МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів морфологічного аналізу цифрових зображень
(тема)

Виконав:
Студент 2 курсу, групи ІПЗздм-19-1
Бурий М.В.
(прізвище, ініціали)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник проф. Шостак І.В.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« ____ » _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Бурого Максима Вікторовича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів морфологічного аналізу цифрових зображень

затверджена наказом університету від 26.03.2021 № 34 Стз

2. Термін подання роботи до екзаменаційної комісії 12 05 2021р.
3. Вихідні дані до роботи *проаналізувати існуючі алгоритми, що використовуються для вимог підтримки прийняття рішень, мови розробки програмного забезпечення*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, опис запропонованих варіантів оптимізації, використовувані методи та алгоритми, опис розробленої програмної системи, опис застосованих програмних рішень, аналіз можливих застосувань*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, слайдів, ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Мета завдання, обґрунтування доцільності розробки, постановка задачі, базові моделі, методи й алгоритми, структурно-логічна схема взаємодії даних, інтерфейс програмної системи, результати дослідної експлуатації програмної

системи, висновки

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
спецчастина	проф. Шостак І.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	26 березня 2021 р.	виконано
2.	Огляд існуючих методів	31 березня 2021 р.	виконано
3.	Розробка алгоритмів, проектування та розробка ПЗ	15 квітня 2021 р.	виконано
4.	Підготовка пояснювальної записки	28 квітня 2021 р.	виконано
5.	Спецчастина	30 квітня 2021 р.	виконано
6.	Підготовка презентації та доповіді	05 травня 2021 р.	виконано
7.	Попередній захист	10 травня 2021 р.	виконано
8.	Нормоконтроль, рецензування	10 травня 2021 р.	виконано
9.	Занесення роботи в електронний	11 травня 2021 р.	виконано
10.	Допуск до захисту в зав. кафедри	12 травня 2021 р.	виконано

Дата видачі завдання ____ _____ 2021р.

Студент _____
(підпис)

Керівник роботи _____ проф. Шостак І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ /ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра: 105 с, 46 рис., 6 дод., 34 джерела

МЕТОД ШМІДТА, СТИСКАННЯ ЗОБРАЖЕНЬ, МОРФОЛОГІЧНИЙ АНАЛІЗ, ВІДЕО-ПОСЛІДОВНІСТЬ, ДЕТЕКЦІЯ РУХУ.

Об’єкт досліджень – морфологічні моделі та аналіз цифрових зображень.

Метою роботи є дослідження алгоритмів компресії, застосовувані безпосередньо до растрових зображень,

Методи дослідження – сучасні підходи до аналізу даних, методів обчислення руху в відео послідовностях, методи паралельного програмування.

Результат – обраний алгоритм відстежування об’єктів у відео потоці на основі переміщення фрагментів реалізований з урахуванням архітектури паралельних систем.

SCHMIDT METHOD, IMAGE COMPRESSION, MORPHOLOGICAL ANALYSIS, VIDEO-SEQUENCE, MOTION DETECTION.

The object of research is morphological models and analysis of digital images.

The aim of the work is to study the compression algorithms applied directly to raster images,

Research methods – modern approaches to data analysis, methods of calculating motion in video sequences, methods of parallel programming.

The result – the selected algorithm for tracking objects in the video stream based on the movement of fragments is implemented taking into account the architecture of parallel systems

Я, Бурий Максим Вікторович, студент гр. ІПЗздм-19-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів морфологічного аналізу цифрових зображень», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ ……...……..8
1 Аналіз стану розв'язання проблеми та обґрунтування
цілей дослідження ……..12
1.1 Аналіз сучасних методів стискання зображень …….…12
1.2 Технологія стискання JPEG …….……17
1.3 Сучасні стандарти й методи стискання цифрової інформації ….……..…20
1.4 Аналіз кодування інформації за алгоритмами Хафмана ……...23
1.5 Аналіз сучасних методик вейвлет-стискання ……26
1.6 Обґрунтування цілей дослідження ……..29
2 Опис проведених теоретичних досліджень ……..….....32
2.1 Розкладання Шмідта функції двох змінних …….…….....32
2.2 Методи аналізу цифрових зображень у завданнях ідентифікації …….36
2.3 Виділення гармонійної складової функції ……..39
2.4 Компоненти кодованого відео та рекомендації Н.264 ……..40
3 Алгоритми виявлення та відстеження рухомих об’єктів ……45
3.1 Алгоритм відстеження центра мас об’єкту ……45
3.2 Метод детекції зображень у стиснутій послідовності зображень ……49
3.3 Алгоритм переміщення фрагментів об’єктів
3.4 Розширена версія адаптивної суміші Гаусової моделі фону ……59
4 Організація паралельних обчислень при детекції об'єктів ……63
5 Програмна реалізація паралельного алгоритму аналізу зображень ……69
Висновки ……77
Перелік джерел посилання …….....79
Додаток А Перелік джерел посилання за науковими напрямами керівника
та науковців кафедри програмної інженерії ……82
Додаток Б Звіт результатів перевірки на унікальність тексту ……...…….....83
Додаток В Слайди презентації ……85

Додаток Г Листінг модуля 94
Додаток Д Апробація роботи……..……...……103
Додаток Е Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ ……..104

ВСТУП

Сучасний прогрес розвитку високих технологій дозволяє вирішувати найрізноманітніші завдання. В основному всі вони зводяться до реалізації передачі, обміну, зберігання, обробки й аналізу різних класів цифрових даних. При цьому необхідність збільшення обсягів ресурсів стимулює вдосконалення наявних і створення принципово нових апаратних і програмних засобів обробки цифрового масиву, породжує нові вимоги до питань раціонального використання технічних можливостей обчислювальних систем. Цим підтверджується актуальність проблеми зберігання цифрових даних – при величезних потоках оброблюваної інформації, від грамотного й оптимального використання сукупного обсягу пам'яті ЕОМ залежить швидкодія програмних продуктів, а виходить, і продуктивність праці.

У пошуках відповідей на поставлені питання особливої уваги вимагають графічні об'єкти, оброблювані обчислювальними системами. Інтерес до математичного аналізу й алгоритмів компактного зберігання зображень пов'язаний, по-перше, зі зручністю наочності графічної інформації, і, по-друге, з тим, що тексти й усілякі програми для зберігання й передачі по мережах вимагають порівняно невеликих обсягів пам'яті, тоді як оригінал зображення розміром 640x480 може займати майже 1 Мегабайт. Особливу цінність представляє собою оптимізація цифрової обробки інформації в системах охоронного телебачення й відеоспостереження, які стали затребуваними останнім часом. Для режиму реального часу типове відцифроване відео з високим дозволом може зажадати 1312992 байт (=1282 Кбайт) для формування одного повного телевізійного кадру без стиску. Таким чином, для системи кольорового телебачення PAL швидкість передачі 25 кадрів на секунду повинна становити приблизно 250 Мб/с (аналогічно й для NTSC) . У реальності навіть для сучасних стандартів така швидкість потоку даних є дуже високою, а іноді й неможливою. Популярне сьогодні використання мережевих відеосерверів, які працюють по

звичайним комп'ютерним мережам Ethernet , накладає певні обмеження, які самі по собі є технічно граничними.

Побудова математичного апарату, що відповідає якісному рішенню прикладних завдань цифрових технологій обробки інформації, неможлива в рамках єдиного виділеного наукового напрямку. Так, сформований підхід до проблем оптимізації кодування інформації, в XX столітті став приводом для формування не тільки окремих наукових теорій, але й цілих наук, сформувавши свої парадигми. Як відзначив Р.Е. Кричевський : «...завдання стиску придбало таку важливість, що для розгляду різних його варіантів виникла нова наука – теорія інформації...».

Досить великий спектр питань цифрової обробки зображень успішно вирішується в роботах сучасних учених, послужили науковою базою для багатьох сучасних математичних методів, зокрема, розробка проблеми відновлення змазаних і розфокусованих зображень, питання морфологічного аналізу цифрових зображень. Реалізувавши математичні методи й власні алгоритми на базі обчислювальних лабораторій у серіях комплексних програмних продуктів, вони заклали не тільки технологічну базу, але й надали можливість використання своїх результатів у навчальному процесі. Праці вітчизняних учених відповідають усім вимогам сучасності, а іноді й випереджають технічні можливості апаратного забезпечення. Іноземні фахівці пропонують свої досить продуктивні алгоритми, в основному орієнтовані на сучасні параметри й продуктивність електронної й обчислювальної техніки [1].

Усі відомі сучасні методи стиску орієнтовані на видалення деякої надмірності, що зберігається в інформації. При цьому під надмірністю звичайно розуміється часта повторюваність окремих однакових ділянок інформації або присвоєння однакової кількості біт для кодування всіх елементів, незалежно від того, як часто ці елементи можуть бути використані. При компресії графічних зображень в основному застосовується усунення специфічної надмірності, яка утворюється за рахунок кореляції по кольоровості і яскравості сусідніх пікселів.

Принципово алгоритми стиску цифрової інформації прийнято розділяти на два типи: стиск без втрат і стиск із втратами інформації [2].

У першому випадку перетворення є оборотним і по стислому образу інформація може бути відновлена з точністю до біта. Традиційно використовуються такі методи, як:

– методи кодування довжин серій, наприклад, RLE, коли послідовні елементи з однаковим значенням кодуються за допомогою пари чисел, що включають довжину серії й значення самих елементів;

– факсимільний стиск;

– словникові методи, наприклад LZW згідно із цим алгоритмом при стиску даних ведеться словник, який містить послідовності значень елементів, що вже зустрічалися. Стислий потік буде містити коди, що вказують на елементи словника.

Другий тип стиску (із втратами) звичайно використовується при компресії графічної інформації. Інформація, яка не суттєва при сприйнятті людським зором, відкидається. Зовні картинка буде виглядати як оригінал або трохи відрізнятися погіршенням якості зображення, але в цілому такий результат може вважатися цілком прийнятним. Такий стиск задовольнить потребу у відносно невеликому обсязі пам'яті для зберігання інформації. Але це перетворення завжди однобічне, і по стислому образу відновити точну копію оригінального зображення не вийде, – стисла картинка буде скоріше просто схожа на оригінал.

Іншою, не менш важливою особливістю наочної інформації, є її більш простий аналіз (у сенсі кількісних або якісних характеристик), порівняння й ідентифікація. Питання про належність даного об'єкта до окремого описаного класу представляє собою сутність окремого підходу при роботі із графічними зображеннями, який у загальному можна назвати теорією розпізнавання образів.

Досягнення в області розпізнавання образів уже досить тривалий час використовуються в медицині, діяльності правоохоронних органів, при дешифруванні космічних знімків, у більшості промислових сфер.

У кваліфікаційній роботі досліджуються алгоритми компресії, застосовувані безпосередньо до растрових зображень, які в цифровому виді зазвичай представляють собою двовимірні масиви числових даних, де кожний елемент є характеристикою окремо взятого пікселя. При цьому обсяг пам'яті, необхідний для зберігання такої інформації, фактично визначається колірною насиченістю картинки.

1 АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

1.1 Аналіз сучасних методів стискання зображень

Для концептуального аналізу основних принципів стиску зображень, необхідно розглянути алгоритм їх цифрової обробки. При цьому визначається споконвічна природа графічного об'єкта, а потім способи зберігання цифрової графіки, виходячи з формування й передачі кольору при візуалізації цифрових картинок. Часто компресії передують технології попередньої цифрової обробки вихідного графічного оригіналу.

У довільному класі аналогових зображень розглянемо процес дискретизації. У дії процедура дискретизації може бути описана як якась разова обробка зображення, після якої графічному об'єкту зіставляється двовимірний масив, елементами якого будуть дійсні числа [3].

Наступною провідною процедурою цифрової обробки для оптимального використання можливостей обладнання для оцифровування і подальшого кодування зображення, є квантування – це процедура заміни дискретного відліку найближчим значенням з набору фіксованих величин (рівнів квантування).

У загальному випадку дана операція може бути представлена як східчаста функція: якщо яскравість $\boldsymbol{x}$ дискретного відліку зображення укладена в числовому проміжку, то вихідний відлік заміняється рівнем квантування kj. Квантований сигнал ухвалює кінцеве число значень (звичайно з діапазону від 0 до 255), які, як правило, збігаються за порядковим номером з рівнем квантування.

Встановлено, що людський зір більше реагує на зміну яскравого, ніж колірного тону в зображенні, що пояснюється функціональними особливостями зору: око містить особливі нервові клітини, які називаються паличками (їх кількість переважає) і колбочками – фізіологічний апарат яскравого й колірного сприйняття відповідно. Виходячи із природи людського зору й можливостей технічних реалізацій, з'явився ряд колірних моделей, які принципово

відрізняються один від одного способами формування кольорів при передачі зображень на обладнання виводу інформації. Далі розглянемо найбільш популярні з них.

Система кольорів RGB – у даній колірній системі всі кольори визначаються як результат змішування червоного, зеленого й синього компонентів. Дана модель представляє собою адитивний синтез трьох основних кольорів і є апаратно-залежною, кольори, які реально можуть бути отримані, утворюють «колірне охоплення» обладнання відображення. Недоліком у цьому випадку можна вважати й той факт, що не існує джерел основних кольорів, які дозволять одержувати всі можливі кольори шляхом позначеного вище способу змішання червоного, зеленого й синього. В RGB для зберігання картинки в електронному виді потрібно 3 байта на кожний піксель зображення тільки для того, щоб зберегти інформацію про колір.

YCbCr – трикомпонентний колірний простір: Y – компонента яскравості, Cb і Cr – компоненти, що визначають кольоровість. Значенням Cb задається синявість зображення, Cr – червоність. Такий підхід до передачі кольору подібний колірній моделі, використовуваної в телевізійних приймачах. При цьому Y сам по собі є півтоновим представленням кольорового зображення.

На відміну від RGB, де всі компоненти рівнозначні, колірна модель YCbCr концентрує найбільш важливу інформацію в одному з компонентів, що дозволяє досягти більшого стиску при компресії зображення шляхом більшого обсягу даних по компоненту яскравості Y, ніж по червоній й синій.

Між колірними моделями RGB і YCbCr у комп'ютерному моніторі взаємозв'язок виражається відомими математичними рівняннями,

$$Y=0.299R+0.587G+0.114B$$

CMYK – є чотирьохкомпонентною колірною моделлю (Cyan, Magenta, Yellow, Black – блакитний, пурпурний, жовтий, чорний), що використовується при кольоровому друку. У розглянутих вище колірних моделях компонента додавали колір у зображення. Чим вище значення компонента, тим ближче колір

до білого. Однак в CMYK більші значення компонентів наближають колір до чорного. При рівності значень C, M, і Y колір являє собою відтінок сірого.

CMYK також може бути апроксимована через RGB, але являє собою той випадок, коли між колірними просторами не існує взаємооднозначної відповідності – на одне й теж значення RGB відображається множина значень CMYK.

При створенні й зберіганні цифрових зображень сучасний користувач може самостійно вибирати графічні формати, під якими прийнято розуміти певні правила запису цифрових даних у файл. Графічні формати розроблялися для того, щоб ефективно організовувати, зберігати й відновлювати графічні дані. В основному, при роботі із цифровими картинками використовують растровий, векторний або метафайловий формати.

Растрові формати раціонально використовувати при зберіганні реальних зображень (відео зображення, фотографії й т.п.) – вони представляють собою «піксельну карту зображення», за якою програма візуалізації відновить зображення на обладнанні виводу інформації.

Векторні формати, на відміну від піксельних, містять математичні описи елементів зображення, за якими програма візуалізації реалізує зображення. Хоча по своїй структурі такий формат простіше, чим більшість відомих растрових, для реалізації зображень у векторних форматах можуть вимагатися більш великі обчислювальні ресурси, і, відповідно, час.

Метафайлові формати можуть зберігати як растрові, так і векторні дані.

Вибір формату звичайно погоджений з видом графіки. Відповідно до характерних відмітних рис графічну інформацію, оброблювану за допомогою комп’ютерної техніки, можна умовно класифікувати на:

– зображення з невеликою кількістю кольорів (наприклад, ділова графіка);

– зображення із плавними переходами кольорів, реалізовані за допомогою можливостей ЕОМ;

– фотореалістичні зображення;

– фотореалістичні зображення з накладенням ділової графіки (наприклад,

реклама) і т.ін.

Становлення різних графічних форматів згодом стало супроводжуватися впровадженими алгоритмами компресії. Сьогодні більшість сучасних способів зберігання зображень містять програмні модулі, що дозволяють більш компактно використовувати ресурси пам'яті ЕОМ при зберіганні цифрової графіки [5].

Однак, не дивлячись на досягнення істотних результатів в області цифрової обробки зображень, дотепер не існує єдино універсальної відповіді на запитання, що стосуються раціонального використання ресурсів ЕОМ у сукупності з оптимальною швидкодією й максимальним заощадженням якості інформації. Принципово ж, основні напрямки подібних досліджень фактично представляють собою або одне з позначених нижче, або їх синтез.

Наразі застосовується методика, коли вирішується завдання про вибір найбільш оптимального способу кодування для конкретного зображення. Даний підхід звичайно супроводжується створенням і використанням відповідних програмних засобів. При вирішенні подібних завдань, наприклад, можна виділяти деякі «особливості» зображень, що можна стиснути, і з урахуванням такої інформації будувати модель оцінки максимально можливого коефіцієнта стиску для різних методів компресії.

Використовуючи алгоритми аналізу типу зображень, можна досягти оптимізації стиску не більше ніж на 10 %. Для інтерактивного режиму роботи в обчислювальній мережі така оптимізація може бути цілком задовільною, однак наведений підхід в основному реалізується за рахунок наявних алгоритмів стиску, і, скоріше спрямований на відновлення власних версій, ніж реалізацію нових алгоритмів компресії.

Створення нових і вдосконалювання наявних алгоритмів стиску постійно стимулює інтерес до проблеми встановлення особливостей алгоритмів компресії. На сьогоднішній день уже виділяються лідери в даній області, застосовність яких була впроваджена й удосконалюється вже не тільки алгоритмічно, але й на технологічному рівні виробництва.

Таким чином, з початком застосування досягнень цифрових технологій у сучасному світі, питання розпізнавання й ідентифікації графічних зображень стали найважливішими для більшості систем моніторингу. Це призвело до того положення, коли завдання з візуалізації інформації, її обробки, транспортування через обчислювальні мережі й аналіз стали одними із центральних серед усієї різноманітності поставлених завдань компактного й раціонального зберігання цифрової інформації. Цифрові технології сьогодні займають одне із привілейованих місць серед основних завдань, з якими доводиться мати справу сучасним фахівцям практично у всіх сферах діяльності. Глобальне впровадження обчислювальної й комп'ютерної техніки в системи діагностики, спостереження й контролю прямо пов'язане з розвитком високих технологій в області цифрової обробки інформації, чим пояснюється невгасаючий інтерес до питань опису усе більш оптимальних методів компресії медіаданих, значну частину яких представляють окремі класи цифрових зображень.

