МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних систем та мереж

_____ Жуков І.А.
(підпис) (ПІБ)

"_____" _____ 2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ "МАГІСТР"
напрям підготовки - 6.050102 "Комп'ютерна інженерія"

Тема: Обробка результатів дискретно-косинусного перетворення інформації

Виконавець: _____ Сипко Р.В.
(підпис) (ПІБ)

Керівник: _____ Лукашенко В.В.
(підпис) (ПІБ)

Нормоконтролер: _____ Андрєєв В.І.
(підпис) (ПІБ)

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Напрям 6.050102 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем та мереж

_____ Жуков І.А.
(підпис) (ПІБ)

« ____ » _____2020 р.

ЗАВДАННЯ

на виконання дипломного проекту

Сипко Романа Вікторовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): "Обробка результатів дискретно-косинусного перетворення інформації"
затверджена наказом ректора від "25" вересня 2020 року № 1793/ст.

2.Термін виконання проекту (роботи): з 05.10.2020 по 31.12.2020

3. Вихідні дані до проекту (роботи): визначення границь ефективного застосування ДКП для зберігання та передачі зображення
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) Системний аналіз представницького рівня еталонної моделі взаємодії відкритих систем;
2) Вибір і обґрунтування інструментарію для визначення ефективних способів передачі інформації;
3) Дослідження способу зберігання та передачі зображення з метою визначення границь його ефективного застосування.

5. Перелік обов'язкового графічного матеріалу:
Презентація *PowerPoint*. 1. Базова модель об'єкту дослідження. 2. Параметри вибраного інструментарію дослідження. 3. Результати дослідження.

6. Календарний план-графік

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Ознайомитись з постановкою задачі дипломного проектування	05.10 – 07.10	
2	Вивчити спеціальну літературу і технічну документацію	08.10 – 15.10	
3	Проаналізувати технології передачі даних	16.10 – 19.10	
4	Написати розділ дипломного проекту щодо еталонної моделі взаємодії відкритих систем	20.10 – 23.10	
5	Проаналізувати інструментарій для дослідження	24.10 – 31.10	
6	Написати розділ дипломного проекту з описом інструментарію для визначення ефективних способів передачі інформації	01.11 – 04.11	
7	Провести дослідження	05.11 – 15.11	
8	Написати розділ дипломного проекту з описом результатів проведеного дослідження	16.11 – 21.11	
9	Підготувати графічний демонстраційний матеріал	22.11 – 28.11	

Дата отримання завдання_____

Керівник дипломного проекту_____Лукашенко В.В.
(підпис)

Завдання прийняв до виконання_____Сипко Р.В.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту “Обробка результатів дискретно — косинусного перетворення інформації”: 88 сторінок, 67 рисунків, 21 літературних джерел.

ДКП, ЗБЕРІГАННЯ ІНФОРМАЦІЇ, ПЕРЕДАЧА ІНФОРМАЦІЇ, ЗІСТАВЛЕННЯ ЗОБРАЖЕНЬ.

Мета дипломного проекту – визначити границі ефективного застосування дискретно - косинусного перетворення для зберігання та передачі інформації.

Завдання дипломного проектування – дослідити існуючі границі застосування ДКП для зберігання та передачі інформації за допомогою створеної програми для роботи з графічним зображенням із використанням ДКП та алгоритмами зіставлення (схожості) зображень.

Об’єкт проектування – використання дискретно - косинусного перетворення для зберігання, передачі інформації та визначення схожості зображень.

Практична значимість полягає у зберіганні та передачі інформації з використанням дискретно - косинусного перетворення, а також алгоритмів для визначення схожості зображень.

Основні конструктивні показники – дослідження графічних зображень з використанням ДКП, графіки та алгоритмів визначення їх схожості.

Отримані результати та їх новизна – встановлено, що використання способу (дискретно - косинусне перетворення) для зберігання та передачі інформації, дозволяє зменшити об’єм інформації для зберігання та зменшити час її передачі.

Результати дипломного проектування рекомендується використовувати для збереження та передачі інформації з бортового комп'ютера, а також для визначення схожості відновлених зображень.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

OSI – *Open Systems Interconnection*

NFS – *Network File System*

DCT – *Discrete Cosine Transform*

PICT – формат даних для комп'ютерів *Macintosh*

TIFF – *Tagged Image File Format*

JPEG – *Joint Photographic Experts Group*

Wi-Fi – *Wireless Fidelity*

NASA – *National Aeronautics and Space Administration*

WiMAX – *Worldwide Interoperability for Microwave Access*

LAN – *Local Area Network*

IEEE – *Institute of Electrical and Electronics Engineers*

Ad Hoc – однорангова мережа

SSID – Service Set Identification

FHSS – *Frequency-hopping spread spectrum*

EDR – *Enhanced Data Rate*

SIG – *Special Interest Group*

HCI – *Host Controller Interface*

DoG – *Difference of Gaussian*

SIFT – *Scale Invariant Feature Transform*

ORB – *Oriented FAST and Rotated BRIEF*

FLANN – *Fast Library for Approximate Nearest Neighbors*

MB – *Megabyte*

kB – *Kilobyte*

ВСТУП

У сучасному світі є багато проблем з передачею та зберіганням даних, та чим далі плине час, тим об’єми даних збільшується, збільшується і швидкість передачі, але для зберігання дуже важливо, який саме об’єм пам’яті буде займати той чи інший об’єкт. Тому однією із основних нагальних проблем із зберіганням та передачею даних є те, як зменшити об’єм пам'яті для об’єкта і тим самим пришвидшити його передачу.

Така проблема потребує вирішення, бо, наприклад, якщо розглядати цю проблему з точки зору зберігання і передачі інформації з бортового комп’ютера, то можна зрозуміти, що бортовий комп’ютер, наприклад, БПЛА не є хранилищем великих об’ємів даних, а має скінченні ресурси. Взагалі в залежності від типу та класу БПЛА він має певні задачі, тобто його призначення. Наприклад, це може бути БПЛА для розвідки, і задачі у нього провести фото та відеофіксацію та передати цю інформації на приймальний пункт. Стає питання в якому форматі передавати цю інформацію. Питаннями форматів даних займається один із рівнів моделі *OSI*, а саме представницький рівень (рівень представлення). На цьому рівні відбувається стиснення, або розпаковка даних, шифрування та їх дешифрування. Для графічних зображень доцільно використовувати алгоритми стиснення, та таким чином перетворювати їх у графічні зображення з інакшим релевантним форматом, але меншим об’ємом пам’яті для їх зберігання та передачі. На сьогодні є різні алгоритми стиснення, такі як із втратами та без втрат інформації, різної компресії даних, від чого залежить і якість отриманих даних в результаті.

У своєму дипломному проекті я буду розглядати один із етапів алгоритма стиснення *JPEG – DCT,* а також алгоритми визначення схожості зображення. *DCT* можна вважати ключовим етапом із семи інших, всього етапів вісім, у тому числі з *DCT*. Дослідження буде проводитися на вияв похибок відновлення графічного зображення від спектральних особливостей цього зображення. Також хочу проаналізувати наскільки ефективним є використання такого способу, як

DCT, для стиснення графічного зображення, адже *DCT* є одним із етапів алгоритму стиснення *JPEG*, а сам *JPEG* є алгоритмом стиснення із втратами. Тобто, чим більше коефіцієнт компресії, тим більше втрати якості зображення.

Метою даного дипломного проекту є визначення границь ефективного застосування дискретно-косинусного перетворення для зберігання, передачі інформації, а також визначення схожості зображення.

Об’єкт дослідження – використання дискретно-косинусного перетворення для зберігання, передачі інформації та визначення схожості зображення.

Предмет дослідження – зберігання та передача інформації з використанням дискретно-косинусного перетворення та визначення схожості зображення.

РОЗДІЛ 1
СИСТЕМНИЙ АНАЛІЗ ПРЕДСТАВНИЦЬОГО РІВНЯ ЕТАЛОННОЇ МОДЕЛІ ВЗАЄМОДІЇ ВІДКРИТИХ СИСТЕМ

1.1. Суть системного аналізу взагалі із критичним аналізом еталонної моделі

1.1.1. Еталонна модель взаємодії відкритих систем

Комп'ютерні мережі розробляються за принципом багаторівневості. Щоб організувати взаємодію між комп’ютерами для початку треба розробити набір визначених заздалегідь правил за якими буде організована їх взаємодія, тобто треба визначити, якщо так можна сказати, «мову» їх спілкування (зв’язку), або ж визначити зміст сигналів за допомогою яких далі буде побудована взаємодія комп’ютерів. Визначення, які були згадані вище, та правила називаються протоколом. Для правильної роботи комп’ютерної мережі використовується низка протоколів, таких як наприклад: протокол керування фізичним зв'язком, встановлення зв'язку в комп’ютерній мережі, протокол доступу до ресурсів та багато інших. Така система з багатьма рівнями була створена з метою спрощення та організації великої кількості існуючих протоколів та зв'язків в комп’ютерній мережі. Модель взаємодії, яка має багато рівнів, передбачає інтерфейси (зв’язок) для конкретного рівня лише із верхнім та нижнім рівнями і віртуальну взаємодію лише із таким же рівнем приймача зв’язку. Під інтерфейсом мається на увазі передача даних. В цьому випадку дані залишаються незмінними та в пункт призначення надходять у такому ж вигляді, як і були у пункті відправлення.

Якщо ж порівнювати з віртуальною взаємодією, то цей вид взаємодії це - передача даних за допомогою посередника і ці посередники можуть змінюватися заздалегідь встановленими способами.

Основним набором правил, які були описані вище, що визначають взаємодію комп'ютерів у мережі є модель *OSI*. Дана модель має сім рівнів – фізичний, канальний, мережевий, транспортний, сеансовий, представлення та прикладний.

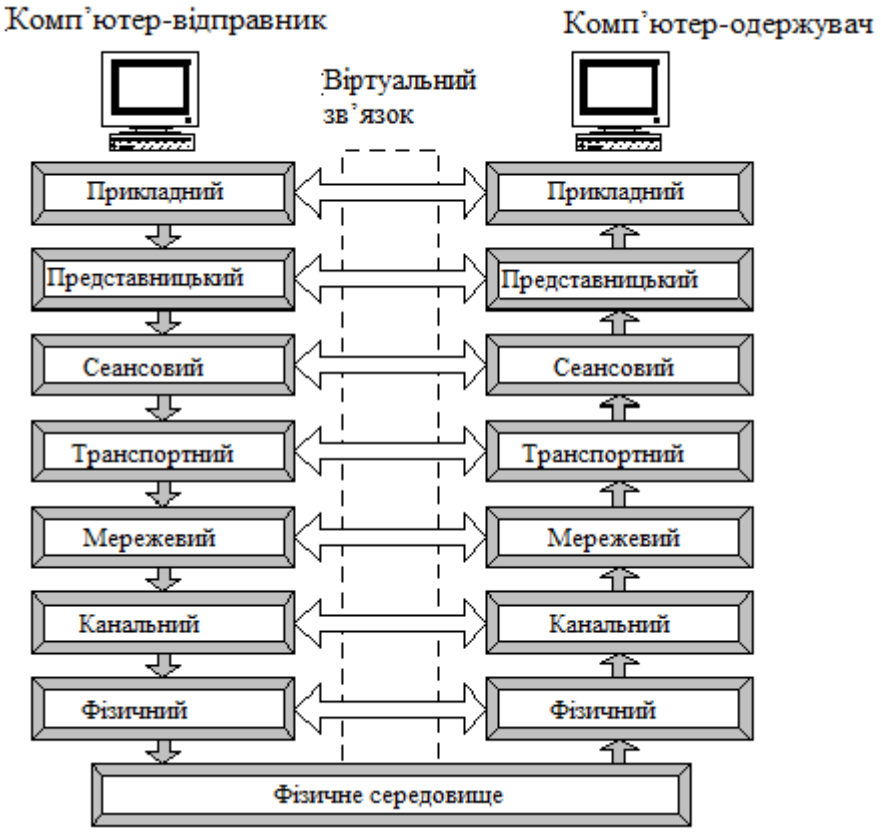

Рис. 1.1. Схема еталонної моделі взаємодії відкритих систем (*OSI*)

- Рівень каналу зв'язку. Цей рівень пов'язаний виключно з фізичним середовищем за допомогою якого передається інформація, а також на цьому рівні визначаються характеристики посередників, що організовують передачу даних. Це можуть бути кабелі, радіолінії, тощо. Як правило, рівень каналу зв'язку не визначається як окремий елемент в моделі взаємодії відкритих систем, але знання про особливості цього рівня можуть допомогти налаштувати стабільний зв'язок, наприклад на рівні простих фізичних з'єднань.
- Фізичний рівень. Цей рівень містить в собі саме фізичні моменти, які призначені для передавання інформації у двійковому вигляді по лінії зв'язку.

Також даний рівень дає опис таким явищам, як, наприклад, напруга, частота, або ж взагалі природа середовища в якому відбувається передача інформації. Він є відповідальним за підтримку, налаштування зв'язку та передачу і прийом інформації у вигляді бітів.

