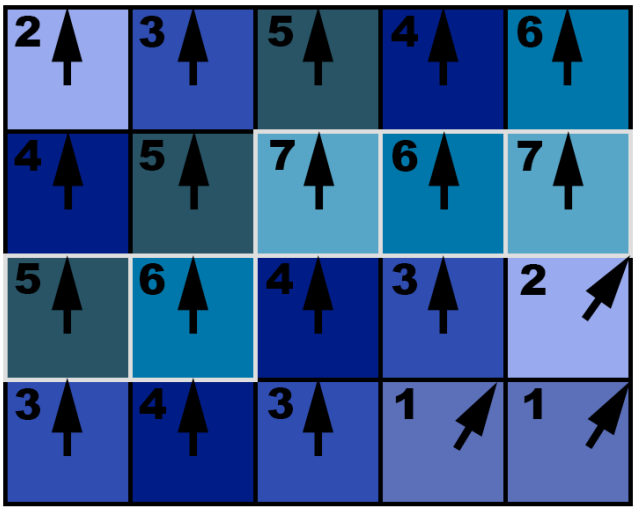


Рисунок 3.3 – Принцип подавлення «не максимум»

Майже всі пікселі в цьому прикладі мають орієнтацію «вгору», тому значення градієнта в цих точках буде зрівняно з нижче і вище-розташованими пікселями. Обведені білим контуром пікселі залишаться в результуючому зображенні, інші - будуть пригнічені. Такий підхід дозволяє істотно знизити виявлення неправдивих країв і забезпечує товщину кордону об'єкта шаром в один піксель [51].

Наступний крок - застосування порога для визначення знаходження кордону в даній точці зображення. Чим менше поріг, тим більше меж буде знаходитися, але тим більше сприйнятливим до шуму стане результат, виділяючи зайві дані зображення. І навпаки, високий поріг може проігнорувати слабкі краї або отримати кордон фрагментами. Виділення меж використовує два порога фільтрації: якщо значення пікселя вище верхньої межі - він приймає максимальне значення (межа вважається достовірною), якщо нижче - піксель пригнічується, точки зі значенням, що потрапляють в

діапазон між порогами, приймають фіксоване середнє значення. В результаті задача зводиться до виділення груп пікселів, які отримали на попередньому етапі проміжне значення, і віднесенню їх до кордону (якщо вони пов'язані з однією з встановлених меж) або їх придушення (в іншому випадку). Піксель додається до групи, якщо він стикається з нею по одному з 8-ми напрямків. Фінальний результат роботи алгоритму над першим кадром відео-послідовності представлено на рис. 3.4.

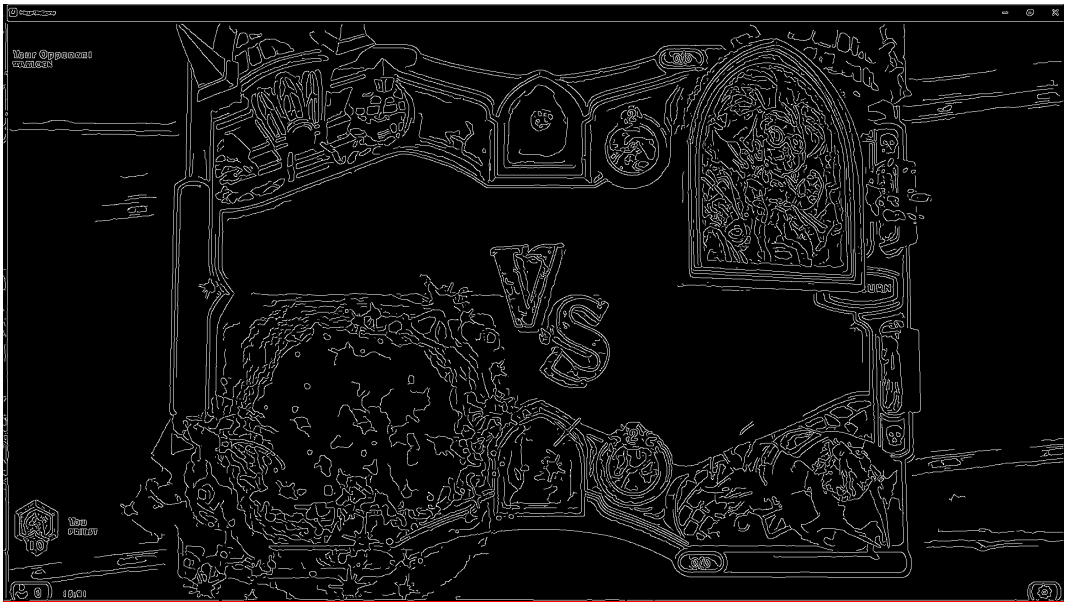


Рисунок 3.4 – Результат обробки першого кадру відео-послідовності алгоритмом Canny.

3.1.3 Фільтр лапласіана-гаусіана [52]

Лапласіан 1-го і 2-го порядку - метод орієнтований на підвищення різкості графічних даних. Головна мета підвищення різкості полягає в тому, щоб підкреслити дрібні деталі або поліпшити ті деталі, які виявилися розфокусуваними внаслідок помилок або недосконалості методів попередньої обробки. Метод заснований на використанні першої або другої похідних і, отже, перша похідна повинна бути:

1. Рівною нулю на плоских ділянках (областях з постійним рівнем яскравості).
2. Ненульовою на початку і в кінці сходинки або схилу яскравості.
3. Ненульова на схилах яскравості.

Аналогічно друга похідна повинна бути:

1. Рівною нулю на плоских ділянках.
2. Ненульовою на початку і в кінці сходинки або схилу яскравості.
3. Рівною нулю на схилах постійної крутизни.

Якщо порівняти похідні 1-го і 2-го порядку, можна отримати наступні висновки:

1. Перша похідна зазвичай дає в результаті більш товсті контури;
2. Друга похідна дає більший за величиною відгук на дрібні деталі - як на окремих точках, так і на тонких лініях;.
3. Відгук на сходинку у першій похідній, як правило, вище, ніж у другій
4. На похилих контурах друга похідна дає подвійний відгук.

При поліпшенні якості кадрів відеопослідовності краще використовувати похідну другого порядку, так як вона має більше посилення дрібних деталей. Найпростішим ізотропним оператором, заснованим на похідних, є Лапласіан (оператор Лапласа), який для двох змінних визначається наступним чином[53]:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} \tag{3.14}$$

Оскільки похідні будь-якого порядку є лінійними операторами, то Лапласіан є лінійним оператором. Дискретна формула двовимірного Лапласіана, заданого рівнянням (14), виходить об'єднанням часткових похідних другого порядку щодо змінних *x і y*:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y) \tag{3.15}$$

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y) \qquad (3.16)$$

звідки можемо отримати формулу для лапласіану:

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 2f(x, y) \qquad (3.17)$$

Це рівняння може бути реалізовано за допомогою маски, представленої на рис. 3.5(а), і формуюче ізотропний результат для поворотів на кути, кратні 90°. Діагональні напрямки враховуються у формулі дискретного Лапласіана додаванням двох членів, тобто по одному для кожного з діагональних напрямків. Оскільки кожна діагональ додає член *-2f(x,y)*, то сумарний член різниці дорівнює - 8 f (x, y). Ізотропна маска для поворотів на кути кратні 45° показана на рис 3.5 б)

0	1	0
1	-4	1
0	1	0

(a)

1	1	1
1	-8	1
1	1	1

(b)

Рисунок 3.5 - Маски Лапласіана для поворотів на кути кратні а) 90°, б)45°

Оскільки оператор Лапласа по суті є другою похідною, його застосування підкреслює розриви рівнів яскравості на зображенні і пригнічує області зі слабкими змінами яскравості. Це призводить до отримання зображення, що містить сіруваті лінії на місці контурів і інших розривів, накладені на темний фон без особливостей. Однак фон можна «відновити», зберігши при цьому ефект підвищення різкості, що досягається лапласіаном. Для цього достатньо скласти вихідне зображення і лапласіан. І для отримання

остаточного результату лапласіан-зображення віднімається з вихідного зображення [54].

Таким чином, узагальнений алгоритм використання Лапласіан для поліпшення зображень зводиться до вирішення наступних рівнянь:

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y), якщо & w(0,0)<0 \\ f(x,y)-\nabla^2 f(x,y), якщо & w(0,0)\geq 0 \end{cases} \quad (3.18)$$

де $w(0,0)$ - значення коефіцієнтів маски лапласіана.

Отже, описаний метод має доволі високу швидкодію і відносно невелику складність обчислення. Крім того, для подібних методів легко можна застосувати розпаралелювання процесів обчислення. Висока швидкодія, простота обчислень, розпаралелювання процесів дає велику перевагу в застосуванні цих методів для обробки відеопотоку. Основними деталями лапласіан-зображення є контури і різкі перепади яскравості різного рівня. Деталі на результуючому зображенні видно значно чистішими і різкими, ніж на оригінальному документі. Об'єднання вихідного зображення і лапласіан відновило загальний діапазон зміни яскравостей на зображенні, а лапласіан посилив контрасти в ділянках розривів яскравості. Таким чином, дрібні деталі поліпшені, а фонові півтони збережені. Перш ніж діяти на зображення лапласіаном, зображення необхідно згладити. лапласіан - це лінійний оператор, тобто дія лапласіан рівносильно згортці зображення з певним ядром (яке позначають $2\nabla K$). З асоціативності згортки випливає, що:

$$K_{\nabla^2} **(G_\sigma **I)=(K_{\nabla^2} * G_\sigma) **I=(\nabla^2 G_\sigma) **I \quad (3.19)$$

Це важливо, оскільки, як і для першої похідної, згладжування з подальшим застосуванням лапласіана рівнозначно згортці зображення з лапласіаном ядра, яке застосовувалося при згладжуванні (рис. 3.6).

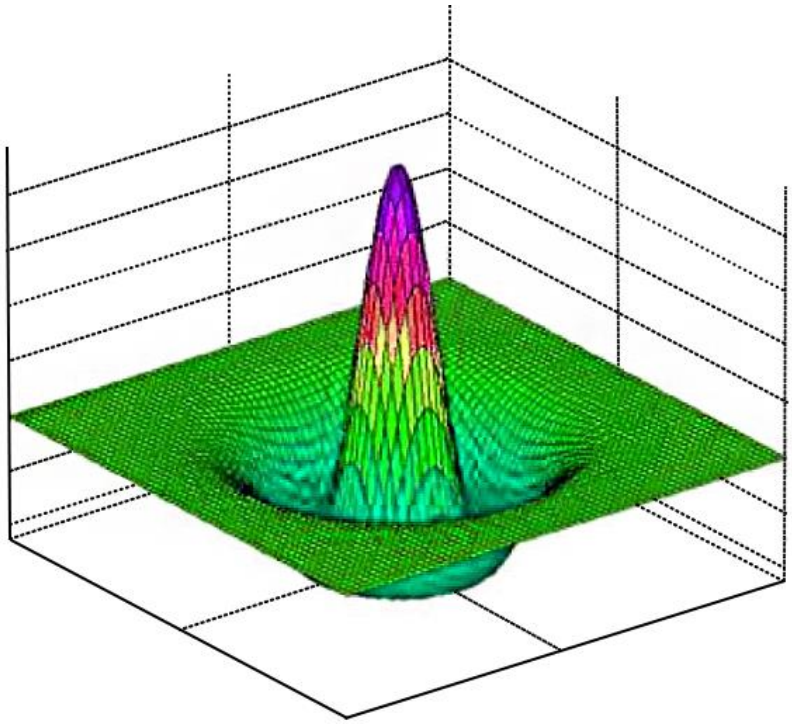


Рисунок 3.6 – Лапласіан-гаусового ядра фільтра

Лапласіан-гаусового ядра фільтра, показаний на малюнку для σ в один піксель, можна вважати результатом вирахування центрального пікселя з зваженого значення оточення. Досить часто це ядро замінюють різницею двох гаусіанів: один з маленьким значенням σ, інший - з великим значенням σ. Оператори Гаусса і Лапласа лінійні, а це означає, що вони комутативні, і їх можна об'єднати в один, змінивши порядок виконання операцій [22]:

$$L*(G*I) = (L*G)*I = LoG*I \qquad (3.20)$$

Тут x - оператор згортки. Замість того, щоб спочатку згладити зображення, а потім взяти лапсасіан від результуючого зображення, спочатку виконується згортка - обох операторів, а потім отриманий оператор застосовується до зображення. Згортка операторів виконується один раз на самому початку, і зображення обробляється один раз. При послідовному згладжуванні і диференціюванні зображення, воно обробляється двічі. Отриманий таким чином оператор отримав назву лапласіан-гаусіана (LoG -

Laplacian of Gaussian): Двовимірний оператор Лапласа було визначено раніше, а двовимірний оператор Гаусса має наступний вигляд:

$$G = \exp\left[-\frac{x^2 + y^2}{2t^2}\right] \qquad (3.21)$$

Формула згортки може бути представлена в матричному вигляді, з якої необхідно згортати зображення. Після цього відзначаються точки, в яких функція дорівнює нулю - переходи через нуль. Далі слід перевірити, чи дійсно в цих точках градієнт досить великий. Реакція лапласіан-гаусового фільтра позитивна з одного боку краю і негативна з іншого. Це означає, що додаток деякої частини цієї реакції до вихідного зображення дає картинку, на якій краї чіткіше, а деталі побачити набагато легше. Визначення країв за допомогою лапласіан-гауссіана не використовується досить часто, оскільки лапласіан-гаусовий фільтр не орієнтований, його характеристика з перпендикулярно орієнтованої, паралельної до краю складової. Це означає, що на кутах, де змінюється напрямок паралельної складової, виникають деякі проблеми. Межі гострих кутів позначаються дуже неточно. У тригранних і більш складних кутах це створює великі труднощі для правильного запису топології кутів. Так як паралельні компоненти фільтра схильні реагувати на шум, а не тільки на краї, це означає, що перехід через нуль не обов'язково точно вказує на край [55].

Переходи через нуль Лапласіан-Гауссових даних можуть незвично вести себе в кутах. По-перше, при прямому куті перехід через нуль випинається(але проходить через вершину). Це явище виникає не через оцифрування (або квантування), воно також зустрічається і в безперервному випадку. У кутах, де сходяться три або більше лінії, контури поводяться незвично, деталі залежать від структури алгоритму позначення контуру - в результаті дій цього алгоритму виникають артефакти викривлення країв. Приклад роботи алгоритму представлений на рис.3.7.

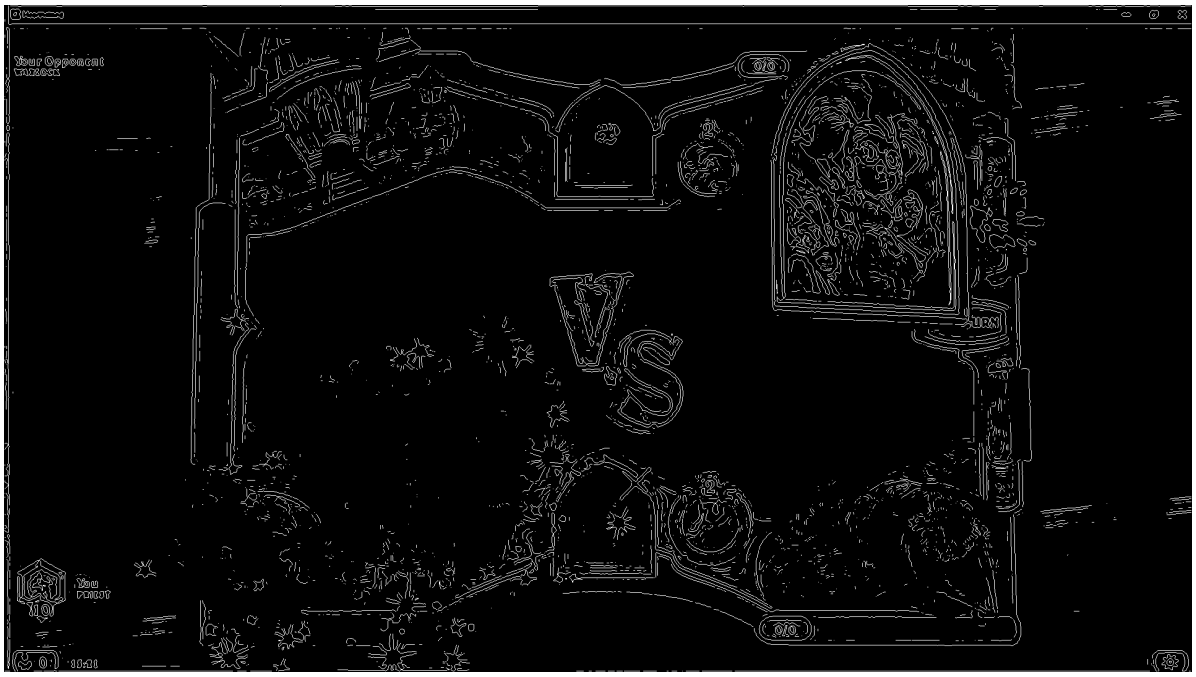


Рисунок 3.7 – Приклад роботи лапласіана-гаусіана на першому кадрі відео-послідовності

3.1.4 Вибір методу виділення країв

При виборі методу вибору країв зазвичай керуються наступними факторами:

1. Відносним рівнем шуму зображення.
2. Бажаною точністю отриманого виділення країв.
3. Швидкістю роботи алгоритму.

Варто відзначити, що в нашому випадку шум на зображенні буде відносно незначним, оскільки виділення країв проводиться після першого з трьох етапів обробки зображення на яких виникають шуми:

1. Етап оцифровки зображення, на якому виникають шуми, викликані зовнішніми впливами (електромагнітними, тепловими) на сенсори і аналого-цифрові перетворювачі (АЦП) системи реєстрації.
2. Етап кодування, на якому шуми, викликаються квантуванням і стисненням даних з втратами;

3. Етап передачі, шуми викликані частковим спотворенням або втратою даних в результаті дії перешкод.

Таким чином можна розглядати методи, які сильно залежні від шуму зображення, такі як наприклад LoG, що має достатньо слабку стійкість до шуму.

Попередню оцінку бажаної точності алгоритму виділення країв можемо провести порівнявши результати роботи описаних вище алгоритмів на прикладі тестового зображення (рисунки 3.2, 3.4, 3.7). Як бачимо оператор Собеля дає далеко не найкращий результат на тестовому зображенні, зовсім не покриваючи проблемні області. Також варто відзначити, що оператор Canny дає доволі велику надлишковість в плані виділення країв, що може суттєво вплинути на оцінку обраної метрики.

3.1.5 Оцінка дії обраних алгоритмів виділення країв на тестовому зображенні.