У даній роботі розглядаються деякі математичні алгоритми ідентифікації й алгоритми стиску зображень, пропонуються нові шляхи вирішення проблеми компресії й аналізу цифрових півтонових зображень класу фотоаанфас.

Алгоритми стиску й відновлення, що розглянуті в роботі, опираються на використання відомого методу Шмідта розкладання функції двох змінних, а також методу сингулярного розкладання матриць. У чисельних експериментах для різних півтонових чорно-білих зображеннь отримано коефіцієнт стиску, порівнянний з коефіцієнтом, який дає стандартна технологія JPEG.

Рішення завдання ідентифікації вимагає побудови такої операції, яка різним функціям прообразам ставить у відповідність різні образи, і обсяг інформації для образів суттєво менше обсягу інформації для прообразів, при цьому така відповідність повинна бути взаємоднозначною (принаймні для деяких основних складових, основних проекцій функцій прообразів і їх образів).

Перший розділ має загальний характер. У розділі 1.1 коротко викладено традиційний алгоритм JPEG. Саме JPEG викладається майже у всіх навчальних і монографічних виданнях, присвячених проблемам компресії й декомпресії

цифрових зображень. Основним елементом його є дискретне косинусне перетворення Фур'є (ДКП). Цей математичний алгоритм дає основний ефект стиску, що досягається даної технологією. Велика кількість робіт присвячена модифікації й прискоренню його основної складової – алгоритму ДКП.

Широко ведеться розробка принципово нових математичних і комп'ютерних методів і алгоритмів для рішення завдання відновлення - відновлення-відбудови-стиснення-відновлення .

Одним з нових перспективних напрямків у дослідженні проблем стиску й відновлення цифрових зображень є морфологічний аналіз.

Вирішення спектрального завдання є складною комп'ютерною процедурою, яка використовує ітераційні алгоритми, що вимагає порівняно великого часу. У роботі для цього використовується новий метод рішення алгебраїчного спектрального завдання, розроблений в останні роки метод максимізації стовпців.

Найважливішим етапом аналізу певних класів зображень є дослідження їх інваріантів щодо деяких перетворень для розглянутого класу зображень.

Наразі особливу увагу дослідників привертає чисельний алгоритм стиску цифрового зображення, заснований на розкладанні Шмідта. Для визначення кожного доданка потрібно визначити головне власне число і його власний вектор для відповідного інтегрального оператора [6].

У дискретному випадку функції зображення це зводиться до спектрального алгебраїчного завдання для матриць.

1.2 Технологія стискання JPEG

Однією із самих затребуваних сучасних технологій стиску цифрової графіки є JPEG, розробкою якого займалися фахівці з машинної обробки фотозображень (від назви об'єднаного комітету експертів з машинної обробки фотозображень – Joint Photographic Experts Group). Фактично, для сучасних програм перегляду й

перетворення цифрової графіки ця технологія є невід'ємною частиною. Незважаючи на відомих розроблювачів, технологія JPEG довгий час існувала без стандартизації, що призвело до неоднозначності її реалізації в програмних модулях і користувацьких інтерфейсах.

Однак, у загальних етапах виконання, алгоритм роботи JPEG представляє собою поетапну реалізацію наступних дій:

– перетворення колірної схеми RGB у яскравно-колірну YCbCr, після чого здійснюється розбивка зображення на сукупність блоків 8x8 пікселів;

– до кожної отриманої цифрової матриці (8x8) перетвореного зображення застосовується дискретне косинусне перетворення Фур'є;

– матриці коефіцієнтів Фур'є зазнають процесу квантування, який для jpeg зводиться до почленного перемноження матриць коефіцієнтів на матриці дільників; матриці дільників рекомендуються розроблювачами;

– отримані результати кодуються методом Хаффмана.

Споконвічно JPEG був орієнтований на компресію нерухливих графічних об'єктів і розроблявся як метод стиску без втрат (JPEG-LS), однак більшу популярність набув його варіант, що реалізує стиск стаціонарної графіки із втратами. Технологія JPEG забезпечує компресію в десятки, а іноді й навіть у сотні раз, чим явно виправдовує себе при роботі з базами даних, орієнтованими на зберігання фотографічних зображень, різного роду повнобарвних картинок і т.п.

Сьогодні як похідний від JPEG, у системах охоронного телебачення реалізований алгоритм M-JPEG, який можна представити як швидко відтворений потік окремо стислих нерухливих jpeg – зображень. У цьому випадку саме висока швидкість відтворення стаціонарних кадрів створює ілюзію руху.

Останнім часом з'являється інформація про новий стандарт JPEG 2000, основна відмінність якого полягає в заміні косинуса перетворення Фур'є перспективним вейвлет-перетворенням. В інших кроках стиску даний алгоритм близький зі звичайним JPEG. Переваги й недоліки розглянутих jpeg технологій освітлені в літературі досить широко. Однак, не дивлячись на вдосконалення, популярність «класичного» JPEG продовжує залишатися досить високою.

Далі розглянемо кожний етап цифрової обробки зображень, що випереджають роботу алгоритму JPEG.

Фотореалістичні зображення перетворююься до цифрового виду й ухвалюють первісний вид піксельного масиву тільки після дискретизації.

Заміну безперервного зображення дискретним можна виконати різними способами. У практичному досвіді найбільш зручним, хоча й не єдино можливим, є періодична дискретизація із прямокутним растром [7]. Вона може розглядатися як один з варіантів застосування ортогонального базису.

Основна проблема, що виникає при заміні безперервного зображення дискретним, полягає у визначенні таких умов, при яких заміна є повноцінною. Вважається, що втрати відсутні, якщо, маючи дискретний сигнал, можна відновити безперервний. Наприклад, відповідно до точного інтерполяційного відновлення безперервного зображення по відомій множині дискретних двовимірних відліків.

Відповідно, розрізняють два види квантування – скалярне й векторне [8]. При скалярному квантуванні округлення чисел проводиться інтуїтивним способом. У результаті можливі втрати не тільки незначної інформації. Варто так само відзначити, що в більшості випадків неможливо передбачити коефіцієнт стиску після скалярного квантування.

Векторне квантування зображення представляє собою процес порівняння рівних між собою блоків пікселів зображення (які й прийнято називати векторами) з кодовою книгою програми кодувальника (кодера). Зображення розбивається на блоки, кожний з яких порівнюється з усіма блоками з кодової книги, і визначається «найближчий» до нього. Потім у вихідний файл записується покажчик на «найближчий» блок кодера. Якщо розмір показника менше розміру блоку, то досягається стиск. При векторному квантуванні коефіцієнт стиску відомий заздалегідь.

Підводячи підсумок, можна визначити квантування як округлення дійсних чисел матриці зображення до фіксованих цілих чисел. Так само квантування може проводитися й із цілими числами, для перетворення останніх у менші.

Поняття цифрової фільтрації зображень вважається одним із центральних при обробці й стиску комп'ютерної графіки, однак не є однозначно сталим за змістом. Так, під фільтрацією в цифровій обробці зображень можна розуміти функцію, що діє з множиначю дійсних чисел на задану підмножину. У цьому випадку дана функція представляє собою відображення на числову підмножину, і її образ несе якісну характеристику цифрового зображення, на якому задано відображення [9].

Технологія фільтрації застосовується при компресії зображень, коли від функції зображення відділяється значна частина інформації, що несе якісну характеристику (наприклад, частотна фільтрація в косинусі перетворення Фур'є).

Інший зміст має поняття фільтрації при обробці цифрових зображень, якщо мається на увазі цифрова обробка при візуалізації графіки на обладнання виводу інформації. У випадку неякісного образу зображення при відновленні графіки використовується адитивна фільтрація з додаванням шуму для надання більш плавних яркісно-колірних характеристик, за допомогою чого підвищується візуальна якість.

1.3 Сучасні стандарти й методи стискання цифрової інформації

За принципами реалізації сучасні методи стиску визначаються як неадаптивні, адаптивні, і напівадаптивні.

– метод неадаптивного стискання – принцип дії алгоритму стиску виключає можливість змінювати параметри своєї роботи залежно від вхідних даних. Алгоритми даного методу якісно стискають однотипні дані;

– метод адаптивного стиску. Робота зі стиску здійснюється у два етапи. Спочатку тестуються вихідні дані, потім, залежно від результатів перевірки, відбувається підстроювання параметрів виконання алгоритму;

– метод напів-адаптивного стиску визначає виконання алгоритмів

компресії як двопрохідних. На першому проході по файлу акумулюється статистика по стиснутих вхідних даних; на другому проходженні виконується стиск із обліком отриманої раніше інформації [10].

Цифрова графіка в безпосередньому первісному виді може займати великі обсяги пам'яті. Звичайне застосування алгоритмів компресії до графічних зображень (що особливо мають аналогову природу) попередньо супроводжується декількома, попередніми етапами цифрової обробки вхідного сигналу. При цьому якісною характеристикою результату компресії є коефіцієнт стиску, під яким прийнято розуміти ступінь продуктивності конкретного алгоритму стиску. Визначимо коефіцієнт стиску K як відношення розміру файлу, отриманого на виході після застосування стиску, до розміру вихідного «нестиснутого» файлу в шкалі обсягу пам'яті, необхідні для зберігання відповідно вихідного й вихідного файлів.

Відсутність компресії буде очевидною при $K = 1$. Значення K, що є більшими за одиницю, свідчать про помилкове застосування даного алгоритму до конкретного зображення – відбудеться розширення. У деяких випадках, крім коефіцієнта стиску, може бути визначена зворотна величина – фактор стиску, яка представляє собою відношення розміру вихідного файлу до розміру файлу, отриманого на виході після застосування процесу компресії [11].

У ряді випадків зображення, оброблювані за допомогою сучасних обчислювальних систем, споконвічно представлені аналоговими джерелами. Їхнє перетворення до цифрового виду є обов'язковим для подальшої можливості їх обробки, зберігання й передачі по обчислювальних мережах. Таким чином, первинний аналоговий малюнок при оцифровуванні з необхідністю зазнає дискретизацію й квантування, після чого до нього можуть бути застосовані комп'ютерні програми аналізу, компресії-декомпресії.

В основі ефективного кодування сигналу при оцифровуванні зображення закладено наведений вище принцип усунення надмірності. У теорії коливань інформаційний зміст сигналу аналізується легше, якщо вихідний сигнал можна розкласти на частотні складові, що в певному сенсі можна розглядати як

процедуру аналізу вхідної інформації. У цьому випадку усунення надмірності буде розглянуто як відділення істотних інформаційних характеристик від менш значимих складових у змісті відновлення зображення. Завдяки видаленню несуттєвої інформації із сигналу, спрощується запам'ятовування й зменшується необхідний обсяг пам'яті. Наведений підхід становить основу одного з найважливіших у цей час методів стиску растрових зображень – дискретного косинусного перетворення (ДКП), що лежить в основі таких стандартів стиску, як JPEG і M-JPEG [12].

ДКП представляє собою оборотне відображення масиву речовинних даних (в JPEG операція проводиться над уже квантованими числовими даними) у масив тієї ж розмірності, елементами якого будуть коефіцієнти косинусних функцій зі зростаючими частотами. Для стиску графіки реалізується двовимірне ДКП. У результаті виходить матриця, основна вага якої розподілена по спаданню з верхнього лівого кута до нижнього правого. Таким чином, основна інформація зосереджується в певній частині масиву. До отриманого результату застосовується квантування, яке й забезпечить основний ефект стиску. Простіше всього воно виконується як матричний добуток на матрицю дільників, які визначаються виходячи з орієнтира на якість одержуваного після компресії зображення. Для відновлення стислих даних позначені кроки виконуються у зворотному порядку.

Особлива проблема при стиску за допомогою ДКП пов'язана з розбивкою зображення на блоки, тому що при дуже сильному стиску блокова структура зображення може виявитися видимою й погіршувати візуальну якість. Не менш важливе й те, що матричне перемножування в ДКП вимагає досить більших витрат ресурсів ЕОМ. Алгоритми, засновані на цьому методі, мають не дуже високу швидкодію [13].

Наведене дискретне косинусне перетворення може бути використане для стиску RGB даних зображень у дійсних кольорах. Однак на практиці вчиняють трохи інакше: перед ДКП скорочують кількість даних, відокремлюючи колірну інформацію від яркостної інформації й стискаючи її шляхом формування

особливих усереднених значень. Яркісткову інформацію стискають безпосередньо. Так само вчиняють і з півтоновими зображеннями, які містять тільки інформацію про яркість.

1.4 Аналіз кодування інформації за алгоритмами Хафмана

Під кодуванням інформації прийнято розуміти функцію, що ставить у відповідність вхідному потоку повідомлень деякий певний код.

Оптимізація кодування є досить неоднозначним поняттям, тому що може розглядатися як спосіб оцінки таких параметрів роботи архіваторів, як ступінь стиску й швидкість роботи, часто також важливим виявляється обсяг використовуваної пам'яті. У теорії інформації інформація визначається тільки імовірнісними, точніше, статистичними властивостями потоку повідомлень [14].

Кількістю власної інформації (власною інформацією) називається величина $I(Xi) = -log(p(x_i)),\ i = 1,2,\ ...,\ L,$ де $p(Xi)$ – ймовірність і-го повідомлення в ансамблі повідомлень $\{X,\ p(x)\},\ X=\{x_1, x_2,...,x_l\}$.

Таким чином, оптимізація може бути визначена через величину власної інформації вхідного потоку (ансамблю повідомлень), і її математичного очікування – ентропію, яка представляє собою середню кількість власної інформації в повідомленнях ансамблей.

Оптимальним визнається таке кодування, при якому досягається найкращий стиск, тобто найменша в середньому кількість біт, необхідних для представлення символів вхідного потоку повідомлень. При виборі методу кодування (стиску) розглядається питання про типи джерел вхідних даних. Залежно від кореляційно статистичних особливостей джерело вхідної інформації називається джерелом Бернуллі, якщо чергове повідомлення або код *не залежить* від усіх попередніх, і джерелом Маркова, коли передбачається, що ймовірність появи чергового сигналу

залежить від *n* попередніх сигналів. Часто, при компресії цифрових зображень використовуються саме властивості джерел Маркова.

Алгоритм кодування, запропонований Давидом Хаффманом на початку п'ятдесятих років двадцятого століття, сьогодні є одним із найпопулярніших для побудови кодів змінної довжини. Цим виправдовується різноманіття застосування його принципу в комп'ютерних програмах стиску різного роду інформації. У тому числі, даний метод кодування широко застосовується в різних версіях JPEG і M-JPEG.

Сутність розглянутого алгоритму полягає в побудові кодового дерева в напрямку знизу нагору. Символи, що зустрічаються в потоці інформації, що кодується, рідше всього, перебувають далі від кореня. Таким символам приписуються більш довгі коди. Символам, які часто зустрічаються, привласнюються коди більш короткої довжини [15].

Розглянемо принцип кодування й декодування, що становить основи алгоритму Хаффмана.

– кодування починається зі складання списку символів алфавіту й відповідних їм частот, якими визначають вузли майбутнього дерева;

– в отриманому списку перебувають два вузли з найменшими частотами. Вони поєднуються, і створюється новий вузол, який представляється як допоміжний символ. Йому приписується частота, рівна сумі частот об'єднаних вузлів. Результат міститься вгорі часткового дерева. Далі буде розглянутий список з недоторканих символів і отриманого допоміжного. До списку застосовуються ті ж дії, процедура побудови гілок дерева триває доти, поки список не скоротиться до одного допоміжного символу, що представляє весь алфавіт, що кодується.

Після того, як отримано останній символ, для кожного вузла дерева призначається значення «1» для однієї з гілок, що виходить із даного вузла, і значення «0» – для іншої. Спускаючись зверху вниз по дереву, будуються всі коди для символів.

Побудовані коди повинні відповідати властивості префікса. Варто відзначити, що наведений варіант побудови списку символів алфавіту й

відповідних їм частот, не є єдино можливим. На практиці існують більш прості способи одержання кодів Хафмана, наприклад, можна генерувати для всіх символів алфавіту довжини кодів (для чого складається окремий, проміжний список), і на підставі цього формувати коди. У такому методі використовуються символи, пов'язані з довжинами кодів, і списки значень із пов'язаними з ними частотами [16].

Важливою стороною при побудові кодів методом Хафмана є обмеження їхньої довжин. Обмеження довжин кодів, наприклад, 16 бітами, змушує використовувати коди менші, ніж при оптимальному кодуванні. Такого результату домагаються шляхом зрушення вузлів у дереві. Проводиться пошук символу з довжиною коду, хоча б на 2 меншої довжини, ніж в коду, який потрібно зменшити. Такий символ заміняється гілками, а більш довгі коди зрушуються по горизонталі.

Декодування здійснюється в напрямку від кореня. Декодер читає початок файлу й будує дерево Хаффмана для алфавіту, після чого починається інтерпретація кодів – читається перший біт стислого файлу, і якщо це нуль, то спуск піде по нижній гілці дерева, а якщо значення дорівнює одиниці, то по верхній. Далі читається другий біт і відбувається аналогічний спуск. Процедура повторюється доти, поки декодер не досягне листа дерева – код першого нестиснутого символу отриманий. Процес буде повторений для наступних бітів, знову починаючи від кореня.

У реалізаціях JPEG не вказується точний спосіб генерування кодів Хаффмана. Більше того, позначена схема виконання стандарту JPEG не є, по суті, єдиною. Так, існують реалізації, у яких використовується арифметичне кодування, коли весь потік вхідних даних кодується одним числом із проміжку [17].

Однак унікальність алгоритму Хаффмана полягає в різних варіаціях його застосування. По суті виконання він може бути реалізований і як адаптивний, і як напів-адаптивний метод.

1.5 Аналіз сучасних методик вейвлет-стискання

Разом з величезними темпами розвитку всієї індустрії інформаційних технологій еволюціонують і її складові. Так, гарні перспективи прогнозують новому стандарту стиску JPEG-2000. Його основна відмінність від класичного JPEG полягає в тому, що замість ДКП при стиску використовується Wavelete Метод.

Сам термін "Wavelet" (метод елементарних хвиль або М-метод) з'явився порівняно недавно; його ввели в середині 80-х років у зв'язку з аналізом властивостей сейсмічних і акустичних сигналів. У цей час сімейство аналізаторів, названих вейвлетами, починає широко застосуються при аналізі зображень різної природи, у завданнях розпізнавання образів, при обробці й синтезі різних сигналів, наприклад, мовних, для вивчення властивостей турбулентних полів, для згортки (упакування) більших обсягів інформації й у багатьох інших випадках [18].