- Канальний рівень. Канальний рівень відповідає за транспортування пакетів даних через фізичний рівень, на якому ці дані мають шанс спотворитись (для цього використовується завадостійке кодування даних) та визначає границі пакету даних в потоці інформації.
- Мережевий рівень. Цей рівень використовує можливості, які надані другим рівнем, для забезпечення стабільного зв'язку двох будь-яких точок в комп’ютерній мережі. Робота з адресами та визначення маршруту пакетів (маршрутизація) також відбувається на даному рівні моделі.
- Транспортний рівень. Він є кінцевою точкою для організації передачі даних, який має контроль над стабільною передачею пакетів, коректною і своєчасною доставкою в визначений пункт призначення, їх збережений зміст та зберігає послідовність пакетів під час їх надходження у точку прийому даних. Відновлює дані шляхом з’єднання пакетів. Має в собі надійну схему адресації для підтримки коректного зв'язку через велику кількість інших мереж та шлюзів. Шлюз – можна сказати є основним пунктом зв'язку із довільною комп’ютерною мережею. Транспортний рівень бере на себе відповідальність за проблеми, які можуть бути пов'язані з передачею даних. Також він організовує взаємодію між рівнями моделі, які знаходяться вище, з інформацією незалежно від процесу передачі інформації.
- Сеансовий рівень. Цей рівень призначений для організації «спілкування» користувачів мережі, та має відповідальність за встановлення зв'язку, відновлення сеансів, які були завершені аварійно. Має відношення до картографії мережі, тобто перетворює доменні імена у числові адреси і у зворотньому напрямку. Підтримує взаємодію процесів у комп’ютерній мережі.
- Рівень представлення даних. Керує синтаксисом і семантикою даних, що передаються через мережі. Таким чином організовує взаємодії та розуміння двох комп’ютерів між собою. На даному рівні вирішуються задачі, які можуть

бути пов’язані із кодуванням, або декодуванням інформації у вигляд зрозумілий для звичайного користувача. Також саме цей рівень має відповідальність за формати зображень, їх стисненням та розпакуванням файлів, а також на цьому рівні організовується взаємодія між різними мережними файловими системами, таких, наприклад, як *NFS*, тощо.

- Прикладний рівень. Організовує взаємодію між кінцевим користувачем і комп’ютерною мережею в якій знаходиться його девайс, та має відповідальність за можливість доступу до багатьох послуг мережі, якими користується звичайний користувач. На цьому рівні працюють не менше ніж п'ять прикладних служб таких як: передача даних у мережі, віддалений доступ, передача електронних листів, служба для довідок та управління самою комп’ютерною мережею. Все це користувач має змогу визначати для себе залежно від його потреб.

1.1.2. Місце двомірного *DCT* в представницькому рівні

Повернемося до рівня шість для того, щоб розглянути його більш детально. На рівні представлення запити додатків або застосунків, які були передані з прикладного рівня, перекодовуються в необхідний коректний формат, і в цьому форматі дані передаються у мережі, а отримані дані з мережі перетворюється в формат зрозумілий для додатків або застосунків. На рівні який розглядається здійснюється стиснення та розпакування інформації, а бо ж її шифрування та дешифрування, а також запити, які адресовані іншому мережевому ресурсу перенаправляються на адрес цього ресурсу, якщо ці запити не можна обробити локально. Рівень представлення керує не тільки перетворенням даних у необхідний формат та їх поданням, він також займається самою структурою даних, які використовуються іншими додатками та застосунками. Тому такими маніпуляціями, які були описані вище, цей рівень забезпечує організацію даних при їх передачі по мережі. На цьому рівні існують і інші підпрограми, які можуть стискати тексти до необхідного розміру і перетворюють зображення у поток бітів, для того щоб ці дані могли передаватися у мережі. Стандарти

властиві цьому рівню визначають способи, якими можна представляти будь-які графічні зображення.

Для такої мети може використовуватися формат, який має назву *PICT*. Це формат графічних зображень, який можна використовувати для передачі графіки, яка має назву *QuickDraw*. Також одним із форматів зображень є *TIFF*, який використовується для зображень з растровою графікою, які мають високу роздільну здатність. Іншим стандартом, який має застосування для перекодування зображень, є стандарт, розроблений Об'єднаної експертною групою по фотографії, такий формат має абревіатуру *JPEG*.

На цьому рівні (рівень представлення) для перетворення графічних зображень у формат *JPEG* має актуальність застосування двовимірного *DCT*, це перетворення інформації є одним з ключових етапів алгоритму стиснення *JPEG*.

1.2. Визначення проблем реалізації функції передачі інформації з бортового комп`ютера та зберігання інформації із визначенням критеріїв якості системи

1.2.1. *ALOHAnet*

Мережі передачі даних можуть бути наступних видів:

- телефонна
- комп'ютерна
- бездротова
- конвергентна

Варто згадати *ALOHAnet*, яка є основою для протоколів та технологій передачі даних в майбутньому.

ALOHAnet - перша комп'ютерна мережа передачі даних з пакетною комутацією, що використовувала бездротове з’єднання в якості середовища доступу до неї. Була розроблена і введена в експлуатацію в 1968-1970-х роках групою вчених Гавайського університету під керівництвом Нормана Абрамсона

в рамках дослідницького проекту *THE ALOHA SYSTEM*, основною метою якого було вивчення можливостей використання радіопередачі як альтернативи на той час провідним комунікаціям. Концептуальні напрацювання і рішення, які були реалізовані в ході цього проекту, багато в чому лягли в основу таких технологій і протоколів як *Ethernet, Wi-Fi* і стільникових мереж. У 1973 році в мережу з використанням супутникових каналів зв'язку *NASA ATS*-1 були об'єднані обчислювальні центри Гавайського університету, Дослідницького центру Еймса (*NASA*), Університету Аляски, Університету Тохоку, Університету електро-комунікацій (Токіо) і Сіднейського університету.

У мережі *ALOHAnet* для з'єднання робочих станцій з головним обчислювальним центром використовувалося радіоз'єднання в дециметровому діапазоні хвиль. Було виділено два радіоканалу шириною 100 КГц зі швидкістю передачі даних по ним 24 000 бод. Один радіоканал на частоті 407.350 МГц використовувався для передачі даних від терміналів до центрального комп'ютера в Гонолулу, а другий канал на частоті 413.475 МГц використовувався для розсилки широкомовних повідомлень від центрального комп'ютера до терміналів (для цього біля центрального комп'ютера була встановлена широкомовна антена, а на віддалених островах - спрямовані антени, які дозволяли приймати повідомлення один від одного - в системі *ALOHA* використовувалася мережева топологія зірка). Оскільки при одночасній спробі передачі по одному частотному діапазону з декількох станцій відбувалися колізії, а колізія призводила до спотворення даних які передаються через мережу, було прийнято інноваційне рішення використовувати метод випадкового доступу до каналу, пізніше названим *ALOHA random access*, який став ключовим нововведенням цієї технології, а також вперше було вирішено розбити дані на «пакети» (по 704 біта: 80 8-бітних символів + 64 біта керуючих).

Першу версію *ALOHA random access* також називають чистим *ALOHA* (англ. *Pure ALOHA*). При використанні цього методу для доступу до каналу зв'язку, комп'ютери, які призначені для звичайного користувача починали передавати пакети даних для центрального комп'ютера відразу після появи призначених для передачі даних. Якщо передача двох або більшого числа станцій збігаються за

часом (хоча б частково), то центральний комп'ютер не може коректно прийняти дані. Щоб дати відправникам можливість виявити колізію, центральний комп'ютер розсилає отриманий пакет даних після прийому. Порівнюючи переданий пакет і прийнятий, відправник може зрозуміти, чи були його дані прийняті коректно або з помилками. Якщо були передані спотворені дані, то відправник очікує випадковий інтервал часу і здійснює повторну спробу передачі.

Оцінка пропускної здатності чистої системи *ALOHA* визначається при наступних пунктах:

- Дані користувачів, які були призначені для передачі, надходять на термінали випадково, створюючи пуассоновский потік;
- Пакети відкинуті через їх спотворення передаються повторно, які також створюють також пуассоновский потік;
- Всі пакети з даними мають однакову довжину і передаються однаковий час τ
- В мережі знаходиться нескінченне число віддалених терміналів (тобто якщо якийсь термінал вже передає дані, це ніяк не впливає на ймовірність передачі даних іншими терміналами).

У 1972 році Лоуренс Робертс запропонував іншу версію системи *ALOHA*, названу слотованою *ALOHA* (англ. *Slotted ALOHA*). Основною відмінністю слотованої *ALOHA* від звичайної була ідея поділу осі часу на дискретні інтервали рівної тривалості τ, названі слотами. Кожен термінал послідовно відміряв границю слотів. Для синхронізації границь слотів використовувався спеціальний синхронізуючий сигнал, який передається з широкомовної антени всім терміналам. При появі призначених для передачі пакетів даних термінал затримував передачу до початку наступного слота. Тривалість слотів вибиралася так, щоб за час одного слота термінал встиг передати свій пакет даних і отримати від центрального комп'ютера підтвердження успішного отримання.

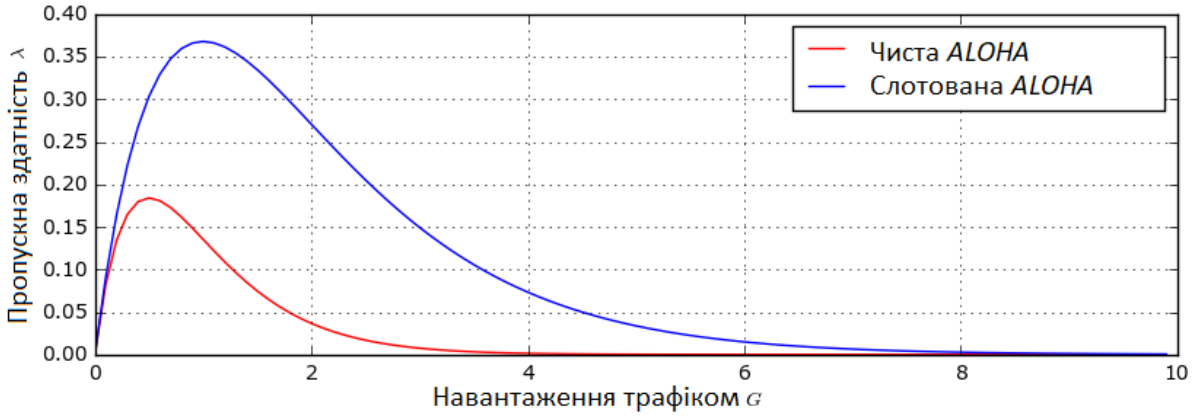

Рис. 1.2. Порівняння пропускної здатності *ALOHA*

Таким чином, використання слотованої *ALOHA* замість чистої дозволило збільшити максимальну пропускну здатність мережі в два рази. Як видно з графіків, поки значення навантаження менше тієї критичної величини, при якій досягається максимум, пропускна здатність мережі зростає зі збільшенням трафіку - система не використовується на 100%. Однак після перевищення критичної величини навантаження, пропускна здатність системи падає - занадто багато пакетів потрапляє в колізії і передається з спотворенням.

Поява в мережі *ALOHAnet* радіоретрансляторів дозволило розширити і упорядкувати її структуру. У 1973 році *ALOHAnet* була приєднана до мережі *ARPAnet* з використанням супутникового каналу зв'язку. Як розвиток ідеї випадкового конкурентного доступу до каналу зв'язку, вперше застосованої в системі *ALOHA*, був створений метод *CSMA*. Модифікації цього методу *CSMA / CA* і *CSMA / CD* лягли в основу протоколів канального рівня мереж *Ethernet* і *Wi-Fi*. *ALOHA random access* використовується в мобільних голосових і пакетних мережах. Зокрема, при встановленні голосового, СМС або інтернет-з'єднання, так перший пакет даних відправляється мобільним пристроєм з використанням *ALOHA random access*. *ALOHA random access* також був використаний в супутникових мережах.

В телекомунікації присутні два види передачі даних:

- Послідовна. В даному випадку передача інформації у вигляді символів і інших об'єктів даних відбувається в послідовному режимі. Такі цифрові мережі передачі даних відправляють біти по одному дроту, оптичному шляху або ж частоті. У зв'язку з цим даний процес вимагає меншого часу на обробку самого

сигналу, а сама швидкість передачі даних збільшується. Послідовна мережа має застосовування і на більш далеких відстанях, бо це обумовлено легкою передачею біта парності і цифри.

- Паралельна. Це одночасна передача інформації (елементів сигналу одного символу). Застосування великої кількості дротів у цифровому зв'язку допомагає здійснити передачу одночасно декількох біт. Все це дозволяє досягти високої швидкості передачі даних. Даний спосіб використовується всередині самого комп'ютера (у внутрішніх шинах даних, наприклад).

Єдиним недоліком тут є «перекіс». Обумовлений він тим, що дроти можуть відрізнятися між собою своїми характеристиками. Саме тому один біт може бути отриманим раніше ніж інший. А це, в свою чергу, буде негативно позначатися на цілісності самого повідомлення.

За принципом комутації мережі можуть бути:

- З комутацією пакетів. Усі дані в цьому випадку передаються невеликими посилками. Їх ще називають пакети, комутація яких відбувається незалежно. На такому принципі побудована велика частина комп'ютерних мереж в нинішній час. Але для роботи тут необхідно більш складне обладнання.

- З комутацією каналів. Для передачі між пристроями виділяється спеціальний канал (логічний або фізичний). Інформація по ньому передається безперервно.

Сукупність усіх видів передачі даних називається канал передачі даних. У нього входять такі засоби передачі даних, як: інтернет мережі, стаціонарні лінії, точки прийому і передачі даних. Канали передачі даних поділяють на два види: аналогові і дискретні. Основна відмінність полягає в тому, що аналоговий тип являє собою безперервний сигнал, а дискретний, в свою чергу, являє собою потік даних який може бути перерваний у будь який час. Для забезпечення оптимальної роботи всі пристрої працюють з іншими пристроями в дискретному вигляді. У дискретному вигляді застосовуються цифрові коди, які перетворюються в електричні сигнали. А для передачі дискретних даних за допомогою аналогового сигналу потрібно модуляція дискретного сигналу.

При використанні інформації на пристрої відбувається зворотнє перетворення сигналу. Зворотнє перетворення сигналу називається демодуляцією. Таким чином, існує два процеси перетворення сигналу: модуляція і демодуляція. В процесі модуляції інформація являє собою синусоїдальний сигнал з певною частотою.