Проводити виділення країв на наборі випадкових даних немає жодного сенсу, оскільки велика ентропія значень в комірках зробить роботу алгоритмів виділення країв абсолютно неефективною (різкий перехід між значенням буде вважатися за краї), тому перейдемо відразу до обробки кадрів відео-потоку. Так як і раніше, проводимо обробку першого та другого кадрів відео послідовності методами HSAD та TSAD, після чого з сформованих масивів векторів кандидатів відкидаємо ті, значення метрики яких близьке до 0 (статичні блоки) і зануляємо їх значення в масиві зображення, що б скоротити час виконання операцій виділення краю (виділяємо краї тільки для блоків у яких є невизначеність з векторами кандидатами). Після цього використовуємо обрані методи для виділення країв на зображеннях, але лише для тих блоків, для яких в є хоча б один піксель значення якого не дорівнює 0 (таким чином не враховуються ті блоки, які лежать не на краях, виключаючи похибку порівняння двох

нульових блоків), та проводимо для цих блоків визначення остаточного кандидату алгоритмом SAD, після чого отримуємо нову кількість блоків кандидатів. Варто відзначити, що використання SAD в якості метрики на даному етапі є обґрунтованим низькою ентропією отриманих карт країв(на них можливі 2 значення 0 – чорні пікселі, 1 – білі пікселі країв).

Таблиця 3.1 – Кількість векторів кандидатів після оцінки зображення з виділеними краями

	Середня кількість кандидатів		Максимальна кількість блоків кандидатів	
	Canny	LoG	Canny	LoG
HSAD	1.131	1.08	2	2
TSAD	1.107	1.093	2	2

Як видно з отриманих значень було отримано суттєве зниження кількості невизначених блоків, після чого можемо визначити решту блоків за допомогою все того ж SSD+PSNR. Варто відзначити, що після остаточної оцінки блоків кандидатів через SAD та карту країв у нас залишилось 13.1% невизначених краєвих блоків для HSAD і 10.7% невизначених блоків для TSAD.

Варто відзначити, що при збільшенні розміру блоків до 32x32 і 64x64 пікселів кількість невизначених блоків на етапі обробки стартового зображення за допомогою TSAD і HSAD як і область для виділення країв зменшується, за рахунок програшу в кількості операцій для повторної оцінки SAD невизначених блоків. В результаті маємо картину розподілу векторів кандидатів, після операції виділення країв та оцінки через SAD для послідовності розбитої на 32x32 і 64x64 піксельні блоки представлену в таблиці 3.2.

Таблиця 3.2 – Кількість векторів кандидатів після оцінки зображення з виділеними краями для блоків 32x32 і 64x64

	Середня кількість кандидатів		Максимальна кількість блоків кандидатів	
	Canny	LoG	Canny	LoG
HSAD 32	1.0463	1.0378	2	2
TSAD 32	1.0392	1.0283	2	2
HSAD 64	1.0133	1	2	1
TSAD 64	1.0095	1	2	1

Як показано в таблиці 3.2, кількість невизначених блоків кандидатів для блоків розміру 64x64, після роботи з виділенням країв та алгоритмом LoG стала рівною як для HSAD так і для TSAD. При цьому звичайно не варто забувати що ці алгоритми не дають 100% перекриття алгоритму SAD, що призводить до спотворення зображення.

Що ж до результатів роботи запропонованих алгоритмів, можна зробити наступний висновок, TSAD при меншій кількості контрольних точок показує кращі результати як в плані похибки суми абсолютних різниць оціночних блоків, результуючої кількості невизначених блоків так і в часі витраченому на операції алгоритму, але дещо програє в перекритті блоків кандидатів отриманих від SAD, що тим не менш, не призведе до критичної зміни якості зображення, тому для візуальних оцінок роботи отриманого алгоритму буде взято TSAD з розміром 16x16 і 64x64 блоки.

Для оцінки візуальних змін в зображені, візьмемо абсолютну нескомпенсовану різницю між першим та другим кадром тестової відео-послідовності кодованим/декодованим на основі класичного SAD та аналогічну різницю для кадру кодованим/декодованим TSAD16 і TSAD64, а також понизимо контрастність отриманих різниць для більш наглядної візуалізації (рис 3. 8).

(a)

(b)

Рисунок 3.8 – Різниця абсолютних нескомпенсованих різниць для яскравісної компоненти кадрів, отриманих через кодування/декодування на основі SAD і (a) TSAD16, (b) TSAD64

Як можна побачити з рис.3.8, в нас все ж залишились деякі проблеми на краях зображення, але ця різниця є мінімальною і малопомітною навіть при послідовному розгляданні цих кадрів.

3.2 Тестування запропонованих алгоритмів згідно стандартів MPEG-4

Запропонований алгоритм оцінки схожості зображень було вбудовано в програмне забезпечення кодера MPEG-4 і протестовано з використанням комплексних умов тестування, запропонованих Спеціальною групою MPEG з оцінки складності відеоінструментів [28] на прикладі 11 відеопослідовностей.

Пошук блоків буде проводитись з використанням алгоритму повного перебору. При цьому пришвидшення дії алгоритму буде відраховуватись від часу роботи повного перебору для SAD, що б виключити вплив програмної реалізації та характеристик комп'ютерної техніки.

Для початку проведемо тестування запропонованих алгоритмів на тестових послідовностях з малим бітрейтом (10-112 біт/с), для блоків розміром 16x16 пікселів та областю пошуку ±16пікселів в усі сторони(таблиця 3.3).

Таблиця 3.3 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим бітрейтом

Показник	SAD	HSADL	HSADC	TSADL	TSADC
Відеопослідовність Hall-Monitor(10).					
PSNR	30.36	29.61	29.37	29.28	29.04
Seed-Up	1	1.28	1.02	1.98	1.53
Відеопослідовність Mother & Daughter(24)					
PSNR	34.81	33.05	32.81	33.72	33.47
Speed-Up	1	1.29	1.03	1.99	1.53
Відеопослідовність Coastguard (48)					
PSNR	28.89	28.2	27.95	27.87	27.65
Speed-Up	1	1.31	1.04	2.01	1.55

Продовження таблиці 3.4 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим бітрейтом

Показник	SAD	HSADL	HSADC	TSADL	TSADC
Відеопослідовність Foreman (112)					
PSNR	30.04	29.34	29.1	29.01	28.85
Speed-Up	1	1.34	1.06	2.06	1,57
Відеопослідовність Coastguard (112)					
PSNR	27.04	26.35	26.16	26.03	25.87
Speed-Up	1	1.31	1.04	2.04	1.55

Як можна побачити з даних таблиці, алгоритм виділення країв LoG показує кращі результати як в плані швидкості виконання для послідовності Foreman(112) HSADL – 1.34, + TSADL – 2.06, canny + HSADC – 1.06, +TSADC – 1.57, так і в плані отриманого приросту PSNR, в середньому алгоритми що використовують LoG показали результати на 9.8% кращі ніж ті, що показали алгоритми з використанням canny, що пояснюється надлишковістю виділення країв методом canny. Варто відзначити, що використання алгоритмів виділення границь об’єктів зменшило загальний виграш в прискоренні з 3 до максимальних 2.06 для TSAD і з 2 до максимальних 1.34 для HSAD, але при цьому отримуєму помітний приріст в значеннях PSNR, так, наприклад, для відеопослідовності Foreman(112) для шаблону HSAD втрата PSNR зменшилась з 1.48dB до 0.7dB, що трохи більше ніж в два рази, для шаблону TSAD 2.09dB до 1.03dB.

Перейдемо до розгляду результатів роботи впливу використання алгоритмів виділення країв на кодування кадрів, на прикладі тестових відеопослідовностей з середнім бітрейтом (512-1024 кбіт/с) з розміром блоку та області пошуку 16x16пікселів (табл 3.4).

Таблиця 3.4 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім бітрейтом

Показник	SAD	HSADL	HSADC	TSADL	TSADC
Відеопослідовність Foreman (512)					
PSNR	34.51	33.83	33.6	33.51	33.35
Speed-Up	1	1.36	1.09	2.09	1.63
Відеопослідовність Foreman (1024)					
PSNR	35.47	34.82	34.58	35.48	34.33
Speed-Up	1	1.38	1.11	2.11	1.67
Відеопослідовність Table tennis (1024)					
PSNR	34.98	34.32	34.07	33.99	33.83
Speed-Up	1	1.36	1.09	2.09	1.64

Як бачимо з таблиці на прикладі відеопослідовності Foreman при зростанні бітрейту, росте й максимальне прискорення від алгоритму з використанням виділення країв з 2.06 для 112 біт/с до 2.11 для 1024 біт/с, це пояснюється більш ефективною роботою алгоритму при зменшенні шумів як на стороні виділення границь так і в загальній оцінці зображення. Так само при збільшенні бітрейту, зменшується спотворення PSNR з 0.7dB для 112 біт/с до 0.66dB для 1024 біт/с.

Перевіримо як на роботу запропонованих алгоритмів вплине збільшення розміру блоків і області пошуку до 32х32пікселів, для відеопослідовностей з середнім та низьким бітрейтом (112-1024біт/с)(табл.2.11).

Таблиця 3.5 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім та низьким бітрейтом для області пошуку 32x32 пікселя

Показник	SAD	HSADL	HSADC	TSADL	TSADC
Відеопослідовність Coastguard (112)					
PSNR	27.04	26.46	26.35	26.23	26.12
Speed-Up	1	1.32	1.03	2.05	1.54
Відеопослідовність Foreman (112)					
PSNR	30.38	29.79	29.68	29.56	29.45
Speed-Up	1	1.35	1.06	2.07	1,56
Відеопослідовність Foreman (512)					
PSNR	34.85	34.31	34.19	34.07	33.96
Speed-Up	1	1.37	1.09	2.08	1.64
Відеопослідовність Foreman (1024)					
PSNR	35.56	35.02	34.93	34.83	35.69
Speed-Up	1	1.39	1.11	2.11	1.67
Відеопослідовність Table tennis (1024)					
PSNR	35.01	34.48	34.35	34.28	34.11
Speed-Up	1	1.35	1.09	2.09	1.65

Як бачимо, збільшення розміру блоку і області пошуку мало впливає на прискорення виконання операції і доволі помітно на значення PSNR. Вже навіть при обробці послідовності за допомогою класичного SAD отримано приріст PSNR в окремих випадках на 0.53dB. При цьому, наприклад, втрата в PSNR для послідовності Foreman (1024) зменшилась з 0.65dB до 0.51dB. Таким чином, можна зробити висновок, що збільшення бітрейту і розмірів блоків призводить до покращення роботи алгоритму в плані мінімізації спотворень відеоряду. При цьому варто відзначити, що збільшення блоку

впливає на загальне зменшення втрати PSNR, але якщо брати відсоткове зменшення відносно використання чистих шаблонів порівняння, зменшення втрати PSNR залишаються на рівнях 46.8% для блоків 16x16 і 41.5% що є доволі незначним покращенням.

Оскільки збільшення розміру блоку мало впливає як на зміну швидкості роботи алгоритму так і на відносне зменшення втрати PSNR, проводити обробку відеопослідовностей з середнім та малим бітрейтом, блоками 64x64 в даному розділі проводитись не буде, натомість проведемо обробку відеопослідовності «Hearthstone» з бітрейтом 24822 кбіт/с, що дозволить нам перевірити вплив збільшення блоків на відеопослідовність з максимальним бітрейтом серед розглянутих, а також найменшим рівнем шумів.

Таблиця 3.6 – Значення параметрів PSNR та пришвидшення для відепослідовності «Hearthstone» з бітрейтом 24822 кбіт/с,

Розмір блоку	Показник	SAD	HSADL	HSADC	TSADL	TSADC
16x16	PSNR	37.43	36.74	36.47	36.42	36.23
	Speed-Up	1	1.49	1.19	2.26	1.81
32x32	PSNR	37.97	37.36	36.21	36.15	36.94
	Speed-Up	1	1.5	1.19	2.25	1.81
64x64	PSNR	38.31	37.82	37.73	37.7	37.55
	Speed-Up	1	1.51	1.21	2.26	1.83

Як бачимо з даних таблиці 3.6, було отримано доволі помітний приріст прискорення відносно відеопослідовностей з малими бітрейтами Hall-Monitor – 1.92, «Hearthstone» з бітрейтом 24822 кбіт/с – 2.26, за рахунок спрощеного алгоритму виділення країв на зображеннях з чітким контрастом. Також на прикладі відеопослідовності «Hearthstone» було підтверджено гіпотезу про

вплив розміру блоку на прискорення роботи алгоритму(розмір блоків не впливає на швидкість роботи алгоритму).

Висновки до розділу

В розділі було проведено розгляд основних алгоритмів виділення країв серед яких найкраще для відеопослідовності «Hearthstone» найкращі результати показали LoG(Лапласіан гаусіана) і Canny. На основі використання цих алгоритмів, було сформовано алгоритми уточнення блоків кандидатів при використанні шаблонів порівняння, які націлені на усунення невизначеності в областях з високою ентропією і, як наслідок, зменшення загальної втрати PSNR при стисненні відеопотоку.

Запропоновані алгоритми було протестовано згідно рекомендацій, запропонованих спеціальною групою MPEG з оцінки складності відеоінструментів. Було проведено тестування на 11 відеопослідовностях, з результатів яких можна зробити наступні висновки:

1. З усіх запропонованих алгоритмів виділення країв, найкраще себе показав алгоритм LoG, так середнє прискорення комбінації TSAD+LoG для блоків розміром 16x16 пікселів складає 51.3% відносно алгоритму повного перебору, при максимальній втраті PSNR - 1.09dB. Максимальне прискорення склало 55.8%, для відеопослідовності з бітрейтом 24822 кбіт/с, з максимальною втратою 1.01dB, що на 53.5% менше ніж для алгоритму без використання виділення границь об’єктів - 2.17dB.
2. Запропонований алгоритм показує себе більш ефективно для відеопослідовностей з більшим бітрейтом, що є наслідком простішого процесу виділення країв за відсутності шумів, а також зменшення впливу випадкових величин на розрахунок TSAD.

При цьому, навіть з використання алгоритмів виділення країв, не дозволило зменшити рівень втрати PSNR до рівня прийнятного для стиснення відеопослідовності. Наступним кроком вдосконалення розроблюваної методики виділення країв є використання вдосконаленого методу блочного пошуку.

4 МЕТОДИ БЛОЧНОГО ПОШУКУ

4.1 Існуючі алгоритми обходу області пошуку векторів кандидатів

4.1.1 Алгоритм повного перебору [56]

Оскільки область пошуку O скінченна, то найбільш очевидним методом мінімізації функції співвідношення блоків є повний перебір всіх значень аргументу $v \in O$.

Даний підхід має свої переваги і недоліки. Перевагою даного методу є гарантоване знаходження глобального мінімуму функції відповідності для кожного блоку. Однак, як було відмічено раніше, не у всіх випадках важливо знайти саме глобальний мінімум. В обробці відео критичне значення має визначення "істинних" векторів руху, незалежно від величин відповідних їм значень функції відповідності.

Очевидним недоліком є обчислювальна складність даного методу. Навіть в світлі високої потужності сучасних процесорів, повний перебір може бути неприйнятний для обробки в режимі реального часу в разі високого розширення відео і великій області пошуку.

4.1.2 Методи шаблонного пошуку[57]

Дані методи є дискретним варіантом покоординатного спуску. Вони ґрунтуються на припущенні, що функція відповідності досить гладка для кожного блоку, без локальних мінімумів.

Перед початком опису загальної схеми алгоритмів даного класу слід ввести поняття шаблону. Шаблон - це набір координат точок, причому координати точок відраховуються від центру шаблону. Таким чином, для довільно заданої точки за шаблоном можна визначити набір координат декількох точок, число цих точок залежить від використовуваного шаблону.

Пошук вектору в кожному блоці є ітеративним процесом. На кожній ітерації обчислюється координата центру шаблону, координати всіх точок

шаблону, і потім значення функції відповідності в кожній з точок шаблону. Центр шаблону на першій ітерації називають центром пошуку. Він зазвичай дорівнює ⟨0,0⟩. В якості центру шаблону для наступної ітерації вибирається та точка шаблону, в якій був досягнутий мінімум функції відповідності. Потім перевіряється умова зупинки пошуку, і в залежності від результату проводиться перехід до наступної ітерації або завершення пошуку вектору в даному блоці. При цьому в якості результату вибирається вектор, відповідний точці мінімуму функції відповідності на шаблоні останньої ітерації [58].

Таким чином, шаблонний алгоритм оцінки руху визначається використовуваним шаблоном. Найбільш часто використовуваними на практиці шаблонами є великий, малий і еластичний ромби, великий і малий квадрати, великий і малий хрести, а також діагональні шаблони рис. 4.1.

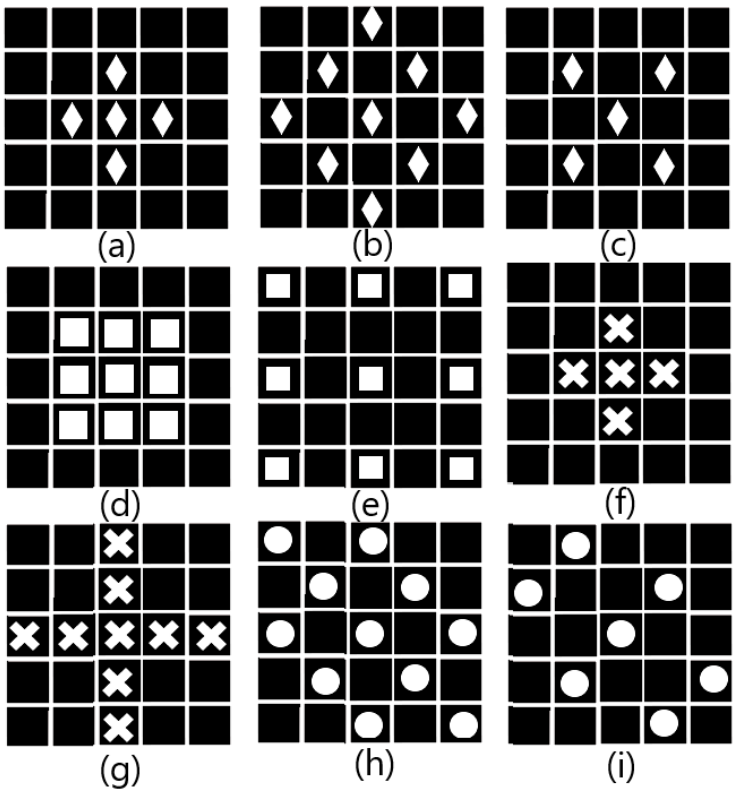


Рисунок 4.1 – Шаблони пошуку (a) малий ромб (b) великий ромб (c)еластичний ромб (d) малий квадрат (e) великий квадрат (f) малий хрест (g)великий хрест (h) велика діагональ (i) мала діагональ

Основним недоліком методів даного класу є їх схильність до знаходження локальних мінімумів функції відповідності замість глобальних. У кожному блоці, як правило, вибирається вектор, відповідний одній з найближчих до центру пошуку точок локального мінімуму функції відповідності, а зовсім не «істинному» руху або її глобальному мінімуму. Однак у даного класу методів є істотна перевага: вони значно скорочують перебір можливих векторів руху, тим самим прискорюючи алгоритм.