Мета перетворення одновимірного сигналу полягає в його розкладанні по базису, що має певні властивості самоподібної функції (хвилі) за допомогою масштабних змін і переносів. Кожна з функцій цього базису характеризує як певну просторову (або тимчасову) частоту, так і її локалізацію у фізичному просторі (або часі).

Таким чином, на відміну від традиційно застосовуваного для аналізу сигналів перетворення Фур'є, М-перетворення забезпечує двовимірне розгортання досліджуваного одномірного сигналу, при цьому частота й координата розглядаються як незалежні змінні. Це дає можливість аналізувати властивості сигналу одночасно у фізичному (час, координата) і в частотному просторах. Усі ці судження узагальнюються на не одномірні сигнали або функції.

Більшість обмежень, що накладаються на вейвлет, пов'язані з необхідністю мати зворотне перетворення. Вибір, що аналізує вейвлет, як правило, визначається тим, яку інформацію необхідно витягти із сигналу. Кожний вейвлет має характерні риси в тимчасовому й у частотному просторі, тому іноді за

допомогою різних вейвлетів можна повніше виявити й підкреслити ті або інші властивості аналізованого сигналу [19].

Часто М-аналіз називають "математичним мікроскопом", маючи на увазі чудову властивість методу, зберігати гарне розширення на різних масштабах. Дана властивість успішно використовується в сучасних пакетах прикладних програм обробки зображень.

Одномірне перетворення Фур'є дає також одномірну інформацію про відносний внесок (амплітуди) різних тимчасових масштабів (частот). Результатом М-перетворення одномірного ряду є двовимірний масив амплітуд М-перетворення. Розподіл цих значень у просторі подає інформацію про еволюцію відносного внеску компонент різного масштабу в часі й називається спектром коефіцієнтів М-перетворення, масштабно-часовим спектром або М-спектром (time-scale spectram або wavelet spectrum на відміну від single spectram перетворення Фур'є) [19].

Якщо порівняти високий ступінь компресії при ДКП і Wavelett методі, то у першому випадку артефактом буде явна блокова структура зображення, у другому – вид брижі поблизу різких границь. Wavelete-метод знайшов широке застосування в цифрових системах відеоспостереження, розробляються альтернативи існуючим алгоритмам стиску.

Наприкінці вісімдесятих років Майклом Барнслі був описаний метод, що представляє собою альтернативу методам стиску зображень у вигляді бітових матриць [20]. Саме поняття фракталів і фрактальних нескінченностей було введено Мандельбротом –було запропоновано одержувати складні зображення шляхом реалізації простих алгоритмів. Одним з найважливіших моментів даного підходу є те, що зображення у вигляді бітової матриці може генеруватися за допомогою декількох формул. Тому тут можна говорити про деякого роду аналогії між векторним зображенням і зображенням у вигляді бітової матриці, з однієї сторони, і фрактальним зображенням і зображенням у вигляді бітової матриці – з іншої. Наприклад, нехай є чорний прямокутник на білому тлі розміром 800x600 при глибині кольору 24 біт. Для запису такого зображення буде

потрібно пам'яті 800x600x3 або, приблизно, 1,3 Мегабайта. Якщо ж записати тільки координати прямокутника разом з товщиною лінії й назвою кольору, буде потрібно всього трохи байт. По суті, у цьому й полягає принцип фрактального стиску. За допомогою певного алгоритму в зображенні у вигляді бітової матриці відшукуються структури, які можуть бути компактно записані у вигляді математичних виражень, тобто мова йде про знаходження подоби. Математичний метод, використовуваний при фрактальному стисканні зовсім інший, але результати виходять порівнянними [21].

Разом з тим, даний метод не знайшов широкого застосування, навіть незважаючи на те, що заснована Майклом Барнслі фірма Inrected Systems за кілька років довела фрактальний метод до реалізації й запропонувала до продажу бібліотеку функцій, які можуть застосовуватися для стиску й відновлення зображень. Причина цього очевидна. Основним недоліком фрактального алгоритму є потреба в значних обчислювальних ресурсах при здійсненні компресії зображення, у той час як розархівація відбувається дуже швидко. Більші ресурси потрібні для того, щоб підібрати для кожного блоку, на які розбивається зображення, максимально подібний йому (з точністю до афінного перетворення). Якщо зображення у вигляді бітової матриці стискується методом JPEG за час у межах секунди, то для фрактального стиску в найкращому разі потрібно кілька хвилин. Хоча отримане зображення часто більш високої якості, величезні тимчасові витрати виправдовують стиск лише у виняткових випадках. Таким виключенням є мультимедіа продукт високої якості, у якому утримується багато зображень.

Особливе місце у вирішенні проблем аналізу зображень займає порівняно новий напрямок – математична морфологія. Сучасні методи даної наукової галузі дозволяють успішно вирішувати такі завдання семантики зображень, як виділення нових або зникнення вже зафіксованих об'єктів при відстеженні охоронними системами, аналіз медичних зображень, і т.п. Однак, не дивлячись на затребуваність проблем, що вирішуються, варто відзначити незначну наявність

публікацій даного предмета в літературі, чим пояснюються можливі розбіжності у формулюванні стандартних визначень і операцій математичної морфології.

Головною метою при аналізі зображень справедливо можна вважати створення адекватного математичного опису, що передає «основний зміст зображення», який у рамках поставленого завдання буде максимально інваріантний таким характеристикам, як умови реєстрації зображень деякого об'єкта або сцени, параметри апаратури, що реєструється і т.ін. [22] .

Розглянемо як приклад ситуацію, характерну різною природою обладнань формування зображення або всілякими перетвореннями яскравості вихідного зображення. На шляху дозволу завдання морфологічного аналізу зазвичай необхідно визначити максимальний інваріант такого класу перетворень, який називається формою. Тому що цей інваріант форму всієї сцени повністю не визначає, його назвемо формою зображення сцени. Так, відмінності останніх будуть відбивати такі зміни, як поява нових об'єктів або зникнення старих.

Іншим важливим класом завдань, що передують завданням порівняння за формою, є завдання сполучення й виділення заданих фрагментів (або структурних елементів).

1.6 Постановка задач дослідження

Можна сказати, що в морфологічному аналізі зображень формою є оператор проектування, певний на заданому класі зображень, що діє в множина усіляких перетворень яскравості й умов реєстрації сцени.

Діапазон яскравостей зображення може бути розбитий на k інтервалів, у кожному з яких буде не більш однієї області постійної яскравості зображення.

При аналізі завдань зберігання, аналізу й передачі цифрових даних обчислювальними мережами найбільш актуальне питання раціонального використання обсягів пам'яті ЕОМ, що породило ряд наукових напрямків. У

рамках даного дослідження інтерес представляють області, де вирішуються завдання компресії і декомпресії різного роду медіаданих і класів цифрової інформації, важливу частину яких становлять зображення. У цьому розділі докладно розглядаються етапи цифрової обробки зображень, алгоритм реалізації популярної технології стиску JPEG, основним елементом якої є.

Саме математичний метод дискретне косинусне перетворення Фур'є невеликих фрагментів зображення дає основний ефект стиску, що досягається даної технологією. Застосування такого перетворення в завданні стиску зображень обумовлене тим, що функція зображення мало змінюється на малих інтервалах, і коефіцієнти Фур'є стають зневажливо малі для високочастотних складових зображення. Однак у цьому випадку аналіз стисливої графіки допускає тільки частотну фільтрацію, не враховуючи тимчасові параметри.

Останнім часом впроваджується стандарт JPEG2000, у реалізації якого замість косинусного перетворення використовується вейвлет-перетворення. Вейвлетт-стиск допускає розкладання, що враховує частотно-часові характеристики зображення, чим становить явну конкуренцію класичному JPEG. Квантування й кодування методом Хаффмана є заключними в огляді Jpeg технології. Фрактальний метод стиску, принципи якого знаходять свою реалізацію в сучасних цифрових й навіть природничо-наукових напрямках.

Важливою особливістю наочної інформації є її більш простий аналіз, порівняння або ідентифікація. Питання про належність даного об'єкта до окремого описаного класу представляє собою сутність окремого підходу при роботі із графічними зображеннями, який у загальному можна назвати теорією розпізнавання образів.

Концепції математичних методів морфологічного аналізу цифрових зображень.

Широко ведеться розробка принципово нових математичних і комп'ютерних методів і алгоритмів для рішення завдання стиснення-відновлення. У розробку методів і алгоритмів компресії і декомпресії величезний внесок

внесли вітчизняні вчені, зокрема, у дослідженні завдання про представлення функції двох змінних.

Великого впливу на вартість комп’ютеризованих систем відеоспостереження також завдає підтримка відеопотоків, бо це породжує операції з великими за різноманітністю інформаційними наборами даних, що збільшує обчислювальні витрати. Саме через це зменшення даних, що передаються та зберігаються, у відепотоках є, фактичним засобом здешевлення вартості систем відеонагляду в цілому. Основним способом зниження розмірності даних є використання методів компресії. За компресією відео та аудіосигналів розрізняють системи, що базуються на програмній та апаратній реалізації. Основним недоліком систем з програмною компресією є завантаження центрального процесора ПК. Це означає, що окрім виконання інших завдань, значна кількість процесорного часу відводиться на обробку кадрів. Навіть у випадку коли, не усі кадри мають однаковий розмір, наприклад коли частота кадрів відеопотоку з одних камер становить 12 кадрів/с, а з інших 25, завантаження ЦП буде змінюватись у часі залежно від використання програмного декодера.

Під час кодування процесорний час зазвичай використовується для таких завдань як оновлення графічного інтерфейсу користувача, функцій відтворення відеопотоків, передача їх мережею і збереження на жорсткі диски, детектування руху [9]. У системах з програмною компресією значне завантаження ЦП призводить до втрати кадрів, спотворення інформації записаної у файли тощо. У системах з апаратною компресією усі ці ефекти відсутні.

Основною метою роботи є дослідження та покращення методів стискання у послідовностях зображень на основ морфологічних ознак, що надходять з цифрових камер у режимі реального часу.

Об’єктом дослідження виступає процес розпізнавання у відеорядах. Основними засобами дослідження є: методи теорії обробки цифрових зображень та машинного зору, теорії алгоритмів, комп’ютерного моделювання.

2 ЗАВДАННЯ АНАЛІЗУ Й ІДЕНТИФІКАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ

2.1 Розкладання Шмідта функції двох змінних

Розглянуто представлення функції *F(x,y)* у вигляді:

$$F(x,y) = f_1(x)g_1(y) + f_2(x)g_2(y) + \ldots + f_n(x)g_n(y) + p_{n+1}(x,y) \qquad (2.1)$$

Одержимо алгоритм визначення складових цього розкладання й оцінку залишку.

Знайти $\mu = inf\|F(x,y) - f(x)g(y)\|^2_{L_2(Q)}$ і мінімізуючу функцію.

За методом Шмідта рішення завдання існує і єдино. Точна нижня межа *ju* існує, отже, існує послідовність $f_n(x)g_n(y)$ така, що

$$\mu = \lim_{n\to\infty}\|F(x,y) - f_n(x)g_n(y)\|^2$$

Ця послідовність фундаментальна:

$$\|f_{n+p}(x)g_{n+p}(y) - fg\| = \|f_{n+p}g_{n+p} - F + F - f_n g_n\| \le \|F - f_{n+p}g_{n+p}\| + \|F - f_n g_n\| \to 0$$

при $n \to \infty$ й будь якому *p*. Отже, існує ліміт

$$\lim_{n\to\infty} f_n(x)g_n(y) = F_0(x,y),$$

тобто рішення завдання існує. Одиничність випливає зі строгої опуклості кулі в гільбертовому просторі.

Позначено $\|F_0(x,y)\| = C_0$, тоді

$$\|f_n(x)\|\|g_n(y)\| \to C_0.$$

На одиничній сфері S_1 отримано послідовності

$$u_n(x) = f_n(x)\|f_n(x)\|^{-1},$$
$$v_n(y) = g_n(y)\|g_n(y)\|^{-1}$$

Мають місце рівності

$$\mu = \lim_{n\to\infty}\|F(x,y) - C_0 u_n(x) v_n(y)\|^2,$$

$$\|F(x,y) - C_0 u_n(x) v_n(y)\|^2 = \|F(x,y)\|^2 - 2C_0(F, u_n v_n)_D + C_0^2.$$

Завдання, що розглянуте, еквівалентно наступному завданню. Знайти

$$\nu = \sup_{S_1}(F(x,y), u_n(x) v_n(y))_D$$

і максимізуючі функції $u_0(x), v_0(y)$.

Далі представляють, що $D = (0,1) \times (0,1)$. Перетворимо скалярний добуток:

$$(F(x,y), u(x)v(y))_D = \iint_D F(x,y)u(x)v(y)dxdy =$$
$$= \int_0^1 v(y)\int_0^1 F(x,y)u(x)dxdy = (v, Au)_{(0,1)}$$

де $(v, Au)_{(0,1)}$ – скалярний добуток в $L_2(0,1)$,

A – інтегральний оператор в $L_2(0,1)$,

$$Au = \int_0^1 F(x,y)u(x)dx.$$

Для рішення завдання досить максимізувати норму виразу Au на одиничній сфері $u \in S_1$ та в якості $v(y)$ взяти $Au\|Au\|^{-1}$.

Завдання III: знайти $\Lambda = \sup_{S_1} \|Au\|^2$ і максимізуючу функцію $u_0(x)$.

Так як

$$\Lambda = \sup_{S_1} \|Au\|^2 = \sup_{S_1} (A^* Au, u), \quad u \in S_1,$$

то рішення зводиться до стандартного спектрального завдання для інтегрального оператора A^*A, а Λ –його найбільше власне число, $u_0(x)$ – відповідна власна функція.

Якщо покласти

$$v_0(y) = \frac{Au_0}{\|Au_0\|} = \frac{\int_0^1 F(x,y)u_0(x)dx}{\left[\int_0^1 \left(\int_0^1 F(x,y)u_0(x)dx\right)^2 dy\right]^{1/2}},$$

то

$$v = \sup_{u,v \in S_1} (v, Au) = (v_0, Au_0) = \|Au_0\| = \sqrt{\Lambda}.$$

Звідси випливає, що

$$\mu = \|F(x,y)\|^2 - 2C_0\sqrt{\Lambda} + C_0^2$$

і мінімізуючи це вираження по C_0, одержують, що $C_0 = \sqrt{\Lambda}$.

Остаточно, рішення завдання має вигляд

$$F_0(x,y) = C_0 u_0(x) v_0(y) = \sqrt{\Lambda} u_0(x) v_0(y)$$

і має місце оцінка

$$\left\|F(x,y) - \sqrt{\Lambda} u_0(x) v_0(y)\right\|^2 = \|F(x,y)\|^2 - \Lambda$$

Оцінка погрішності.

$$F_1(x,y) = F(x,y),$$

Якщо $A_к$ – інтегральний оператор з ядром $F(x,y)$, Λ_k – максимальне власне число оператора $A_k^* A_k$, u_k – відповідна власна функція цього оператора,

$$v_k = A_k u_k \|A_k u_k\|^{-1},$$

$$F_{k+1} = F_k - \sqrt{\Lambda_k} u_k v_k .$$

Так як

$$\|A_k\|^2 = \sup_{S_1} \|A_k u\|^2_{(0,1)} = \|A_k u_k\|^2_{(0,1)} =$$

$$= \int_0^1 (\int_0^1 F_k(x,y) u_k dx)^2 dy \le \int_0^1 (\int_0^1 F_k^2(x,y) dx) dy = \|F_k(x,y)\|^2_D$$

Оцінка (2.1) переписується наступним чином

$$\left\|F_k - \sqrt{\Lambda_k} u_k v_k\right\|^2_D = \|F_k\|^2 - \Lambda_k$$

або, враховуючи наші міркування,

$$\|F_{k+1}\|^2 = \|F_k\|^2 - \Lambda_k$$

Тому що $F_1(x, y) = F(x,y)$, то рівність (1.1) остаточно запишеться у вигляді

$$F_1(x,y) = \sum_{k=1}^{n} \sqrt{\Lambda_k} u_k(x) v_k(y) + F_{n+1}(x),$$

тобто

$$\rho_{n+1}(x,y) = F_{n+1}(x,y),$$

і використовуючи рекурентне співвідношення (2.3), маємо

$$\|\rho_{n+1}(x,y)\|^2 = \|F_k\|^2 - \Lambda_k = \|F_{k-1}\|^2 - \Lambda_{k-1} - \Lambda_k = \ldots = \|F(x,y)\|^2 - \Lambda_1 - \Lambda_2 - \ldots - \Lambda_n .$$

Звідси випливає, що $\|\rho_{n+1}(x,y)\| \to 0$ при $n \to \infty$.

Наведено доказ методу Шмідта для представлення функції

$$f(x,y),\quad f(x,y)\in L_2(Q),\quad (x,y)\in Q=(0,1)\times(0,1)$$

у вигляді

$$f(x,y)=\sum_{k=1}^{N}\lambda_k a_k(x)b_k(y)+\rho_N(x,y),$$

де функція $\rho_N(x,y)$ прагне до нуля при $N\to\infty$.

2.2 Методи аналізу цифрових зображень у завданнях ідентифікації

Завдання розпізнавання цифрових зображень повинно формулюватися для певного класу зображень, тому що вирішальна процедура повинна використовувати властивості цього класу. При цьому саме поняття класу зображень звичайно визначається інтуїтивно й описово.

Вирішальна процедура може містити, зокрема, алгоритм компресії, який виділяє деякий числовий набір $\bar{r}$ ознак зображення. Це відображення $f(x,y)\to\bar{r}$ повинне бути однозначно оборотне (з точністю до малих варіацій для $f(x, y)$).

Попередньо можуть бути проведені процедури стандартизації, які перетворять $f(x, y)$ до більш зручного виду. Наприклад, це може бути перетворення певного згладжування, що допускає зворотне перетворення, і це відображення також повинне бути взаємно однозначним.

Нехай класами зображень є класи фотозображень у фас, фотозображень у профіль і т.п. Із числових ознак особливо ефективні ті, які інваріантні при основних перетвореннях, що можуть зазнавати зображення даного класу, наприклад, зрушення, повороти, розтягання й інші.

Для таких класів і таких перетворень інваріантами є, зокрема, центр ваги (при цьому $f(x, y)$ трактується як щільність мас) і головна вісь інерції.

Ідентифікацію зображень зручно проводити в тому випадку, коли чітко виділені основні інваріанти зображення – центр маси й головна вісь інерції. Тому що дані величини не залежать від «якості» зображення, пропонується провести операцію відкидання тих затінених (прояснених) областей, які несуть несуттєву інформацію в плані ідентифікації.