Для перетворення даних використовуються такі способи модуляції:

- Амплітудна модуляція даних
- Частотна модуляція даних
- Фазова модуляція даних

Для передачі даних дискретного типу по цифровому каналу використовується система кодування. В основному, розрізняють два типи кодування:

- Потенційне кодування
- Імпульсне кодування

Для обміну даними між обчислювальними мережами використовують три основні методи передачі інформації:

- Симплексний (односпрямована)
- Напівдуплексному
- Дуплексная (двунаправленная)

Перед тим, як відправити інформацію в обчислювальну мережу, відправник розділяє інформацію на маленькі блоки, які найчастіше називають пакетами даних. На кінцевому пункті відправки всі пакети збираються в єдиний послідовний список. Потім відбувається процес перетворення всіх частин в єдиний вихідний матеріал.

Для правильної роботи з пакетом даних повинна бути вказана така інформація, як:

- Передані файли
- Посилання на файл, інформація про фото
- Коди управління файлом. Являють собою список відомостей про файл

Існують три типи комутації обчислювальної системи:

- Комутація каналів;
- Комутація пакетів;
- Комутація повідомлень.

Комутація каналів служить для створення безперервного каналу з послідовно з'єднаних ліній. Після того як даний канал утворився, вся інформація і файли можуть передаватися на високій швидкості. Комутація повідомлень, щоб виконувати завдання поштовими файлами і серверами. Ця операція включає в себе ряд можливостей таких як: передача, прийом, зберігання. Велика кількість повідомлень, як правило, передається блоками. При відправці групи повідомлень блок переходить від одного комунікаційного вузла до іншого і в кінцевому підсумку доходить до адресата. Якщо сталася помилка передачі блоку (збій зв'язку, технічні неполадки і т.д.), то весь блок повідомлень почне передаватися заново. До того моменту поки весь блок повідомлень не досягне одержувача, буде неможливо здійснити нову передачу.

Процес передачі пакетів повідомлень повністю ідентичний процесу передачі повідомлень. Завдяки меншому розміру, пакет з інформацією швидко проходить комунікаційні вузли. Тому канал зайнятий тільки при передачі пакетних даних, а після завершення звільняється для подальших завантажень. Подібний тип передачі даних є визнаним стандартом для мережі Інтернет. Сучасні комунікаційні мережі мають технологію цифрової передачі даних, що дозволять передавати будь-який тип інформації з даного каналу. А новітні сучасні матеріали і високу якість установки дозволяють домогтися високих швидкостей з'єднання. У нинішній час дуже проблематично собі уявити будь - яку компанію, яка не використовувала бездротові технології. Це обумовлено перевагами, які мають бездротові мережі Це може бути: *Wi-Fi, Bluetooth* і *WiMAX*. Всі вони працюють на радіохвилях, за допомогою радіоканалів певної частоти. Відрізняються вони між собою частотою і шириною самих хвиль. Ну і звичайно ж швидкістю передачі даних. За допомогою технологій, які були зазначені вище, вдається сформувати комп'ютерні групи, де немає місця для кабелів.

Конвергентні мережі для передачі даних являють собою обчислювальні мережі, в яких об'єднана передача голосових даних та самих даних. Все це забезпечує наступне:

- Можливість здійснювати роботу з різнорідною інформацією, такий як відео, голос, файли і електронна пошта, на загальному призначеному для користувача терміналі.
- Істотне спрощення корпоративних комунікацій. Тут кілька незалежних мереж замінюються загальною мережею.
- Додаткову високу функціональність, а також спрощення робіт під час обміну різнорідними даними і їх обробки.

Сам термін конвергенція дуже часто можна почути на телекомунікаційних семінарах, конференціях і навіть виставках. Розглянемо такі технології, як *Wi-Fi*, *Bluetooth* і *WiMAX*.

1.2.2. *Wi-Fi*

Wi-Fi - це назва стандарту *IEEE* 802.11, який стосується передачі потоків даних, які були оцифровані по радіоканалах. Будь-яке обладнання, яке відповідає цьому стандарту, може бути перевірено на *Wi-Fi Alliance*, і якщо тест успішний, то це обладнання отримує сертифікат і права на створення логотипу *Wi-Fi*. Найбільш поширеним сьогодні є протокол *IEEE* 802.11*n*. Створення бездротової локальної мережі буде придатним для розвитку мереж, де встановлення кабельної системи неможливо або економічно невигідне. Сучасні реалізації *Wi-Fi* дозволяють здійснювати швидкість передачі даних понад 100 Мбіт / с, тоді як кожен користувач такої мережі може переміщатися в будь-якому місці у зоні покриття цієї бездротової мережі і може використовувати будь-які пристрої, які мають вбудований модуль *Wi-Fi* і мають можливість для доступу до Інтернету.

Рис. 1.3. Офіційний логотип *Wi-Fi*

Як правило, схема мережі *Wi-Fi* може включати в себе щонайменше одну точку доступу і може легко масштабуватися. Також можливе одночасне підключення декількох клієнтів в режимі «точка-точка» (*Ad-hoc*), коли точка доступу не використовується, і клієнти з'єднуються з мережевими адаптерами «безпосередньо».

Точка доступу повинна завжди передавати свій ідентифікатор мережі (*SSID*) за допомогою спеціальних сигнальних пакетів на 0,1 Мбіт / с на 100 мс. Тому 0,1 Мбіт / с - це найменша швидкість передачі даних для *Wi-Fi*. Знаючи мережевий *SSID* (мережеве ім'я), користувач може дізнатися, чи може він підключитися до цієї точки. Коли користувач входить в зону роботи двох точок доступу з однаковими іменами, користувач може вибирати між ними, виходячи з якості рівня сигналу. Стандарт *Wi-Fi* надає клієнту повну свободу у виборі критеріїв підключення. Однак стандарт не описує всі аспекти побудови бездротових локальних мереж *Wi-Fi*. Тому кожен виробник обладнання вирішує цю проблему по-своєму, застосовуючи підходи, які він вважає найкращими з тієї чи іншої точки. Тому існує необхідність класифікувати способи побудови бездротових локальних мереж. За допомогою комбінування точок доступу в єдину систему можна вибрати:

- Автономні точки доступу (також називаються незалежними, децентралізованими, інтелектуальними)
- Точки доступу, що працюють під контролем контролера (також звані "легкі", централізовані)
- Неконтрольований, але не автономний (керований без контролера)

За допомогою організації та управління радіоканалами можна ідентифікувати бездротові локальні мережі:

- За допомогою статичних установок радіоканалу
- За допомогою динамічних (адаптивних) установок радіоканалу
- з «багатошаровою» структурою радіоканалів

***IEEE* 802.11** - набір стандартів для зв'язку через пропускну здатність бездротової локальної мережі 0,9; 2.4; 3.6; 5 і 60 ГГц. При описі стандарту в дужках вказано рік його прийняття. Швидкість вказана приблизно.

- 802.11 - початковий 1 Мбіт / с і 2 Мбіт / с, 2.4 ГГц та ІК стандарт (1997).
- 802.11*a* - 54 Мбіт / с, стандарт 5 ГГц (1999, випуск продукту в 2001 році).
- 802.11*b* - оновлення до 802.11 для підтримки 5.5 та 11 *Mbps* (1999).
- процедури 802.11c; включені в стандарт *IEEE* 802.1*D* (2001).
- 802.11*d* - розширення міжнародного роумінгу (2001).
- 802.11*e* - удосконалення: *QoS*, пакетний розрив (2005).
- 802.11*F* - Протокол між точками доступу (2003).
- 802.11*g* - 54 Мбіт / с, стандарт 2.4 ГГц (*b*-сумісний) (2003).
- 802.11*h* - розподілений за 802.11*a* (5 ГГц) для сумісності в Європі (2004).
- 802.11*i* - Покращена безпека (2004).
- 802.11*j* - розширення для Японії (2004).
- 802.11*k* - вдосконалене вимірювання радіоресурсів.
- 802.11*l* - зарезервовано.
- 802.11*m* - виправлення та виправлення для всієї групи стандартів 802.11.
- 802.11*n* (*WiFi* 4) - Збільшення швидкісті передачі даних (600 Мбіт / с). 2.4-2.5 або 5 ГГц. Зворотна сумісність з 802.11*a* / *b* / *g* (вересень 2009).

- 802.11*o* - зарезервовано.
- 802.11*p* - *WAVE* - безпровідний доступ для середовища транспортного засобу (бездротовий доступ до транспортних засобів).
- 802.11*q* - зарезервований, іноді плутаний з 802.1*Q*.
- 802.11*r* - швидкий роумінг.
- 802.11*s* - бездротова мережа *ESS* (розширений набір послуг; мережева мережа - багатоадресна мережа).
- 802.11*T* - прогнозування бездротового зв'язку (*WPP*, прогнозування продуктивності бездротового обладнання) - методи випробувань та вимірювання.
- 802.11*u* - взаємодія з не-802 мережами (наприклад, стільниковий).
- 802.11*v* - управління бездротовою мережею.
- 802.11*w* - захищені кадри управління (захищені контрольні кадри).
- 802.11*x* - зарезервовано і не використовуватиметься. Не плутати із стандартом контролю доступу *IEEE* 802.1*X*.
- 802.11*y* - додатковий стандарт зв'язку, що працює на частотах 3,65-3,70 ГГц. Забезпечує швидкість до 54 Мбіт / с до 5000 м у відкритому просторі.
- 802.11*ac* (*WiFi* 5) - новий стандарт *IEEE*. Швидкість передачі даних до 6,77 Гбіт / с для пристроїв з 8 антенами. Затверджено у січні 2014 року.
- 802.11*ad* - новий стандарт з додатковим діапазоном 60 ГГц (частота не вимагає ліцензування). Швидкість передачі даних до 7 Гбіт / с
- 802.11*ax* (*WiFi* 6) - новий стандарт (до 10747 Мбіт / с)
- 802.11*ay* - в розробці (до 20 Гбіт / с). Стандартне затвердження очікується у листопаді 2019 року.
- 802.11*az* є перспективним стандартом, який має бути реалізований у березні 2021 року.

1.2.3. *Bluetooth*

Bluetooth **-** це бездротова технологія, створена в 1998 році. Основною метою Bluetooth є забезпечення економічної ефективності та комунікації між різними

електронними пристроями, такими як стільникові телефони, смарт-годинники, бездротові гарнітури, планшети, принтери, ноутбуки та інше. Також можна використовувати Bluetooth у невеликих пристроях, таких як годинник.

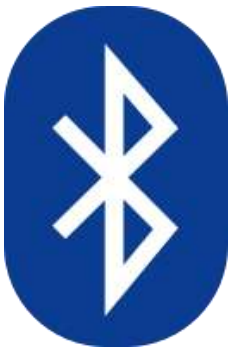

Рис. 1.4. Логотип *Bluetooth*

Інтерфейс *Bluetooth* дозволяє передавати як голос (зі швидкістю 64 Кбіт / с), так і дані. Для передачі даних можуть використовуватися асиметричні (721 Кбіт / с в одному напрямку і 57,6 Кбіт / с в іншому) і симетричні (432,6 Кбіт / с в обох напрямках) методи. При роботі на частоті 2,4 ГГц приймач (*Bluetooth*-чіп) може підключатися в межах 10 або 100 метрів. Велика різниця в відстані, але з'єднання в межах 10 метрів дозволяє зберегти низьке енергоспоживання, компактні розміри і низьку вартість компонентів. Так, передавач малої потужності споживає лише 0,3 мА в режимі очікування і в середньому 30 мА під час обміну інформацією. Стандарт *Bluetooth* забезпечує шифрування даних, що передаються за допомогою ключа ефективної довжини від 8 до 128 біт і можливість вибору односторонньої або двосторонньої аутентифікації. Крім того, шифрування на рівні програми можна використовувати для шифрування на рівні протоколу. Технологія *Bluetooth* працює за принципом *FHSS*. Коротше кажучи, це можна пояснити таким чином: передавач розбиває дані на пакети і передає їх до псевдовипадкового алгоритму для скасування частоти (1600 разів на секунду) або шаблон (малюнок), що складається з 79 суб-частот. "Зрозуміти" можуть тільки ті пристрої, які налаштовані на один і той же шаблон передачі - для пристроїв сторонніх виробників, передана інформація буде нормальним шумом. Основним структурним елементом мережі *Bluetooth* є так звана "пікомережа"

(piconet) - колекція від 2 до 8 пристроїв, що працюють на одному шаблоні. У кожній пікомережі один пристрій працює як активний, а інший - як пасивний (підлеглий) пристрій. Активний пристрій визначає шаблон, який буде запускати всі підлеглі пристрої і синхронізувати його роботу. Стандарт *Bluetooth* передбачає підключення незалежної і навіть несинхронізованої швидкої допомоги (до 10) в так званому "*scatternet*" (розсіювання звучить як "*scatter*"). Для цього кожне сполучення в парі має мати принаймні одне спільне пристрій, який буде активним в одному і пасивно в іншому. Таким чином, в окремому scatternet з інтерфейсом *Bluetooth*, максимум 71 пристрої можуть бути підключені за один раз, але ніхто не обмежує використання шлюзів, які використовують той же Інтернет для більш віддаленого з'єднання. Діапазон частот *Bluetooth* у більшості країн вільний від ліцензування, але у Франції, Іспанії та Японії, через правові обмеження, необхідно використовувати різні частоти з вищезазначених.

Bluetooth 1.0 Версія 1.0 (1998) і 1.0*B* пристрої мали погану сумісність між продуктами різних виробників. В 1.0 і 1.0*B* відбувалася обов'язкова передача адреси пристрою (*BD_ADDR*) під час фази установки, що унеможливило реалізацію анонімності з'єднання на рівні протоколу і було основним недоліком даної специфікації.

Bluetooth 1.1. *Bluetooth* 1.1 виправив купу помилок, знайдених у 1.0*B*, додав підтримку незашифрованих каналів, індикацію рівня сигналу (*RSSI*).

Bluetooth 1.2 У версії 1.2 була додана технологія адаптивного регулювання частоти (*AFH*), яка має поліпшену стійкість до електромагнітних перешкод (перешкод) за допомогою декількох частот у послідовності перебудови.

Також збільшилася швидкість передачі і додана технологія *eSCO*, що поліпшило якість передачі мови шляхом повторення пошкоджених пакетів. *HCI* додала підтримку для трипровідного *UART*-інтерфейсу.