В сучасних алгоритмах оцінки руху шаблонний пошук, в основному, використовується для фінального уточнення векторів, отриманих на попередніх етапах. При цьому на різних ітераціях можуть використовуватися різні шаблони.

4.1.3 Алгоритми ієрархічного пошуку [59]

Ідея алгоритмів даної групи полягає в наступному. Перед початком пошуку проводиться обчислення N - 1 зменшених "копій" поточного і опорного кадрів, при цьому кожна чергова копія в 2n (n - натуральне число) разів менше попередньої (рис. 4.2). Пари кадрів однакового розміру будемо називати рівнями. На одному рівні опорний і поточний кадри однакового розміру. Тоді всю множину пар кадрів можна представити N рівнями. Пронумеруємо рівні відповідно до розміру внутрішніх кадрів від меншого до більшого: 1-й рівень буде містити кадри мінімального розміру, N-й - кадри вихідного розміру. Процес оцінки руху складається з N ітерацій, на кожній з яких обробляється пара кадрів з рівня з відповідним номером. Обробка йде від кадрів меншого розміру до більшого. На кожній ітерації проходить оцінка руху яким небудь з відомих методів, наприклад, шаблонним пошуком. При цьому в якості стартової точки на кожній ітерації вибирається векторне поле, отримане з попередньої ітерації. Іншими словами, кожна чергова ітерація проводить уточнення векторів, обчислених на попередній ітерації. При переході на чергову ітерацію розміри області пошуку і блоків, для яких

оцінюються вектори, зазвичай збільшуються в 2n раз, для того щоб число блоків в кадрі на кожній ітерації не змінювалося [60].

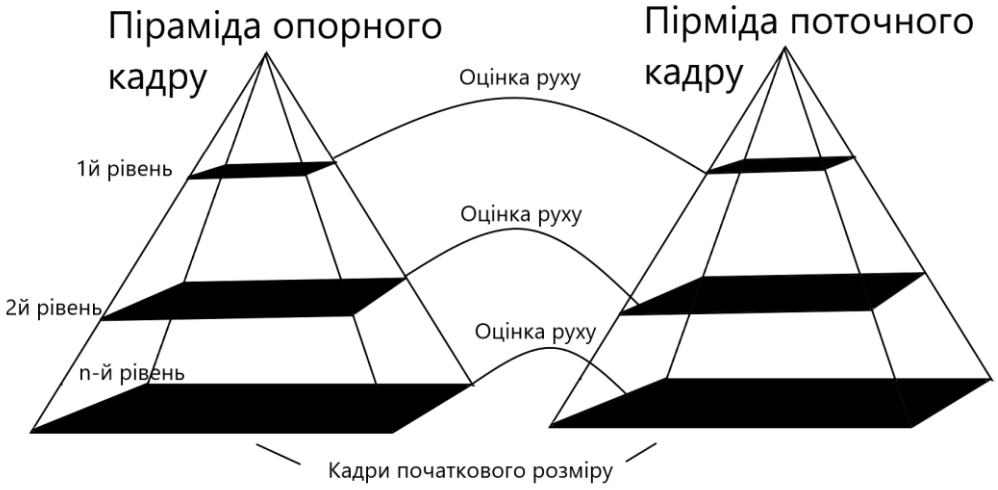


Рисунок 4.2 – схема алгоритмів ієрархічного пошуку

Перевагою алгоритмів даної групи є перебір зменшеної кількості векторів і як наслідок підвищена обчислювальна ефективність. Однак ця перевага нівелюється в разі використання постійного розміру блоків і областей пошуку при переході з одного рівня на інший. Тим не менш, є аргумент на користь того, щоб фіксувати розмір блоків і областей для всіх ітерацій. При фіксованому розмірі блоків підвищується стійкість векторів в гладких областях, оскільки ймовірність потрапляння контрастних деталей зростає разом з ростом площі блоку. Це дозволяє частково вирішити одну з основних проблем блокових методів оцінки руху - проблему пошуку векторів у гладких областях. Для блоків з таких областей функція відповідності приймає дуже близькі (або навіть однакові рівні, в разі абсолютно гладких областей) значення для різних векторів руху. Ефективність використання її значень в якості основного критерію вибору вектору зводиться до мінімуму. Підвищити ймовірність успішного знаходження вектору руху дозволяє використання блоків більшого розміру, але це ефективно лише у випадках,

коли розмір блоку більше розміру гладких областей. Повертаючись до ієрархічного підходу, можна сказати, що виконання оцінки руху при фіксованому розмірі блоку на рівнях з зменшеними кадрами дозволяє підвищити ймовірність попадання в блок контрастних деталей (і тим самим збільшити ймовірність успішного знаходження вектору) в порівнянні з випадком зміни розміру блоку при переході на наступний рівень.

Постійний розмір області пошуку дозволяє збільшити максимальну амплітуду векторів руху в порівнянні з випадком, коли розмір області пошуку зменшується вдвічі при переході на черговий рівень.

Додатковою перевагою даного методу є стійкість до шуму, оскільки під час зменшення зображень, як правило, видаляються високочастотні шуми. Однак разом з високочастотними шумами можуть пропасти і дрібні деталі, що призведе до неправильного визначення руху в деталізованих областях.

Методи даного класу знаходять широке застосування в багатьох задачах обробки відео. Початковий рівень ієрархії вибирається на основі припущення про близькість значень функції відповідності для векторів з невеликої околиці. Початковий рівень ієрархії обирається на основі аналізу даних, отриманих при обчисленні значень функцій відповідності для векторів з невеликої околиці.

4.1.4 Методи, що використовують вектори кандидати.

Для більшості відео-послідовностей справедливим є твердження, що вектори руху сусідніх блоків дуже схожі, оскільки ці блоки найчастіше належать одному рухомому об’єкту. Це твердження призвело до появи цілого класу методів ОД, що використовують вектори-кандидати.

Основна ідея алгоритмів цієї групи дуже проста. Перед обчисленням інформації про рух для поточного блоку формується набір, що складається з уже обчислених векторів руху сусідніх блоків. При цьому сусідні блоки можуть вибиратися як в просторовій так і часовій областях. Сформований

набір векторів називається набором кандидатів. Як результуючий вектор руху в кожному блоці, вибирається найкращий вектор з набору кандидатів. Як критерій пошуку зазвичай використовується функція відповідності. Найбільш яскравими представниками алгоритмів даної групи є 3DRS(3D recursive search) і E3DRS (Enhanced 3DRS).

Метод 3DRS формує набір векторів-кандидатів зі знайдених векторів руху зі зсувами ⟨1, -1⟩ і ⟨1, -1⟩ в поточному кадрі і ⟨2,2⟩, ⟨2,2⟩ в попередньому кадрі. До перших двох векторів-кандидатів додається рівномірно розподілений випадковий вектор з амплітудою до ± 3 пікселів. Після цього з отриманих кандидатів вибирається вектор з найменшою SAD. Використання векторів-кандидатів, взятих з різних напрямків, дозволяє методу 3DRS досить швидко сходитися до реального напрямку руху поблизу кордонів об'єктів в порівнянні з більш простими методами.

У методі E3DRS використовується схожий набір векторів-кандидатів, однак тут є стадія додаткового пошуку вектору з найкращим SAD за шаблоном "малий квадрат" навколо обраного вектору-кандидата. Це забезпечує кращі величини SAD, ніж у методу 3DRS [61].

Методи, які використовують вектори-кандидати, часто мають низьку обчислювальну складність, але при цьому забезпечують гладкість векторного поля, що робить їх придатними для використання в апаратурі реального часу.

4.1.5 Комбіновані методи

У більшості сучасних блокових алгоритмів знаходження руху використовуються різні комбінації базових підходів, описаних вище.

Найбільш популярною комбінацією є спільне використання підходу, що використовує вектори-кандидати, і шаблонного пошуку. Ідея методів даної групи полягає в уточненні кращого вектору набору за допомогою шаблонного пошуку. Завдяки простоті і обчислювальній ефективності алгоритми даної групи досить часто стають предметом інтересу дослідників.

Як приклад розглянемо найбільш популярний з сучасних представників цієї групи алгоритм FAME (Fast Adaptive Motion Estimation).

Цей алгоритм використовує прийоми швидкого визначення нерухомих блоків, підвищення стійкості пошуку векторів у гладких областях, ранню зупинку пошуку для економії обчислювальних ресурсів, а також підвищення ефективності використання наборів кандидатів і шаблонів. В рамках даного розділу найбільший інтерес представляють остання пара прийомів. Розглянемо їх більш детально.

Підвищення ефективності використання наборів кандидатів досягається за рахунок додавання в набір додаткових векторів. Стандартний набір включає в себе вектори з 3 блоків, що знаходяться ліворуч, зверху і праворуч зверху щодо поточного блоку. Крім них в набір додаються ще два вектори. Перший обчислюється як середнє значення векторів стандартного набору. Другий називається інерційним кандидатом і дорівнює вектору того блоку попереднього кадру, проекція якого на поточний кадр має найбільшу область перетину з поточним блоком. При цьому пошук здійснюється уздовж вектору попереднього кадру. Варто зауважити, що мається на увазі саме попередній кадр, а не опорний. Інерційний вектор вибирається з векторного поля, обчисленого алгоритмом оцінки руху для попереднього кадру. Фактично цей прийом використовує припущення про рівномірний рух об'єктів, а сам інерційний кандидат є продовженням траєкторії руху блоку попереднього кадру. Наявність зазначених двох додаткових кандидатів дозволяє підвищити точність пошуку векторів.

Новизна в використанні шаблонних методів полягає в наступному. Алгоритм використовує шаблони «великий ромб», «малий ромб» і шаблон «еластичний ромб». Використання цих шаблонів при пошуку вектору в кожному блоці залежить від декількох характеристик локальної околиці

поточного блоку. Зокрема, використовується величина, яка характеризує гладкість векторного поля, і помилки компенсації сусідніх блоків. Залежно від умов, справедливих для цих величин, проводиться вибір однієї з 3 стратегій пошуку, при цьому в рамках кожної стратегії можуть бути використані не всі 3 шаблони.

Слід зазначити, що алгоритми кожної з розглянутих груп мають свої переваги і недоліки. Жоден з базових методів, використаних в чистому вигляді, не дає прийнятних результатів на практиці. Для побудови ефективного методу оцінки руху необхідно комбінувати прийоми, використані в алгоритмах різних типів, так, щоб недоліки одного методу компенсувалися перевагами іншого. Прикладом такого методу є алгоритм FAME, що поєднує використання векторів-кандидатів, методів шаблонного пошуку, умов ранньої зупинки пошуку, способів швидкого визначення нерухомих блоків і інші [62].

Блокові алгоритми оцінки руху є досить вигідним компромісом по співвідношенню обчислювальна складність/точність знайдених векторів. Комбінування прийомів з алгоритмами різних категорій в рамках класу блокових методів дозволяє побудувати універсальні алгоритми ОР, що володіють заданими властивостями і легко реалізуються апаратно.

4.2 Модернізовані методи та техніки блочного пошуку

4.2.1 Техніка адаптивного пошуку поля векторів руху

Техніка адаптивного пошуку поля векторів руху (MVFAST - Motion Vector Field Adaptive Search Technique) є вдосконаленням існуючого шаблону великого ромбу (diamond search - DS), який дозволяє отримати виграш як у візуальній якості так і в швидкості обрахунку порівняно з DS.

Спочатку розглядається невеликий набір кандидатів, а саме вектор руху (0,0) та вектори руху з трьох просторово сусідніх блоків (ліворуч, зверху, праворуч) як можливих кандидатів на прогнозування вектору руху [63].

На відміну від DS, де розглядався лише великий ромб, що рухається (рис. 4.1-b), MVFAST використовує менший рухомий ромб (рис. 4.3). Ромбовий малюнок продовжує рухатись у напрямку поточного мінімуму (рис 4.3-b,c,e), поки в центрі не буде знайдено мінімальний SAD. Якщо використовувати малий ромб, пошук на цьому закінчиться. Якщо ж використовувати шаблон великого ромбу, фінальний пошук проводиться навколо центру використовуючи шаблон малого ромбу

У MVFAST вибір між використанням малого ромбу та великого алмазу здійснювався на основі характеристик типу руху, який, швидше за все, матиме поточний блок. Рух характеризується як малий, коли довжина найбільшого вектору руху з сусідніх блоків, що використовувалась для формування вектору кандидату, менша деякого порогу L_1, середнім коли довжина лежить між L_1 і наступним порогом L_2, великим коли довжина вектору більша за L_2. Якщо рух малий, для уточнення вектору руху використовується малий ромб з вектором руху (0,0) в якості центру. Якщо рух середній, центром обирається вектор руху (0,0), після чого використовується шаблон великого ромбу. Коли ж рух визначається як великий, досліджується вектор руху (0,0) та додаткові вектори кандидати, сформовані суперпозицією сусідніх блоків, після чого вектор кандидат з найнижчим SAD використовується як центр для шаблону малого ромбу.

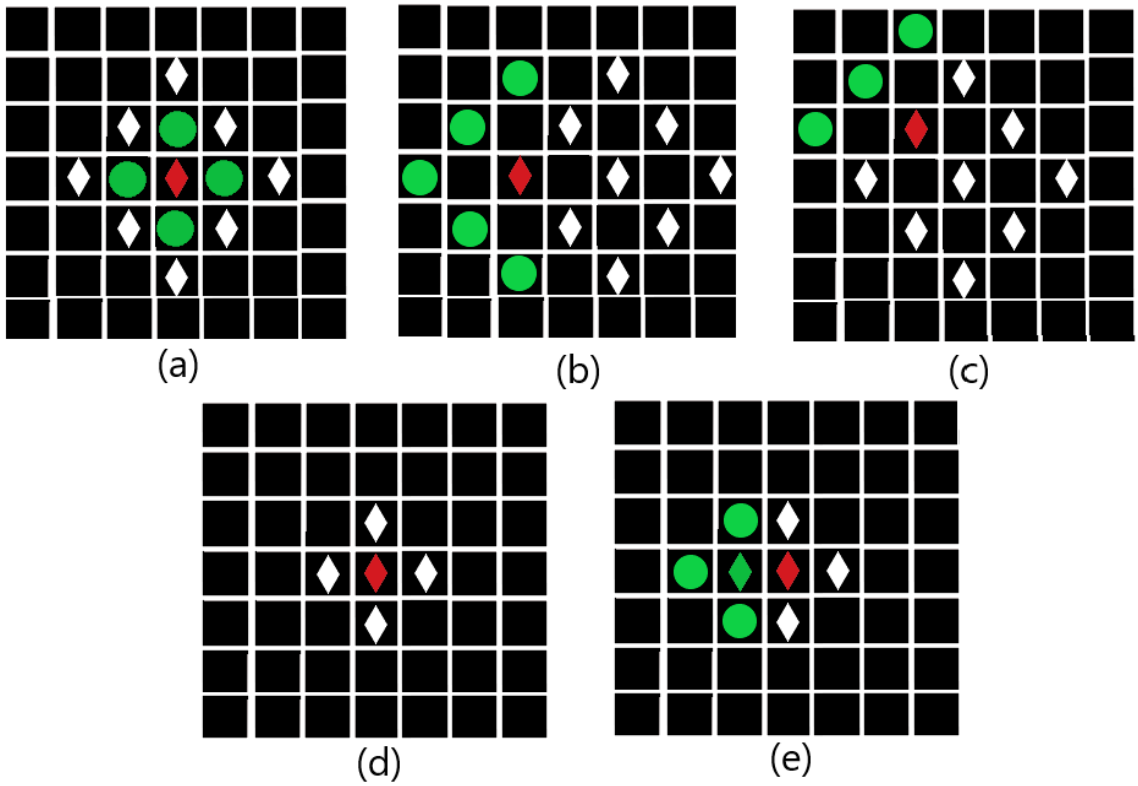


Рисунок 4.3 – Кроки пошуку MVFAST для шаблонів великого ромбу a-c, малого ромбу d-e.

Для зменшення викривлення зображення, додатково як центр пошуку можна використати вектор з SAD, найближчим до мінімального, так що б при цьому їхня різниця не перевищувала фіксовану похибку T = 512. Після дослідження альтернативного центру, береться той вектор руху для якого SAD буде найменшим. Якщо ж SAD для жодного з сусідніх векторів руху, не задовольняє вимогу відмінності на T, уточнення по альтернативному центру не проводиться.

4.2.2 Узагальнений вибір вектору кандидату та використання адаптивної поправки

Алгоритм MVFAST дозволяє значно покращити продуктивність порівняно з DS як в плані прискорення, так і PSNR. Для подальшого вдосконалення алгоритму, потрібно детальніше розглянути характеристики

закодованої послідовності. Зокрема, як видно з опису алгоритму, MVFAST як і DS працює на основі нульового припущення – довжина вектору руху даного блоку дорівнює нулю (V(0,0)). Це також основа більшості інших алгоритмів, включаючи NTSS (New three step Search – новий алгоритм трьох крокового пошуку) [64], і оригінальних шаблонних алгоритмів. Підставою цього припущення є те, що для тестових відеопослідовностей які використовувались при розробці MPEG-4(відеопослідовність Foreman), вектори руху були зосереджені на невеликій території навколо центру пошуку. На жаль, для деяких послідовностей це не завжди відповідає дійсності (рис.4.4).

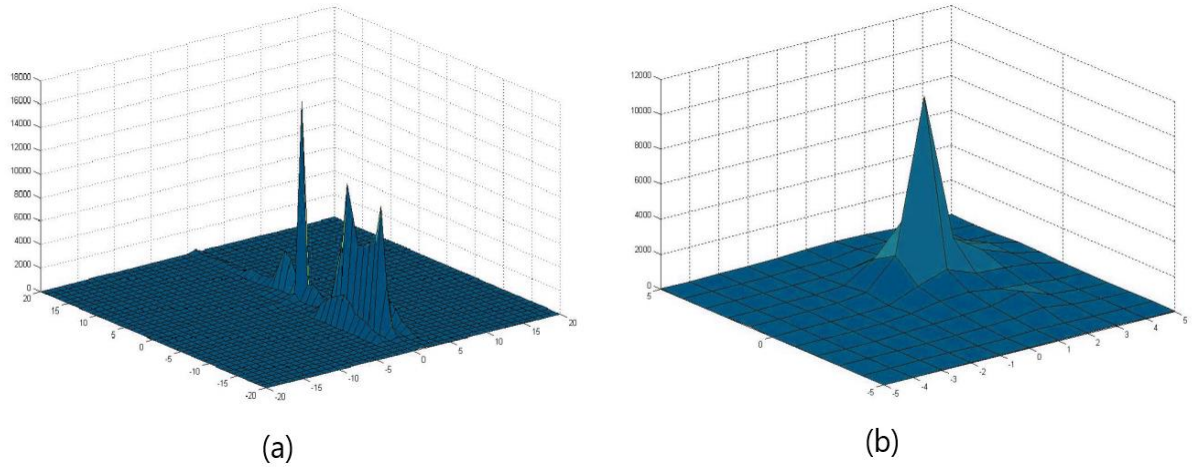


Рисунок 4.4 – Розподіл векторів руху у відеопослідовностях (a)Foreman і (b)Bus, отриманих алгоритмом повного перебору, в порівнянні з нульовими векторами.