Визначимо нівелювання як процедуру відкидання лінійної частини в завданні функції зображення – виділення лінійної частини $ax + by + з$ функції $f(x, y)$ і вирахування її з $f(x, y)$. Нехай $f(x, y)$ визначена в області Q. Коефіцієнти a, b, c обчислюються мінімізацією різниці

$$ax + by + c - f(x, y)$$

у нормі простору $L_2(Q)$. Позначимо через $F(a,b,c)$ наступну функцію

$$F(a,b,c) = \| f(x,y) - (ax + by + c) \|^2_{L_2(Q)}.$$

Завдання мінімізації функції $F(a,b,c)$ приводить до системи рівнянь

$$\frac{\partial F}{\partial a} = \frac{\partial F}{\partial b} = \frac{\partial F}{\partial c} = 0$$

і одержують лінійну алгебраїчну систему з невирожденою матрицею для визначення коефіцієнтів a, b, c.

Центр ваги (x_1, y_1) визначається стандартно.

Момент інерції m_L пластини Q із щільністю $f(x, y)$ щодо осі L, що проходить через (x_1, y_1), визначається інтегралом

$$m_L = \iint_Q f(x, y)\, p(x, y)\, dxdy,$$

де $p(x, y)$ – відстань від (x, y) до осі L .

Положення осі L залежить від кута φ з віссю Ox, $m_L = m(\varphi)$. Функція $m(\varphi)$ періодична з періодом $T = \pi$, максимальному значенню $m(\varphi)$ відповідає головна вісь інерції.

Розглянемо алгоритм визначення головної осі. Замість повороту осі L будемо обертати зображення.

Нехай центр ваги визначений, помістимо його в початок координат $(x_1, y_1) = (0,0)$. (L збігається з Ox), $(x, y)=(r \cos t, r \sin t)$,

$$f\,(r\cos t, r\sin t) := g\,(r, t),$$

тоді

$$m\,(\varphi) = \iint_Q g(r, t+\varphi) r \cos t\, r\, dtdr\,.$$

Можна довести, що має місце рівність

$$m''\,(\varphi) = -4\,m\,(\varphi),$$

отже, $m(\varphi)$ представляється у вигляді

$$m\,(\varphi) = a\cos 2\varphi + b\sin 2\varphi\,.$$

Для визначення $m(\varphi)$ досить обчислити $m(\varphi)$ для двох значень кута і визначити коефіцієнти a, b із системи

$$\begin{cases} a\cos 2\varphi_1 + b\sin 2\varphi_1 = m(\varphi_1) \\ a\cos 2\varphi_2 + b\sin 2\varphi_2 = m(\varphi_2) \end{cases}.$$

Кут φ_0 головної осі інерції визначається далі рівністю

$$m'(\varphi_0) = 0\,.$$

2.3 Виділення гармонійної складової функції

Нехай Q – обмежена область із границею Ляпунова S,

$$S \in C^{1+\alpha}, \alpha > 0.$$

Позначено

$$Q^{+} = R^{2} \setminus \overline{Q}, \qquad Q^{-} = Q.$$

Позначимо також $\Gamma(Q)$ – підпростір в $L_2(Q)$, що полягає з гармонійних в Q функцій, а через $N(Q)$ – ортогональне доповнення до нього.

Таким чином, якщо Q – обмежена область в R^n, $(n=2, 3,...)$ із границею Ляпунова ∂Q, то простір $L_2(Q)$ має наступне розкладання в пряму суму

$$L_2(Q) = \Gamma(Q) \oplus N(Q),$$

де $\Gamma(Q)$ – підпростір гармонійних в Q функцій, а функція $p(x)$ належить $N(Q)$ тоді й тільки тоді, коли

$$\int_Q p(y)E(x-y)dy = 0, \quad x \in Q^{+}.$$

Розглянуто в $L_2(Q)$ множину функцій $p(x)$, що задовольняють умові

$$\int_Q p(y)E(x-y)dy = 0, \quad x \in S,$$

область Q є обмеженою з досить гладкою границею S.

Показано, що ця множина утворює підпростір в $L_2(Q)$. Воно є, очевидно, лінійним різноманіттям. Позначимо його $N(Q)$, потрібно довести його замкнутість.

Нехай $p_n(x) \in N(Q)$, $(n = 1,2,..)$. , і послідовність $p_n(x)$фундаментальна в $L_2(Q)$. Позначити граничну функцію $p_0(x)$ і розглянемо функцію

$$f(x) = \int_Q p_0(y)E(x-y)dy.$$

У граничній точці $x^1 \in S$, ми маємо

$$\left| f(x^1) - \int_Q p_n(y)E(x^1 - y)dy \right| = \left| \int_Q (p_0(y) - p_n(y))E(x^1 - y)dy \right| \le$$

$$\le \|p_0 - p_n\|_{L_2(Q)} \left(\int_Q E^2(x^1 - y)dy \right)^{\frac{1}{2}} \xrightarrow[n\to\infty]{} 0.$$

2.4 Компоненти кодованого відео та рекомендації H.264

Знаходження відмінностей є важливою операцією обробки зображень, яка ідентифікує зміни пікселів, пов’язаних з рухом або зміною об’єкта в заданому фоні (до фону ми відносимо усі об’єкти, що не рухаються). Як результат цієї операції, отримуємо пікселі або області, які можна класифікувати як змінені або без змін на основі заданого критерію.

Так як одним з найпопулярніших стандартів стиснення відео, призначеним для досягнення високого ступеню стиснення відеопотоку при збереженні високої якості, є H.264, тому основні етапи саме цього стандарту буде розглянуто.

Одним з основних алгоритмі, що використовуються при обробці та стисненні відео, являється алгоритм компенсації руху. Даний алгоритм використовую схожість сусідніх кадрів у відео послідовності та знаходить вектори руху окремих частин зображення (зазвичай використовуються блоки розміром 16 на 16 або 8 на 8 пікселів). Використання компенсації руху при стисненні дає змогу багатократно збільшити ступінь стиснення за рахунок видалення надлишковості у вигляді частин кадрів, що збігаються.

Рішення проблеми стиснення стало найважливішим завданням з самого з’явлення цифрового відео, бо для зберігання відео файлу, як звичайного набору

картинок, потрібно було багато пам’яті. Ідея алгоритму полягає в тому, що практично в любій відео послідовності сусідні кадри схожі, мають спільні об’єкти, що, як правило, зміщуються відносно один одного. Тому цілком доцільно закодувати відео так, щоб об’єкти не кодувалися багаторазово, а просто описувалось деяке їх зміщення.

Під час стиску відеоданих у форматі H.264 застосовується алгоритм компенсації руху [16], який дозволяє точно знаходити відмінності на зображенні у відеопотоці. Формат H.264 є ефективним стандартом кодування відео. Одна з ключових переваг методів визначення відмінностей полягає у простоті їхньої реалізації. Проте вони мають недоліки — це висока чутливість до шуму та освітлення.

У кодері, що описується в рекомендації H.264, у режимі прогнозування з компенсацію руху підтримуються розміри блоків від 16x16 до 4x4 для складової яскравості зображення в різних поєднаннях. Складова яскравості кожного макроблоку розміром 16x16 може бути розбитий чотирма способами (див. рис. 2.1)

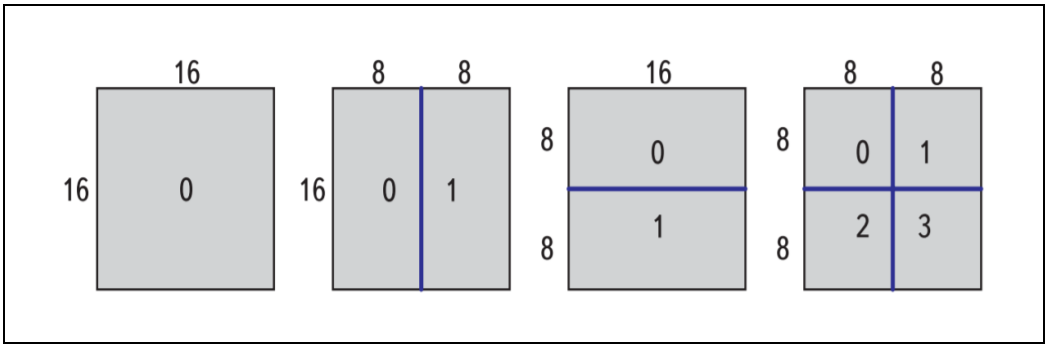

Рисунок 2.1 – Розбиття макроблоку на блоки 16x16, 16x8, 8x16, 8x8

Кожна з областей меншого розміру являється частиною макроблоку. Якщо вибрано режим пошуку векторів руху для блоків 8x8, то кожен з чотирьох блоків 8x8 може бути поділено на блоки чотирма способами, як показано на рисунку 2.2

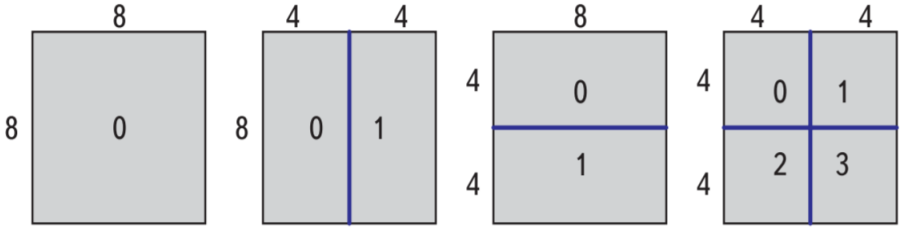

Рисунок 2.2 – Розбиття блоку 8x8 на субблоки 8x8, 8x4, 4x8, 4x4

Можливість ділити макроблок на блоки, а ті в свою чергу на субблоки, дозволяє отримати велику кількість комбінацій їх об’єднань в межах кожного макроблоку. Цей метод розбиття макроблоків на блоки та субблоки в режимі компенсації руху отримав назву структурного дерева компенсації руху. Для кожного блоку і субблоку потрібен окремий вектор руху. Кожен вектор руху повинен бути закодований і переданий, крім того потрібно закодувати і передати конкретний вид структурного дерева для кожного макроблоку.

Вибір великого розміру блоку означає, що кількість біт для передачі даних векторів руху та структурного дерева мінімальна, проте при цьому залишкові коефіцієнти можуть містити суттєву кількість енергії в високочастотних областях кадру (з високою дальністю). Натомість, при виборі маленького розміру блоків можемо отримати більш низьку енергію залишкових коефіцієнтів після компенсації руху, але потрібна більша кількість біт для кодування та передачі даних про вектори руху та структурі дерева. Тому вибір розміру блоків суттєво впливає на ефективність стиснення. Взагалі, великий розмір блоків відповідає низькочастотним областям кадру, а маленький – може бути вигідний для високочастотних.

Блок може бути передбачений іншим блоком з попередніх фреймів, якщо різниця між блоками є мінімальною. Різниця між блоками описується компенсацією руху. Блоки не змінюються, а зсуваються в положення передбаченого блоку. Цей зсув зображено вектором руху (див. рис. 2.3).

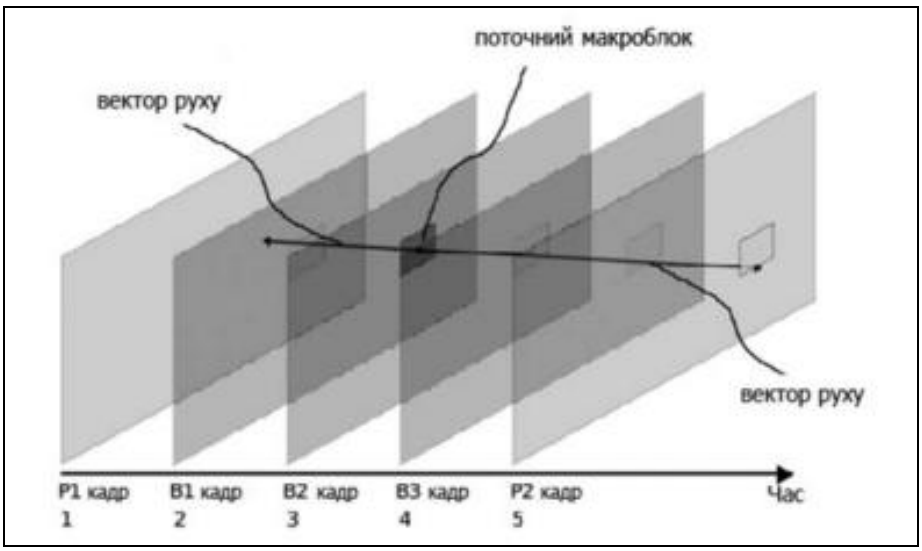

Рисунок 2.3 – Застосування компенсації руху до макроблоку B-кадру

Алгоритм компенсації руху дозволяє точно виділяти області руху. Кадри відеопотоку відрізняються за типами залежно від способу кодування, зберігання інформації та наявності або відсутності залежностей цього кадру від попередніх і наступних.

Залежно від типу кадру існують також і різні типи макроблоків. I-кадри відеопотоку, або ключові кадри [17], можуть складатися тільки з intra-кодованих макроблоків – це незалежно стислі макроблоки, без використання компенсації руху. Тоді як P-кадри, «різницеві» кадри можуть містити intra- і inter-кодовані макроблоки, з використанням компенсації, з посиланням на інший I- або P-кадр. B-кадри, або «двонаправлені» кадри, містять теж intra- і inter-кодовані макроблоки з посиланням на один кадр або на два найближчих кадри I-, P або B-кадри. Багато «помилкових» блоків руху з’являється, якщо враховувати всі присутні вектори руху, навіть коли немає руху та об’єкт не присутній. Блоки руху з ненульовими векторами руху знаходяться у тій області зображення, де знаходиться об’єкт, що рухається. Чим більше значення вектору руху, тим ближче розташований об’єкт або тим швидше рухається.

Кожен різницевий макроблок перетворюється, сканується та квантується. Попередні стандарти, такі як MPEG-1, MPEG-2, MPEG-4 та H.263, використовували в якості базового перетворення дискретне косинусне

перетворення для масивів розміром 8x8. У рекомендації H.264 визначені три види перетворення в залежності від типу оброблюємих блоків:

– перетворення блоку розміром 4x4 нульових коефіцієнтів яскравості, отриманих з макроблоків яскравості розміром 16x16, сформованих у режимі INTRA;

– - перетворення блоку розміром 2x2 нульових кольорорізністних блоків любого типу;

– перетворення для всіх останніх різністних блоків розміром 4x4.

Дані у межах одного макроблоку передаються у порядку, зображенному на рисунку 2.8. Якщо макроблок розміром 16x16 обробляється в режимі INTRA, то спочатку передається блок коефіцієнтів розміром 4x4. Він маркується значенням «-1», після чого передаются останні блоки (0-15) у порядку, зображенному на рисунку 2.4.

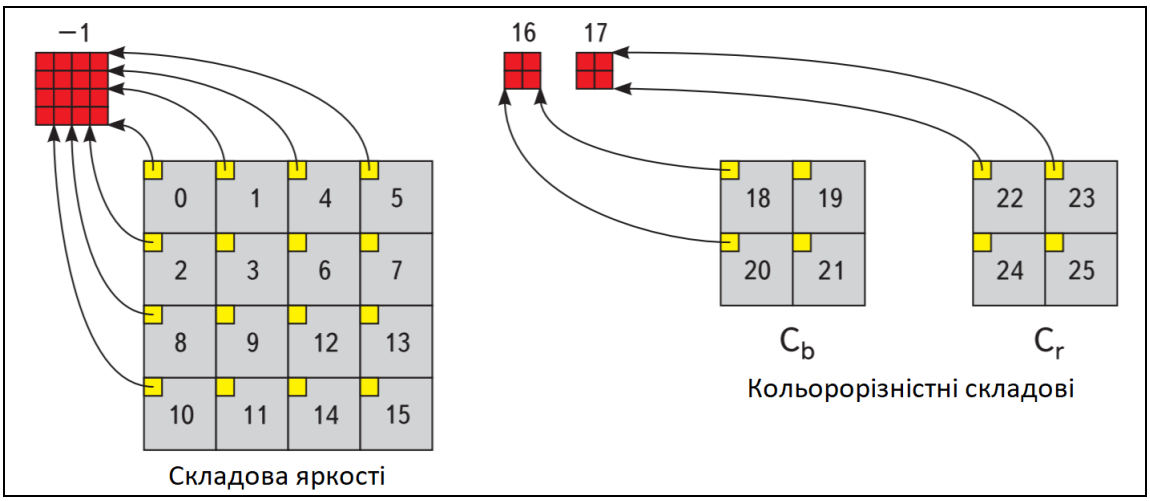

Рисунок 2.4 – Порядок сканування блоків у макроблоці

Блоки 16 та 17 містять масиви розміром 2x2, що містять коефіцієнти з кольорорізницевих C_b- та C_r- складових відповідно. І в самому кінці передаються кольорорізницеві блоки 18 та 25.

3 АЛГОРИТМИ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ РУХОМИХ ОБ'ЄКТІВ

3.1 Алгоритм відстеження центра мас об'єкту

Теорія розпізнавання об'єктів – це розділ кібернетики, який розвиває теоретичні основи та методи класифікації та ідентифікації предметів, явищ, процесів, сигналів, ситуацій та інших об'єктів, що характеризуються кінцевим набором деяких властивостей та ознак. Такі задачі розв'язуються доволі часто, наприклад, при переході вулиці за сигналами світлофора. Розпізнання кольору лампи світлофора, що загорілася та знання правил дорожнього руху дозволяють прийняти правильне рішення про те, можна чи ні переходити вулицю в даний момент.

У розпізнаванні об'єктів можна виділити два основних направлення: вивчення властивостей до розпізнавання у живих істот, їхнє пояснення і моделювання; а також розвиток теорії та методів побудови пристроїв, що призначені для розв'язання окремих завдань у прикладних задачах.

Розпізнавання об'єктів – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані із загальної маси неіснуючих даних. При постановці задач розпізнавання намагаються користуватися математичною мовою, намагаючись замінити експеримент логічними міркуваннями та математичними доведеннями. Найчастіше в задачах розпізнавання об'єктів розглядаються монохромні зображення (зображення, що містять колір лише одного спектру, що сприймається як один відтінок), що дає можливість розглядати зображення як функцію на площині. Якщо розглядати точкову множину на площині T, де функція x(x, y) виражає в кожній точці зображення його характеристику – яскравість, прозорість, оптичну щільність, то така функція є формальним записом зображення.