Основні покращення:

- Швидке з'єднання та виявлення.
- Адаптивне регулювання частоти з розширеним спектром (*AFH*), що підвищує стійкість до радіоперешкод.
- Швидше, ніж 1.1, швидкість передачі даних досягає 721 кбіт / с.
- *Advanced Synchronous Connections* (*eSCO*), які покращують якість передачі голосу в аудіопотоці, дозволяючи повторні передачі пошкоджених пакетів і, при необхідності, можуть збільшити затримку звуку для забезпечення кращої підтримки паралельної передачі даних.
- Інтерфейс хост-контролера (*HCI*) додає підтримку для потрійного інтерфейсу *UART*.
- Затверджено як стандарт стандарту *IEEE* 802.15.1-2005.
- Впроваджено режими контролю потоку та повторної передачі для *L2CAP*.

Bluetooth 2.0 + *EDR*. 10 листопада 2004 року вийшла версія *Bluetooth* 2.0, яка має зворотну сумісність з попередніми версіями 1.*x*. Основною інновацією була підтримка *Enhanced Data Rate* (*EDR*) для прискорення передачі даних. Номінальна швидкість *EDR* становить близько 3 Мбіт / с, але на практиці це дозволило збільшити швидкість передачі даних лише до 2,1 Мбіт / с. Додаткова продуктивність досягається за допомогою різних технологій радіозв'язку для передачі даних. Стандартна (основна) швидкість передачі даних використовує *GFSK*-модуляцію радіосигналу при швидкості передачі 1 Мбіт / с. *EDR* використовує комбінацію *GFSK* і *PSK* модуляцій з двома варіантами, π / 4-*DQPSK* і 8*DPSK*. Крім *EDR*, є й інші незначні вдосконалення специфікації 2.0, а продукти можуть відповідати "*Bluetooth* 2.0 *Technologies*" без підтримки більш високих швидкостей передачі даних. Принаймні один комерційний пристрій, *HTC TyTN Pocket PC*, використовує "*Bluetooth* 2.0 без *EDR*" у своїх технічних специфікаціях.

Відповідно до специфікації 2.0+ *EDR, EDR* надає такі переваги:

- У деяких випадках збільшити швидкість передачі в 3 рази (2,1 Мбіт / с).
- Зменшення складності декількох одночасних з'єднань через додаткову пропускну здатність.
- Менше споживання енергії через зменшення навантаження.

Bluetooth 2.1 У 2007 році була додана розширена технологія запиту функцій для пристрою (для додаткової фільтрації списку спільно), енергозберігаючої технології *Sniff Subrating*, що дозволяє збільшити тривалість пристрою від одного заряду батареї 3 -10 разів. Крім того, оновлена специфікація значно спрощує і прискорює з'єднання між двома пристроями, дозволяє оновлювати ключ шифрування без порушення з'єднання, а також робить ці з'єднання більш безпечними, завдяки використанню технології *Near Field Communication*.

Bluetooth 2.1 + *EDR*. У серпні 2008 року компанія *Bluetooth SIG* представила версію 2.1 + *EDR*. Нове видання *Bluetooth* зменшує споживання електроенергії до 5 разів, покращує захист даних і полегшує розпізнавання та підключення пристроїв *Bluetooth* за рахунок зменшення кількості кроків, які він виконує.

Робоча група по розробці бездротових даних *Bluetooth* 21 квітня 2009 року випустила специфікацію *Bluetooth* 3.0. Модулі, що підтримують нову специфікацію, об'єднають дві радіосистеми. Перший, малопотужний, забезпечує передачу даних до звичайної для другої версії швидкості *Bluetooth* трьох мегабіт в секунду. Інший, високошвидкісний і сумісний зі стандартом *IEEE* 802.11, забезпечує швидкість, порівнянну зі швидкістю мереж *Wi-Fi*.

Варто відзначити, що Bluetooth 3.0 використовує стандартний 802.11 без суфікса, тобто формально несумісний з такими специфікаціями *Wi-Fi* як 802.11*b* / *g* або 802.11*n*. 802.11 є найбільш поширеним стандартом. Використання радіосистеми залежить від розміру переданого файлу. Невеликі файли будуть передаватися на повільному каналі, а великі - на високошвидкісній основі. Після передачі модуль повертається в режим низького енергоспоживання. Крім того, *Bluetooth* 3.0 має функцію "*Enhanced Power Control*". Це запобігає розриву з'єднання, якщо пристрій поміщений в сумку або в кишеню.

Bluetooth 4.0

У грудні 2009 року консорціум *Bluetooth SIG* оголосив про стандарт *Bluetooth* 4.0 для електронних пристроїв. Новий стандарт призначений для передачі коротких пакетів даних розміром 8-27 байт зі швидкістю 1 Мбіт / с. Для порівняння, *Bluetooth* 3.0, розробка якого була завершена в квітні 2008 року, дозволяє передавати дані зі швидкістю до 24 Мбіт / с, але призначена для іншої області застосування. *Bluetooth* 4.0 використовується в мініатюрних датчиках, розміщених на тілі пацієнтів, в спортивному взуття, тренажерах тощо. Датчики на основі нового стандарту можуть передавати різну інформацію від зовнішнього світу - температуру, тиск, вологість, швидкість руху і так далі - різним пристроям управління, включаючи мобільні телефони. За словами представників консорціуму, окремий стандарт був розроблений завдяки тому, що *Bluetooth* 3.0 і більш ранні версії не змогли забезпечити необхідне низьке енергоспоживання. Перший чіп з одночасною підтримкою *Bluetooth* 4.0 і 3.0 випустив *ST-Ericsson*. У липні 2010 року специфікація була схвалена компанією *Bluetooth Special Interest Group*.

Bluetooth 4.2

Грудень 3, 2014 Група спеціальних інтересів *Bluetooth* (*SIG*) представила специфікацію *Bluetooth* 4.2. Нове видання збільшило швидкість прийому даних. Поліпшення конфіденційності та безпеки. Реалізовано можливість підключення до Інтернету.

Bluetooth 5

Bluetooth SIG офіційно представив *Bluetooth* 5 16 червня 2016 року. *Samsung Galaxy S*8 підтримує *Bluetooth* 5.0.

1.2.4. Визначення реалізації функції передачі інформації для БПЛА

Повернемося до розгляду БПЛА з точки зору зберігання інформації. Якщо БПЛА використовується для отримання, збереження і передачі інформації на пульт оператора, в ньому додатково встановлюються карта пам'яті і передавач.

Також треба зазначити, що якість системи зберігання та передачі інформації визначається за певними критеріями, розглянемо декілька з них:

- Релевантний об’єм пам’яті для зберігання інформації – такий максимальний об’єм пам’яті, який необхідний для тих чи інших цілей та який буде повністю забезпечувати потреби.
- Коли подій (похибок) багато і вони слідують одна за одною, то вони утворюють потік. Інтенсивність потоку похибок вказує скільки в середньому відбувається таких похибок за одиницю часу. Якщо інтервал між подіями(похибками)дорівнює константі або визначений якою-небудь формулою, то потік називається детермінованим. Інакше потік називається випадковим. Випадкові потоки бувають: ординарні(ймовірність одночасної появи двох і більше подій дорівнює нулю), стаціонарні, без післядії(ймовірність появи випадкової події не залежить від моменту появи попередніх подій).

Висновки до розділу

У даному розділі розглянуто еталонну модель взаємодії відкритих систем (*OSI*) та існуючі реалізації передачі даних. Також було проаналізовано характеристики засобів для передачі даних та їх версії, які є актульними на данимй момент часу. У ході аналізу було закцентовано увагу на один із рівнів еталонної моделі взаємодії відкритих систем, а саме рівень представлення.

Основним показником якості системи зберігання та передачі інформації з бортового комп’ютера визначено такий показник, як релевантний об’єм пам’яті для зберігання та передачі інформації.

РОЗДІЛ 2
ВИБІР І ОБҐРУНТУВАННЯ ІНСТРУМЕНТАРІЮ ДЛЯ ВИЗНАЧЕННЯ ЕФЕКТИВНИХ СПОСОБІВ ПЕРЕДАЧІ ІНФОРМАЦІЇ

2.1. Обґрунтування вибору показників ефективності інструментарію

2.1.1. Показники ефективності інструментарію

Швидкість обробки інформації комп'ютером обмежена. Саме тому дуже важливо оцінювати час виконання алгоритму за допомогою комп'ютера, так як якщо алгоритм не є оптимізованим, то він може виконуватися тривалий час. Але так як швидкість роботи комп'ютерів різна, то оцінюють не час роботи конкретного алгоритму, а його трудомісткість. Для одержання трудомісткості алгоритму, представленого у формальній системі абстрактної машини необхідно уточнювати які сама операції вважаються «елементарними», та співвідносити їх з мовою програмування високого рівня. «Елементарні» операції можуть бути такими:

- Просте присвоєння.
- Доступ за індексом.
- Арифметичні операції.
- Операція порівняння.
- Логічна операція.

Аналіз трудомісткості алгоритму робиться по різному для різних конструкцій. Розберемо їх.

Конструкція проходження. Це крок алгоритму і його трудомісткість визначається трудомісткістю всіх операцій на цьому кроці. Наприклад, якщо треба порахувати 2 + 2 і прирівняти результат до якоїсь змінної, то тут будуть дві операції: "складання" і "прирівнення".

Конструкція розгалуження. У цьому випадку трохи складніше, ніж з конструкцією проходження, так як не завжди відразу зрозуміло, по якій гілці програма продовжить своє виконання. Більш того, в різних ситуаціях програма може йти по різних гілках. Тому для обчислення трудомісткості алгоритму розгалуження обчислюють ймовірність того, що програма піде по тій чи іншій гілці, і підсумовують трудомісткість всіх гілок, помножену на ймовірність того, що програма піде по цій гілці. Наприклад, нехай у нас перевіряється умова рівності якоїсь змінної числу 3. Припустимо, ми знаємо, що ця змінна може приймати значення від 1 до 10. Значить, ймовірність того, що умова виконається 10%, а що не виконається 90%. Правда, не завжди буває так просто визначити ймовірність. Розглянемо, наприклад, таку ситуацію: програма запитує вік користувача. Якщо введене значення менше 18, то вона працює в одному режимі, якщо більше або дорівнює то в іншому режимі.

Конструкція циклу. Для визначення трудомісткості циклу необхідно трудомісткість однієї ітерації циклу помножити на кількість повторень.

Тепер перейдемо до часових оцінок. Порівняння двох алгоритмів по їх трудомісткості може занести не дуже велику похибку в результати, які ми отримаємо. Однією із таких похибок є різна частота зустрічання «елементарних» операцій, що породжується різними алгоритмами і відмінність у часі виконання цих «елементарних» операцій у реальному житті. Отже, виникає необхідність переходу від трудомісткості до оцінки часу роботи алгоритму на конкретному процесорі.

Під час розгляду часових оцінок можна наткнутися різного роду проблем, огляд яких може викликати вагомі труднощі. Зазначимо основні з цих проблем:

- некоректність формальної системи яка описується для запису алгоритму та реальної системи команд процесора

- деякі особливості архітектури процесора можуть суттєво впливати на час виконання програми
- різний час виконання реальних машинних команд
- різниця у часі при виконанні однієї і тієї ж команди, але в залежності від різних значень операндів
- відмінність у часі реального виконання однотипних команд в залежності від їх типів даних
- неоднозначна компіляція вихідного тексту, причиною чого може бути, як сам компілятор, так і його визначені налаштування

2.2. Системний аналіз існуючих засобів дослідження ефективності систем зберігання та передачі інформації з бортового комп'ютера із обґрунтуванням вибору інструментів дослідження

2.2.1. Алгебра

Алгебра — це розділ математики, який вивчає математичні операції та їх відношення. Утворення, які базуються на математичних операціях та їх відношеннях можуть бути такі: многочлени, алгебраїчні рівняння, алгебраїчні структури. У сучасному житті алгебра є однією із найважливіших частин математики, що може використовуватися, як і у теорії, так і на практиці у різних галузях науки. Існують такі підрозділи алгебри:

- Елементарна алгебра — включає в себе лінійні рівняння, квадратні рівняння, кубічні рівняння та рівняння 4 степеня
- Абстрактна алгебра — спеціалізується на алгебраїчних структурах, наприклад як групи, кільця
- Булева алгебра — є однією із алгебраїчних структур
- Лінійна алгебра — допомагає вивчати вектори та матриці
- Універсальна алгебра — вивчає алгебраїчні властивості, які є спільними для всіх існуючих алгебраїчних структур
- Алгебраїчна теорія чисел — спеціалізується на цілих числах у різних представленнях алгебраїчних структур

- Комутативна алгебра — займається вивченням комутативних кілець, їх модулів та ідеалів
- Алгебраїчна комбінаторика — застосовує різні методи абстрактної алгебри та теорії груп для вирішення задач комбінаторики
- Алгебра Кодда — займається вивченням теорії множин, та така алгебра лягла в основу логіки для роботи існуючих баз даних

2.2.2. Теорія графів

Теорія графів — це розділ математики, який займається вивченням властивостей та особливостей графів. Визначення самого графу є абстракним поняттям, тому терміном «граф» можна описати дуже багато подій навколо та навіть об'єкти, які ми можемо зустріти у реальному житті. Тому узагальнення та абстракція дає можливість застосовувати алгоритми для вирішення задач, які є нетиповими, наприклад у комп'ютерних мережах та інших комп'ютерних моделюваннях. Теорія графів може гарно себе зарекомендувати, якщо розглядати її у контексті геоінформаційних систем (ГІС). Використання багатьох обчислень може дати можливість знаходження найкоротшого шляху та дає можливість планувати оптимальний маршрут. Де має застосування теорія графів:

- Теорія графів є основою хемоінформатика. У цьому випадку вона має змогу визначати кількість ізомерів у органічних сполуках та вуглеводах.
- Програмування
- Транспортні системи
- Економіка
- Логістика
- Схемотехніка (наприклад, різні топології для з’єдання елментів.