Однак в інших алгоритмах, таких як зональні алгоритми [64], в якості вектору кандидату використовується суперпозиція векторів руху трьох просторово сусідніх блоків(ліворуч, зверху та праворуч угорі). Що стосується сучасних стандартів кодування відео таких як MPEG-4 і H.263, вони використовують схожу схему отримання вектору кандидату, щоправда вектори з яких розраховується суперпозиція можуть бути різними. В

порівнянні з нульовим припущенням, метод суперпозиції векторів сусідніх блоків дає значно кращі результати. Для підтвердження цього розглянемо всі ті ж тестові послідовності Foreman та Bus, а саме кореляцію векторів руху знайдених методом повного перебору відносно методу суперпозиції сусідніх векторів(медіанного пошуку)(рис.4.5). Я к бачимо, медіанні кандидати, мають набагато вищу кореляцію з «істинними» векторами руху, навіть для нецентрованих послідовностей, як наприклад послідовність Bus. Таким чином є сенс, замість вивчення початкової позиції (0,0) можна досягти більш точних результатів використавши для початку пошуку суперпозицію векторів руху сусідніх блоків.

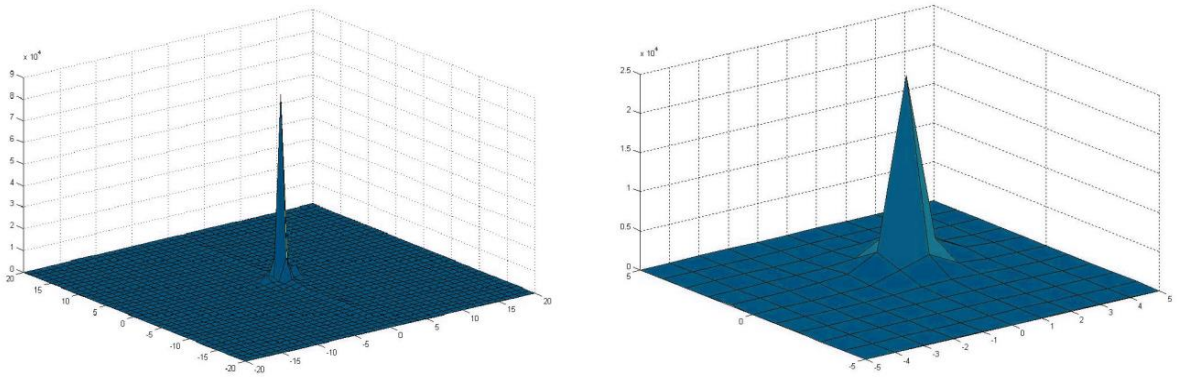


Рисунок 4.5 – Розподіл векторів руху для відеопослідовностей a)Foreman і (b)Bus, отриманих алгоритмом повного перебору, в порівнянні з медіанними векторами.

Крім того, інші можливі предикати також мають дуже високу кореляцію з вектором руху поточного блоку. В MVFAST, крім (0,0), та векторів руху трьох просторово сусідніх блоків, що використовуються для медіанного обчислення (лівий, верхній та верхній правий блоки) можуть бути використані й інші предикати, які, призводять до набору з 4 початкових предикатів.

Розглянемо вибірку з більшої кількості таких предикатів, яка крім попередніх чотирьох згаданих та медіанного предикату також включає кілька інших, кандидатів з високою ймовірністю співпадіння. Такі кандидати можуть включати вектор руху інших просторово або часово суміжних блоків, лінійні та нелінійні комбінації цих векторів руху, такі як середнє або зважене середнє значення всіх цих кандидатів. Такий набір предикатів називають загальним набором. При роботі з такими наборами, визначають вектори руху макроблоків зліва, зверху та зверху праворуч, їх суперпозицію, (0,0) вектор руху та вектор руху блоку розміщеного в попередньому кадрі в момент часу t-1 як набір кандидатів A. Ці вектори руху найпростіші у виборі та не суттєво збільшують обчислення. З рис 4.6, очевидно, що з використанням цього набору векторів руху кореляція ще більше зросла навіть у порівнянні з медіанним предикатом (близько 24% для відеопослідовності Foreman).

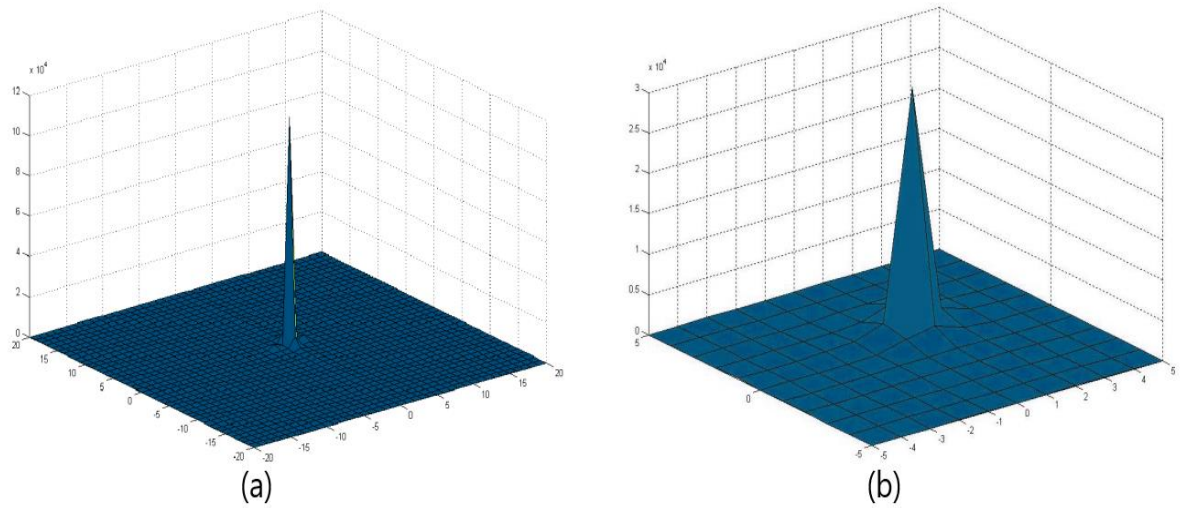


Рисунок 4.6 – Розподіл векторів руху для відеопослідовностей a)Foreman і (b)Bus, отриманих алгоритмом повного перебору, в порівнянні набором A.

Для кращої візуалізації розглянемо ентропію векторів руху, розраховану відносно всіх можливих векторів кандидатів табл.4.1

Таблиця 4.1 – Ентропія векторів кандидатів для різних наборів предикатів відносно повного перебору

Відеопослідовність	(0,0)	Медіана	Набір A
Foreman	6.81	5.22	4.66
Coastguard	7.11	5.01	4.47
Bus	6.82	5.29	4.34

Окрім вибору предикату, було згадано, що MVFAST та зональні алгоритми також використовують набір порогових параметрів. Через певні інтервали поточне мінімальне спотворення порівнюється з цими пороговими значеннями і залежно від результату, обирається припинити пошук негайно чи припинити його після вивчення додаткової кількості контрольних пунктів або зон, або просто продовження пошуку. Одним з важливих питань, є техніка вибору цих порогів оскільки використання фіксованих значень, що не враховують особливості закодованої послідовності може вплинути як на складність оцінки – якщо поріг буде занадто високим, так і на різке зниження візуальної якості відеопослідовності – якщо поріг буде занизьким. Вибір такого порогу для відеопослідовності може бути доволі складним процесом і залежати від особливостей кодування таких як наприклад, шум, тип руху, роздільної здатності тощо), тому є сенс розглянути адаптивний розрахунок порогових значень, оскільки це могло б потенційно покращити ефективність алгоритму що їх використовує.

Розрахунок порогових значень проводиться адаптивно на основі кожного блоку або навіть набору блоків, що в певному сенсі дозволяє нам адаптуватися до локальних характеристик кадру і, таким чином, підвищити ефективність порогового значення. Пороговими значеннями можуть бути лінійні або нелінійні функції декількох доступних параметрів, таких як критерії спотворення тимчасово або просторово сусідніх блоків або навіть значень їх вектору руху. Для використання метрики SAD n-кількості

предикатів, то обчислення параметра порогового значення T_j можна розглядати як:

$$T_j = f_k(mSAD_1,..., mSAD_n, \vec{MV}_1,..., \vec{MV}_n) \quad (4.1)$$

де *mSAD* - мінімальне значення SAD, $\vec{MV}$ - вектор руху відповідного предиката.

Також можна зважити ці значення або компенсувати їх фіксованим або адаптивним параметром, або навіть дозволити цим порогам адаптуватися або змінюватися під час процесу оцінки руху поточного блоку. Для подальших розрахунків буде використано мінімальне значення SAD блоків вектори руху яких використовуються для визначення області пошуку (рис.4.8).

Більш загальна формула для порогу вибору, при використанні цих лімітів:

$$T_j =, \min\{\max[\ f_k(mSAD_1,..., mSAD_n, \vec{MV}_1,..., \vec{MV}_n), T_{j-1}], T_k\} \quad (4.2)$$

де на значення поточного порогу будуть впливати T_{j-1} попереднього блоку і середній з усіх розрахованих до цього T_k [65].

4.2.3 Техніка передбачуваного адаптивного пошуку поля векторів руху (PMVFAST)

Техніка передбачуваного адаптивного пошуку векторів руху (PMVFAST) - покращує MVFAST, використовуючи набір A предикатів та адаптивні методи визначення порогу, розглянутих в попередньому розділі. PMVFAST використовує в основному ту саму архітектуру та шаблони, що і MVFAST. Спочатку досліджується набір предикатів, який на відміну від MVFAST включає медіанний предикат та тимчасові кандидати, а набір параметрів порогового значення адаптивно визначається. На відміну від MVFAST, де пріоритет надається вектору руху (0,0), пріоритет у PMVFAST надається медіанному предикату, первинне дослідження якого, робить

PMVFAST медіанно-упередженим алгоритмом. Потім застосовується порогове значення проти малого порогу (наприклад, T_j = 256), що дозволяє алгоритму припинити пошук після вивчення лише однієї точки. Після визначення найкращого серед предикатів застосовується порогове значення. Цей поріг обчислюється адаптивно як мінімальне значення трьох сусідніх блоків (лівий, верхній і верхній правий). Якщо поточний SAD задовольняє висунуті значення, алгоритм зупиняється. В іншому випадку найкращий предикат використовується як центр пошуку малим ромбом. Нижня та верхня межі для T_j встановлені відповідно на 512 та 1024. Можливий варіант розрахунків з жорстким значенням без меж [66].

Істотною відмінністю PMVFAST порівняно з MVFAST є спосіб вибору малого та великого алмазу [65].

На відміну від MVFAST, де рух характеризувався як малий, середній або великий, враховуючи найбільшого кандидата на вектор руху, у PMVFAST використовується інша стратегія відбору. Ця стратегія може покращити загальну швидкість алгоритму, рідше використовуючи великий ромб. На першому кроці обчислюється $\|\text{MedianMV}\|_1$, де MedianMV є медіаною предиката трьох сусідніх блоків до поточного положення. Якщо це значення дорівнює нулю, а індикатор спотворень T_j є відносно великим, використовується великий ромб. В іншому випадку використовується маленький ромб. На відміну від MVFAST, усі прогнози завжди перевіряються, що підвищує точність прогнозу. Поріг T_j в основному є ще одним параметром порогового значення, який використовується лише тоді, коли медіанний предикат знаходиться в центрі вікна пошуку, і може розглядатися як ще одне передбачення SAD поточного блоку. Якщо T_j досить малий, менше деякого значення K, SAD поточного блоку також, ймовірно, буде малим. У цьому випадку вектор руху, швидше за все, можна знайти в невеликому районі навколо центру, що говорить про те, що малий

ромб може бути більш ефективним при використанні. В іншому випадку замість цього слід використовувати великий ромб.

Ще одною перевагою PMVFAST є додаткові критерії оцінки руху, що дозволяють зменшити час виконання пошуку[66,67]. Вони засновані на можливість тому, що деякі або навіть усі використовувані предикати вектору руху - однакові або дуже близькі один до одного. У такому випадку також є велика ймовірність того, що цільовий вектор руху знаходиться в невеликому околі предикатів. Наприклад, якщо всі просторово сусідні вектори руху однакові, або медіана передбачення однакова з вектором руху розміщеного блоку в попередньому кадрі, існує більша, ніж зазвичай, впевненість у точності предикату, і пошук може бути виконаний після N ітерацій пошуку малим ромбом, для деяких малих N. Чим більше таких критеріїв справджуються тим більша ймовірність правильності обраного предикату, за рахунок чого кількість ітерацій пошуку шаблоном малого ромба буде помітно зменшена, в і навіть зведена до одної, або повної зупинки знаходження вектору руху на блоці отриманому з предикату, не зважаючи на інші критерії дострокової зупинки.

Очевидно, що PMVFAST є узагальненням MVFAST. Корегування різних параметрів та використання різних предикатів, критеріїв порогового значення та функцій, дозволяють використовувати кілька варіацій алгоритму PMVFAST. PMVFAST також є частковим випадком зональних алгоритмів [67-69].

4.3 Розробка нового методу блочного пошуку векторів руху

4.3.1 Загальна ідея методу

Як було зазначено вище, в сучасних системах компресії відео-потоку використовується комбінація алгоритмів що базуються на використанні

векторів-кандидатів та шаблонів пошуку. Так при розробці методу пошуку нам потрібно визначитись з наступними параметрами:

1. Попередня підготовка даних до обробки
2. Комбінацію яких векторів кандидатів використовувати для знаходження вектору-руху для конкретного блоку
3. Параметри визначення порогових значень для оцінки та умов ранньої зупинки пошуку
4. Шаблон, що буде використано для уточнення пошуку вектору руху.

4.3.2 Попередня обробка зображення за умовою ранньої зупинки руху

В 2 розділі даної роботи було визначено метрику, яку буде використано для оцінки схожості блоків. Таким чином, умова ранньої зупинки пошуку зводиться до визначення значення метрики, при якому можна вважати, що знайдений блок відповідає бажаним умовам якості зображення. Так бля блоку розміром 16x16 пікселів, зі значенням яскравісної компоненти, яка знаходиться в межах 0-255, можна виділити значення, при зміні на яке, людське око не буде помічати майже ніякої різниці. Так як показано в дослідженні чутливості людського ока та його впливу на якість зображення [70], зміна яскравісної компоненти на 2%(5 при заданій глибині кольору), не призводить до зміни сприйняття зображення. Звичайно це не значить, що при значенні SAD *256 – 1280 можна вважати блоки схожими і приймати до розрахунку блоків кандидатів. Це твердження справедливе тільки тоді, коли зміна яскравісної компоненти кожного окремого пікселя не перевищує заданий поріг. Тобто має сенс ввести перший етап оцінки оброблюваних зображень, на якому буде взято різницю між кожним пікселем опорного та поточного кадру, після чого в залежності від значень цих різниць в середині блоків, можна виключити з подальшого розгляду блоки, для яких можна вважати вектори руху нульовими. Це дозволить нам, суттєво зменшити час обрахунку всіх наступних етапів компенсації руху, а також зменшити

невизначеність в областях з низькою ентропією зображення. Також на цьому етапі можна усунути ефект різкого зростання значення пікселів: якщо в середині монотонної області, значення різниці пікселів не перевищує заданий поріг, з'являються одиночні пікселі, різниця в значенні яких є доволі значною, можна компенсувати це значення шляхом усереднення значень оточуючих пікселів.

Таким чином отримано перший етап, після якого можна отримати карту нескомпенсованої різниці зображень, та прибрати з оброблюваних матриць блоки, максимальна значення різниці в яких не перевищує заданий поріг в 2% від глибини пікселя. Для цих блоків вектори руху будемо вважати нульовими, та виключаємо їх зі подальшої обробки занулюючи їх значення.

4.3.3 Вибір блоків впливу на пошук вектору кандидату для поточного вектору.

В методах, описаних в розділі 4.2 для визначення векторів кандидатів використовується припущення про те, що різні частини одного об'єкту рухаються по одній траєкторії, що дозволяє знайти вектор руху блоку, шляхом суперпозиції векторів руху сусідніх блоків а також твердженням про однорідність руху на сусідніх кадрах(оскільки при частоті близькій до 30кадрів/секунда, зміна в кадрах представляє собою мінімальні зміни в положенні а також напрямку об'єктів). Таким чином, є сенс, для отримання вектора кандидата розглядати вектори руху відповідних блоків на попередньому фреймі зображення, що дозволяє підвисити початкову точність визначення векторів кандидатів. Як було показано в розділі 3.2.3, на практиці часто використовуються вектори руху попереднього зображення для формування набору векторів предикатів. При цьому, для збільшення точності формування предикатів є сенс, на кожному зображенні розраховувати ту частину векторів руху, яку не розраховано на попередньому

(рис.4.7), що дозволить використати уточнення за вектором руху конкретного блоку.