Графічні дані поділяються на два класи: векторні та растрові [18]. Оскільки суть роботи програми полягає у розбитті потоку відеоданих на кадри, що за замовчуванням мають растровий формат .bmp (це має сенс, адже файли такого

типу особливо добре підходять для зберігання зображень реального світу, наприклад цифрованих фотографій), до уваги бралися саме ті алгоритми, котрі давали змогу якомога швидше й краще аналізувати растрові зображення. Наприклад, алгоритм детектора руху, який порівнює два bitmap-файли, що були захоплені з потокового відео і шукає між ними невідповідності або ж алгоритм відстеження руху об’єкта, в якому необхідно розрахувати так званий, центр мас рухомого тіла.

Наразі найбільш популярною моделлю для передачі кольорів є модель RGB. RGB (абревіатура від англ. Red, Green, Blue – червоний, зелений синій) – адитивна кольорова модель, що описує способи синтезу кольорів. Вибір основних кольорів для цієї моделі обумовлюється особливостями їх сприйняття людською сітківкою ока. Адитивною вона ще називається тому, що кольори отримуються шляхом додавання основних кольорів різної інтенсивності до чорного. Кольорова модель RGB спочатку була розроблена для опису кольорів на кольоровому моніторі, але оскільки монітори різних моделей та виробництва відрізняються між собою, було запропоновано декілька альтернативних кольорових просторів, що відповідають «усередненому» монітору. До таких відносяться, наприклад, sRGB и Adobe RGB.

З точки зору механіки центр мас – це геометрична точка, що характеризує рух тіла або системи часток як єдиного цілого. Ідея алгоритму – щоразу знаходити центр об’єкту і слідкувати за його подальшим переміщенням досить вдала, тому що таким чином програмою фактично ігнорується обертання об’єкта навколо власної осі в тому випадку, якщо такі обертання дійсно мають місце [19]. Суть даного алгоритму полягає в тому, що кольорові компоненти усіх пікселів кадру порівнюються підряд із тим пікселем (назвемо його еталонним), на якому раніше програмою був зафіксований рух за формулою (див. рис. 3.1).

$$A_{i,j} \uparrow 30\ (r \acute{H} rc)^2 \acute{G} 59\ (g \acute{H} gc)^2 \acute{G} 11\ (b \acute{H} bc)^2,$$

Рисунок 3.1 – Формула зафіксованого руху

R, G, B – кольорові компоненти еталонного пікселя; rc, gc, bc – кольорові компоненти поточного пікселя, що обробляється. У зв’язку з тим, що кольорові компоненти не бажано сприймати, як рівноправні (у них різні відтінки основних кольорів, різна кольорова температура, показник гамма-корекції тощо), Міжнародний союз електрозв’язку емпіричним шляхом установив і рекомендував для використання у подібних формулах оптимальні числові коефіцієнти для кожної кольорової компоненти окремо (30 – для червоної, 59 – для зеленої, 11 – для синьої). Результати розрахунків формули для кожного пікселю записуються у матрицю пікселів A. Після обробки кадру та обчислення всіх значень, обирається декілька елементів масиву A, що мають найменше значення, і серед відповідних їм координат (i,j) обчислюється середнє значення, що й буде координатами центра мас. Цей алгоритм є досить точним і зручним у використанні, проте занадто велика кількість обчислень, які необхідно виконати, не дозволяє використовувати цей алгоритм в реальному часі.

Але існує декілька варіантів спростити цей алгоритм. Один з них це перевіряти не всі пікселі кадру, а брати лише деяку вибірку з пікселів і шукати серед них еталонний колір. У результаті, серед координат знайдених точок визначаються координати лише одної, що й буде центром мас. Кожен раз, в міру того, як обчислюються нові координати центра мас, будується траєкторія руху об’єкта.

Алгоритм побудови траєкторії руху об’єкту активізується лише тоді, коли на попередньому етапі роботи програми був знайдений рух і став відомим колір точки рухомого об’єкту (назвемо його еталонним). Алгоритм безпосереднього відстежування полягає у постійній вибірці з кадру випадковим чином деякої кількості точок (у співвідношенні швидкодії та точності, достатньо взяти вибірку з 300 пікселів). Далі для кожної з точок виділяємо їх червону, зелену та синю компоненти та перевіряємо наступні умови (див. рис. 3.2).

$$|r1 - r2| \le 16, \quad |g1 - g2| \le 16, \quad |b1 - b2| \le 16$$

Рисунок 3.2 – Умови перевірки після виділення компонент

В формулі *r1, g1, b1* – кольорові компоненти поточного пікселя; r2, g2, b2 – кольорові компоненти еталонного пікселя; 16 – значення, яке визначає допустимий ступінь наближення кольорового пікселя до еталонного. У тому випадку, якщо умови виконуються, то колір поточного пікселю дуже подібний до кольору самого об’єкта, отже існує висока ймовірність того, що знайдений піксель належить цьому об’єкту. У цьому випадку програма запам’ятовує координати цього пікселю. Після того, коли усі точки будуть перевірені, програма звертається до координат тих пікселів, які пройшли умову, і знаходить серед них середнє арифметичне, що й буде координатами необхідного центра мас рухомого об’єкта. Отримана точка наноситься на графік.

Ідея алгоритму полягає в наступному. Спочатку обчислюється середнє значення інтенсивності пікселів (Im), що знаходяться в двох граничних стовпчиках і рядках матриці розмірністю MxN пікселів. Порогова межа шуму визначається як 3*Im. Далі сканується матриця розмірністю M-2xN-2, визначається піксель, інтенсивність якого вище, ніж вибраний порог, а після цього сканується деяка область S, центром якої є даний піксель (розміри можуть варіюватись). Якщо кількість пікселів, інтенсивність яких більше ніж порогове значення, всередині обраного вікна S менше деякого значення R, то припускається, що це не пляма, і сканування продовжується. У тому випадку, якщо підходящий піксель був знайдений, визначається так звана «Область фокусу» (ОФ), розмірністю AxA, центром якої є цей піксель. Шляхом усереднення значень інтенсивності пікселів, що знаходяться в двох граничних строках і стовпцях ОФ, визначається новий шумовий поріг. Сканується область розмірністю A-2xA-2 із метою виявлення пікселів, інтенсивність яких нижча за пороговий рівень. Якщо інтенсивність пікселя менше або дорівнює пороговому значенню, вона прирівнюється до нуля, а якщо більше, то обчислюється як «інтенсивність-поріг». Наприкінці, коли вся область просканована, координати її центра мас обчислюються в системі координат (X,Y).

Фактично, ніяке обмеження на розмірність матриці MxN не встановлюється, вона може бути довільною. Експерименти, що були проведені, показують, що не варто брати велику область S, тому що це ускладнює роботу алгоритму та зменшує точність результату. Щоб мати достовірність результату хоча б 92%, доцільно брати область S розмірністю 5x5. За таких умов, найоптимальніше значення R, що є кількістю пікселів, інтенсивність яких більша, ніж порогове значення, становитиме 18. Це значення було обране з припущення, що величина R повинна становити близько ¾ від загальної кіль- кості пікселів області S.

3.2 Метод детекції руху у стиснутому відеопотоці

Нашим завданням є детекція змін на зображенні за наявністю векторів руху. Кожний inter- кодований блок з P- і B-кадрів має вектор руху, цей блок будемо називати блоком руху, оскільки саме в цьому блоці відбулися зміни. Багато «помилкових» блоків руху з’являється, якщо враховувати всі присутні вектори руху, навіть коли немає руху та об’єкт не присутній (див. рис. 3.3). Для вирішення цієї проблеми застосовують фільтрацію.

Фільтрація векторів руху є дуже важливим завданням при аналізі руху в області стиснення, оскільки від цього залежить надійність результату. Найчастіше використовують порогове значення. Якщо відстань вектора менше заданого порогового значення, вектор руху не враховують. Додатково застосовують медіанний фільтр разом з усереднюючим. У запропоновано деякий модифікований фільтр Mean-AccumulatedThresholded (MAT) filter. У використовують каскадний фільтр, який складається з двох послідовних фільтрів Гауса та медіанного.

Досліджено, що при застосуванні модифікованого фільтра, каскадного фільтра та окремо при застосуванні гаусового, медіанного, усереднюючого

фільтра та порогового значення, залишаються «помилкові» блоки та губляться дрібні об’єкти, що повільно рухаються.

Рисунок 3.3 – Приклад розташування усіх блоків руху на зображені

Це пов’язано з тим, що об’єкти рухаються з різною швидкістю. Якщо рух повільний, відстань векторів буде незначною, а через недоліки освітлення та передачу даних відстань векторів може бути значною.

Блоки руху з ненульовими векторами руху знаходяться у тій області зображення, де знаходиться об’єкт, що рухається. Чим більше значення вектору руху, тим ближче розташований об’єкт або тим швидше рухається.

Для знаходження «істинних» блоків руху ми знаходимо лише довжини векторів руху в блоках 8x8, 16x8 та 8x16, оскільки виявили, що «помилкові» блоки частіше знаходяться в неподілених макроблоках 16x16 (див. рис. 3.4). А «істинні» блоки руху представлено векторами в блоках 8x8, 16x8 та 8x16. Додатково ми використовуємо фільтрацію.

Вектор руху повинен мати більше двох ненульових сусідніх векторів і мати відстань більше двох. Це випливає з розміру досліджуваного об’єкта. Після фільтрації можемо проаналізувати зображення за таким параметром, як сума довжин усіх векторів руху.

Рисунок 3.4 – Приклад розташування помилкових блоків після фільтрації

Якщо сума довжин усіх векторів більше відповідного порогового значення h, можна стверджувати, що на зображенні присутні рухомі об’єкти, відповідно можна знайти області рухомих об’єктів на зображенні. Порогове значення *h* задається залежно від необхідності не враховувати дрібний рух, наприклад рух птахів.

3.3 Алгоритм переміщення фрагментів об’єктів

Кожен рухомий об'єкт розглядається як сукупність рухомих фрагментів. Кожен фрагмент є областю кадру довільного розміру і форми, що містить частину об'єкту або цілий об'єкт. Відстежування переміщень об'єктів може бути зведене до двох основних кроків:

– відстежування переміщення всіх рухомих фрагментів у відеопотоці;

– побудова траєкторій руху об'єктів на основі результатів відстежування переміщення їх фрагментів. Групи близько розташованих фрагментів, які рухаються узгоджено, об'єднуються в об'єкти.

Запропонований підхід дозволяє:

– виконувати відстежування об'єктів, які частково перекриті іншими об'єктами. Для цього достатньо успішно відстежити переміщення хоча б одного фрагмента об'єкту. По траєкторії цього фрагмента можливо відтворити траєкторію всього об'єкту;

– виконувати відстежування множини об'єктів, які протягом всього часу свого існування є частково перекритими і утворюють одну “пляму”. Таку група об'єктів може бути сегментовано на окремі об'єкти за траєкторіями руху їх частин;

– виконувати відстежування об'єктів, що сильно змінюють свою форму. Для цього достатньо, щоб у об'єкту існувала видима частина, яка міняє свою форму відносно слабо і може бути відстежена;

Дії, що відбуваються при аналізі одного кадру (загальний крок алгоритму), представлені на рисунку 3.5. Овалами показані дані, прямокутниками — виконувані дії, стрілками — потоки даних.

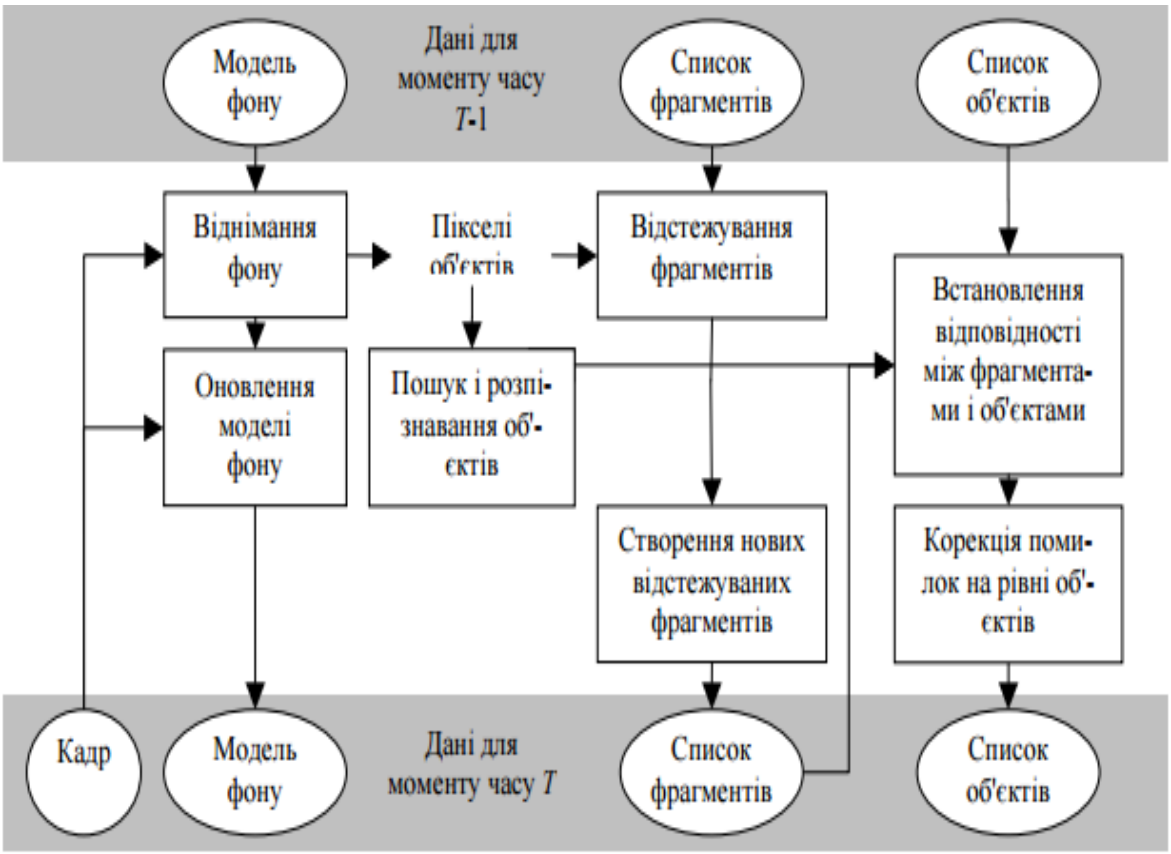

Рисунок 3.5 - Загальний крок алгоритму відстежування об'єктів

Знаходження об'єктів і їх координат на основі траєкторій фрагментів можна виконати один раз після завершення відстежування фрагментів у всьому відеозапису, або виконувати в кожному кадрі після відстежування фрагментів в цьому кадрі. У першому випадку, маючи більше інформації, можна допустити менше помилок. Проте, у багатьох випадках необхідне отримання координат об'єктів у реальному часі. У кожен момент часу в пам'яті зберігаються усі об'єкти, фрагменти і їх координати за останні T_{window} секунд. З кожним новим кадром відбувається отримання нових координат і, якщо це необхідно, уточнення одержаних раніше. Таким чином, запропонований підходи дозволяє отримувати координати як у реальному часі, так і після закінчення відстежування.

Віднімання фону і моделювання фону. Віднімання фону, побудова і оновлення моделі фону є поширеним підходом до виділення рухомих об'єктів у відеопотоці. Це дозволяє виділити в кожному кадрі пікселі, які відрізняються від статичного фону і належать рухомим об'єктам.

Пошук і розпізнавання об'єктів групуючи фрагменти, які рухаються узгоджено, запропонований алгоритм може виділяти об'єкти довільного виду [20]. Тому пошук і розпізнавання об'єктів в кадрах не є обов'язковим. Для угрупування фрагментів, що мають схожі траєкторії, необхідно, щоб вони рухалися протягом деякого часу. Через це об'єкти, що щойно з'явилися, не можуть бути виділені відразу. Пошук і розпізнавання об'єкту, що щойно з'явився, дозволить об'єднати всі його фрагменти відразу після появи об'єкту.

Визначення фрагмента і об'єкту. Відстежуваний фрагмент f_k характеризується наступними властивостями:

– $Widthk$, $Heightk$ — ширина і висота прямокутника, що містить фрагмент;

– Ik — матриця розміром $Heightk \times Widthk$, що містить пікселі фрагмента. Значення пікселів, що не відносяться до фрагмента, можуть бути довільними;

– Mk — матриця розміром $Heightk \times Widthk$, яка показує, які пікселі з Ik і з якою впевненістю відносяться до даного фрагменту, $m_{kij} \in [0,1]$;

– $[T_k^{min} \quad T_k^{max}]$ — проміжок часу, в якому відомі координати даного фрагмента, тобто його час існування;

– X (k)t, Y (k)t — координати лівого верхнього кута фрагмента у момент часу $t \in [T_k^{min}, T_k^{max}]$;

– Idk дорівнює унікальному ідентифікатору об'єкту, до якого відноситься даний фрагмент, або особливому значенню None , якщо фрагмент не відноситься до жодного об'єкту.

Різні фрагменти можуть перетинатися. Більшість пікселів об'єктів відносяться як мінімум до одного фрагмента. В процесі відстежування відбувається оновлення матриць I і M .

Відстежування переміщення фрагментів. Для відстежування переміщення фрагментів можна застосовувати швидкі алгоритми пошуку прямокутних фрагментів, розроблені для кодування відео. Відстежування переміщень фрагментів при відстежуванні об'єктів має дві особливості: фрагмент може мати довільну форму, необхідно оцінювати надійність відстежування (ймовірність того, що відстеження переміщення даного фрагменту відбулося вірно).

Нехай *V* – область пошуку фрагмента *fk* у кадрі *A*. Розглянемо клас алгоритмів, які в процесі відстежування обчислюють різницю між фрагментом *fk* і областю кадру *A* за формулою

$$D(x, y) = \sum_{dx=0}^{Width_k - 1} \sum_{dy=0}^{Height_k - 1} F\left(a_{x+dx, y+dy}, i_{k,dx,dy}\right), \tag{3.1}$$

і визначають координати фрагменту кадру *A* , найбільш схожого на фрагмент *fk* як

$$\left(x_{\min}, y_{\min}\right) = \arg \min_{(x, y) \in V} D(x, y) \tag{3.2}$$

де F — деяка функція (наприклад, модуль різниці яскравостей).

Такі алгоритми можуть бути легко модифіковані для відстежування фрагментів довільної форми. Для визначення надійності відстежування фрагмента використовуються наступні критерії:

– вибраний фрагмент повинен бути достатньо схожий на фрагмент *fk*;

– усі інші фрагменти з області пошуку *V* повинні бути достатньо несхожі на шуканий фрагмент *fk* та задовольняти умові (3.3)

$$D_{\min} = \min_{(x,y)\in V,\,|(x,y),(x_{\min},y_{\min})|\geq r} D(x,y), \qquad D_{\min} \geq sD(x_{\min}, y_{\min}) \tag{3.3}$$

де Dmax , r і s — константи.