2.2.3. Математичне програмування

Математичне програмування — це наука, яка займається вивченням задач, які займаються пошуком *max* або *min* та створює методи для розв'язання такого роду задач. Вони ще називаються оптимізаційними задачами. Залежно від типу функції та напряму, математичне програмування поділяють на:

- Лінійне програмування. Функція, яка є лінійною (прикладом є рівняння першого порядку)
- Нелінійне програмування. Функція, є нелінійними (прикладом є рівняння вищих порядків)
- Дискретне програмування — одна змінна підпадає під умову цілочисельності
- Динамічне програмування — параметри змінюються за плином часу, та рішення може мати декілька варіантів

В залежності від інформації, математичне програмування в залежності від галузі може поділятися на:

- Стохастичне програмування — відомої інформації дуже мало, параметри функції є випадковими.
- Детерміноване програмування — у цьому випадку інформації для роботи відома заздалегідь

2.2.4. Теорія ймовірностей

Теорія ймовірностей — такий розділ математики, що вивчає закономірності випадкових явищ у нашому житті та і в цілому. Моделі, які побудовані в теорії ймовірності описуються з використанням певного ступеня точності, та результати, які ми отримаємо визначаються самими умовами під час випробування. Теорія ймовірностей використовує комбінаторику та теорію міри. Теорія ймовірностей є основою мат. статистики.

Основними об'єктами досліджень є:

- Будь-яка випадкова подія та ймовірність цієї події
- Будь-яка випадкова величина та функція її розподілу

- Будь-який випадковий процес та ймовірнісна характеристика цього процесу.

Маю зазначити, що у теорії ймовірностей випадкову змінну вважають завжди відомою.

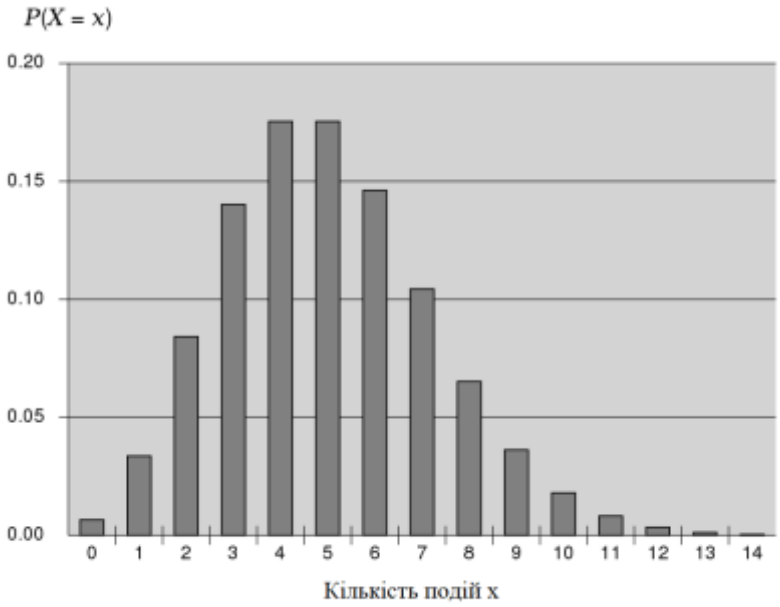

Рис. 2.1. Розподіл Пуассона, дискретний розподіл ймовірностей

Неперервні розподіли ймовірностей. Неперервна теорія ймовірностей вивчає події, що трапляються у неперервному часі.

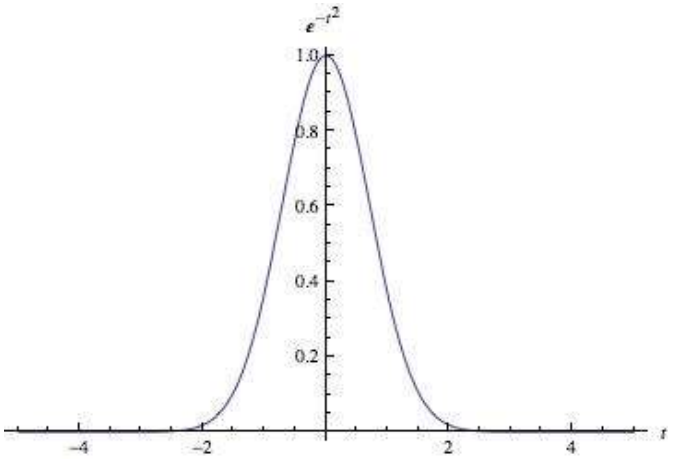

Рис. 2.2. Розподіл Гауса, неперервний розподіл ймовірностей

2.2.5. Теорія рядів

Числовий ряд — числовий ряд є одним із основних понять мат. аналізу. Ряд має вигляд як нескінчена сума чисел. Класифікація рядів:

- Позитивний ряд — такий ряд, у якого всі коефіцієнти цього ряду є невід'ємними.
- Знакозмінний ряд — такий ряд, у якому знаки коефіцієнтів цього ряду чергуються між собою.
- Збіжний ряд

Загальним поняттям ряду є подвійний ряд. В аналізі досліджуються ряди не тільки з чисел, але і з ступеневих рядів, функціональних рядів, рядів Фур'є, рядів Лорана та інші.

2.2.6. Математична статистика

Математична статистика — у такому розділі, на використовуючі вхідні дані вивчається ймовірність закономірностей явищ. Основною задачею мат. статистики є оцінка розподілу ймовірностей та параметрів цього розподілу, дослідження статистичної залежності, вивчення та визначення числових характеристик вибірок, якими є: дисперсія, середнє, стандартне відхилення. Прикладом цього може бути перевірка гіпотез для дослідження питань про зміну процесу з плином часом. Прикладом для оцінювання параметрів є оцінка середнього для статистичної змінної у дослідженні. Для дослідження залежності у статиці використовуються методи кореляції.

Мат. статистика загалом може використовувати методи з теорії ймовірності для дослідження (побудова, перевірка) математичних моделей. Методи дозволяють значно розширити можливості наукового прогнозу і оптимального прийняття рішення для задач, де важливі параметри не можуть бути визначені чи під контролем з гарною точністю.

Методи мат. статистики загалом мають застосування у організації багатьох виробництв, різній радіотехніці, у військових цілях, теоріях напівавтоматичного керування, напрямках біології, економіці тощо. Мат. статистику також можна

використовувати для розв'язання різного роду задач у кібернетиці. Новим напрямком для розвитку мат. статистики вже зараз є лінійний (послідовний) аналіз та теорія прийняття статистичних рішень.

2.2.7. Теорія масового обслуговування

Теорія масового обслуговування — такий розділ теорії ймовірностей, який має за мету раціональний вибір структури системи обслуговування та процесу обслуговування. У теорії масового обслуговування застосовується різні методи із теорії ймовірностей та мат. статистики.

- Однорідний потік. Потік заявок однорідний, якщо всі заявки мають рівні права, розглядаються тільки моменти часу надходження заявок, тобто факти заявок без уточнення деталей кожної конкретної заявки.
- Потік без післядії. Потік без післядії, якщо число подій за будь-який інтервал часу не залежить від числа подій на будь-якому іншому інтервалі часу.
- Стаціонарний потік. Потік заявок стаціонарний , якщо ймовірність появи n подій на інтервалі часу не залежить від часу, а залежить тільки від довжини цієї ділянки.
- Найпростіший потік. Однорідний стаціонарний потік без післядії є найпростішим або пуассонівським потоком.

2.2.8. Імітаційне моделювання

Імітаційне моделювання — це метод, що дозволяє імітувати будь які процеси, які неможливо, або за якихось певних умов не можна виконати. Такого роду модель можна створювати, як і один раз, так і багато разів за наявністю такої потреби та при цьому результати будуть випадковими.

Імітаційне моделювання — являється розділом мат. моделювання. Має існування класифікація об'єктів, для яких по різним причинам на даний момент часу немає аналітичних моделей або не створені методи для розв'язання задач такого роду моделей. Для таких випадків статична модель може бути замінена моделлю, яка імітує поведінку, тобто імітація роботи звичайної моделі.

Імітаційна модель — це опис, який буде використовуватися за допомогою комп'ютера для дослідження, аналітики та оцінки об’єкта, який був змодельований.

Існують такі різновиди імітації:

- метод Монте-Карло
- метод імітаційного моделювання
- ігрове моделювання
- агентне моделювання
- дискретно - подійне моделювання
- системна динаміка

2.2.9. Фізичне моделювання

Фізичне моделювання — метод який вивчає фізичні явища у експериментальному вигляді. Таке моделювання може використовуватися у випадках коли:

- немає математичної моделі явища, або ж ця модель є дуже складною
- відтворення об’єкта у реальних масштабах не є можливим

Цей метод дозволяє створювати таку ж фізичну модель явища, але у зменшеному вигляді, після цього проводяться експерименти за допомогою побудованої моделі. Результати які були одержані з таких випробуваннях переносяться та мають значення для об’єкта у реальному житті. Такий метод може давати гарні результати тільки у випадках з ідентичними особливостями середовища. Дані, які були отримані під час експерименту, мають вплив на існуючий об'єкт з урахуванням всіх критеріїв масштабування. Розглянемо приклади можливого застосування фізичного моделювання:

- Аналіз аеродинаміки різних об’єктів
- Аналіз гідродинамічні показників з імітації моделей кораблів
- Аналіз сейсмічних показників щодо стійкості будівель на етапі їх проектування.
- Дослідження стійкості різного роду конструкцій за участі впливу силових навантажень.
- Вимірювання потоків та розсіювання тепла для систем, що використовуються в умовах значних теплових навантажень.
- Дослідження на аналіз причин стихійних явищ та випливаючих з цього наслідків.

2.3. Системний аналіз існуючих засобів дослідження та вирішення задач зіставлення зображень для передачі інформації з бортового комп’ютера із обґрунтуванням вибору інструментів дослідження

Почнемо з тих випадків, коли взагалі вирішується завдання зіставлення зображень. Можна перерахувати наступні: створення панорам, створення стереопари і реконструкція тривимірної моделі об'єкта по його двовимірним проекція в принципі, розпізнавання об'єктів і пошук за зразком з якоїсь бази, стеження за рухом об'єкта з кількох знімків, реконструкція афінних перетворень зображень. Можливо якесь застосування в цьому списку я упустив або цього застосування ще не придумали, але, напевно, і так вже можна скласти уявлення про те яке коло завдань покликане вирішувати застосування дескрипторів. Слід

зазначити, що область знань, яка розглядає такого роду завдання (комп'ютерний зір) досить молода, з усіма наслідками, що випливають звідси. Немає певного універсального методу, який вирішує всі перераховані вище проблеми в повному обсязі, тобто для всіх вхідних зображень. Однак, не все так погано, просто треба знати, що існують методи вирішення різного роду вужчих завдань, і розуміти, що багато в виборі методу розв'язання задачі залежить безпосередньо від типу самого завдання, типу об'єктів і характеру сцени, на якій вони зображені.

2.3.1. Побудова *SIFT* дескрипторів і завдання зіставлення зображень

Людина може порівняти зображення і виділяти на них об'єкти візуально, на інтуїтивному рівні. Однак, для машини зображення - всього лише набір даних. У загальних рисах можна описати два підходи, щоб машина хоча б змогла б це зробити на мінімальному рівні. Існують методи для порівняння зображень, які в основі працюють на зіставленні знань про ціле зображення.

В основному у загальних випадках це може виглядати таким чином: для кожної точки на зображенні певним чином обчислюється значення деякої функції, після цього на підставі значень з обчислення можна створити для зображення характеристику і тоді задача з порівняння двох зображень зводиться до задачі з порівняння характеристик зображень, які досліджуються. Ці методи настільки ж не ефективні, наскільки й прості та працюють вони майже тільки в ідеальних ситуаціях. Причин для цього може бути більш ніж достатньо: поява нових об'єктів на зображенні, перекриття одних об'єктів іншими, шуми, зміни масштабу, положення об'єкта на зображенні, положення камери в тривимірному просторі, висвітлення, аффінні перетворення і т.д. Власне, погані якості цих методів обумовлені їх основною ідеєю, тобто тим, що в характеристику вносить вклад кожна точка зображення, яким би поганим цей вклад не був. Тому треба якимось чином обійти ці проблемт: треба вибирати точки, що мають значний вплив на характеристику, або ж ще краще, виділяти деякі особливі (ключові) точки і порівнювати їх.

На цьому ми і підійшли до ідеї зіставлення зображень по ключових точках. Можна сказати, що ми замінюємо зображення якоюсь моделлю — набір

ключових точок цього зображення. Ото ж одразу можна відзначити, що ключовою буде називатися така точка об'єкта, яка з великою часткою ймовірності може бути знайдена на іншому зображенні. Детектором будемо називати метод вилучення особливих (ключових) точок з зображення. Детектор має забезпечувати багато варіантів для знаходження схожих ключових точок щодо досліджуваних зображень. Єдино, що залишається незрозумілим - яким чином визначати яка ключова точка одного зображення відповідає ключовій точці іншого зображення. Адже після застосування детектора можна визначити тільки координати особливих точок, а вони на кожному зображенні різні. Тут в справу і вступають дескриптори. Дескриптор - ідентифікатор ключової точки, що виділяє її з іншої маси ключових точок. У свою чергу, дескриптори повинні забезпечувати варінти для знаходження відповідності між особливими точками щодо перетворень зображень. Але є питання, що залишається не дуже зрозумілим — який спосіб треба використовувати для визначення того яка ключова точка першого зображення майже відповідає точці другого зображенняю. Адже після застосування детектора можна визначити тільки координати особливих точок, а вони на кожному зображенні різні. Тут в справу і вступають дескриптори. Дескриптор - ідентифікатор ключової точки, що виділяє її з іншої маси ключових точок. У свою чергу, дескриптори повинні забезпечувати інваріантність знаходження відповідності між особливими точками щодо перетворень зображень.