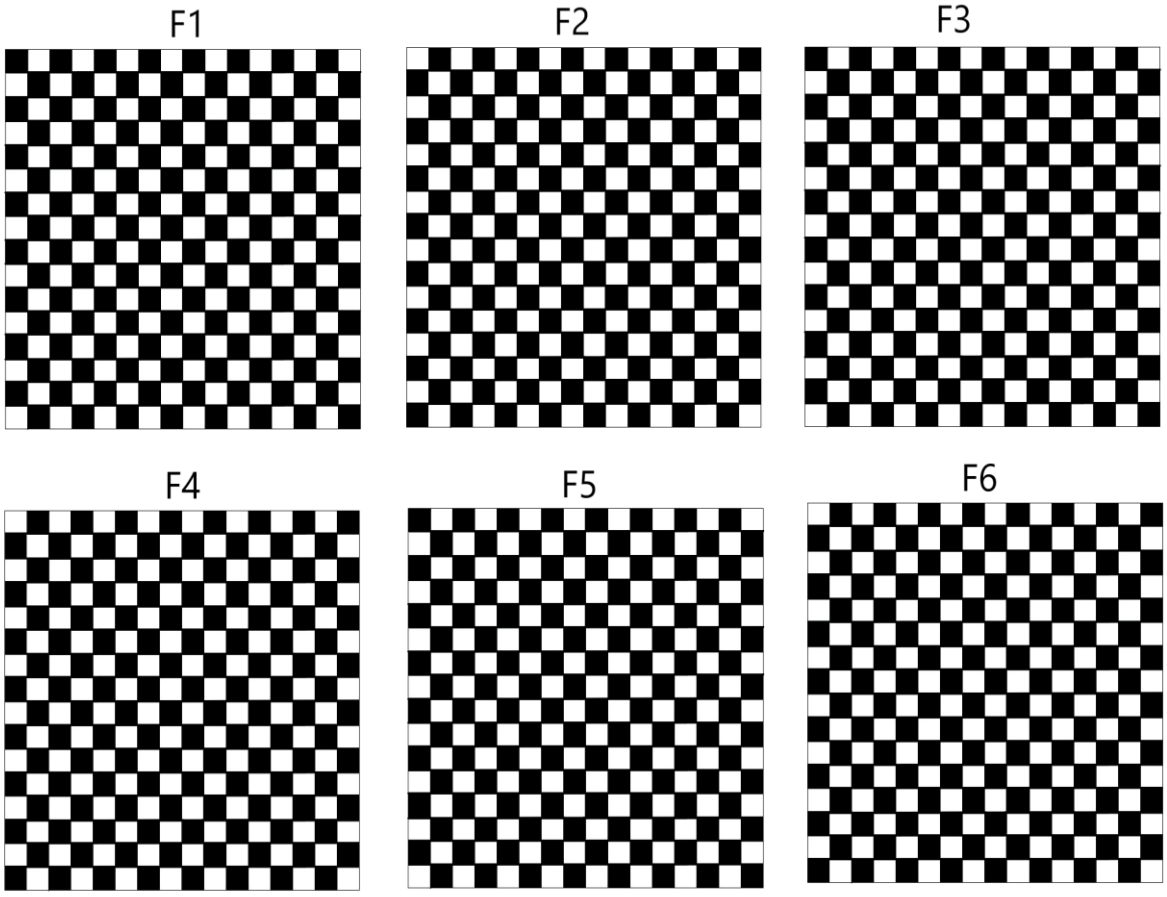


Рисунок 4.7 – Шаблон «шахової дошки» для пошуку векторів руху блоків, через кадр.

При цьому варто відзначити, що схема представлена на рис 3.3 не завжди буде давати найкращий результат в плані перекриття блоків, для яких не було знайдено векторів руху на попередньому зображенні. Крім того, при використанні шаблону шахової дошки отримано надлишок векторної інформації. Також, однією з ідей нашого методу є скорочення кількості блоків які обраховуються на першій ітерації. Перед тим як обирати шаблон, за яким буде знайдено вектори руху для блоків на першій ітерації, розглянемо загальну ідею методу на прикладі «шахової дошки»:

1. Визначаємо вектори руху для другого кадру відео-послідовності за шаблоном «шахової дошки».
2. Прибираємо з розрахунків третього кадру ті блоки, на які вказують вектори руху знайдені для другого кадру.
3. Визначаємо вектори руху для третього кадру відео-послідовності, для блоків, що не були визначені на попередньому кадрі
4. Повторяємо процедуру для четвертого і наступних кадрів відео-послідовності.
5. Після знаходження половини блоків на першій ітерації, повертаємось до другого кадру, де для формування набору предикатів використовуємо вектори руху наступного кадру.

В результаті такого перебору отримано карту руху половини блоків на парних кадрах і половини на непарних кадрах. Після першої ітерації алгоритму повертаємось до другого кадру відео-послідовності, де знайдено половину блоків (на практиці їх буде значно менше, оскільки було виключено блоки з нульовими векторами) і повторяємо алгоритм. Залишилось визначитись набір яких предикатів буде використано для знаходження вектору руху поточного блоку.

Для цього буде використано Набір А, доповнений новими предикатами утвореними суперпозицію векторів руху суміжних блоків для кадру в момент часу t-1 або t+1 в залежності від ітерації. Також не буде задіяно нульовий вектор (як було описано в розділі 4.3.2, першим кроком пошуку векторів руху є виключення блоків з SAD = 0 і блоків, для яких максимум різниці між пікселями буде менше порогового значення). Таким чином фінальний набір для утворення вектору предикату буде утворюватися суперпозицією векторів руху, декількох суміжних блоків на поточному кадрі і суміжних блоків на попередньому кадрі (рис.4.8).

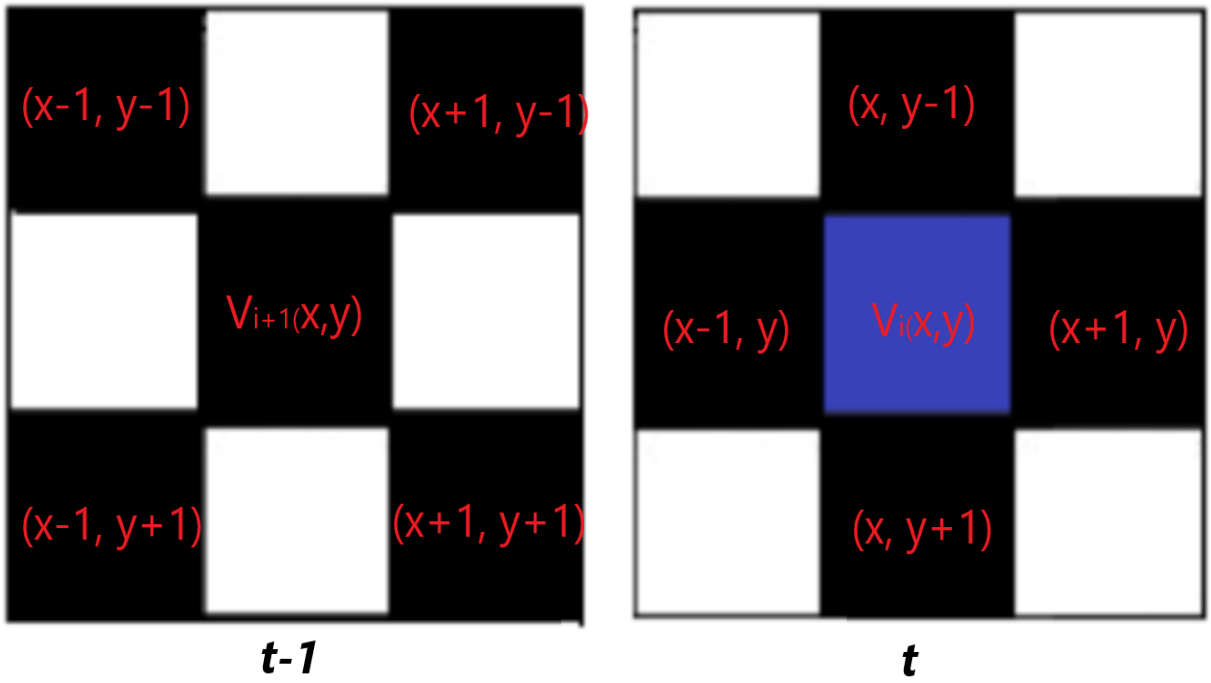


Рисунок 4.8 – Блоки вектори руху яких будуть впливати на формування медіанних векторів для пошуку поточного блоку для шаблону «шахової дошки».

Як бачимо, в результаті використання шаблону шахової дошки, для кожного конкретного блоку можна отримати суперпозицію векторів руху 4х оточуючих блоків на поточному кадрі і з 5блоків на попередньому та наступному. В розділі 4.2.2 було визначено, що для формування предикату медіанного вектору сусідніх блоків, нам достатньо буде суперпозиції 3х векторів за рахунок чого можна скоротити кількість операцій на першій ітерації методу. Таким чином має сенс використати шаблон «кожен третій», який було використано в шаблоні TSAD, рис 4.9.

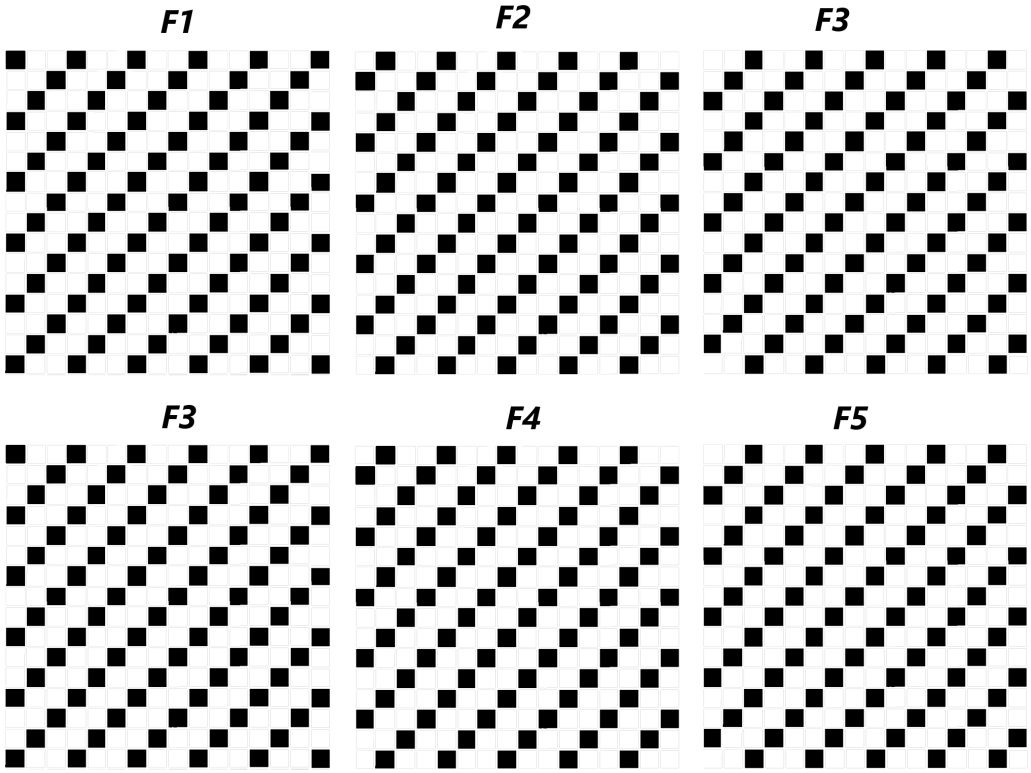


Рисунок 4.9 – Шаблон «кожен третій» для пошуку векторів руху блоків, через кадр.

Таким чином, на кожному кадрі послідовності буде відомо мінімум 3 сусідніх блоки для кожного з блоків не рахуючи краєві. Розташування блоків, які впливають на формування предикатів поточного блоку представлені на рис 4.10.

Як результат, буде отримано набір з п’яти додаткових векторів-кандидатів:

1. Медіанний вектор суміжних блоків на поточному, попередньому і наступному кадрів.
2. Медіанний вектор суміжних блоків на поточному і наступному кадрі.
3. Медіанний вектор суміжних блоків на поточному і попередньому кадрі.
4. Вектор руху даного блоку з попереднього кадру

5. Медіанний вектор суміжних блоків на наступному кадрі

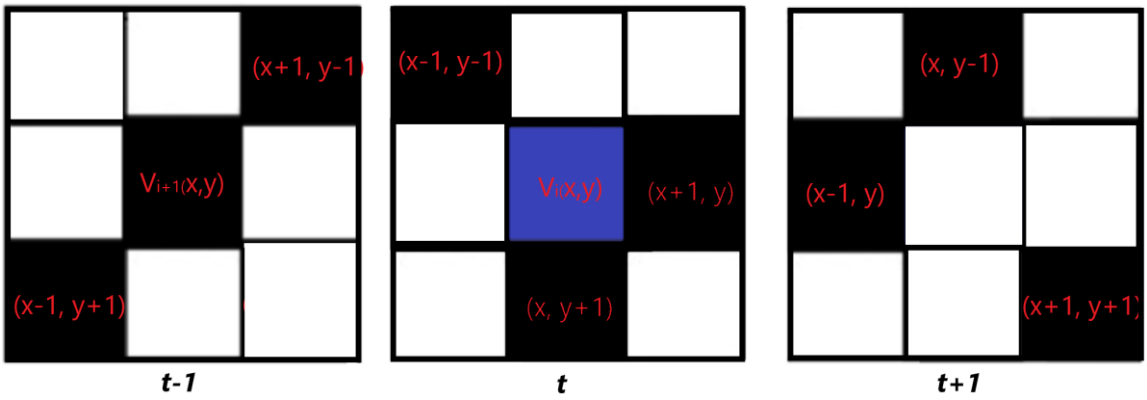


Рисунок 4.10 – Блоки вектори руху яких будуть впливати на формування медіанних векторів для пошуку поточного блоку для шаблону «через три».

При цьому першим проводимо пошук в області, для якої SAD блока предиката є мінімальною, після чого проводимо перевірку відповідності умови відмінності наступних SAD від мінімальної на значення T. Якщо після розгляду п’яти основних векторів кандидатів не буде знайдено вектор руху який буде задовольняти виставлені нами умови пошуку, пошук буде продовжуватись по набору A. Порівняння розподілів векторів руху отриманих повним перебором і останнім з запропонованих методів представлено на рис.4.8. Алгоритм формування такого набору назвемо TF (three by frame – три на кадр). Звичайно блоки представлені на рис.4.10 це частковий випадок і розташування блоків може змінюватись місцями, але для поточного буде завжди наступна композиція:

1. Три блоки навколо(лівий, верхній, нижній правий/правий, нижній, верхній правий).
2. На попередньому блоці верхній правий, нижній лівий.
3. На наступному(правий, нижній, верхній правий/ лівий, верхній, нижній правий).

Розташування блоків на попередньому і наступному кадрах можуть змінюватись місцями.

Розглянемо розподіл векторів руху предикатів відносно «істинних» векторів руху для відеопослідовностей Foreman і Bus (рис 4.11).

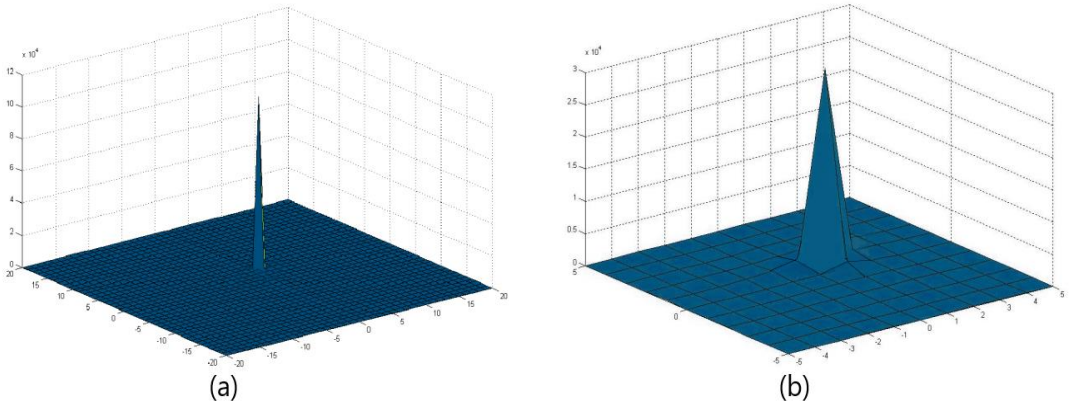


Рисунок 4.11– Розподіл векторів руху для відеопослідовностей (a)Foreman і (b)Bus, отриманих алгоритмом повного перебору, в порівнянні з набором TF.

Як бачимо, знову було отримано приріст кореляції. Для кращої візуалізації додамо значення ентропії для методу TC в табл. 4.1

Таблиця 4.2 – Ентропія векторів кандидатів для різних наборів предикатів відносно повного перебору

Відеопослідовність	(0,0)	Медіана	Набір A	TF
Foreman	6.81	5.22	4.66	4.21
Coastguard	7.11	5.01	4.47	4.28
Bus	6.82	5.29	4.34	4.13

4.3.4 Адаптивна оцінка порогу зупинки додаткового пошуку

При формуванні набору предикатів за допомогою методу ТС є сенс там, де можливо, використовувати значення порогового параметру з попереднього кадру. Таким чином можемо переписати формулу 4.2 в наступному вигляді

$$T_j =, \min\{\max[\ f_k(mSAD_1,..., mSAD_n, \vec{MV}_1,..., \vec{MV}_n), T_{j-1}], T_{j(t-1)}\} \qquad (4.3)$$

тобто замість T_k - середнього порогового значення по попереднім блокам кадру, буде використано $T_{j(t-1)}$ - порогове значення даного блоку, з попереднього кадру відеопослідовності на першій ітерації алгоритму. На другій же ітерації, за рахунок знаходження різних блоків для різних кадрів можливе використання значення параметру $T_{j(t+1)}$ з наступного кадру:

$$T_j = \min\left\langle \max\{T_{j(t-1)}, T_{j(t+1)}\}\{\max[\ f_k(mSAD_1,..., mSAD_n, \vec{MV}_1,..., \vec{MV}_n), T_{j-1}],\}\right\rangle \qquad (3.4)$$

Таким чином, якщо SAD знайдене в процесі перевірки одного з предикатів менше за мінімальне SAD блоків, що впливали на формування предикату і менше за SAD для цього блоку з попереднього кадру або наступного кадру, можна зупинити пошук на поточному блоці.

4.3.5 Шаблон уточнення пошуку

Одним з найбільш часто використовуваних шаблонів пошуку є шаблони великого та малого ромбів, особливості використання яких було описано в розділі 4.2.1. Також одним з перспективних шаблонів на нашу думку, може буди еластичний ромб рис.4.1. Варто відзначити, що у випадку використання медіанних предикатів є сенс корегувати використання шаблону пошуку в залежності від характеристик руху, а саме куту напрямку, оскільки в різних випадках малий і еластичний ромб дає краще перекриття в напрямку руху(рис 4.12).

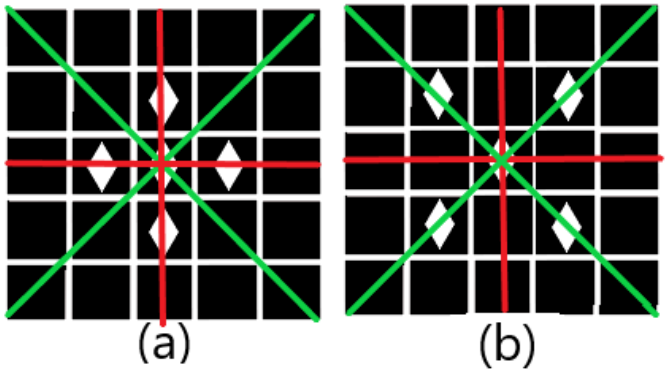


Рисунок 4.12 – Приклад перекриття точок в напрямку руху під кутами кратними π і π/2, для шаблонів (а) малий ромб, (б) еластичний ромб

Таким чином після вибору предикату шаблон уточнення буде сформовано на основі кута предиката.

4.3.6 Загальна структура методу

Запропонований метод складається з наступних частин.