Для перевірки умови (3.3) необхідно виконати порівняння всіх фрагментів з області пошуку *V* з фрагментом *fk*. При цьому відбуваються практично ті ж самі дії, що і при пошуку фрагмента методом повного перебору всіх його можливих положень, що має велику часову складність. Доцільно виконувати перевірку (3.3) один раз в декілька кадрів. Передбачення найбільш вірогідного положення фрагмента і області, в якій він може знаходитися здійснюється на підставі відомих координат фрагмента або об'єкту, до якого він належить, а також заданих максимальної швидкості та прискорення. Відстежування фрагмента припиняється, якщо його не вдалося знайти в Tfragment послідовних кадрів. Якщо фрагмент був втрачений, а потім знайдений, його невідомі координати інтерполюються по відомих.

Створення нових відстежуваних фрагментів. Один раз в декілька кадрів виконується пошук всіх пікселів, які не відносяться ні до фону, ні до наявних фрагментів. Створюються нові відстежувані фрагменти, що покривають всі достатньо великі групи знайдених пікселів. Визначення оптимального положення, розмірів і форми створюваних фрагментів вимагає подальших досліджень.

Встановлення відповідності між фрагментами і об'єктами. Алгоритм встановлення відповідності між фрагментами і об'єктами складається з наступних

кроків, які дозволяють обробляти всі ситуації, пов'язані з виникненням, зникненням, злиттям і розділенням видимих об'єктів:

– відділення фрагментів від об'єктів (включаючи розділення об'єктів), якщо їх рух не узгоджений;

– віднесення фрагментів до існуючих об'єктів, з якими їх рух є узгодженим;

– злиття об'єктів, рух яких є узгодженим;

– видалення об'єктів, у яких не залишилося відстежуваних фрагментів;

– створення нових об'єктів із фрагментів, що не відносяться до існуючих об'єктів.

Побудова траєкторії руху об'єкту. Траєкторія руху будується на базі траєкторій фрагментів, що відносяться до об'єкту, а також положень об'єкту в окремих кадрах, де він був розпізнаний. Введемо наступні позначення (3.4)

$$Displacement(f_i, t) = \begin{cases} (X_i(t) - X_i(t-1), Y_i(t) - Y_i(t-1)), t \in \left[T_i^{\min} + 1, T_i^{\max}\right] \\ (0,0), t \notin \left[T_i^{\min} + 1, T_i^{\max}\right] \end{cases} \quad (3.4)$$

Якщо відомі координати об'єкту *oi* у момент часу $t < t_1$, але не відомі в подальші моменти, то обчислення проводиться по такій формулі

$$(OX_i(t), OY_i(t)) = (OX_i(t_1), OY_i(t_1)) + Displacement(o_i, t_1 + 1, t). \quad (3.5)$$

Якщо відомі координати об'єкту *oi* у момент часу $t > t_2$, але не відомі в попередні моменти, то обчислення проводиться по такій формулі

$$(OX_i(t), OY_i(t)) = (OX_i(t_2), OY_i(t_2)) - Displacement(o_i, t + 1, t_2). \quad (3.6)$$

Якщо відомі координати об'єкту *oi* у моменти часу $t < t_1$ і $t < t_2$ то визначається за наступною формулою:

$$(OX_i(t), OY_i(t)) = \frac{t_2 - t}{t_2 - t_1}(OX_i(t_1), OY_i(t_1)) + Displacement(o_i, t_1 + 1, t) + \frac{t - t_1}{t_2 - t_1}((OX_i(t_2), OY_i(t_2)) \quad (3.7)$$

Якщо координати об'єкту не відомі в усі моменти часу, то вони визначаються для одного будь-якого моменту із прямокутника, що обмежує фрагменти даного об'єкту, а для решти моментів часу визначаються згідно з першим і другим пунктом. Перерахунок координат об'єкту відбувається при кожній зміні в списку його фрагментів на тому інтервалі часу, який зачіпається даною зміною.

Корекція помилок на рівні об'єктів. Корекція помилок на рівні об'єктів може містити ряд правил, що використовують форму, колір і траєкторію об'єктів, і дозволяють виправляти деякі помилки, допущені на попередніх етапах. Реалізовано 2 правила: якщо деякий об'єкт не вдалося відстежити, а поряд з місцем його останнього розташування був знайдений новий об'єкт, то ці два об'єкти об'єднуються в один. Це правило дозволяє у багатьох випадках продовжувати відстежувати об'єкти в умовах повного перекриття, якщо деякий об'єкт, що має вік не менше *del t*, в кожному кадрі знаходився на відстані не більше ніж *del r* від об'єкту, значущість якого перевищує значущість даного об'єкту в *del k* раз, то даний об'єкт видаляється. Це правило дозволяє видаляти помилкові об'єкти, викликані помилками при відстежуванні фрагментів (див. рис. 3.6).

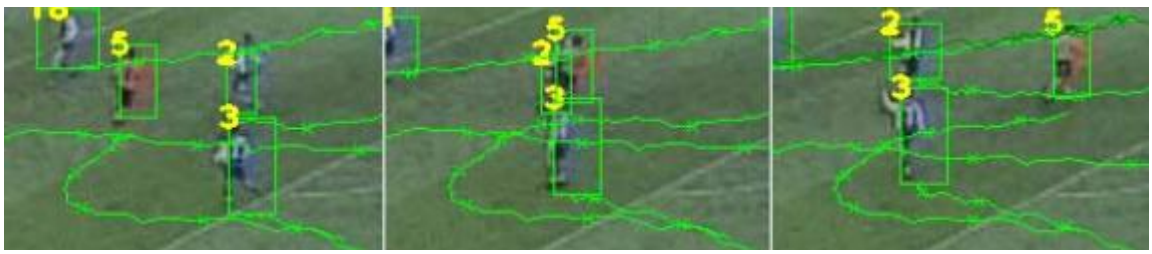

Рисунок 3.6 – Приклад відстежування об'єктів в умовах часткового перекриття

Експерименти показали, що розроблений алгоритм може відстежувати об'єкти в умовах часткового перекриття. На рисунку 3.6 приведені фрагменти трьох проаналізованих кадрів з послідовності зображень, що містить таки ситуації.

Середній час повного аналізу одного кадру відеозапису дозволом 670x367 пікселів, що містить зображення дванадцяти футболістів, на комп'ютері з процесором Intel i5 5400 із задіяним одним ядром, за умови, що в кожному другому кадрі виконується тільки відстежування фрагментів, дорівнює 31 мс. З них в середньому 18 мс витрачається на відстежування фрагментів і 10 мс витрачається на пошук і розпізнавання людей. Така швидкодія дозволяє обробляти відеозапис футбольного матчу у реальному часі.

3.4 Розширена версія адаптивної суміші Гаусової моделі фону

Метод генерування моделі фону та передньопланових об’єктів. Для виділення фону і рухомих об’єктів використовується розширена версія адаптивної суміші Гаусової моделі фону. Зміна значення інтенсивності кожного пікселя у кожному кадрі розглядається як піксельний процес:

$$\{X_1, .. X_t\} = \{I(x_0, y_0, i) : 1 \le i \le t\} \quad (3.8)$$

де I – послідовність кадрів відеопотоку,

t – поточний момент часу,

Xt – значення пікселя з координатами ($x0$, $y0$)

Для моделювання густини розподілу значення пікселя використовано суміш *K* розподілів:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(X_i, \mu_{i,t}, \Sigma_{i,t}),\ t = \overline{1,T}, \tag{3.9}$$

де *K* – кількість Гаусових розподілів,

$\omega i,t$ – ваговий коефіцієнт,

η – Гаусовий розподіл,

$\Sigma\, i,t$ – коваріаційна матриця у момент часу $t \in T$

Нове значення пікселя ставиться у відповідність одній із компонент суміші, для якої ймовірність отримання даного значення є найвищою і використовується для оновлення моделі. Ваговий коефіцієнт, що визначає швидкість оновлення переважно приймається як 0.01. Значення інтенсивності кожного пікселя кадру оцінюється на належність до рухомого об’єкта [21]. Якщо нерівність виконується то піксель кадру з інтенсивністю *It* сегментується як передньоплановий, інакше – як піксель фону.

Якщо значення пікселя збігається з одним із Гаусових розподілів, то оновлюються значення математичного очікування і дисперсії Гаусового розподілу:

$$\begin{aligned} &\mu_t = \rho I_t + (1-\rho)\mu_{t-1}; \\ &\sigma_t^{\ 2} = \rho d^2 + (1-\rho)\sigma^2_{\ t-1}; \\ &d = (I_t - \mu_t)^2, \end{aligned} \tag{3.10}$$

де *It* – значення яскравості пікселя, у момент часу *t*, що аналізується,

σ_2 – значення математичного очікування і дисперсії Гаусового розподілу у момент часу *t*,

ρ – ваговий коефіцієнт, що визначає швидкість оновлення (переважно приймається рівним 0,01),

dt – Евклідова відстань

На етапі сегментації окремі пікселі бінарної маски *Mt* які були класифіковані як передньопланові необхідно об’єднати в регіони за допомогою морфологічної операції умовного нарощування. Структурним елементом *S* при цьому є вікно розміром три пікселі. Позначимо через Mt^* - результуюче зображення бінарної маски. Умовним нарощуванням Mt^* зображення структурним елементом *S* за відношенням до *Mt* є операція:

$$M^*_t \oplus\Big|_{M_t} S = M^*_{t,m}, \tag{3.11}$$

де індекс *m* рівний найменшому значенню, при якому $M_{t,m}* = M_{t,m-1}^*$

Виділення сегментованих рухомих об’єктів виконане на основі обчислення зміни відношення площ кожного об’єкта на попередньому і поточному кадрах. Для обчислення площі рухомого об’єкта використано метод, що базується на ланцюгових кодах

$$A = \sum_{i=1}^{n} c_{ix}\left(y_{i-1} + \frac{1}{2}c_{iy}\right) \tag{3.12}$$

де *n* – довжина ланцюга;

c_{ix}, c_{iy} – *x* та *y*-компоненти *i*-го елемента ланцюга,

y_{i-1} – координата початкової точки елемента ланцюга c_i у заданій системі координат

Сегментований рухомий об’єкт вважається виділеним, якщо відношення площі рухомого об’єкта на поточному кадрі до площі на попередньому не перевищує заданого порогового значення. Для супроводу виділеного рухомого об’єкта приймаємо, що переміщення його центра мас та модуль його швидкості змінюються незначно, тому застосовуємо модель однорідного руху. Коефіцієнти

відхилення траєкторії руху центра мас виділеного об’єкта від оптимальної його траєкторії обчислюється за виразом:

$$C_{ij}^{\ k} = \frac{\left\| \left(x_i^k - x_{\alpha_i^k}^{k-1} \right) - \left(x_j^{k+1} - x_i^k \right) \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{k+1}} \left\| \left(x_p^k - x_{\alpha_p^k}^{k-1} \right) - \left(x_q^{k+1} - x_p^k \right) \right\|} + \frac{\left\| \left(x_j^{k+1} - x_i^k \right) \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{k+1}} \left\| \left(x_j^{k+1} - x_i^k \right) \right\|} \tag{3.13}$$

де x_i^k – координати проекції i-ї точки на k-му кадрі,

M, m – кількість рядків та кількість стовпців матриці відхилень,

α – коефіцієнт.

У реалізації запропонованого методу побудовано програмне рішення супроводу рухомих об’єктів, приклади функціонування якого наведено на рисунку 3.7

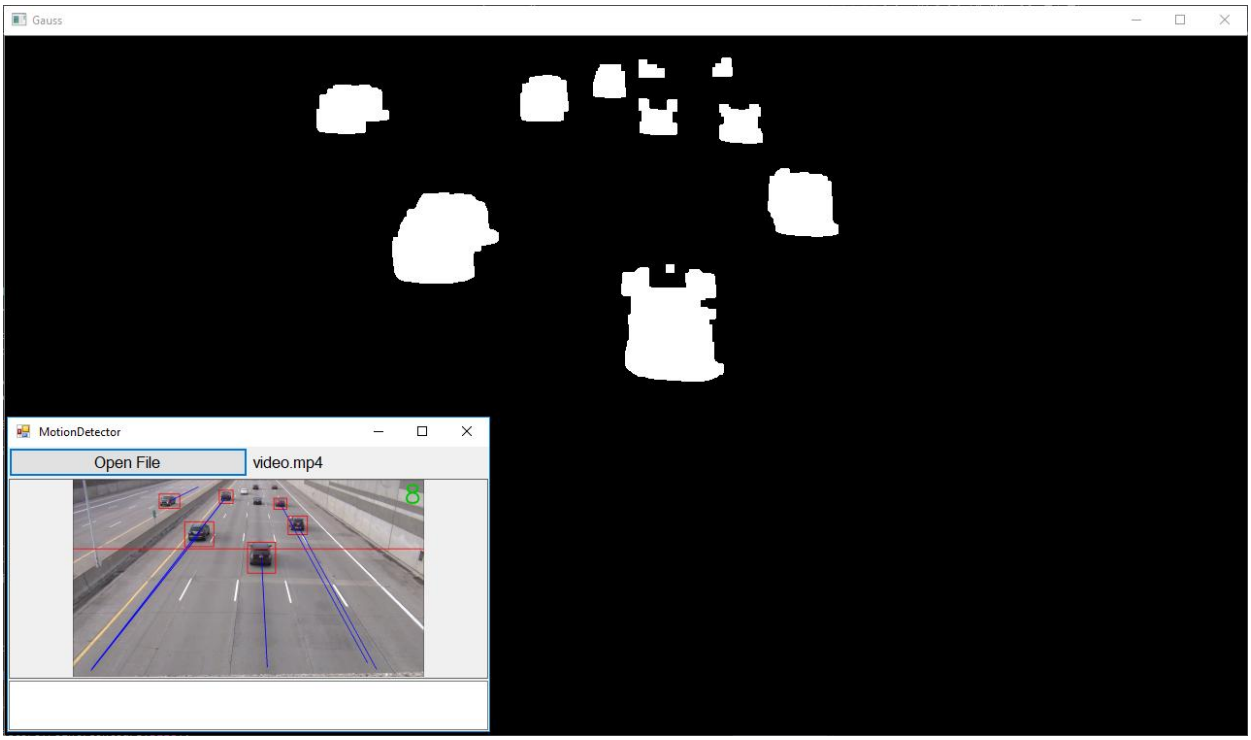

Рисунок 3.7 – Приклад обробки відеопотоку та побудова траєкторії руху

За результатами використання цього програмного рішення можна стверджувати що, динамічне визначення траєкторії є безпомилковим при типових параметрах освітленості.

Метод визначення траєкторії рухомих об’єктів дає можливість розробляти системи реального часу навіть без апаратної реалізації його складових частин. Для розробленого програмного рішення не проводилось жодних оптимізаційних процедур пришвидшення виконання коду, проте отримана система, без суттєвих затримок працює у режимі наближеному до реального часу на платформі Windows.

4 ОРГАНІЗАЦІЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ПРИ ВІДСТЕЖУВАННІ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

У попередньому розділі описано метод відстежування об'єктів у відеопотоці на основі відстежування переміщення фрагментів об'єктів, що поєднує властивості приведених вище методів. При виконанні на сучасному одноядерному процесорі метод не забезпечує швидкодію, достатню для аналізу у реальному часі відео високої роздільної здатності або відео з декількох камер. Для його прискорення доцільно використовувати паралельні обчислювальні системи. Основні типи паралельних обчислювальних засобів, що можуть бути застосовані для паралельного аналізу цифрових послідовностей графіки і відео:

– спеціально розроблені для вирішення конкретних задач обчислювальні структури. Наприклад, алгоритми пошуку фрагментів зображення можуть бути ефективно реалізовані на систолічних масивах, модифікації яких можуть бути використані для пошуку фрагментів. Спецпроцесор для пошуку фрагментів зображення для мобільних пристроїв;

– універсальні паралельні архітектури, орієнтовані на обробку векторів. В інтернеті існує безліч статей де описано успішне застосування архітектури CUDA [22], яка визначається як SIMT (single-instruction, multiple-thread) для відстежування облич;

– використовування паралельних обчислювальних систем з процесорами загального призначення. До них можна віднести комп'ютери з багатоядерним процесором, які фактично стали стандартними і мають паралельну архітектуру SMP (Symmetric Multiprocessing) UMA (Uniform Memory Access) і кластери, складені з них. Достоїнствами таких систем є їх доступність і можливість розпаралелювати виконання алгоритмів, які погано відображаються на векторно-орієнтовану паралельну архітектуру.

Однією з популярних технологій створення паралельних програм є використання бібліотеки MPI (Message Passing Interface), що являє собою

стандартизований набір засобів для обміну повідомленнями і може використовуватись як на SMP системах, так і на кластерах.

Можливий паралелізм в послідовному алгоритмі можна виявити як при незалежній обробці масивів однотипних даних, так і для різнорідних етапів алгоритму, які можуть бути виконані одночасно. Для визначення можливості паралельного виконання етапів складного алгоритму використовується побудова графа інформаційних залежностей алгоритму. У даній роботі аналізуються підходи до послідовного відстежування об'єктів у відео і будується паралельний алгоритм по методу відстежування об’єктів у відеопотоці на основі переміщення фрагментів об’єкті. При реалізації використовуються технологія MPI. Отримуються оцінки ефективності, а також оптимальне балансування навантаження на різних системах.

У методі відстежування об’єктів у відеопотоці на основі переміщення фрагментів кожен рухомий об'єкт розглядається як сукупність рухомих фрагментів (див. рис. 4.1).

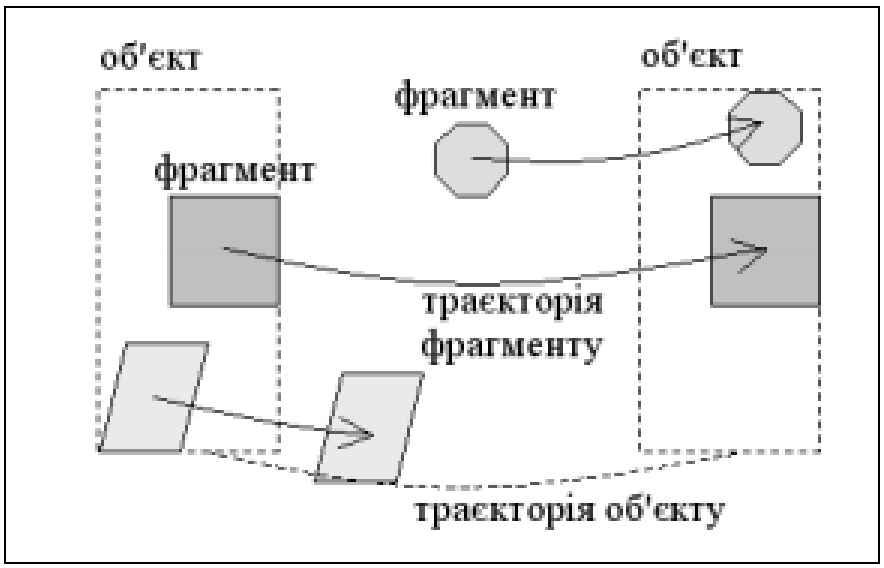

Рисунок 4.1 – Відстежування об'єкту як множини фрагментів

Кожен фрагмент об'єкту є областю кадру довільного розміру і форми, що містить частину об'єкту або цілий об'єкт, і задається своїм зображенням і матрицею, що визначає приналежність пікселів до фрагмента. Такий підхід

дозволяє відстежувати частково перекриті об'єкти. В алгоритмі використовуються модель статичного фону, множина об'єктів і множина їх фрагментів.