У підсумку виходить наступна схема рішення задачі зіставлення зображень:

- На зображеннях виділяють особливі точки та відповідні їм дескриптори.
- За збігом дескрипторів виділяються відповідні один одному ключові точки.
- На основі набору ключових точок, що мають відповідність будується модель для майбутніх перетворень зображень, за допомогою якої з вихідного зображення можна отримати результуюче зображення.

На кожному з цих етапів є свої специфічні питання, проблеми та різні способи їх вирішення, що вносить певне свавілля для вирішення задачі, яка була поставлена спочатку. Далі будемо розглядати першу частину рішення, а саме виділення особливих точок і їх дескрипторів методом *SIFT*. В основному буде описаний алгоритм методу *SIFT*, а не те, чому цей алгоритм працює, і виглядає саме так.

Наостанок перелічимо перетворення, для яких можна було б отримати інваріантність:

- поворот зображення
- зміщення зображення
- зміна положення фотокамери
- зміна яскравості зображення
- зміна масштабу зображення

2.3.2. Знаходження особливих точок для зображень

Основним моментом у знаходженні особливих точок є побудова піраміди гауссіанів (*Gaussian*) і різниць гауссіанів (*DoG*). У двох словах скажемо про масштабовані простори. Масштабованим простором зображення є набір всіляких, згладжених деяким фільтром, версій вихідного зображення. Доведено, що гауссово масштабується простір є лінійним, інваріантним щодо зрушень, обертань, масштабу, що не зміщує локальні екстремуми, і має властивість напівгруп. Для нас важливо, що різна ступінь розмиття зображення гаусовим фільтром може бути прийнята за вихідне зображення, взяте в деякому масштабі.

В цілому багатоваріантність для масштабу може досягається за допомогою визначення особливих точок для початкового зображення, яке було представлене в різних масштабах. Для цього треба побудувати піраміду гауссіанів: масштабований простір розбивається на визначенні ділянки (октави), при тому що частина простору, яка займає наступна октава, є в два рази більшою частини, що займає попередня октава. Також коли робиться перехід від однієї октави до іншої то проводиться ресемплінг зображення, розмір зображення зменшується у

два рази. Кожна октава може охоплювати нескінчену кількість гауссіанів цього зображення, тому для цього береться тільки деяка кількість N, а також визначається крок радіусу розмиття. З цим кроком створюються два інших додаткових гауссіана ($N + 2$), та вони не містяться в октаві. Далі ми зможемо побачити для чого це потрібно. Масштаб першого зображення октави що йде наступною дорівнює тому самому масштабу зображення з попередньої октави яка має номер N. На цьому ж кроці з побудовою піраміди гауссіанів, створюється піраміда різниць гауссіанов, що містить в собі різні сусідні зображення в піраміді гауссіанів. Відповідно, кількість зображень в цій піраміді буде $N + 1$.

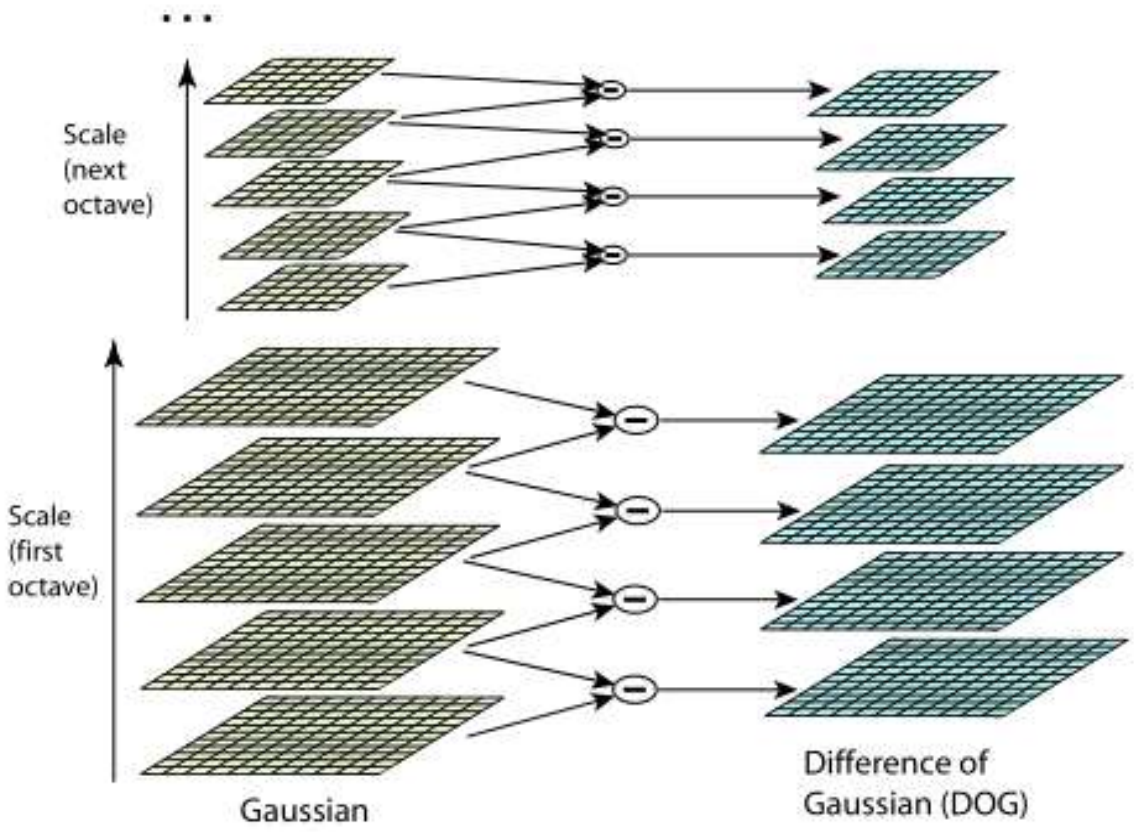

Рис. 2.3. Піраміда гауссіанів та їх різниця

На рисунку (зліва) побудована піраміда гауссіанів, а праворуч різниця цих пірамід. Схематично зображено, що кожна різниця виходить з двох сусідніх гауссіанів, а кількість різниць на вже на 1 менше кількості початкових гауссіанів та після переходу до наступної октави розмір зображень зменшується у два рази. У кожному зображенні з піраміди *DoG* шукаються точки локального

екстремуму. Кожна точка даного зображення *DoG* має порівняння з її сусідами і з сусідами в *DoG*, що знаходяться на рівень вище і нижче в піраміді. Якщо ця точка більше (менше) всіх сусідів, то вона приймається за точку локального екстремуму.

На попередньому пункті пошук ключових точок не закінчено. Наступним кроком буде декілька перевірок на придатність точки екстремуму щоб вона мала роль ключової. На початку визначаються координати ключової точки з підпіксельною точністю. Це можна досягти за допомогою використання апроксимації функції *DoG* многочленом Тейлора другого порядку, який був взятий в точці обчислення екстремуму.

D - функція *DoG*, *X* = (*x*, *y*, *sigma*) — є вектор певного зміщення відносно точки розподілу, перша похідна *DoG* — це градієнт, друга похідна *DoG* - матриця Гессе. Екстремум многочлена Тейлора може визначатися шляхом визначення похідної та прирівняння до нуля. У результаті цього можна отримати зсув точки екстремуму що був обчислений. У підсумку отримаємо СЛАР яке має розмірність 3*x*3, щодо вектора *X* ^. Для сусідньої точки все повторюється заново. Якщо вдалося вийти за межі октави, то слід виключити дану точку з розгляду. Нарешті, остання перевірка. Якщо особлива точка лежить на границі досліджуваного об'єкта або вона може бути погано освітлена, то таку точку можна виключити з розгляду. Ці точки можуть мати великий вигин вздовж границі і малий в перпендикулярному напрямку. Цей великий вигин визначається матрицею Гессе H.

2.3.3. Знаходження орієнтації особливих точок для зображень

Після переконання того, що точка є ключовою, потрібно обчислити її напрямок на зображенні. Така точка може мати не один напрямок. Напрямок ключової точки можна обчислити за допомогою напрямів градієнтів цих точок, сусідніх з ключовою. Усі обчислення проводяться в піраміді гауссіанів, з масштабом який дуже близький до масштабу ключової точки. Насамперед треба визначити околицю ключової точки, в якому будуть досліженні майбутні градієнти. Для гауссового ядра діє так зване правило «трьох сигм». Воно полягає в тому, що значення гауссова ядра дуже близько до нуля на відстані, що більше за 3 * *sigma*. Ото ж радіус вікна обчислюється як 3 * *sigma*. Напрямок ключової точки можна знайти за допомогою гістограми напрямків О. Ця гістограма напрямків містить 36 компонентів, які рівномірно забезпечують покриття в 360 градусів, і формується вона в такий спосіб: точка вікна (*x*, *y*) має значення, рівному *m* * *G* (*x*, *y*, *sigma*), в той компонент гістограми напрямків, яка має покриття проміжку, що маж напрям градієнта *theta* (*x*, *y*). Напрямок ключовий точки лежить в проміжку, що покривається максимальною компонентою гістограми. Значення максимальної компоненти (*max*) і двох сусідніх компонент інтерполюються, і точка максимуму для обчисленої параболи визначається як напрямок ключової точки. Якщо ж в гістограмі напрямків містяться компоненти які мають значення не менше ніж 0.8 * *max*, то вони також піддаються впливу інтерполяції і після цього обчислені напрями додаються до ключової точки.

2.3.4. Побудова дескрипторів

Тепер перейдемо безпосередньо до дескрипторів. Дане раніше визначення говорить про те, що повинен робити дескриптор, але не про те, що це таке. В принципі, дескриптором може виступати будь-який об'єкт (аби він справився зі своїми функціями), але зазвичай дескриптором є якась інформація про околиці ключовий точки. Цей вибір зроблений по декількох причинах: маленькі області мають менший вплив ефектів спотворень, зміни взагалі можуть не впливати на дескриптор зображення.

Для так зваого *SIFT* у ролі дескриптора виступає вектор. Як і напрямок ключової точки, дескриптор визначається на гауссіані, який розташований найближче до ключової точки. Перед обчисленням дескриптора це вікно повертають на кут напрямку ключовий точки, чим і досягається інваріантність щодо повороту.

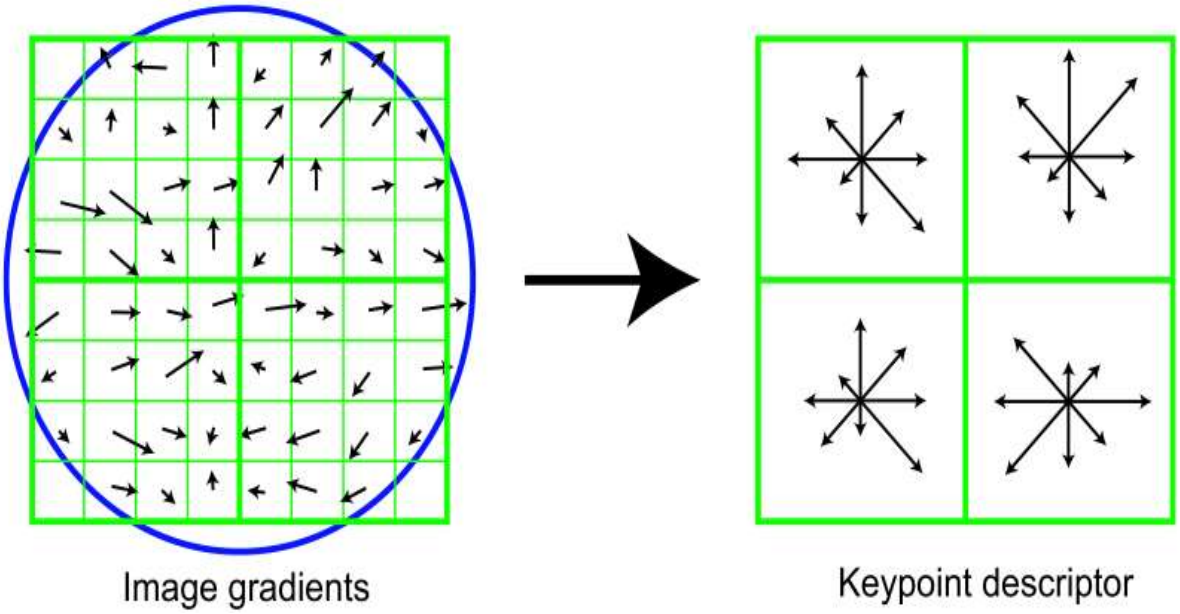

Рис. 2.4. Схематична частина зображення та її дескриптор

На зображені зліва можна побачити пікселі, які позначені маленькими квадратами. Ці пікселі мають походження з вікна дескриптора, і це вікно поділене на чотири рівні частини (регіони). Маленька стрілка, в центрі кожного пікселя позначає градієнт цього пікселя. Цікаво те, що центр цього вікна знаходиться між пікселями. Його треба вибирати якомога ближче до точних координат ключовий точки. Останнє що можна побачити зліва - це коло, що позначає вікно згортки яке містить гаусове ядро. Для цього ядра обчислюється *sigma*, рівну половині ширини вікна обчислюваного дескриптора. Надалі значення кожної точки вікна дескриптора буде помножуватися на значення гауссова ядра в цій точці, як на ваговий коефіцієнт. На зображені зправа можна побачити дескриптор ключової точки, яка має розмірність 2*x*2*x*8. 2 в значенні розмірності - це кількість регіонів які розташовані по горизонталі та вертикалі. Квадрати, які охоплюють визначений регіон з пікселів на лівому зображені,

праворуч вже охоплює гістограми напрямів, які побудовані на пікселях регіонів. 8 в розмірності дескриптора визначає кількість компонентів гістограми для цих регіонів.

Гістограми напрямків в регіонах обчислюються:

- Кожна гістограма має покривати певну ділянку в 360 градусів, але ділити її на 8 частин
- В якості вагового коефіцієнта береться значення ядра гаусса, яке є загальним для дескриптора
- В якості вагомих коефіцієнтів є коефіцієнти трилінійної інтерполяції.