1. Видалення з розрахунків блоків SAD яких рівний нулю, або локальний максимум різниці пікселів не перевищує 2% глибини кольору.
2. Знаходимо всі вектори руху першого кадру для якого проводиться компенсація, одним з методів який дає найкращий результат в тому що стосується PSNR (наприклад PMVFAST або FS), жертвуючи швидкістю.
3. Знаходження третини блоків наступного кадру на основі набору предикатів отриманих за допомогою TF (за основу береться попередній кадр послідовності), з адаптивним підбором порогів зупинки базованих на інформації з попереднього кадру, та шаблоном уточнення, що обирається в залежності від куту вектору предикату.
4. Знаходження іншої половини блоків кожного кадру на основі набору предикатів отриманих за допомогою TF (за основу береться

наступний кадр відеопослідовності), з адаптивним підбором порогів зупинки базованих на інформації з попереднього і наступного кадру, та шаблоном уточнення, що обирається в залежності від куту вектору предикату.

В результаті було отримано новий метод блочного пошуку, який назвемо TFMVFAS (Three by Frame Motion Vector Field Adaptive Search – адаптивний пошук векторів руху заснований на часових шахах).

4.4 Оцінка роботи алгоритму на прикладі тестової послідовності

Запропонований алгоритм TFMVFAS було вбудовано в програмне забезпечення кодера MPEG-4 і протестовано з використанням комплексних умов тестування, запропонованих Спеціальною групою MPEG з оцінки складності відеоінструментів.

У таблиці 4.3 TFMVFAS порівнюється з точки зору PSNR та контрольних пунктів порівняно з алгоритмами FS (full search), TSS (three step search) [67], NTSS (New Three Step Search), DS (diamond search), MVFAST [68], PMVFAST для тестових відеопослідовностей з малим бітрейтом та розміром блоку пошуку 16x16 пікселів і відповідною областю пошуку (таблиця 4.3). При цьому прискорення буде розраховуватись відносно алгоритму повного перебору, що б виключити вплив програмної реалізації та технічних характеристик пристрою.

Таблиця 4.3 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим бітрейтом

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Hall-Monitor(10).							
PSNR	30.36	29.71	29.72	30.33	30.3	30.33	**30.35**
Speed-Up	1	40	56	78	236	379	**406**

Продовження таблиці 4.4 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим бітрейтом

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Mother & Daughter(24)							
PSNR	34.81	34.78	34.78	34.75	34.74	34.8	**34.81**
Speed-Up	1	40	55	73	208	313	**336**
Відеопослідовність Coastguard (48)							
PSNR	28.89	28.78	28.75	28.72	28.86	28.88	**28.88**
Speed-Up	1	40	40	57	103	156	**167**
Відеопослідовність Coastguard (112)							
PSNR	27.04	26.69	26.62	26.43	27.04	27.12	**27.14**
Speed-Up	1	40	36	48	98	121	**125**
Відеопослідовність Foreman (112)							
PSNR	30.04	29.47	29.56	29.59	29.93	29.94	**30.01**
Speed-Up	1	40	36	44	84	105	**113**

З результатів видно, що запропонований алгоритм значно швидший за всі аналоги, одночасно досягаючи подібних і в більшості випадків значно кращих PSNR. В середньому, для послідовностей, досліджених у цьому тесті, TFMVFAS приблизно на 7.46% швидше ніж найближчий аналог, при цьому його PSNR максимум на 0.1dB менше ніж PSNR повного перебору.

Розглянемо результати роботу алгоритму на відепопослідовностях з середнім бітрейтом (таблиця 4.4).

Як видно з таблиці, для середніх бітрейтів, алгоритм TFMVFAS на 7.6% швидше ніж найближчий аналог, при цьому його PSNR навіть стало більше ніж PSNR повного перебору, що пояснюється відносною неточністю метрики SAD.

Таблиця 4.5 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім бітрейтом

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Foreman (512)							
PSNR	34.51	33.90	34.02	34.03	34.4	34.56	**34.55**
Speed-Up	1	40	41	46	96	137	**148**
Відеопослідовність Foreman (1024)							
PSNR	35.47	34.81	34.91	34.98	35.39	35.53	**35.57**
Speed-Up	1	40	43	54	123	202	**218**
Відеопослідовність Table tennis (1024)							
PSNR	34.98	34.73	34.85	34.9	34.91	34.96	**35**
Speed-Up	1	40	51	67	184	311	**333**

Розглянемо, як на роботу запропонованого методу вплине збільшення розміру блоку та відповідно області пошуку до 32x32 пікселів(таблиця 4.5).

Таблиця 4.6 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім та низьким бітрейтом для області пошуку 32x32 пікселя

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Coastguard (112)							
PSNR	27.04	26.71	26.67	26.43	27.09	27.14	**27.15**
Speed-Up	1	154	141	183	374	430	**465**
Відеопослідовність Foreman (112)							
PSNR	30.38	29.53	29.59	29.65	30.17	30.25	**30.32**
Speed-Up	1	154	143	157	305	384	**415**

Продовження таблиці 4.5 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім та низьким бітрейтом для області пошуку 32x32 пікселя

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Foreman (512)							
PSNR	34.85	33.91	34.03	34.07	34.72	34.89	**34.91**
Speed-Up	1	156	153	173	366	526	**566**
Відеопослідовність Foreman (1024)							
PSNR	35.56	34.8	34.93	34.99	35.43	35.61	**35.61**
Speed-Up	1	156	169	208	474	763	**831**
Відеопослідовність Table tennis (1024)							
PSNR	35.01	34.73	34.84	34.91	34.93	34.98	**35.00**
Speed-Up	1	155	191	258	686	1165	**1274**

Як видно з таблиці, при збільшенні розміру блоків та області пошуку, прискорення отримане від запропонованого алгоритму росте до позначки 8.3%, а максимальна різниця між бітрейтом запропонованого методу і алгоритму повного перебору становить 0.06dB, що знову ж таки є мінімальним серед всіх розглянутих алгоритмів. Знову ж таки приріст PSNR для окремих відеопослідовностей пояснюється відносно низькою точністю метрики SAD.

Збільшимо розмір блоків для обробки описаних в таблиці 2.5 відеопослідовностей до 64x64 пікселів (таблиця 4.6)

Як бачимо з даних таблиці, тенденція попередніх розрахунків продовжується і алгоритм TFMVFAS показує найкращі результати як в плані прискорення так і в планів трати PSNR відносно алгоритму повного перебору.

Таблиця 4.7 - Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім та низьким бітрейтом для області пошуку 64x64 пікселя

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Coastguard (112)							
PSNR	27.13	26.09	26.15	26.31	26.83	26.11	**27.13**
Speed-Up	1	596	548	722	1439	1616	**1751**
Відеопослідовність Foreman (112)							
PSNR	30.69	29.55	29.58	29.65	29.83	30.68	**30.69**
Speed-Up	1	601	556	621	1174	1443	**1555**
Відеопослідовність Foreman (512)							
PSNR	35.21	33.91	33.94	35.01	34.72	35.21	**35.21**
Speed-Up	1	598	595	667	1409	1981	**2151**
Відеопослідовність Foreman (1024)							
PSNR	35.62	34.26	34.34	34.42	34.82	35.61	**35.63**
Speed-Up	1	596	657	825	1824	2861	**3112**
Відеопослідовність Table tennis (1024)							
PSNR	35.15	33.81	34.89	34.96	34.75	34.14	**35.14**
Speed-Up	1	599	743	1021	2641	4380	**4757**

Проведемо останній тест запропонованого нами алгоритму TFMVFAS для відеопослідовностей з високим бітрейтом, також збільшимо розмір блоку та області пошуку до 64x64 пікселів (таблиця 4.7).

Як видно з таблиці, для високих бітрейтів і розміру блоку 64x64пікселя, середнє прискорення алгоритму становить в середньому, 9% відносно найближчого аналогу. Варто відзначити, що зі збільшенням бітрейту, різниця в PSNR між TFMVFAS та алгоритмом повного перебору виросла до 0.34dB, при цьому це найнижчий показник серед всіх аналогів.

Таблиця 4.8 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з високим бітрейтом

Показник	FS	TSS	NTSS	DS	MVFAST	PMVFAST	**TFMVFAS**
Відеопослідовність Basket (4000)							
PSNR	26.72	25.91	25.94	26.04	26.41	26.51	**26.64**
Speed-Up	1	605	526	587	1323	1922	**2093**
Відеопослідовність Basket (9000)							
PSNR	30.73	29.89	29.93	29.94	30.43	30.49	**30.61**
Speed-Up	1	604	527	593	1364	2043	**2231**
Відеопослідовність Bus (4000)							
PSNR	29.93	27.03	26.93	27.13	29.12	29.48	**29.59**
Speed-Up	1	601	543	625	1517	2687	**2926**
Відеопослідовність Bus (9000)							
PSNR	34.04	31.04	30.97	31.13	33.35	33.92	**34.01**
Speed-Up	1	602	543	644	1569	2913	**3178**

Висновки до розділу

В розділі було проведено аналіз останніх досліджень, та розібрано основи роботи найновіших методів блочного пошуку, на основі яких було запропоновано новий метод TFMVFAS (Three by Frame Motion Vector Field Adaptive Search – адаптивний пошук векторів руху заснований на часових шахах).

TFMVFAS був створений за рахунок оптимізації трьох основних кроків алгоритмів блочного пошуку:

1. Алгоритм формування набору предикатів на основі використання попереднього і наступного кадрів.
2. Алгоритм оцінки порогових умов зупинки пошуку, на основі використання інформації попереднього і наступного кадрів.
3. Адаптивне використання шаблонів пошуку для уточнення предикатів, в залежності від кута вектору предиката.

Порівняння роботи запропонованого алгоритму з існуючими алгоритмами блочного пошуку такими як FS, TSS, NTSS, DS, MVFAST, PMVFAST на основі рекомендацій тестування, запропонованих Спеціальною групою MPEG з оцінки складності відеоінструментів. Всього було проведено 12 тестів, для відеопослідовностей з малим, середнім та великим бітрейтом, для розміру блоку 16x16, 32x32 та 64x64 пікселі. При цьому, в кожному з тестів запропонований алгоритм показував кращі результати ніж його аналоги як в тому що стосується PSNR так і в тому, шо стосується швидкості знаходження векторів руху. Так для блоків розміру 16x16 та відеопослідовностей з малим бітрейтом (10-112), TFMVFAS показав приріст швидкості 7.46% від найближчого аналогу яким виявився PMVFAST, для блоків розміру 16x16 та відеопослідовностей з середнім бітрейтом (512-1024) приріст склав - 7.6%, для блоків розміром 32x32 та відеопослідовностей з низьким та середнім бітрейтом – 8.3%, і для блоків розміром 64x64 для відеопослідовностей з великим бітрейтом(4000-9000) приріст склав 9%. При цьому, для кожної з відеопослідовностей, запропонований алгоритм показав найкращі показники PSNR з максимальним відхиленням для відеопослідовності Foreman(112) – 0.06dB що на 0.07dB більше ніж у найближчого аналога. Високих показників PSNR вдалось досягти за рахунок використання суперпозиції більшої кількості векторів руху суміжних блоків, в той час як основне прискорення отримано за рахунок ранніх зупинок пошуку на етапі другої ітерації – врахування інформації не тільки попереднього, а й наступного кадру. Таким чином, було сформовано метод

блочного пошуку, який дозволить усунути проблему невизначеності блоків та впливу випадкових співпадінь значень блоків на формування вектору руху.

Одним з можливих недоліків TFMVFAS є додаткова пам'ять, яка може знадобитися для зберігання векторів руху або значень SAD. У випадку векторів руху немає необхідності в додатковій пам'яті, оскільки можна повторно використовувати поточну пам'ять вектору руху, виділену в кодері. Досить важливо, щоб ця пам'ять не ініціалізувалась повторно в кінці кожного кадру, таким чином забезпечуючи доступ до значення вектору руху кодокованого блоку. У досить екстремальному випадку з роздільною здатністю відео 2048 × 2048 пікселів, і якщо для зберігання одного вектору руху у формі (MV_x , MV_y) потрібні 4 байти, то загалом потрібно 64 КБ. Для цієї роздільної здатності для значень SAD кадру потрібно додатково 32 КБ. Однак це число є відносно невеликим, особливо якщо врахувати, що в цьому крайньому випадку 3 кадри потребують загалом 18 МБ. Якщо потрібен відносно невеликий діапазон пошуку (менше ± 128), пам’ять, необхідна для зберігання векторів руху, може зменшитися вдвічі. При цьому варто відзначити, що автор не ставляв перед собою ціль створити найбільш ефективну програмну реалізацію продукту, залишаючи це для подальших досліджень.

5 ФОРМУВАННЯ МЕТОДИКИ КОМПЕНСАЦІЇ РУХУ НА ОСНОВІ АЛГОРИТМУ TSADL І МЕТОДУ TCMVFAS

Як було показано в розділі 4.4 метод TFMVFAS дозволяє отримати значення PSNR близькі до значень отриманих алгоритмом повного перебору, за рахунок використання розширеного набору предикатів, що дуже важливо для використання шаблонів порівняння TSAD і HSAD. Наступним кроком формування результуючої методики компенсації руху є використання шаблону TSAD для обрахунку метрики схожості зображень, умов ранньої зупинки пошуку а також використання алгоритмів виділення границь об’єктів для обробки областей з високою ентропією зображення. Назвемо результуючу методику - TFEMVFAS(Three by Frame Edge Map Motion Vector Field Adaptive Search – адаптивний алгоритм визначення векторів руху один через три за картою границь), яка включає в себе наступні ітерації:

1. Обробка другого кадру відеопослідовності для формування опорного набору векторів руху.
    1.1. Попередня оцінка кадру. На цьому етапі, алгоритм знаходить абсолютну нескомпенсовану різницю першого і другого кадрів відеопослідовності, після чого вектор руху блоків, максимальна різниця в піксельному значенні кожного окремого пікселя яких не перевищує порогу $D_{2\%}$, вважаються нульовими і виключаються з подальшого розгляду.
    1.2. Знаходимо вектори руху для решти блоків методом, який дає найкращі результати в плані PSNR і не спирається на значення векторів руху блоків рознесених в часі, наприклад PMVFAST без використання медіанного вектору суміжних блоків з попереднього кадру.

2. Обрахунок векторів руху для кожного третього блоку, кожного кадру(зі зміщенням шаблону кожен третій) на основі інформації з попереднього кадру на основі метрики що дає найкращий результат по значенню PSNR – наприклад SAD.
   2.1. Попередня обробка кадру, аналогічна до тої що проводилась на першій ітерації методики.
   2.2. Обраховуємо третину векторів руху поточного кадру(за шаблоном кожен третій) з предикатами за «Набором А», на основі метрики SAD.
   2.3. Зміщуємо шаблон та повторяємо пункт 2.2. для решти кадрів.
3. Виділення країв для всіх кадрів відеопослідовностей, для областей з ненульовими векторами руху.
4. Обрахунок векторів руху невизначених блоків на основі методу TFMVFAS(інформацією попереднього і наступного кадрів) і метрики TSAD. Варто відзначити, що дана ітерація може бути здійснена за одним з двох сценаріїв, перший з яких дає більший степінь прискорення але гірші показники PSNR, другий навпаки:
   4.1. На карті країв, знаходимо вектор руху, шляхом почергового перебору предикатів з набору TFMVFAS, з умовою ранньої зупинки пошуку представленою формулою 4.2. Таку модифікацію методики будемо позначати - TFEMVFASS (-TFEMVFA speed search – швидкий пошук).
   4.2. На карті країв, проводимо порівняння векторів руху, на які вказують перші п’ять предикатів з набору TFMVFAS і зупиняємось на тому, метрика TSAD є мінімальною і задовольняє умову ранньої зупинки пошуку (формула 4.2), якщо такого не знаходимо проводимо порівняння блоків на які

вказують решта предикатів. Таким чином, було проведено обрахунок як мінімум метрик TSAD(в шаблонах малий і еластичний ромб 5 точок перевірки). Таку модифікацію методики будемо позначати - TFEMVFAAS (- TFEMVFA accuracy search – точний пошук).

Також варто відзначити, що описана вище методика, підходить як для обробки вже існуючою відеопослідовності, так і для роботи з відеопослідовностями в реальному часі, пункт 2-4 буде повторюватися для кожної пари кадрів, а не для всієї відеопослідовності відразу, що не має вплинути на час чи значення PSNR для методики – TFEMVFASO (- TFEMVFAS online).

Проведемо тестування запропонованих модифікацій методики TFEMVFAS. Тестування як і до цього будемо проводити на відеопослідовностях з різними бітрейтами та на основі блоків трьох різних розмірів 16x16, 32x32 і 64x64 пікселів з відповідними розмірами області пошуку. При цьому приберемо з порівняння методи які показали себе найгірше в розділі 3.4 а саме TSS, NTSS, DS і MVFAST, залишивши лише методи з найкращими показниками прискорення або PSNR: FS, PMVFAST, TFMVFAS, при цьому в якості метрики всіх методів крім запропонованих нами в цьому розділі буде виступати SAD(розрахована по всіх точках блоку). Також в даному розділі будемо проводити порівняння часу та значень PSNR запропонованих методик не тільки відносно еталонного алгоритму FS а й відносно найближчих аналогів, оскільки алгоритм FS майже не використовується при реальному стисненні відеопослідовностей.

Спочатку проведемо розрахунок відеопослідовностей з малим бітрейтом(10-112) для блоків та області пошуку розміром 16x16 пікселів(таблиця 5.1).

Таблиця 5.1 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим бітрейтом

Показник	FS	PMVFAST	**TFMVFAS**	**TFEMVFAAS**	**TFEMVFASS**
Відеопослідовність Hall-Monitor(10).					
PSNR	30.36	30.33	**30.35**	**29.99**	**29.72**
Seed-Up	1	379	**406**	**437**	**547**
Відеопослідовність Mother & Daughter(24)					
PSNR	34.81	34.8	**34.81**	**34.41**	**34.09**
Seed-Up	1	313	**336**	**361**	**398**
Відеопослідовність Coastguard (48)					
PSNR	28.89	28.88	**28.88**	**28.55**	**28.25**
Seed-Up	1	156	**167**	**181**	**199**
Відеопослідовність Coastguard (112)					
PSNR	27.04	27.12	**27.14**	**26.73**	**26.45**
Seed-Up	1	121	**130**	**139**	**154**
Відеопослідовність Foreman (112)					
PSNR	30.04	29.94	**30.01**	**29.71**	**29.41**
Seed-Up	1	105	**113**	**122**	**134**

Як бачимо з даних таблиці, використання шаблонів порівняння дозволило скоротити час обрахунку на 15.6% для методики TFMVFAAS і на 27.5% для методики TFMVFASS. При цьому середня втрата PSNR для TFMVFAAS склала 1.15% тоді як для TFMVFASS 2.1%. При цьому максимальна втрата склала 0.4dB для TFMVFAAS і 0.63dB для TFMVFASS відповідно для відеопослідовності Mother & Daughter(24), варто відзначити що аналогічна втрата при використанні алгоритму повного перебору для шаблону TSAD з виділенням країв становила 1.09dB, тобто приріст склав 63.3 і 42.3% відповідно.