На початку роботи алгоритму множини фрагментів і об'єктів порожні. В процесі відстежування по черзі аналізуються кадри відео, що приходять.

Віднімання фону (C1) використовується для визначення пікселів кадру, які відносяться до об'єктів.

Розпізнаванню об'єктів (C3) на багатошаровому персептроні [9] (наприклад, силуету людини) підлягають області кадру, що містять групи близько розташованих пікселів, відмінних від фону (C2).

Для кожного фрагмента, положення якого відоме в попередніх кадрах, на основі моделі його руху визначається область (C4), в якій він може знаходитися в поточному кадрі, і в ній шукається його положення (C5).

Нові відстежуванні фрагменти (C6) створюються один раз в декілька кадрів, щоб покрити пікселі, які не відносяться ні до фону, ні до фрагментів.

Встановлення відповідності між фрагментами і об'єктами та побудова траєкторій об’єктів (C7) здійснюється на основі траєкторій руху їх фрагментів. У об'єкти об'єднуються групи близько розташованих фрагментів, які рухаються узгоджено. Обробляються ситуації, пов'язані з виникненням, зникненням, об'єднанням і розділенням об'єктів.

Розпаралелювання можна виконати наступними способами:

– при аналізі даних з декількох відеокамер доцільно виконувати аналіз і операції введення-виведення для кожної камери на окремому процесорі;

– при аналізі відеозапису можливо розбити його за часом на декілька відрізків і проаналізувати їх паралельно, після чого визначити відповідності між об'єктами на межах сусідніх відрізків;

– кожен кадр відеопотоку можна розділити за площею на декілька частин і створити з цих частин декілька окремих відеопотоків меншої роздільної здатності, кожен з яких аналізується на окремому процесорі;

– можливо провести розпаралелювання всередині алгоритму аналізу одного кадру.

UML діаграма діяльності для послідовного алгоритму аналізу одного кадру приведена на рисунку 4.2.

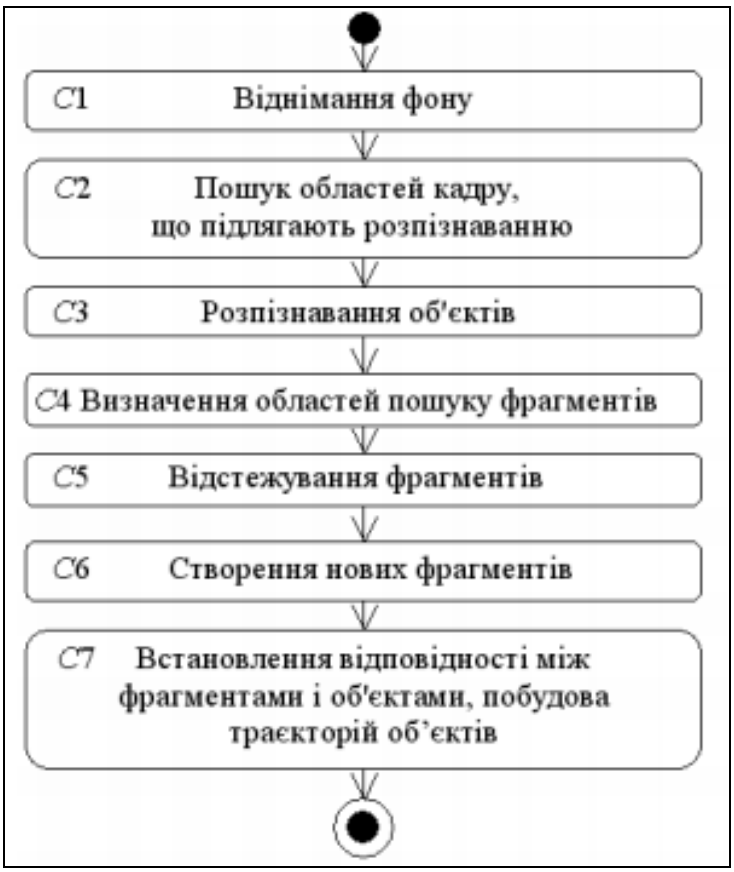

Рисунок 4.2 – Послідовний алгоритм аналізу кадру

Перші три способи розпаралелювання не потребують модифікації послідовного алгоритму відстежування, процеси незалежно виконуються на архітектурі типа MIMD і не обмінюються повідомленнями під час аналізу. Це дозволяє досягти прискорення, близького до числа використовуваних процесорів. Якщо необхідно аналізувати у реальному часі відео високої роздільної здатності, для роботи з яким не вистачає продуктивності одного процесора, то необхідно використовувати третій або четвертий спосіб розпаралелювання.

Третій спосіб має істотні недоліки: щоб не втратити об'єкти на межах областей, на які поділено кадр, ці області повинні перетинатись. Також потрібно визначення відповідності між об'єктами на межах областей. Це збільшує загальне необхідне число операцій і підвищує ймовірність помилок.

Четвертий спосіб є найбільш складним в реалізації, але потенційно дозволяє одержати максимально можливу швидкодію при аналізі відеопотоку у реальному часі. Він розглядається нижче детальніше.

Доцільно виконувати паралельно ті етапи алгоритму на рисунку 4.2, які займають багато часу. Експериментально отриманий середній час виконання основних груп операцій при аналізі кадру футбольного матчу розміром 672x368 на одному ядрі процесора Intel i5 5400 приведено в таблиці 4.1.

Таблиця 4.1 – Середній час виконання операцій на одному ядрі процесору

етап	C1	C2	C3	C4	C5	C6	C7
час, мс	2.36	0.41	1.84	2.24	16.1	0.92	2.81

Операціями розробленого алгоритму для паралельного виконання є:

– відстежування фрагментів об'єктів (C5) – найбільш ресурсоємний етап алгоритму. Пошук кожного фрагмента може бути виконаний незалежно. Можливо виконувати пошук однакової кількості фрагментів на кожному процесорі. Для кращого балансування можна оцінити число операцій, що потрібне для пошуку кожного фрагменту і розподілити фрагменти по процесорах так, щоб сумарне очікуване число операцій на кожному з них відрізнялося мало;

– віднімання фону (C5) може бути розпаралелено шляхом розбиття кадру на області. Кожен процесор зберігає модель фону і віднімає фон в одній такій області. При балансуванні навантаження можна вважати, що час віднімання фону пропорційний площі зображення;

– розпізнавання об'єктів (C3). При використовуванні складного алгоритму розпізнавання час виконання цього етапу може зрости. Розпізнавання кожного об'єкту може бути виконане незалежно; всі області, що підлягають розпізнаванню в кожному кадрі, можна розподілити між різними процесорами.

Операції, які виконуються послідовно:

– пошук областей кадру, що підлягають розпізнаванню (C2) і створення нових фрагментів (C6) займають відносно малу частку процесорного часу;

– визначення областей пошуку фрагментів (С4) потрібно виконати до розподілу фрагментів по процесорах для виконання балансування навантаження етапу С5;

– багато кроків алгоритму встановлення відповідності між фрагментами і об'єктами та побудови траєкторій об’єктів (С7) потребують наявності у кожного процесора повної множини об'єктів і їх фрагментів, одержаної на попередньому кроці, що при розпаралелюванні викликає необхідність багатократних обмінів даними та синхронізацій.

Назвемо один з процесорів основним. На ньому виконуватимуться операції введення і виведення, етапи С2, С4, С6 і С7 алгоритму на рисунку 4.2, а також управляючий код, що зв'язує послідовні і паралельні етапи алгоритму.

Основний процесор розподіляє завдання між рештою процесорів і отримує від них результати. Решта процесорів не обмінюється повідомленнями між собою.

Для побудови паралельного алгоритму замінимо у алгоритмі на рисунку 4.2 кожний з етапів С1, С3 і С5 сукупністю трьох операцій:

– розподіл завдань по процесорах;

– паралельне обчислення на всіх процесорах, включаючи основний;

– об'єднання результатів на основному процесорі.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ АНАЛІЗУ КАДРУ

У програмній реалізації окремо описані потоки команд для основного і додаткових процесорів. UML діаграма діяльності для отриманого паралельного алгоритму аналізу кадру приведена на рисунку 5.1. Дії етапів послідовного алгоритму С1, С3 і С5, що виконуються на основному процесорі, позначені як М1, М3 і М5, а на інших процесорах – О1, О3 і О5 відповідно. Не показані допоміжні операції по балансуванню даних між процесорами, які виконуються безпосередньо перед пересилання даних.

Перед початком аналізу кадру в основному процесі є множини об'єктів і їх фрагментів, отримані на попередньому кроці, а в кожному процесі – модель фону у відповідній частині кадру.

Для мінімізації простою процесорів через очікування результатів від інших процесорів, бажано асинхронно відправляти результати, необхідні іншим процесорам, як тільки вони будуть обчислені, і відкладати синхронне отримання даних від інших процесорів до тих пір, поки вони не знадобляться на даному процесорі.

Граф інформаційних залежностей алгоритму приведений на рисунку 5.1. Прямокутниками показані операції, стрілками – залежності між ними. Як видно з рисунку, етапи С2 і С3 можна виконувати паралельно з етапами С4, С5 і С6, а етап С4 – паралельно з етапом С1.

Отримаємо оцінку часу роботи алгоритму на рисунку 5.1 на різних паралельних системах. Для спрощення приймемо, що навантаження на усіх процесорах, окрім основного, для всіх етапів алгоритму, що виконуються паралельно, збалансоване. При паралельному виконанні найбільш навантаженим є основний процесор і загальний час паралельного аналізу кадру практично дорівнює часу його аналізу на основному процесорі.

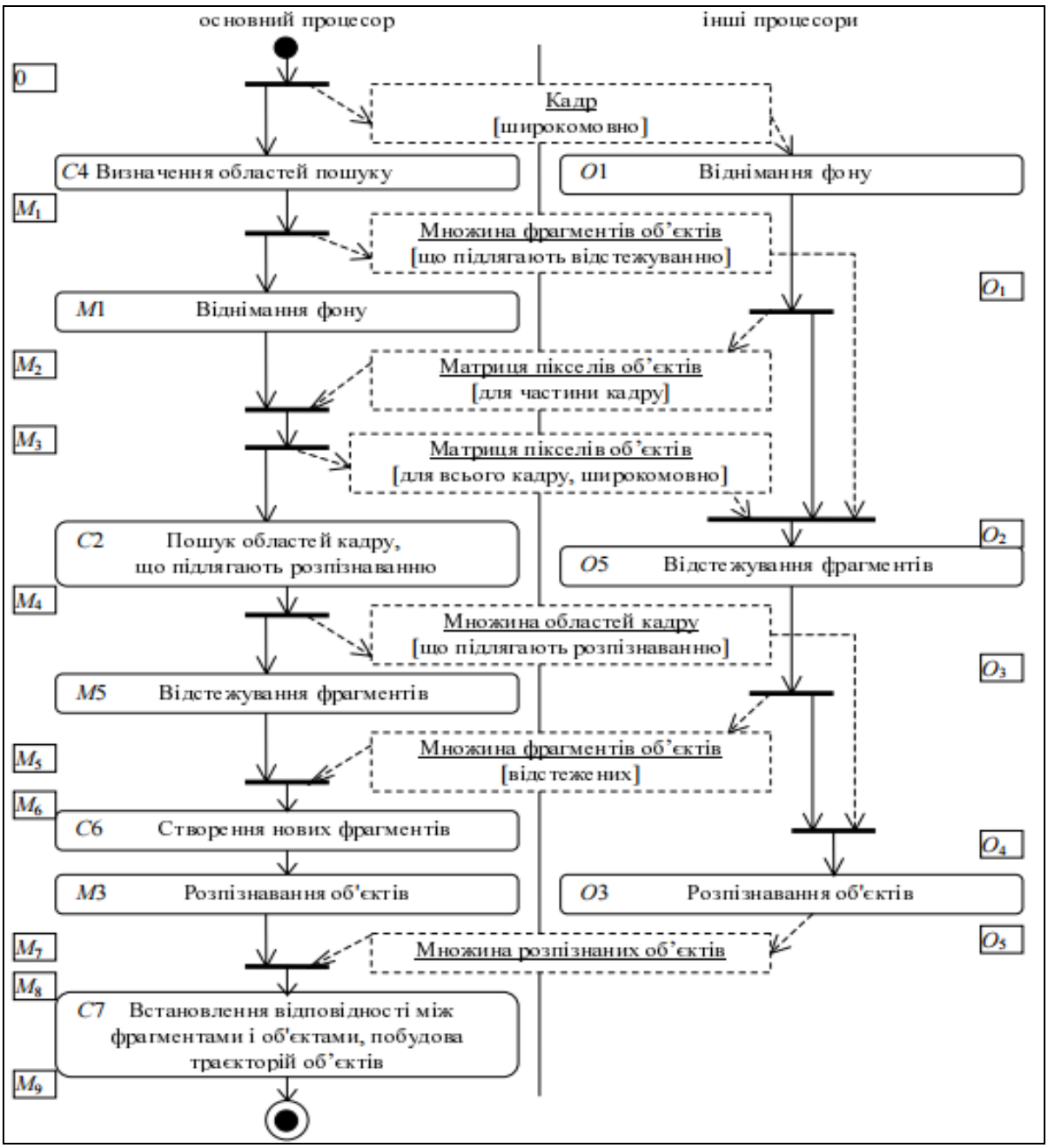

Рисунок 5.1 – Паралельний алгоритм аналізу зображення

Нехай використовується N процесорів, причому на кожному комп'ютері, що входить в кластер, розташовані N_l процесорів. Позначимо час виконання етапу X алгоритму через T_X. Час виконання паралельних етапів алгоритму на основному та інших процесорах дорівнює наступним формулам

$$T_{Mi} = T_{Ci}(1 - A_i),$$
$$T_{Oi} = T_{Ci} A_i / (N - 1) \qquad (5.1)$$

де $i \in \{1,3,5\}$ – номер етапу,

A_1, A_3 и A_5 – доля обчислень етапів С1, С3 і С5, які виконуються на усіх процесорах, окрім основного.

Слід вважати, що упакування або розпакування S байтів даних, що передаються, займає K_lS одиниць процесорного часу. Для спрощення приймемо, що час передачі S байт по мережі (не включаючи упакування і розпаковування) дорівнює KnS, а час передачі між процесорами одного обчислювального вузла дорівнює нулю.

Розглянуто асинхронну індивідуальну розсилку S/(N-1) байт від основного процесора до кожного з інших. Приклад виконання такої розсилки при N=4 і N_l=2 показаний на рисунку 5.2 у першій частині з позначенням літерою а.

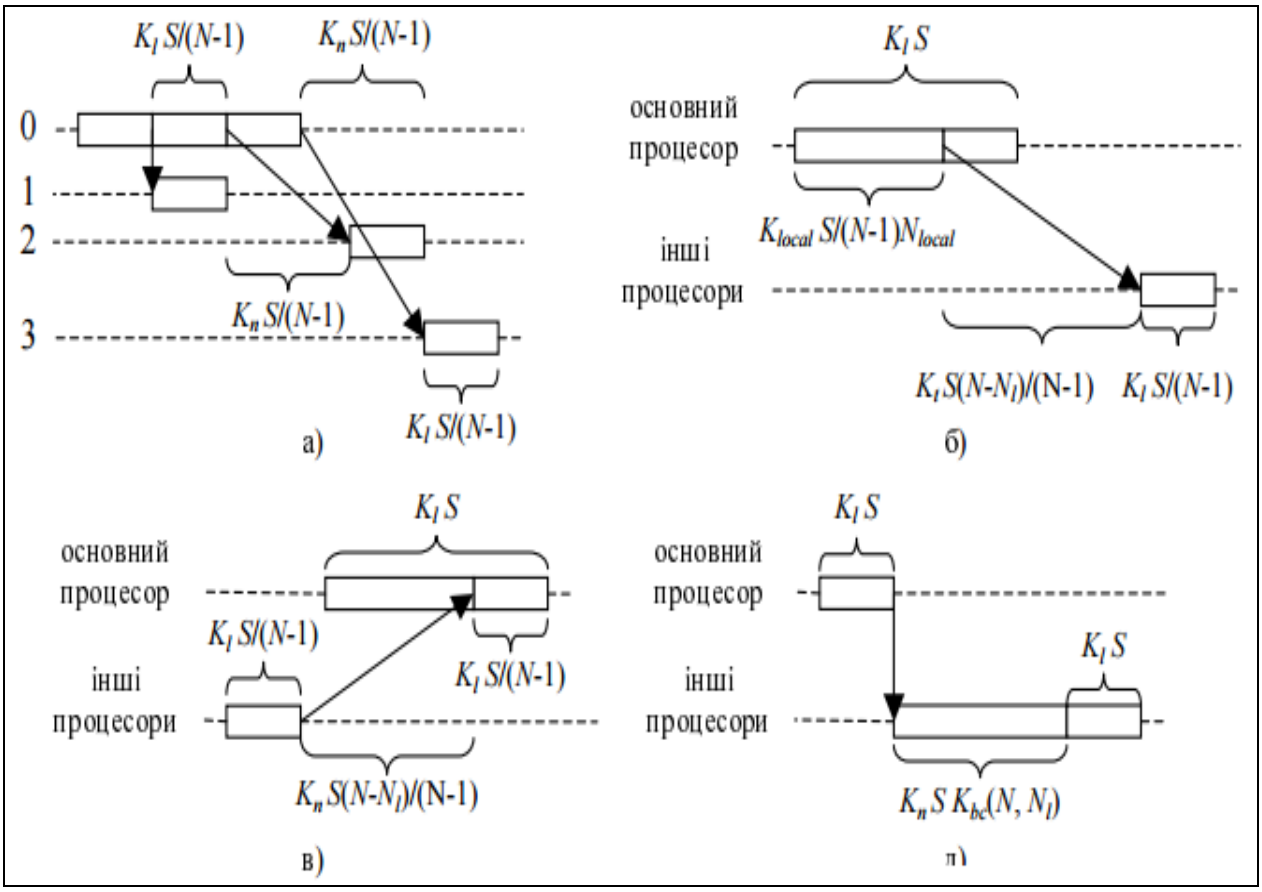

Рисунок 5.2 – Затримки при обміні даними

Пунктирними лініями позначені осі часу для кожного з процесорів, цифрами – ранги процесорів (0-основний), стрілками – передача даних, прямокутниками – процесорний час. При використанні мережі Ethernet (N-Nl)

асинхронних надсилань конкурують за використання каналу і їх сумарний час передачі дорівнює

$$K_n S(N\text{-}N_l)/(N\text{-}1) \tag{5.2}$$

При $K_{local} < K_{net}$ можна вважати, що пересилання всіх даних по мережі буде завершено через

$$K_l S(N - N_l)/(N - 1) \tag{5.3}$$

Оскільки пересилання є асинхронним, моменти читання і розпакування даних можуть бути перенесені в майбутнє на будь-який час. Тоді для визначення затримок можна використовувати спрощену схему, представлену на рисунку 5.2 під літерою б. Аналогічно отримано спрощені схеми для визначення затримок при асинхронному зборі даних основним процесором (див. рис. 5.2в) і широкомовної розсилки (див. рис. 5.2д).