Кожному градієнту в вікні дескриптора можна приписати три речові координати (x, y, n), де x - відстань до градієнта по горизонталі, y - відстань по вертикалі, n - відстань до напрямку градієнта в гістограмі (мається на увазі гістограма напрямків дескриптора яка є відповідною до тої на яку має вплив цей градієнт). В якості точки відліку береться нижній лівий кут вікна дескриптора і початкове значення гістограми. За поодинокі відрізки беруться розміри регіонів які розташовані горизонтально і вертикалі для x і y відповідно, і кількість градусів в компоненті гістограми для n. Коефіцієнт трилінійної інтерполяції визначається для кожної координати (x, y, n) градієнта як 1-d, де d дорівнює відстані від координати градієнта до середини того одиничного проміжку в який ця координата потрапила. Кожне входження градієнта в гистограмму множиться на три вагових коефіцієнта використованої трилінійної інтерполяції. Дескриптор ключової точки створюється з всіх отриманих раніше гістограм напрямків. Як було наведено у прикладі, розмірність дескриптора на рисунку складається з 32 компонентів (2x2x8), але загалом під час практичи мають використання дескриптори, які мають розмірність 128 компонентів (4x4x8). І тільки після таких маніпулювань дескриптори для зіставлення зображень можуть бути використані у подальших дослідженнях.

Висновки до розділу

У даному розділі розглянуто показники ефективності інструментарію. Такими показниками є: конструкція проходження, конструкція розгалуження, конструкція циклу, трудомісткість та наглядність розрахунків. В моїй дипломній роботі найбільш придатними інструментами для дослідження є: алгебри (в тому числі булева), теорія графів, математичне програмування, теорія ймовірностей, теорія рядів, математична статистика, теорія масового обслуговування, імітаційне моделювання та фізичне моделювання. В якості показника ефективності інструментарію визначено трудомісткість та наглядність розрахунків. Також були розглянуті *SIFT* дескриптори. *SIFT* дескриптори не позбавлені недоліків. Не всі отримані точки і їх дескриптори будуть відповідати вимогам, що пред'являються. Природно це буде позначатися на подальшому вирішенні завдання зіставлення зображень. У деяких випадках рішення може бути не знайдено, навіть якщо воно існує. Наприклад, при пошуку афінних перетворень (або фундаментальної матриці) за двома зображенням цегляної стіни може бути не знайдено рішення через те, що стіна складається з повторюваних об'єктів (цегли), які роблять схожими між собою дескриптори різних ключових точок. Незважаючи на цю обставину, дані дескриптори добре працюють в багатьох практично важливих випадках. Тож в результаті аналізу в якості інструментарію дослідження обрано теорію ймовірностей, математичну статистику та теорію рядів, а також побудову дескрипторів зіставлення зображень для оцінки їх візуальної вірності.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ СПОСОБУ ЗБЕРІГАННЯ ТА ПЕРЕДАЧІ ЗОБРАЖЕННЯ З МЕТОЮ ВИЗНАЧЕННЯ ГРАНИЦЬ ЙОГО ЕФЕКТИВНОГО ЗАСТОСУВАННЯ

3.1. Розробка структурної схеми алгоритму дослідження

3.1.1. Основні елементи структурної схеми алгоритму

Блок-схема - представлення алгоритму розв'язування або аналізу задачі за допомогою геометричних елементів (блоків), які позначають операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних — прямокутником, блок прийняття рішень — ромбом, еліпсом — початок та кінець алгоритму.

Основні елементи схем алгоритму:

- **Термінатор** - елемент відображає вхід у зовнішнє середовище або вихід з нього (найчастіше застосування - початок і кінець програми). Всередині фігури записується відповідна дія.

Рис. 3.1. Термінатор

- **Процес** - елемент відображає одну або кілька операцій, обробку даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рис. 3.2. Процес

- **Рішення -** елемент відображає обробку умови, рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижньої). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижньої) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Рис. 3.3. Рішення

- **Зумовлений процес** - елемент відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Рис. 3.4. Зумовлений процес

- **Дані** - елемент відображає перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

Рис. 3.5. Дані

- **Цикл з параметром** - елемент відображає заголовок циклу з параметром. У ньому через крапку з комою вказуються ім'я змінної (параметра) з початковим значенням, граничне значення параметра (або умова виконання циклу), крок зміни параметра.

Рис. 3.6. Цикл з параметром

- **Межа циклу** - елемент складається з двох частин - відповідно, початок і кінець циклу - операції, що виконуються всередині циклу, розміщуються між ними. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу. Часто для зображення на блок-схемі циклу замість цього символу використовують символ рішення, вказуючи в ньому умову, а одну з ліній виходу замикають вище в блок-схемі (перед операціями циклу).

Рис. 3.7. Межа циклу

- **З’єднувач** - елемент відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Рис. 3.8. З’єднувач

- **Коментар** - елемент використовується для детальнішої інформації про кроки, процесу або групи процесів. Опис поміщається з боку квадратної дужки і охоплюється нею по всій висоті. Пунктирна лінія йде до описуваного елементу, або групи елементів (при цьому група виділяється замкнутою пунктирною лінією). Також символ коментаря слід використовувати в тих випадках, коли обсяг тексту у будь-якому іншому символі перевищує його обсяг.

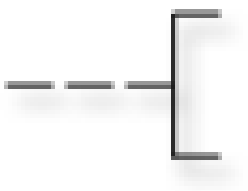

Рис. 3.9. Коментар

3.1.2. Структурна схема алгоритму дослідження

Рис. 3.10. Структурна схема алгоритму дослідження

3.2. Реалізація дослідження на предмет визначення впливу спектральних характеристик зображення на інтенсивність потоку похибок при передачі зображення

3.2.1. Перший дослід зображення

В досліджені, як було вище зазначено, буде використано один із етапів стиснення зображення у формат *JPEG*, а саме *DCT*. Було вирішено провести дослідження обрізаючи зображення до роздільної здатності 8 на 8 пікселів та з кольоровим форматом *grayscale*.

Зображення для цього досліду:

Рис. 3.11. Перше зображення для досліду

Для початку завантажимо зображення для подальшої роботи з ним.

```
def get_image(size=(8, 8)):
    image = Image.open('woman.png')
    img_color = image.resize(size, 1)
    img_grey = img_color.convert('L')
    img = np.array(img_grey, dtype=np.float)
    return img
```

Рис. 3.12. Функція для завантаження зображення

На початку функції завантажимо зображення для дослідження та визначаємо кольорову модель зображення. Конвертуємо наше зображення у кольоровий режим зображення *grayscale* (відтінки сірого), у ході програми ми будемо працювати із зображенням саме у цьому кольоровому режимі. В кінці функції

конвертуємо зображення у двовимірний масив. Нижче приведений фрагмент двовимірної матриці після попередніх перетворень.

	0	1	2	3	4	5
0	84.0	85.0	98.0	116.0	98.0	97.0
1	81.0	76.0	90.0	150.0	167.0	107.0
2	82.0	91.0	104.0	104.0	135.0	164.0
3	84.0	93.0	68.0	56.0	136.0	107.0
4	96.0	61.0	48.0	104.0	130.0	79.0
5	92.0	55.0	63.0	81.0	120.0	57.0
6	85.0	55.0	54.0	53.0	123.0	94.0
7	82.0	37.0	53.0	77.0	125.0	163.0

Рис. 3.13. Фрагмент двовимірної матриці

Наступним кроком є використання *DCT* на двовимірній матриці. Кожен коефіцієнт досліджуваної матриці підлягає *DCT*. Цей крок виконується за допомогою програми, тобто автоматично. *DCT* застосовується до блоків матриці розмірності 8 на 8 елементів.

	0	1	2	3
0	755.5	-93.49278168979802	-50.26819535654379	62.1004052140
1	44.76925885258256	55.582908299115296	-60.986856207948875	-15.84732646184
2	14.774106857318985	-19.422344342136533	-21.59188309203678	15.19318750866
3	-24.01095858097789	4.360312271403707	46.75648849882036	-55.2696401870
4	-17.000000000000004	-1.1000663490792828	16.724335898976676	6.524139035268
5	-13.799250340090838	-4.225478341012166	50.89562510876386	-15.28009567975
6	16.069404673743207	-11.589391550869863	13.303300858899105	-5.10315395766
7	12.610234726781467	-8.216177762551078	0.12211727245620452	6.213018502529

Рис. 3.14. Фрагмент двовимірної матриці після *DCT*

Для того щоб проаналізувати похибку *DCT* відновимо початкову матрицю за допомогою оберненого *DCT*.

	0	1	2	3
0	84.00000000000003	85.00000000000003	98.00000000000003	116.00000000000003
1	80.99999999999999	76.0	90.0	150.0
2	82.0	91.0	104.0	104.00000000000003
3	84.00000000000001	93.00000000000001	68.00000000000001	56.000000000000014
4	96.00000000000006	61.000000000000014	48.000000000000014	104.00000000000003
5	92.00000000000003	54.99999999999999	63.00000000000001	81.00000000000001
6	85.0	55.000000000000014	54.000000000000014	52.99999999999999
7	82.00000000000003	37.000000000000036	53.00000000000002	77.0

Рис. 3.15. Фрагмент відновленої матриці

Як можна побачити, під час перетворення програмний код заокруглював кожен коефіцієнт, тому якщо округлити кожен із коефіцієнтів до двох символів

після коми і порівняти початкову матрицю і відновлену, то можна побачити, що коефіцієнти не змінилися попри те, що для дослідження було використане зображення з використанням *blur*.

3.2.2. Другий дослід зображення

Зображення для другого досліду:

Рис. 3.16. Перше тестове зображення

Рис. 3.17. Друге тестове зображення

Рис. 3.18. Третє тестове зображення

Завантажимо зображення для подальшої роботи з ним.

```
def get_image():
    image = Image.open('test_images/1_full.png')
    img_grey = image.convert('L')
    img = np.array(img_grey, dtype=np.float)
    return img
```

Рис. 3.18. Завантаження першого зображення

```
def get_image():
    image = Image.open('test_images/4_full.png')
    img_grey = image.convert('L')
    img = np.array(img_grey, dtype=np.float)
    return img
```

Рис. 3.19. Завантаження другого зображення

```
def get_image():
    image = Image.open('test_images/7_full.png')
    img_grey = image.convert('L')
    img = np.array(img_grey, dtype=np.float)
    return img
```

Рис. 3.20. Завантаження третього зображення

На початку функції завантажуються зображення з роздільною здатністю зображень:

- Перше тестове зображення 1472 на 870 пікселі
- Друге тестове зображення 1342 на 757 пікселі
- Третє тестове зображення 1305 на 763 пікселі

Конвертуємо тестові зображення у кольоровий режим зображення *grayscale* (відтінки сірого), у ході програми ми будемо працювати із зображенням саме у цьому кольоровому режимі. В кінці функції перетворюємо зображення у двовимірний масив. Нижче приведено фрагмент двовимірної матриці після описаних перетворень.

	0	1	2	3	4	5	6	7
0	99.00000	77.00000	53.00000	56.00000	51.00000	49.00000	53.00000	53.00000
1	93.00000	71.00000	59.00000	66.00000	63.00000	64.00000	67.00000	58.00000
2	82.00000	63.00000	68.00000	74.00000	75.00000	74.00000	75.00000	62.00000
3	34.00000	40.00000	72.00000	69.00000	69.00000	68.00000	66.00000	59.00000
4	41.00000	52.00000	82.00000	67.00000	66.00000	63.00000	57.00000	53.00000
5	64.00000	68.00000	78.00000	66.00000	62.00000	62.00000	52.00000	53.00000
6	79.00000	53.00000	58.00000	56.00000	60.00000	57.00000	51.00000	54.00000
7	112.00000	79.00000	67.00000	48.00000	71.00000	93.00000	61.00000	54.00000
8	106.00000	87.00000	70.00000	43.00000	82.00000	106.00000	73.00000	54.00000
9	63.00000	54.00000	46.00000	94.00000	96.00000	81.00000	90.00000	58.00000
10	87.00000	54.00000	49.00000	100.00000	95.00000	75.00000	66.00000	77.00000
11	111.00000	86.00000	71.00000	97.00000	94.00000	78.00000	69.00000	81.00000
12	124.00000	109.00000	89.00000	83.00000	81.00000	84.00000	111.00000	69.00000
13	108.00000	66.00000	57.00000	77.00000	80.00000	71.00000	88.00000	66.00000
14	90.00000	49.00000	41.00000	87.00000	81.00000	64.00000	74.00000	67.00000
15	85.00000	36.00000	36.00000	82.00000	65.00000	46.00000	76.00000	68.00000
16	83.00000	34.00000	29.00000	50.00000	43.00000	37.00000	52.00000	77.00000
17	98.00000	49.00000	30.00000	43.00000	41.00000	38.00000	40.00000	69.00000
18	107.00000	74.00000	32.00000	42.00000	46.00000	42.00000	41.00000	56.00000
19	120.00000	73.00000	32.00000	43.00000	60.00000	60.00000	34.00000	41.00000