Розглянемо вплив бітрейту на результати роботи запропонованих алгоритмів, для відеопослідовностей з середнім бітрейтом(512-1024) та розміром блоку 16x16пікселів (таблиця 5.2).

Таблиця 5.2 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з середнім бітрейтом

Показник	FS	PMVFAST	**TFMVFAS**	**TFEMVFAAS**	**TFEMVFASS**
Відеопослідовність Foreman (512)					
PSNR	34.51	34.56	**34.55**	**34.19**	**33.76**
Seed-Up	1	137	**148**	**159**	**175**
Відеопослідовність Foreman (1024)					
PSNR	35.47	35.53	**35.57**	**35.07**	**34.73**
Seed-Up	1	202	**218**	**233**	**261**
Відеопослідовність Table tennis (1024)					
PSNR	34.98	34.96	**35**	**34.58**	**34.21**
Seed-Up	1	311	**333**	**360**	**399**

З таблиці 5.2 можна зробити висновок, що зростання бітрейту відеопослідовностей призводить до незначного зростання швидкості знаходження векторів руху, за рахунок зниження загальної кількості шуму на зображенні. При цьому втрати PSNR залишаються на тому ж рівні, що й для відеопослідовностей з низьким бітрейтом. Перевіримо як на результати роботи запропонованих методик вплине збільшення розміру блоку та області пошуку з 16x16 до 32x32 блоків (таблиця 5.3).

Як можна побачити з даних таблиці, збільшення розміру блоку майже не впливає на відносне збільшення швидкості відносно найближчого з аналогів (для відеопослідовності Foreman(1024) обробленої блоками 16x16 приріст швидкості для методики TFMVFAAS – 15% , TFMVFASS – 27% ,

тоді як для цієї ж відеопослідовності обробленої блоками розміром 32x32, 15% і 27.1% відповідно).

Таблиця 5.3 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим та середнім бітрейтом для обробки блоками 32x32пікселя

Показник	FS	PMVFAST	**TFMVFAS**	**TFEMVFAAS**	**TFEMVFASS**
Coastguard (112)					
PSNR	27.04	27.14	**27.15**	**26.93**	**26.55**
Seed-Up	1	430	**465**	**499**	**551**
Foreman (112)					
PSNR	30.38	30.25	**30.32**	**30.14**	**29.85**
Seed-Up	1	384	**415**	**447**	**491**
Foreman (512)					
PSNR	34.85	34.89	**34.91**	**34.69**	**34.25**
Seed-Up	1	526	**566**	**611**	**679**
Foreman (1024)					
PSNR	35.56	35.61	**35.61**	**35.43**	**34.94**
Seed-Up	1	763	**831**	**878**	**984**
Table tennis (1024)					
PSNR	35.01	34.98	**35.00**	**34.87**	**34.36**
Seed-Up	1	1165	**1274**	**1353**	**1498**

Також збільшення блоків доволі помітно вплинуло на значення PSNR для запропонованих метрик (для відеопослідовності Foreman(1024) обробленої блоками з розміром 32x32 втрата в PSNR відносно алгоритму FS склала 0.36% для TFEMVFAAS і 0.1.8% для TFEMVFASS, в той час коли для блоків 16x16 втрати склали 1,13% і 2,1% відповідно. При цьому, при

збільшенні розміру блоків до 64x64, тенденція зменшення втрати PSNR має зберегтись. Максимальні значення PSNR для запропонованої метрики наведено в таблиці 5.4.

Таблиця 5.4 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з малим та середнім бітрейтом для обробки блоками 64x64пікселя

Показник	FS	PMVFAST	**TFMVFAS**	**TFEMVFAAS**	**TFEMVFASS**
Coastguard (112)					
PSNR	27.13	27.11	**27.13**	**27.04**	**26.82**
Seed-Up	1	1616	**1751**	**1877**	**2073**
Foreman (112)					
PSNR	30.69	30.68	**30.69**	**30.57**	**30.36**
Seed-Up	1	1443	**1555**	**1678**	**1859**
Foreman (512)					
PSNR	35.21	35.21	**35.21**	**34.13**	**34.83**
Seed-Up	1	1981	**2151**	**2308**	**2548**
Foreman (1024)					
PSNR	35.62	35.61	**35.63**	**35.55**	**35.25**
Seed-Up	1	2861	**3112**	**3331**	**3685**
Відеопослідовність Table tennis (1024)					
PSNR	35.15	34.14	**35.14**	**35.03**	**34.75**
Seed-Up	1	4380	**4757**	**5087**	**5641**

Знову ж таки, як видно з даних, наведених в таблиці 5.4, прискорення роботи алгоритму мало залежить від розміру блоків, при цьому, як було зазначено вище, мінімальне спотворення для запропонованих методик складає: 0.19% для TFEMVFAAS і 1.1% для TFEMVFASS.

Розглянемо результати обробки запропонованими методиками відеопослідовностей з високим бітрейтом, блоками 64x64 пікселя.

Таблиця 5.5 – Значення параметрів PSNR та пришвидшення для тестових відеопослідовностей з великим бітрейтом для обробки блоками 64x64пікселя

Показник	FS	PMVFAST	**TFMVFAS**	**TFEMVFAAS**	**TFEMVFASS**
Basket (4000)					
PSNR	26.72	26.51	**26.64**	**26.58**	**26.39**
Seed-Up	1	1922	**2093**	**2243**	**2489**
Basket (9000)					
PSNR	30.73	30.49	**30.61**	**30.55**	**30.36**
Seed-Up	1	2043	**2231**	**2387**	**2663**
Відеопослідовність Bus (4000)					
PSNR	29.93	29.48	**29.59**	**29.53**	**29.32**
Seed-Up	1	2687	**2926**	**3137**	**3469**
Відеопослідовність Bus (9000)					
PSNR	34.04	33.92	**34.01**	**33.97**	**33.63**
Seed-Up	1	2913	**3178**	**3409**	**3779**

Як бачимо для відеопослідовностей Basket і Bus, маємо доволі великі втрати PSNR відносно алгоритму повного перебору, при цьому варто відзначити, що й найближчі аналоги показали не найкращі показники для цих відеопослідовностей, що може буди обумовлено великою міжкадровою різницею. Так якщо порівнювати результати запропонованих методик, відносно існуючого аналогу PMVFAST то методика TFEMVFASS показала навіть кращі результати в плані PSNR при відносному прискоренні на рівні 16.7%, тоді як середнє відставання в значенні PSNR для методики TFEMVFAAS склало 0.15dB, а середній виграш в швидкості склав 29.5%. Такий приріст PSNR є наслідком використання блоків розміром 64x64

пікселі, для яких найкраще проявляється унікальність рисунку шаблону TSAD.

Останнім етапом тестування є обробка відеопослідовності «Hearthstone», яка має максимальний бітрейт серед розглянутих відеопослідовностей: 24822 кбіт/с, запропонованими методиками на основі блоків різного розмірів (таблиця 5.6).

Таблиця 5.6 – Значення параметрів PSNR та пришвидшення для відеопослідовності «Hearthstone» з бітрейтом 24822 кбіт/с при обробці різними розмірами блоків

Показник	FS	PMVFAST	**TFMVFAS**	**TFMVFAAS**	**TFMVFASS**
Hearthstone 16x16					
PSNR	37.43	37.24	**37.33**	**36.98**	**36.64**
Seed-Up	1	203	**221**	**237**	**261**
Hearthstone 32x32					
PSNR	37.97	37.81	**37.92**	**37.71**	**37.32**
Seed-Up	1	789	**859**	**925**	**1034**
Hearthstone 64x64					
PSNR	38.31	38.17	**38.25**	**38.19**	**37.89**
Seed-Up	1	3008	**3281**	**3528**	**3948**

З таблиці 5.6 можна зробити кілька висновків:

1. Збільшення бітрейту відеопослідовності призводить до зменшення часу обрахунку векторів руху запропонованими методиками, оскільки процес виділення країв краще себе показує на зображеннях з низьким рівнем шуму. Максимальне прискорення відносно методики PMVFAST склало 17.2%, з показниками PSNR на 0.05dB вищими і 31.1% при втраті 0.03dB значення PSNR.

2. Збільшення розміру блоків, які використовуються в методиці призводить до зменшення втрат PSNR з максимальних 2.1% для блоків розміром 16x16 до 1.1% для блоків розміром 64x64 методики TFMVFASS і з 1.2% до 0.3% для TFMVFAAS.

Висновки до розділу

В даному розділі було сформовано методику на основі поєднання розробок отриманих в розділах 2,3 і 4, а саме:

1. Алгоритму оцінки схожості зображень TSADL який формується на основі використання шаблону порівняння «кожен третій» для розрахунку метрики SAD(суми абсолютних різниць), що дозволяє суттєво скоротити час обрахунку метрики за рахунок втрати PSNR.
2. Алгоритму виділення країв LoG (Лапласіан Гаусіана), який дозволяє підвищити показники PSNR стисненої відеопослідовності за рахунок усунення невизначеності в областях з високою ентропією зображення.
3. Методу блочного пошуку TFMVFAS, особливістю якого є використання інформації як з попереднього так і з наступного кадрів відеопослідовності для побудови наборів векторів предикатів та формування умов ранньої зупинки пошуку. Такий підхід дозволяє отримати незначний виграш в швидкості обробки відеопослідовності(за рахунок збільшення точності предикатів) та помітний приріст в значення PSNR, що є великим недоліком використання шаблонів порівняння.

В результаті, було сформовано методику з двома модифікаціями:

1. TFEMVFAAS – яка дозволяє отримати суттєвий виграш в значеннях PSNR за рахунок проведення додаткових уточнень серед блоків з отриманих на основі наборів предикатів. Виграш у швидкості

методики залишається на рівні 15-17% відносно найближчого з аналогів PMVFAST з максимальними втратами на рівні 1.2 – 0.3% в залежності від розмірів блоків, що використовуються для розрахунку.

2. TFEMVFASS – яка зосереджена на отриманні максимального прискорення шляхом жертвування значеннями PSNR. Для цієї модифікації, прискорення залишалось на рівні 27-31% відносно PMVFAST з максимальними втратами 2.1-1.1% значення PSNR відносно алгоритму повного перебору.

ВИСНОВКИ

В результаті дисертаційного дослідження вирішено актуальне наукове завдання: підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху.

На підставі проведеного аналізу сучасних технологій стиснення відеопотоку, було виявлено, що окремі частини методів та алгоритмів методики компенсації руху(яка використовується в більшості досліджених систем), використовують надлишковий час для процесу визначення векторів руху на двох основних етапах – процесі оцінки схожості блоків і при обході області пошуку для знаходження векторів руху. Оптимізація методики компенсації руху на цих етапах, дозволить скоротити загальний час кодування відеопотоку і як наслідок затримку при передачі відеофайлів, що підтверджує актуальність наукового дослідження.

В дисертації отримані наступні наукові результати:

1. Удосконалено алгоритм оцінки схожості зображень на основі суми абсолютних різниць, за рахунок використання шаблонів порівняння за характерними точками. Такий підхід дозволяє суттєво скоротити час розрахунку метрики SAD, як наслідок меншої кількості точок. Варто відзначити, що це також призводить до погіршення PSNR стиснутого відеопотоку.
2. Вперше запропоновано використовувати алгоритми виділення країв, для збільшення значення PSNR стиснутого відеопотоку, за рахунок уникнення втрати важливої інформації в областях з високою ентропією зображення(на границях об’єктів та в областях різкого переходу кольору).
3. Удосконалено методи блочного пошуку, за рахунок використання часової постійності руху на коротких проміжках часу і як наслідок використання інформації з наступного кадру відеопослідовності для

формування наборів предикатів та умов раннього пошуку, а також адаптивних шаблонів уточнення, що найкраще враховують особливості руху.

4. Вперше розроблено методику TFEMVFAS (Three by Frame Edge Map Motion Vector Field Adaptive Search – адаптивний алгоритм визначення векторів руху один через три за картою границь), в двох модифікаціях, в двох модифікаціях A (accuracy – точність) яка дає менший приріст в швидкості, при мінімальних втратах значення PSNR стиснутої відеопослідовності і S(speed – швидкість), яка дає суттєвий приріст швидкості за рахунок більших втрат значення PSNR. Результуючу методику протестовано в двох модифікаціях, згідно комплексних рекомендацій тестування міжнародної комісії MPEG по оцінці складних відеоінструментів, на прикладі 13 відеопослідовностей, з різними бітрейтами. Порівняння результатів роботи запропонованої методики було проведено з сучасними методами формування векторів руху: FS, TSS, NTSS, DS, MVFAST, PMVFAST, в результаті чого було визначено що запропонована методика дозволяє отримати прискорення на рівні 15-17%(рівень прискорення залежить від бітрейту оброблюваної відеопослідовності: вищий бітрейт – вище прискорення) від найближчого аналога при втратах на рівні 0.3-1.2%значень PSNR(втрати PSNR залежать від розміру блоків для яких проводився пошуку, максимальна втрата для блоків 16x16пікселів, мінімальна для блоків 64x64пікселів) від значень еталонного методу пошуку векторів руху для модифікації A і прискорення на рівні 27-31% при втраті 1.1%-2.1% в залежності розміру блоку та бітрейту відеопослідовності, для модифікації S.

Результати досліджень прийняті до впровадження в ТОВ «Vega» в ТОВ «НТЦ Енергозв'язок», в ТОВ «СУРВАЙВАЛ, ЕКСПЛОРЕР ЕНД КРАФТ

ГЕЙМЗ» а також використовуються в навчальному процесі кафедри інженерії програмного забезпечення Державного університету телекомунікацій при викладанні дисциплін «Розробка комп'ютерних ігор» та «Програмування мобільних пристроїв» для студентів спеціальності 121 «програмна інженерія» денної форми навчання.

Мета дисертаційних досліджень щодо підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху і всі часткові завдання вирішено. Наукові результати є корисним внеском у розвиток інформаційних технологій, для розроблення технологій стиснення відеофайлів.

Перспективними шляхами подальших досліджень у зазначеному напрямку може бути коло питань з використанням блоків адаптивного розміру для удосконалення сучасних методик знаходження векторів руху для систем відео-стиснення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H.M. Hang, Y.M. Chou, S.C. Cheng, “Motion Estimation for Video Coding Standards”. Journal of VLSI Signal Processing 17, pp.113–136– 2001.
2. A.N. Netravali, B. Prasada. “Visual Communications Systems”, IEEE Press, New York, NY, – 2003.
3. K. Zhang, Y.W. Chen, L. Zhang, W.J. Chien, M. Karczewicz. “An Improved Framework of Affine Motion Compensation in Video Coding”. IEEE Transactions on Image Processing, Volume: 28, Issue: 3, pp.1456 – 1569 – 2019.
4. M. Karczewicz, J. Niewęgłowski, P. Haavisto. “Video coding using motion compensation with polynomial motion vector fields”. Signal Process. Image Commun., vol. 10, no. 1, pp. 63-91 – 2007.
5. H. Yang, S. Liu, K. Zhang. “CE4: Summary Report on Inter Prediction and Motion Vector Coding” – 2018.
6. S. Parker, Y. Chen, D. Barker, P. de Rivaz., D. Mukherjee. “Global and locally adaptive warped motion compensation in video compression” Proc. IEEE Int. Conf. Image Process. (ICIP), pp. 275-279 – 2017.
7. J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand. "Comparison of the coding efficiency of video coding standards—Including high efficiency video coding (HEVC)", IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1669-1684 – 2012.
8. A. Segall, V. Baroncini, J. Boyce, J. Chen and T. Suzuki. “Joint Call for Proposals on Video Compression with Capability Beyond HEVC – 2017
9. Huang H, J. W. Woods, Y. Zhao and H. Bai, "Control-point representation and differential coding affine-motion compensation", IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 10, pp. 1651-1660 – 2013.
10. A.M. Tourapis, O.C. Au, and M.L. Liou. “Fast Block-Matching Motion Estimation using Predictive Motion Vector Field Adaptive Search

Technique (PMVFAST),” in ISO/IEC JTC1/SC29/WG11 MPEG2000/M5866, Noordwijkerhout, NL – 2000.

11. A.M. Tourapis, O.C. Au, M.L. Liou, and G. Shen. “An Advanced Zonal Block Based Algorithm for Motion Estimation”, Proc. of IEEE Int. Conf. On Image Processing (ICIP’99), section 26PO3.1, Kobe, Japan – 1999.
12. A. M. Tourapis, O. C. Au, M.L. Liou, G. Shen, and I. Ahmad, “Optimizing the Mpeg-4 Encoder – Advanced Diamond Zonal Search”, in Proc. of 2000 IEEE Inter. Symposium on Circuits and Systems (ISCAS-2000), Geneva, Switzerland, May, 2000.
13. O.A. Dibrivniy, “Usage of characteristic points to reduce time of performance of sad metric calculation during video flow compression by motion compensation method, Телекомунікаційні та інформаційні технології №1, К.: ДУТ pp. 19 - 19– 2021.
14. O.A. Dibrivniy, V.V. Grebenyuk. “Sum of absolute differences algorithm improvement by characteristic points usage. Modern engineering and innovative technologies”, Vol 15, pp.11-11– 2021.
15. O.A. Dibrivniy, V.V. Grebenyuk. “Usage of edge selection methods to refine candidate blocks in the motion compensation process, based on the SAD algorithm using characteristic points”. Sciences of Europe, VOL 1, No 63, pp. 19-24 – 2021.
16. О.А. Дібрівний, В.В. Гребенюк, О.В. Негоденко, А.П. Бондарчук. “Порівняльний аналіз нереферентних методів оцінки якості відеоматеріалу”. Журнал Зв’язок №6 13-13ст. – 2020.
17. О.А. Дібрівний, Л.П. Крючкова, В.І. Стрельников, М.В. Акулінічева, О.С. Бортник. “Алгоритми виділення контурів зображень об’єктів в інтелектуальних системах відеоспостереження”. Журнал Зв’язок №5 11-12ст. – 2020.
18. О.А. Дібрівний, Д.В. Сорокін, О.В. Сеньков, А.П. Бондарчук “Нові сфери застосування сучасних інформаційних мереж”. Науково

практична конференція «Проблеми комп'ютерної інженерії». Збірник тез К.:ДУТ 73-34ст. – 2020.