Нехай загальний обсяг даних, необхідних для передачі всього кадру, дорівнює S_{frame}, матриці пікселів об'єктів всього кадру – S_{back}, всіх фрагментів – S_{frag} і всіх областей, що підлягають розпізнаванню – S_{rec}. На рисунку 5.1 показані відмітки часу від початку аналізу кадру для основного процесора M1…M9 і для додаткових процесорів O1…O5. Виразимо кожну з відміток часу через попередні відмітки часу, а також тривалість обчислень і передачі даних, необхідних щоб перейти до даної відмітки:

$$
\begin{aligned}
&O_1 = 0 + (K_l + K_n K_{bc} + K_l)\, S_{frame} + T_{C1}\, A_1/(N\text{-}1);\\
&M_1 = 0 + K_l S_{frame} + T_{C4};\\
&M_2 = M_1 + K_l S_{frag}\, A_5 + T_{C1}\,(1\text{-}A_1);\\
&M_3 = \max(M_2 + K_l\, S_{back}\, A_1\,(N\text{-}2)/(N\text{-}1),\ O_1 +\\
&\qquad K_l\, S_{back}\, A_1/(N\text{-}1) + K_n\, S_{back}\, A_1\,(N\text{-}N_l)/(N\text{-}1))\\
&\qquad + K_l\, S_{back}\, A_1/(N\text{-}1);\\
&O_2 = \max(M_3 + K_l\, S_{back} + K_{bc}\, K_n\, S_{back},\ O_1\\
&\qquad + K_l\, S_{back}\, A_1/(N\text{-}1)) + K_l\, S_{back};\\
&M_4 = M_3 + K_l\, S_{back} + T_{C2};\\
&M_5 = M_4 + K_l\, S_{rec}\, A_3 + T_{C5}\,(1\text{-}A_5);\\
&M_6 = \max(M_5 + K_l\, S_{frag}\,(N\text{-}2)/(N\text{-}1),\ O_3 +\\
&\qquad K_l\, S_{frag}\, A_5/(N\text{-}1) + K_n\, S_{frag}\, A_5\,(N\text{-}N_l)/(N\text{-}1))\\
&\qquad + K_l\, S_{frag}\, A_5\,/(N\text{-}1);\\
&O_4 = \max(M_4 + K_l\, S_{rec}\, A_3\, N_l/(N\text{-}1) +\\
&\qquad K_n\, S_{rec}\, A_3\,(N\text{-}N_l)/(N\text{-}1),\ O_3\\
&\qquad + K_l\, S_{frag}\, A_5\,/(N\text{-}1)) + K_l\, S_{rec}\, A_3\,(N\text{-}1);\\
&M_7 = M_6 + T_{C6} + T_{C3}\,(1\text{-}A_3);\\
&O_5 = O_4 + T_{C3}\, A_3/(N\text{-}1);\\
&M_8 = \max(M_7 + K_l S_{rec}\,(N\text{-}2)/(N\text{-}1),\ O_5\\
&\qquad + K_l\, S_{rec}\, A_3/(N\text{-}1) + K_n\, S_{rec}\, A_3\,(N\text{-}N_l)/(N\text{-}1))\\
&\qquad + K_l\, S_{rec}\, A_3\,/(N\text{-}1);\\
&M_9 = M_8 + T_{C7}.
\end{aligned}
\tag{5.4}
$$

Змінні $T_{C1}\ldots T_{C7}$, S_{back}, S_{frag} і S_{rec} залежать від параметрів алгоритму відстежування і його вхідних даних, а K_l, K_n, N_l і N – від конфігурації обчислювальної системи. При незмінних параметрах алгоритму відстежування, вхідних даних і конфігурації обчислювальної системи, можна розглядати M_9 як функцію від A_1, A_3 и A_5.

Нижче наведений лістинг коду для розпаралеленого трекінгу об’єктів у відеопотоці (див. рис. 5.3).

```
1 reference | 0 changes | 0 authors, 0 changes
private void BlobTracing(List<Blob> existingBlobs, IEnumerable<Blob> currentFrameBlobs)
{
    Parallel.ForEach(existingBlobs, blob =>
    {
        blob.IsCurrentMatchFoundOrNewBlob = false;
        blob.PredictNextPosition();
    });

    Parallel.ForEach(currentFrameBlobs, blob =>
    {
        int indexOfLeastDistance = 0;
        double leastDistance = 1000000;

        for (int i = 0; i < existingBlobs.Count; i++)
        {
            if (!existingBlobs[i].IsStillTracked)
            {
                continue;
            }

            double distance = DistanceBetweenPoints(blob.CenterPositions.Last(),
                existingBlobs[i].PredictedNextPosition);
            if (distance >= leastDistance)
            {
                continue;
            }
            leastDistance = distance;
            indexOfLeastDistance = i;
        }

        if (leastDistance < blob.CurrentDiagonalSize * 0.5)
        {
            AddBlobToExistingBlobs(blob, ref existingBlobs, ref indexOfLeastDistance);
        }
        else
        {
            AddNewBlob(blob, ref existingBlobs);
        }
    });
}
```

Рисунок 5.3 – Паралельне обчислювання траєкторії руху об’єкту

Експериментально отримані кількісні величини при аналізі відеозапису руху автомобільного потоку роздільної здатності 1280x720 на імітаційному кластері, що включає комп'ютери з процесорами Intel i5 6260 і мережу 100 Мбіт/с, наведено на рисунку 5.4.

Оцінка часу (5.4) при оптимальному навантажені і реальне значення середнього часу аналізу кадру при різному числі процесорів на імітаційному кластері показані на рисунку 5.4. Час виконання на одному процесорі наведено для послідовної програми, що не використовує MPI.

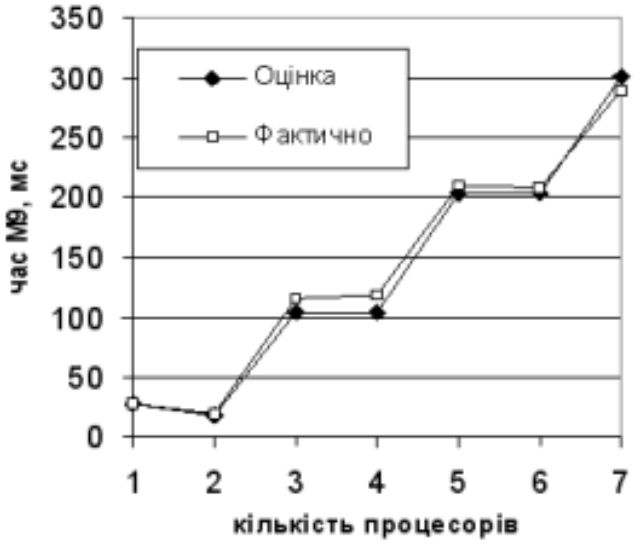

Рисунок 5.4 – Розрахунковий середній час аналізу кадру на імітаційному кластері

Стрибкоподібне збільшення часу роботи відбувається через розсилання даних по біноміальному дереву у функції MPI_Bcast із бібліотеки MPI. Як видно з рисунку 5.4, виконання алгоритму на рисунку 5.1 при аналізі відео руху автомобільного потоку при використовуванні мережі Ethernet 100 Мбіт/с призводить не до прискорення, а до уповільнення аналізу.

Більшого прискорення можна досягти:

– при використанні мережі з більшою пропускною спроможністю;

– при використанні структур даних, що не потребують упаковки при відправці. Така зміна вимагає істотної переробки наявної послідовної програми;

– при реалізації алгоритму на системі UMA без використання MPI (наприклад, стандартними засобами для роботи з потоками ОС Windows). Відправку даних можна звести до передачі їх адреси в пам'яті без упаковки даних в безперервний буфер. В цьому випадку можна вважати $K_l = 0$.

Оцінку часу (5.4) при оптимальному балансуванні навантаження при використанні мережі Ethernet 1 Gbit/s і 10 Gbit/s, а також при використанні системи UMA з $K_l = 0$, показано на рисунку 5.4.

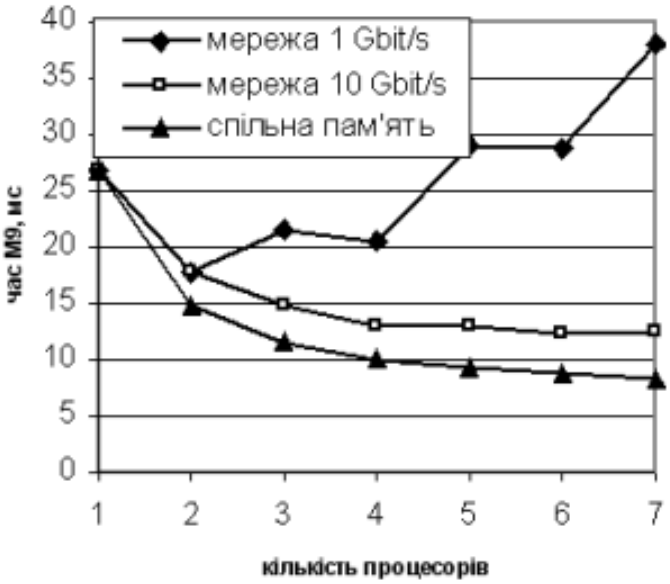

Рисунок 5.5 – Розрахункова оцінка часу аналізу кадру на різних паралельних системах

Як видно з рисунку 5.5, ще більше прискорення при менших апаратних витратах можна отримати при реалізації алгоритму на паралельній системі UMA з $K_l = 0$ (наприклад, як багато потокової програми, яка виконується на багатоядерному процесорі). Таким чином середній час аналізу кадру в два потоки на одному комп'ютері на імітованому кластері склав 17.2 мс.

ВИСНОВКИ

У роботі були досліджені алгоритми визначення параметрів руху у відеопотоці. Виявленні сильні та слабкі сторони кожного з досліджених методів. Була проведенна класифікація методів пошуку руху у відео послідовності. Спроєктована система пошуку та трекінгу руху об'єктів на відео.

Проаналізовано алгоритм Шмідта задля визначення параметрів об’єктів в зображенні. Метод Шмідта й сингулярне розкладання, використані разом з методом Хаффмана, дають для зображень результат стиску, порівнянний з, тими, що представляються стандартною технологією JPEG.

Проаналізовано методику згладжування зображень, заснована на спеціальних рішеннях рівняння Пуассона.

Система має змогу обробляти як відеозаписи так і відеопоток у реальному часі отриманного з камер відеонагляду. Для реалізації був обраний алгоритм відстежування об’єктів у відеопотоці на основі переміщення фрагментів. Цей метод показав непогані результати роботи але визначення руху у реальному часі інколи призводило до затримання відеокадру. З збільшенням якості та розмірів відео його час роботи теж збільшувався на пропорціональну величину, що заважало відстежувати рух об'єктів у реальному часі отримане з камер високої якості.

Для вирішення цієї проблеми було прийнято рішення реалізувати цей алгоритм з урахуванням архітектури паралельних систем. Було виявленно 4 способа розпаралелювання данного алгоритму.

Було виявленно 3 найбільш затратні операції котрі можливо зробити паралельно: відстежування фрагментів об'єктів, віднімання фону, розпізнавання об'єктів.

Найбільш значне прискорення алгоритму довелось отримати за допомогою його реалізації на стандартних засобах роботи з потоками у Windows на одному

багатоядерному процесорі, саме через те, що не витрачається зайвого часу для пересилання обробленої інформації між процесорами у кластері.

Отримані результати мають як теоретичну, так і практичну значимість. Результати можуть бути використані як безпосередньо для оптимізації існуючих методів стискання й ідентифікації цифрових графічних зображень та відео послідовностей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Журавель И.М. Краткий курс теории обработки зображений (MatLab - Image Processing Toolbox) / Журавель И.М. – СПб: Питер, 2015. − 120с.

2. Дэвид А. Форсайт "Компьютерное зрение. Современный подход" / Дэвид А. Форсайт, Беэр Бибо: пер. с англ. - М.: ООО И.Д. Вильямс , 2014. – 416 с.

3. Виды моделирования. Компьютерное моделирование [портал bourabai.kz: Виды моделирования. Компьютерное моделирование – Режим доступу: www/URL: http://bourabai.kz/cm/bahvalov2.htm .

4. Сандерсон С. Обработка изображений и анализ данных. / С. Сандерсон: пер. с англ. - М.: ООО И.Д. Вильямс , 2018. – 560 с.

5. Вапник В.Н. Теория распознавания образов (статистические проблемы обучения) / Вапник В.Н. – СПб: Питер, 2012. − 437с.

6. Коренной А.В. Обнаружение, распознавание и определение параметров образов объектов. Методы и алгоритмы / Коренной А.В. – Либроком: Москва, 2012. – 120с.

7. Емельянов Н.И. Обработка изображений и анализ данных / Емельянов Н.И. – Либроком: Москва, 2014 – 420с.

8. Гончаров, В. А. Детекція руху в реальному часі / В. А. Гончаров, В. М. Кравцов // Журн. вычисл. математики и мат. физики. - 1988. - Т. 28, №12. - С. 1858-1866.

9. Основные направления компьютерного зрения/ С.-Петерб. гос. ун-т, фак. прикладной математики - процессов управления. - Режим доступа : www/ URL: http://apcp.apmath.spbu.ru/ru/staff/tuzov/onakz.html/ .

10. Голицынский Н.И. Основы OpenCV / Голицынский Н.И. – Либроком: Москва, 2010 – 327с.

11. Шапиро Л.А. OpenCV – простейшие алгоритмы распознавания / Шапиро Л.А. – СПб: Питер, 2013. − 274с.

12. Методы поиска движения на видеокадре/ С.-Петерб. гос. ун-т, фак. прикладной математики - процессов управления. - Режим доступа : www/ URL: http://apcp.apmath.spbu.ru/ru/staff/kravc/mpdviz.html/

13. Хофман С. Перемещение объектов на видео / С. Хофман: пер. с англ. - М.: ООО И.Д. Вильямс , 2015. – 226 с.

14. Кравцов Л.И. Параметры объекта в видеокадре / Кравцов Л.И. – Либроком: Москва, 2007 – 327с.

15. Ланец А.П. Прикладные задачи компьютерного зрения / Ланец А.П. – Либроком: Москва, 2011 – 358с.

16. Кормэн А. Томас Основные алгоритмы движения объектов / Кормэн А. Томас, Беэр Бибо: пер. с англ. - М.: ООО И.Д. Вильямс , 2005. – 628 с.

17. Середа К.И. Кодирование видеопоследовательностей / Середа К.И. – Либроком: Москва, 2014 – 529с.

18. Лукавенко Б.С. Основы компьютерной графики / Лукавенко Б.С. – ООО И.Д. Вильямс , 2009. – 377 с.

19. Методы трекинга в видеопотоке.− Виды моделирования. Компьютерное моделирование URL: http://rusnauki.com/cm/comvisionmodel.htm.

20. Распознавание объектов методом Хаара. Распознавание объектов методом Хаара: URL: http://rusnauki.com/cm/haarmethoddetection.htm.

21. Сикора В.А. Движение в системах компьютерного зрения / Сикора В.А. – СПб: Питер, 2013. − 349с.

22. Бойко К.Л. Современные графические процессоры с архитектурой CUDA / Бойко К.Л. – СПб: Питер, 2016. − 576с.

23. Будников Е.Ю., Кукоев И.Ф., Максимов А.В. Вейвлет- и Фурье-анализ электрических флуктуаций в полупроводниковых и электрохимических системах // Измерительная техника. – 1999. – № 11. – С. 40-44.

24. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. – М.: Наука, 1979. – 448 с.

25. Витязев В.В. Вейвлет-анализ временных рядов: Учеб. Пособие. – СПб.: Изд-во С.-Петерб. ун-та, 2011. – 258 с.

26. Григорьев Ал.А., Кондратьев К.Я. Спутниковый мониторинг природных и антропогенных катастроф // Исследование Земли из космоса. – 1996. – № 3. – С. 68-79.

27. Кадена Л., Кириллова С.В., Симонов К.В. Анализ сложных сигналов на основе вейвлет-преобразования // Материалы Всероссийской конференции: СФУ, 2011. – С. 59-63.

28. Kutyniok G., Lim W.-Q. Image Separation using Wavelets and Shearlets, preprint, 2016.

29. Guo K., Labate D., W.-Q Lim, Edge analysis and identification using the Continuous Shearlet Transform // Appl. Comput. Harmon. Anal. 27 (2014), 24- 46.

30. Дремин И.М., Иванов О.В., Нечитайло В.А. Вейвлеты и их использование // Успехи физических наук, 2001. – Т. 171. – № 5. – С. 465-501.

31. Кадена Л. Application of wavelet analysis to the electrocardiogram // Тезисы докладов конференции по математическому моделированию и информационным технологиям – Новосибирск: ИВТ СО РАН, 2011. – С. 45

32. Дьяконов В., Абраменкова И. MATLAB. Обработка сигналов и изображений. Специальный справочник. – СПб.: Питер, 2012.

33. Дудар З.В., Склярук Д.О. Дослідження алгоритмів підвищення надійності відеозв'язку / Информационные системы и технологии (ИСТ-2018): материалы 7-й Международ. Науч.-техн. конф., Харьков-Коблево, 10-15 сентября 2018, – тезисы докладов. – Х.: ХНУРЕ, 2018

34. Shubin, I., Snisar, S., Zhyrnov, V., Slavhorodskyi, V.// Practical Application of Formal Representation of Information for Intelligent Radar Systems 2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings, 2019, pp. 433-436, 8632103