19. О.А. Дібрівний, В.О. Кисельов. "Розробка десктопного додатку для перегляду відеофайлів на мові програмування C#". XI науково-технічна кофнференція студентів та молодих вчених, Збірник тез К.:ДУТ 39-40 – 2020.
20. О.А. Дібрівний, Д.О. Криворученко. "Розробка гри "Layers" в жанрі головоломки за допомогою ігрового двигуна Unity". Всеукраїнська науково-технічна конференція "Сучасний стан та перспективи розвитку IoT". Збірник тез. К.: ДУТ 29-30 – 2020.
21. R. Armes. "On Video Studies in Culture and Communication". Routledge pp. 31-33 – 2006
22. J. D. Slepian, J. K. Wolf. "Noiseless coding of correlated information sources", IEEE Trans. Inf. Theory, vol. IT-19, pp. 471-480 – 1973.
23. L. Yufeng, X. Jufei, W. Wei. "Motion Estimation Based On H.264 Video Coding", International Congress on Image and Signal Processing(CISP), International Conference on IEEE – 2012.
24. Q. Wang, R. J. Clarke, "Motion estimation and compensation for image sequence coding", Image Commun., vol. 4, no. 2, pp. 161-174, - 1992.
25. T. Sikora. "MPEG digital video-coding standards" IEEE Signal Processing Magazine, Volume: 14, Issue: 5, pp. 82 – 100 - 1997
26. Q. Wang, R. J. Clarke, "Motion estimation and compensation for image sequence coding", Image Commun., vol. 4, no. 2, pp. 161-174, - 1992.
27. Р. Н. Квєтний, Є. М. Бочков. "Метод виділення контурів на основі вейвлет-перетворення з використанням двовимірних фільтрів" Інформаційні технології та комп'ютерна інженерія №3, В.:ВНТУ ст. 26-34 – 2010.

28. "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 -10 AVC", Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG JVT-G050 - 2003.
29. S.T. Wiegand, G. J. Sulivan, and A. Luthra. “Draft ITU-T Recommendation H.264 and Final Draft International Standard 14496-10 AVC”. JVT of ISO/IEC JTC1/SC29/WG11 and ITU-T SG16/Q. 6, Doc. JVT-G050r1, Geneva, Switzerland – 2003.
30. T. Stockhammer, M. M. Hannuksela, T. Wiegand, "H.264/AVC in wireless environments", IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 657-673, - 2003.
31. B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, T. Wiegand, “High Efficiency Video Coding (HEVC)” Text Specification Draft 9 - 2012.
32. Draft meeting report for 31st VCEG Meeting (Marrakech, MA, pp.15-16 – 2007.
33. H. Samet, "The quadtree and related hierarchical data structures", Comput. Survey, vol. 16, no. 2, pp. 187-260, - 1984.
34. Joint Call for Proposals on Video Compression Technology - 2010.
35. L. Chen, J. Cao, L. Tang, J.J. Huang, H. Guo X. Dong. “Optimized H.264 Motion Estimation Algorithm Based on UMHexagonS”, International Conference on Intelligent Control and Information Processing(ICICIP),2013 International Conference on. IEEE – 2012.
36. J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including High Efficiency Video Coding (HEVC)", IEEE Trans. Circuits Syst. Video Technol., vol. 22, pp. 1668-1683, - 2012.
37. Wiegand, G. J. Sullivan, G. Bjøntegaard, A. Luthra, "Overview of the H.264/AVC video coding standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 560-576, July 2003.

38. F. Bossen, B. Bross, K. Shring and D. Flynn, "HEVC complexity and implementation analysis", IEEE Trans. Circuits Syst. Video Technol., vol. 22, pp. 1684-1695, - 2012.
39. G. J. Sullivan and J.-R. Ohm, "Recent developments in standardization of High Efficiency Video Coding (HEVC)", Proc. 33rd SPIE Appl. Dig. Image Process., vol. 7798, pp. 7798-7830, - 2010.
40. M. Ezhilarasan, P. Thambidurai. "Simplified Block Matching Algorithm for Fast Motion Estimation in Video Compression " Journal of Computer Science, vol. 4, pp. 282-289 – 2008.
41. K. Mueller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic and T. Oelbaum, "Multi-view video coding on H.264/AVC using hierarchical B-frames", Proc. Picture Coding Symp. - 2006.
42. ITU-T and ISO/IEC, "Advanced Video Coding for Generic Audiovisual Services," H.264, MPEG, 14496-10 – 2003.
43. A. Barjatya, "Block Matching Algorithms for Motion Estimation," DIP 6620 Spring – 2004.
44. Implementation Study Group, “Experimental conditions for evaluating encoder motion estimation algorithms,” in ISO/IEC JTC1/SC29/WG11 MPEG99/n3141, Hawaii, USA, – 2000.
45. W. Gao, X. Zhang, L. Yang, H. Liu. “An improved Sobel edge detection”. 2010 3rd International Conference on Computer Science and Information Technology – 2010.
46. W. Ng, J. Li, S. Godsill, "A review of recent results in multiple target tracking[C]", II Proc. Of the IEEE International Conference on Robotics and Automation, pp. 3807-3812, - 2006.
47. W. Cheney. “Analysis for Applied Mathematics” Springer Science & Business Media/ 112-121 стор – 2001
48. C. Chang-Huang, R. Peishen, T. Shenkeng. “Edge detection based on class ratio”. 152, sec.3, 22202, Taiwan, R.O.C. – 2002.

49. F. Canny, “A computational approach to edge detection. IEEE Trans pattern Analysis and machine Intelligence” 679-698. - 1986
50. William McIlhagga, "The Canny Edge Detector Revisited, " International Journal of Computer Vision, no.91, pp.251-261 – 2011
51. X H Wang, Y J Qian, "An improved Canny edge detection algorithm," Mechanical and Electrical Engineering Magazine, vol.25, no.12, pp.60-63 – 2008
52. M. Sharifi; M. Fathy; M.T. Mahmoudi. “A classified and comparative study of edge detection algorithms”. Proceedings. International Conference on Information Technology: Coding and Computing. Las Vegas, NV, USA – 2002.
53. K.H. Jung, K.Y. Yoo “High-capacity index based data hiding method”. Multimed Tools Appl 74: pp. 2179–2193 – 2015.
54. S.Y. Shen, L.H. Huang. “A data hiding scheme using pixel value differencing and improving exploiting modification directions”. Comput Security 48: pp. 131–141 – 2015.
55. Y.-C. Lin, S.-C. Tai. “Fast full-search block-matching algorithm for motion-compensated video compression”. IEEE Transactions on Communications (Volume: 45, Issue: 5. pp. 527-531 – 1997.
56. S. Zhu, K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation", Proc. Int. Conf. Information Communications and Signal Processing (ICICS), vol. 1, pp. 292-296 - 1997
57. B. Demir, S. Erturk "Block motion estimation using adaptive modified two-bit transform", IET-IPR, vol. 1, pp. 215-222 – 2007
58. L. K. Liu, E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding", IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 419-423 - 1996.

59. S. Zhu, K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation" Image Processing, IEEE Transactions on, vol. 9, pp. 287-290 – 2000.
60. C.H. Cheung, L.M. Po, "A novel cross-diamond search algorithm for fast block motion estimation". IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 1168- 1177– 2002.
61. Y. Nie, K. K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation" IEEE Trans. Image Processing, vol. 11, pp. 1442-1448 – 2002.
62. P.I. Hosur, K.K. Ma, "Motion Vector Field Adaptive Fast Motion Estimation," Second International Conference on Information, Communications and Signal Processing (ICICS '99), Singapore, pp. 7-10 – 2009.
63. R. Li, B. Zeng, M.L. Liou. "A new three-step search algorithm for block motion estimation". IEEE Trans on Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 438-42 – 2004.
64. A. M. Tourapis, O. C. Au, M. L. Liou, "Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation," submitted to Circuits and Systems for Video Technology – 2000.
65. A.M. Tourapis, O.C. Au, and M.L. Liou "Predictive motion vector field adaptive search technique (PMVFAST): enhancing block-based motion estimation". Photonics West 2001 - Electronic Imaging, pp. 883-892 – 2001.
66. A.M. Tourapis, O.C. Au, M.L. Liou. "Core Experiment on Block Based Motion Estimation," in ISO/IEC JTC1/SC29/WG11 MPEG2000/M5867, Noordwijkerhout, – 2000.
67. A.M. Tourapis, O.C. Au, M.L. Liou. "Implementation of the Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) algorithm in

the Optimization Model 1.0," in ISO/IEC JTC1/SC29/WG11 MPEG2000/M6194, Beijing, China – 2000.

68. P. G. Barten. "Contrast sensitivity of the human eye and its effects on image quality". Bellingham, Washington: SPIE Optical Engineering Press, cop. 14-22pp. – 1999.

69. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro. "Motion compensated interframe coding for video conferencing". Proc. Nat. Telecommun. Conf., New Orleans, LA, pp. G5.3.1-G5.3.5 – 2005.

70. K.K. Ma, P.I. Hosur, "Performance Report of Motion Vector Field Adaptive Search Technique (MVFAST)," in ISO/IEC JTC1/SC29/WG11 MPEG99/m5851, Noordwijkerhout, NL, – 2000.

ДОДАТОК А

Наукові праці у яких опубліковані основні наукові результати дисертації:

6. Dibrivniy O.A., Grebenyuk V.V. Usage of edge selection methods to refine candidate blocks in the motion compensation process, based on the SAD algorithm using characteristic points. Sciences of Europe, VOL 1, No 63 (2021) –19-24 p.
7. Dibrivniy O.A., Grebenyuk V.V. Sum of absolute differences algorithm improvement by characteristic points usage. Modern engineering and innovative technologies, Vol 15(2021) –
8. Dibrivniy O.A., Usage of characteristic points to reduce time of performance of sad metric calculation during video flow compression by motion compensation method, Телекомунікаційні та інформаційні технології №1 2021 –
9. Дібрівний О.А., Гребенюк В.В., Негоденко О.В., Бондарчук А.П. Порівняльний аналіз нереферентних методів оцінки якості відеоматеріалу. Журнал Зв'язок №6(2020) –
10. Дібрівний О.А., Крючкова Л.П., Стрельников В.І., Акулінічева М.В., Бортник О.С. Алгоритми виділення контурів зображень об'єктів в інтелектуальних системах відеоспостереження. Журнал Зв'язок №5(2020) –

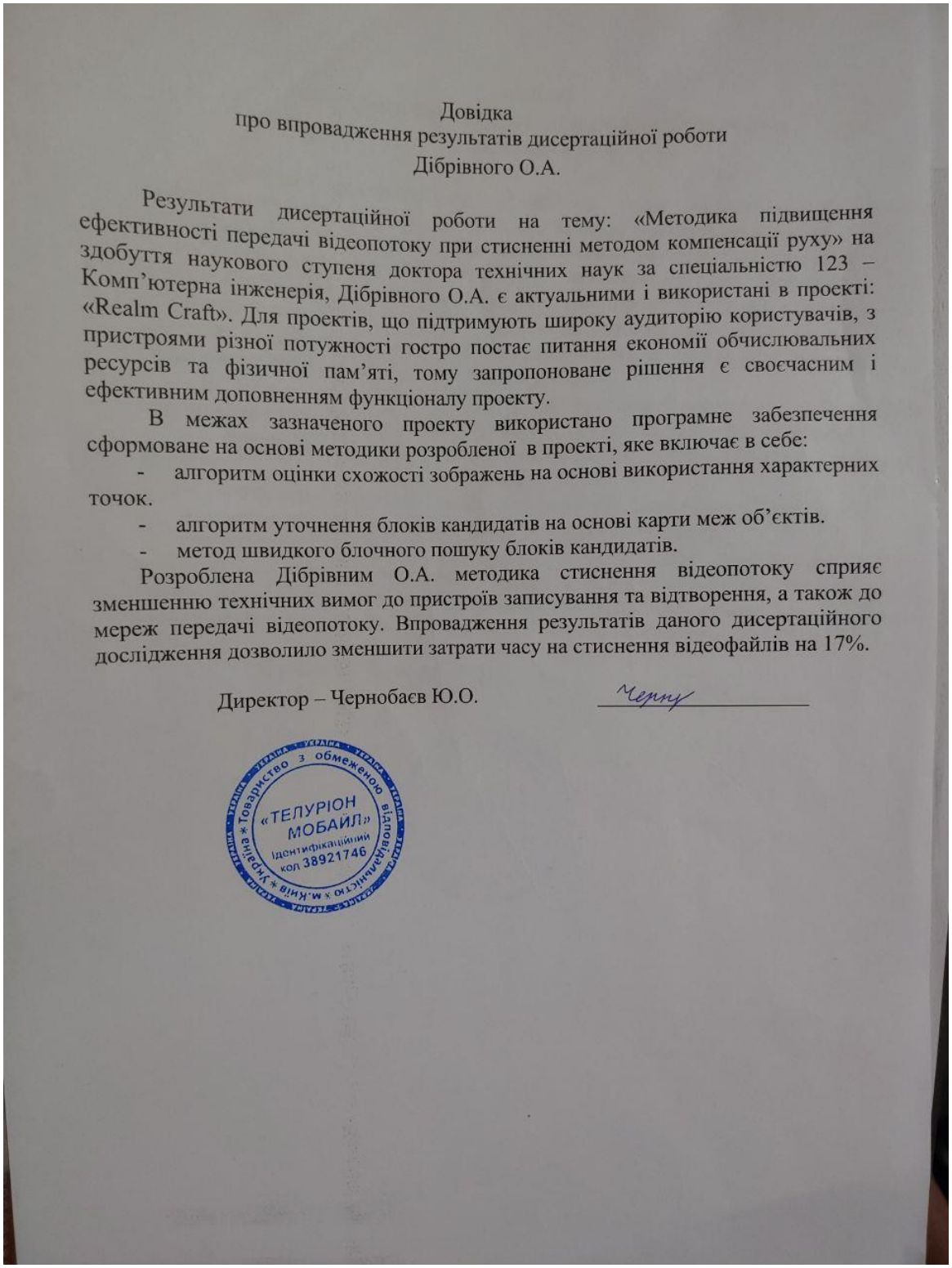
Наукові праці, які засвідчують апробацію матеріалів дисертації:

4. О.А. Дібрівний, Д.В. Сорокін, О.В. Сеньков, А.П. Бондарчук "Нові сфери застосування сучасних інформаційних мереж". Науково практична конференція «Проблеми комп'ютерної інженерії». Збірник тез К.:ДУТ 73-34ст. – 2020.
5. О.А. Дібрівний, В.О. Кисельов. "Розробка десктопного додатку для перегляду відеофайлів на мові програмування C#". XI науково-технічна

кофнференція студентів та молодих вчених, Збірник тез К.:ДУТ 39-40 – 2020.

6. О.А. Дібрівний, Д.О. Криворученко. “Розробка гри “Layers” в жанрі головоломки за допомогою ігрового двигуна Unity”. Всеукраїнська науково-технічна конференція “Сучасний стан та перспективи розвитку IoT”. Збірник тез. К.: ДУТ 29-30 – 2020.

ДОДАТОК Б



Довідка
про впровадження результатів дисертаційної роботи
Дібрівного О.А.

Результати дисертаційної роботи на тему: «Методика підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху» на здобуття наукового ступеня доктора технічних наук за спеціальністю 123 – Комп'ютерна інженерія, Дібрівного О.А. є актуальними і використані в проекті: «Realm Craft». Для проектів, що підтримують широку аудиторію користувачів, з пристроями різної потужності гостро постає питання економії обчислювальних ресурсів та фізичної пам'яті, тому запропоноване рішення є своєчасним і ефективним доповненням функціоналу проекту.

В межах зазначеного проекту використано програмне забезпечення сформоване на основі методики розробленої в проекті, яке включає в себе:

- алгоритм оцінки схожості зображень на основі використання характерних точок.
- алгоритм уточнення блоків кандидатів на основі карти меж об'єктів.
- метод швидкого блочного пошуку блоків кандидатів.

Розроблена Дібрівним О.А. методика стиснення відеопотоку сприяє зменшенню технічних вимог до пристроїв записування та відтворення, а також до мереж передачі відеопотоку. Впровадження результатів даного дисертаційного дослідження дозволило зменшити затрати часу на стиснення відеофайлів на 17%.

Директор – Чернобаєв Ю.О. _____

Товариство з обмеженою відповідальністю «ТЕЛУРІОН МОБАЙЛ» Ідентифікаційний код 38921746 м. Київ Україна

vega

Вих 11.02.2021 №536

АКТ

про розгляд наукових матеріалів Дібрівного Олеся Андрійовича, отриманих при виконанні дисертаційного дослідження

Розглянуті результати дисертаційного дослідження на тему: «Методика підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху» за спеціальністю 123 – Комп'ютерна інженерія.

Було встановлено, що результати дисертаційної роботи Дібрівного Олеся Андрійовича є актуальними та можуть бути впроваджені в практичну діяльність компанії ПрАТ «Фарлеп-Інвест»

Розроблена в дисертаційному дослідженні методика стиснення відепотоку, дозволяє зменшити час кодування відеопотоку і як наслідок вимоги до мережі, якою передається відеопотік. Даний підхід є основою проектування системи кодування і дозволяє скоротити загальний час кодування на 17% відносно класичних підходів.

Розроблені підходи становлять закінчені результати наукових досліджень, теоретично обґрунтовані і представляють практичний інтерес для ПрАТ «Фарлеп-Інвест».

Даним актом ПрАТ «Фарлеп-Інвест» підтверджує обґрунтованість запропонованих рішень і намір реалізувати їх при побудові нових та модернізації існуючих мереж.

Керівник проектів
Департаменту стратегічного розвитку мережі
ПрАТ «Фарлеп-Інвест» Д.В. Сорокін

НАУКОВО-ТЕХНІЧНИЙ ЦЕНТР
"ЕНЕРГОЗВ'ЯЗОК"

ЗАТВЕРДЖУЮ

Директор ТОВ "Науково-технічний центр Енергозв'язок"

_____ Л. Г. Погорілий

"25" лютого 2021 р.

АКТ

з використання результатів дисертаційної роботи Дібрівного Олеся Андрійовича на тему: «Методика підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху» на здобуття наукового ступеня кандидата технічних наук за спеціальністю 123 – Комп'ютерна інженерія.

Системи кодування та декодування відеопотоку використовуються на всіх етапах роботи з відео даними. При цьому, від 40 до 80% часу, що йде на стиснення відеопотоку займає процес знаходження схожих блоків, для визначення векторів руху. Оптимізувавши цей етап кодування відеопотоку, можна отримати значне пришвидшення загального часу обробки відеопотоку і як результат затримку при передачі відеоданих.

Комісія у складі:
голова комісії – Хрущов С.В., завідуючий групою;
члени комісії – Мускеєв О.Є., ведучій спеціаліст,
Древецький А.К., інженер 1ї категорії,
розглянула дисертаційну роботу.

Комісія встановила, що результати дисертаційної роботи Дібрівного О.А. є актуальними і використані в науково-дослідній роботі: «Методика підвищення ефективності передачі відеопотоку при стисненні методом компенсації руху».

Запропонований Дібрівним О.А. алгоритм стиснення відеопотоку дозволяє зменшити час кодування відеоданих за рахунок використання комбінації алгоритмів роботи з зображеннями, за рахунок чого його можна застосовувати до різного типу відеоданих.

Використані результати дозволили зменшити загальний час обробки відеопотоку, за рахунок незначних втрат в якості зображення.

Голова комісії _____ Хрущов С.В.

Члени комісії _____ Мускеєв О.Є.

_____ Древецький А.К.