Рис. 3.21. Фрагмент двовимірної матриці для першого тестового зображення

	0	1	2	3	4	5	6	7
0	97.00000	133.00000	107.00000	116.00000	117.00000	125.00000	96.00000	105.00000
1	91.00000	154.00000	125.00000	124.00000	115.00000	130.00000	107.00000	110.00000
2	97.00000	125.00000	146.00000	142.00000	144.00000	108.00000	100.00000	104.00000
3	93.00000	108.00000	149.00000	161.00000	89.00000	104.00000	87.00000	87.00000
4	96.00000	126.00000	141.00000	122.00000	124.00000	96.00000	107.00000	99.00000
5	98.00000	119.00000	138.00000	121.00000	109.00000	131.00000	155.00000	97.00000
6	111.00000	98.00000	93.00000	100.00000	138.00000	147.00000	94.00000	66.00000
7	121.00000	97.00000	128.00000	116.00000	114.00000	144.00000	91.00000	94.00000
8	134.00000	99.00000	126.00000	116.00000	98.00000	103.00000	118.00000	103.00000
9	149.00000	119.00000	115.00000	137.00000	129.00000	107.00000	138.00000	102.00000
10	121.00000	102.00000	112.00000	128.00000	115.00000	98.00000	110.00000	134.00000
11	87.00000	120.00000	130.00000	98.00000	96.00000	133.00000	154.00000	156.00000
12	125.00000	138.00000	130.00000	100.00000	117.00000	146.00000	140.00000	106.00000
13	111.00000	106.00000	134.00000	135.00000	122.00000	141.00000	131.00000	98.00000
14	115.00000	113.00000	99.00000	101.00000	136.00000	142.00000	130.00000	94.00000
15	108.00000	114.00000	99.00000	118.00000	116.00000	97.00000	130.00000	96.00000
16	114.00000	146.00000	115.00000	128.00000	96.00000	110.00000	146.00000	128.00000
17	125.00000	129.00000	118.00000	117.00000	109.00000	138.00000	136.00000	124.00000
18	113.00000	107.00000	113.00000	115.00000	104.00000	107.00000	109.00000	116.00000
19	97.00000	110.00000	116.00000	103.00000	95.00000	106.00000	122.00000	114.00000

Рис. 3.22. Фрагмент двовимірної матриці для другого тестового зображення

	0	1	2	3	4	5	6	7
0	121.00000	136.00000	147.00000	137.00000	147.00000	126.00000	134.00000	135.00000
1	124.00000	130.00000	116.00000	113.00000	121.00000	120.00000	126.00000	102.00000
2	123.00000	99.00000	142.00000	146.00000	142.00000	115.00000	132.00000	125.00000
3	156.00000	132.00000	126.00000	114.00000	142.00000	144.00000	98.00000	110.00000
4	145.00000	149.00000	140.00000	119.00000	137.00000	137.00000	108.00000	124.00000
5	127.00000	142.00000	148.00000	130.00000	140.00000	140.00000	137.00000	138.00000
6	140.00000	151.00000	155.00000	123.00000	121.00000	122.00000	141.00000	146.00000
7	131.00000	133.00000	152.00000	133.00000	143.00000	128.00000	126.00000	148.00000
8	126.00000	129.00000	142.00000	137.00000	138.00000	150.00000	128.00000	140.00000
9	128.00000	139.00000	167.00000	170.00000	114.00000	130.00000	124.00000	140.00000
10	125.00000	119.00000	171.00000	177.00000	134.00000	161.00000	154.00000	151.00000
11	131.00000	137.00000	153.00000	157.00000	135.00000	160.00000	140.00000	166.00000
12	133.00000	135.00000	142.00000	152.00000	137.00000	159.00000	171.00000	139.00000
13	114.00000	114.00000	110.00000	124.00000	138.00000	142.00000	148.00000	143.00000
14	118.00000	116.00000	119.00000	122.00000	124.00000	135.00000	151.00000	148.00000
15	112.00000	117.00000	126.00000	132.00000	159.00000	147.00000	136.00000	151.00000
16	139.00000	121.00000	140.00000	157.00000	170.00000	147.00000	150.00000	154.00000
17	131.00000	128.00000	135.00000	138.00000	164.00000	153.00000	166.00000	154.00000
18	139.00000	139.00000	145.00000	110.00000	124.00000	131.00000	154.00000	156.00000
19	142.00000	143.00000	137.00000	132.00000	129.00000	119.00000	136.00000	137.00000

Рис. 3.23. Фрагмент двовимірної матриці для третього тестового зображення

Наступним кроком є використання *DCT*. Кожен коефіцієнт досліджуваної матриці підлягає *DCT*. Цей крок виконується за допомогою програми, тобто автоматично. *DCT* застосовується до блоків матриці розмірності 8 на 8 елементів, двовимірна матриця буде поділена на однакові блоки розмірністю 8 на 8 елементів.

	0	1	2	3	4	5	6	7
0	178023.79616	-30842.29971	2697.32576	-7027.93311	-7110.17096	2694.90558	1647.93307	-2268.56536
1	-19973.92268	2671.52205	16581.44051	5593.75021	3495.01790	4194.95863	1246.88384	-4068.33592
2	38.79947	3684.36962	-553.71756	4720.79723	-345.52032	-7952.00714	-4139.93171	-3406.01626
3	-4485.69242	5084.99200	1648.40060	1397.65060	711.14270	35.74718	3535.45431	3730.89678
4	-4784.40556	-1003.59051	109.58924	-1967.36092	-168.93661	59.90436	1331.34934	2932.38550
5	3155.93894	-1461.47703	-3924.73817	-2392.77434	-1557.43332	1474.15356	2101.49934	995.50832
6	-2033.65661	55.07027	834.34190	-447.48812	-1728.82100	2748.68676	840.02797	-1223.15261
7	1215.67401	283.05843	2781.59864	-267.00020	-82.99690	2370.28736	-1638.10252	-4160.42654
8	2436.03472	-1766.86327	1701.30029	4105.75005	987.03742	188.80581	111.14162	1417.86945
9	-2430.20068	-2182.57531	-931.62475	-834.80802	-3269.07971	-1893.67801	878.31496	1422.82320
10	-2092.12402	-2453.82392	-912.07260	-2189.54077	-1478.60092	-1929.77079	-2034.39257	-282.10045
11	3780.51696	473.37833	-1021.69512	1225.58584	1636.20330	116.83371	-1885.99214	-1106.65049
12	839.62522	856.28904	-595.61382	2250.32282	1573.01315	957.63457	1618.46529	-452.31207
13	-1306.23679	-1322.26473	-1114.41168	-1247.81425	-1844.56851	-1061.21389	2019.16157	894.60409
14	-335.63210	-1590.86720	664.16902	1242.02174	-2132.16700	-1540.91608	798.19647	1236.75957
15	458.35830	423.22939	111.62462	951.67302	-586.66159	-173.97098	227.30080	426.78538
16	-668.02397	-618.95852	-950.06673	-595.80482	493.82679	-464.02092	-574.29884	228.71933
17	1343.18164	-117.77090	-412.78424	703.05116	1250.94980	179.17081	-233.99152	764.31111
18	784.86637	926.37232	-332.41247	-826.52339	-640.29701	-78.90876	-42.07171	-39.09119
19	-1111.57101	183.77125	93.23400	-1167.39919	88.56200	466.56370	-60.90028	-975.30380

Рис. 3.24. Фрагмент двовимірної матриці першого тестового зображення після *DCT*

	0	1	2	3	4	5	6	7
0	109464.47621	-1646.21546	438.21890	-1723.31813	-5707.86375	8.69617	-1211.17503	-335.51406
1	7405.85959	19433.66935	-1596.12569	-6050.53467	931.83141	-1632.45753	-1170.18420	4500.28476
2	-6357.63004	2942.50781	18411.54237	1594.53627	-1692.56757	-871.15650	-2052.49894	-2164.30487
3	-5914.54089	-7744.70630	-3265.68671	2355.34134	4148.84336	1234.60431	-1209.06641	-1659.83468
4	-707.90431	-971.53643	-932.91567	-4045.50511	-2954.97408	-1021.98165	4098.72577	823.85770
5	-2955.89976	2768.64975	2057.85335	3718.66987	-1229.54988	-1961.55403	-391.04715	1524.99077
6	1328.18073	-1316.16882	-2074.15984	1606.87633	3071.70317	568.36666	-2226.28771	-969.63377
7	-314.12119	235.76792	-2578.74695	-1115.26056	-447.97979	1270.37446	995.68432	-225.08012
8	-2960.15530	-56.50997	1835.04000	-646.31426	-1067.89406	-171.92948	441.14713	-544.57515
9	1167.41113	-48.84006	488.73275	192.96528	-583.52647	-1500.52966	597.93760	-804.32036
10	-554.95050	240.03046	335.56909	297.51449	-566.27372	1178.59174	855.99914	-106.15929
11	-28.59881	-1212.66699	247.07136	-215.22362	967.04326	-22.47367	-490.63552	450.89296
12	342.32885	641.64825	-651.46524	31.59282	-971.27176	40.06330	-1710.42601	-749.30324
13	-891.65982	-436.36508	-202.42485	457.25746	-23.64218	437.19164	-417.48092	-87.80685
14	463.27974	-833.91242	869.30041	182.14579	353.69602	-664.69380	-600.43059	1029.94790
15	-317.55149	343.93581	409.35455	462.48798	131.39393	654.24522	-131.59619	721.45924
16	-230.77033	232.50293	1045.27490	-721.90460	13.15248	-141.63463	208.58568	-411.60867
17	11.20575	80.26666	-419.99206	196.40814	-1233.29727	491.43234	-496.31635	-395.92312
18	-200.88267	-386.24072	-972.53688	323.88400	-173.85623	696.73907	-182.54259	43.89537
19	737.12658	59.61059	725.92609	158.60664	-191.93287	-97.85502	128.59107	673.05676

Рис. 3.25. Фрагмент двовимірної матриці другого тестового зображення після *DCT*

	0	1	2	3	4	5	6	7
0	127324.02257	1469.89664	-6072.93801	2913.21584	-6336.99662	-2376.95893	-628.02094	-463.71727
1	-2520.43843	8152.07246	-1655.60799	-2187.15953	-1237.85259	3203.38270	-2682.86711	-1683.69175
2	-10719.78888	1894.52633	5076.73234	6380.31067	-5102.28744	-748.53067	462.51118	-2057.64814
3	-3066.15100	1148.02381	1866.65783	5894.79256	1003.76068	-3686.68337	-62.87871	-747.48670
4	-121.77610	1154.18919	-1243.42677	190.46148	2099.04650	1179.31137	-1982.58581	-3811.67465
5	-1692.76948	30.39379	-353.44045	1398.94440	194.48311	1168.31353	2282.15014	535.44461
6	-876.80021	-1718.58750	184.06296	-545.30861	320.05197	740.07189	283.05872	486.48885
7	-165.53897	648.12514	-663.75219	-2046.44244	-1918.93929	1756.47191	2900.40580	858.13512
8	194.24369	822.30111	971.75038	1059.59300	1062.58470	-270.19579	47.94243	1349.43848
9	-874.99032	-1123.66481	-331.78195	265.99384	673.01890	-408.05693	-1723.53056	157.67057
10	-1218.83573	717.50901	1711.43674	444.54225	167.62109	1154.48865	846.89298	35.81141
11	77.72313	349.74668	-272.64996	-516.82740	420.37685	-636.61198	362.80872	-198.32371
12	-1108.30411	-577.24889	-590.11888	292.82841	678.91693	-379.47481	38.64535	631.84827
13	-216.88912	115.58036	-21.63798	-24.44951	696.05766	-283.52557	305.60855	-921.11507
14	-860.62743	-69.00084	466.22433	-197.89155	-157.52374	-782.20067	-315.02443	-362.47182
15	-1.63578	-199.97611	-31.54885	-355.62428	-234.42307	146.92904	416.53484	-63.57249
16	64.00548	-268.19478	-14.23143	-173.70408	-424.38370	-240.49850	4.02416	-107.69765
17	-111.50075	159.97201	-342.80166	-63.78007	-199.62584	-87.51569	-648.14853	-252.23613
18	820.76240	735.56975	609.57126	-25.94762	454.39423	299.14214	858.10396	803.12990
19	-222.66153	-123.80092	-117.87240	31.10139	-899.27764	-381.51666	-284.77385	-520.36689

Рис. 3.26. Фрагмент двовимірної матриці третього тестового зображення після *DCT*

Для того щоб проаналізувати результати після перетворень відновимо початкову матрицю за допомогою оберненого *DCT*.

	0	1	2	3	4	5	6	7
0	99.26544	75.65394	55.99862	52.43066	52.77374	50.23313	50.29967	54.35255
1	90.68513	74.45483	60.02456	60.59416	66.82837	65.83055	62.37157	59.96115
2	77.41849	71.16955	65.55648	69.79610	78.08937	77.11251	70.01326	62.22961
3	29.25050	49.73632	65.51588	69.32200	70.62987	68.99620	63.16642	59.94563
4	38.06124	59.15272	74.42846	71.78961	64.76969	62.08187	57.99646	53.31238
5	63.21474	70.46788	74.23556	69.13472	61.91576	58.67761	56.04570	51.69071
6	75.48031	60.76841	51.73751	57.42251	61.93318	55.30446	50.99417	54.29196
7	109.65157	85.63815	58.24553	53.99264	72.59686	83.96434	71.03110	50.58560
8	106.20625	89.15850	62.07833	53.39973	77.77257	99.84770	83.19640	50.50238
9	66.95523	44.50606	55.67430	88.69647	95.24539	86.84682	82.08110	63.50738
10	89.41284	48.62353	54.00784	96.63235	98.19483	70.76766	69.53683	77.38527
11	113.58730	80.57868	74.85042	96.10595	94.59629	75.06293	73.11324	79.24873
12	121.53567	112.14757	91.73317	76.16257	81.82961	95.61022	93.63366	77.99587
13	102.27770	75.31231	56.83436	67.93776	84.64921	80.70417	71.19953	73.51137
14	86.71549	52.96226	45.70495	75.02733	88.00951	70.02317	61.74208	71.77915
15	80.44847	42.40023	39.22996	69.89426	72.26602	53.48392	60.52761	75.70805
16	81.24815	37.63129	26.89779	48.80656	46.33343	33.97344	53.10837	78.05081
17	97.54204	50.20952	28.66690	42.90333	43.83076	33.12706	44.46113	66.62232
18	110.28095	67.05841	37.34235	40.50676	45.81068	41.04949	42.80441	55.92125
19	119.52144	73.22506	33.67657	39.62905	63.29982	57.76568	35.68993	39.59964

Рис. 3.27. Фрагмент відновленої матриці першого тестового зображення

	0	1	2	3	4	5	6	7
0	104.73659	116.64522	118.99296	113.60673	116.50889	119.26679	108.13120	95.22358
1	100.49126	133.23630	141.55586	119.66390	113.51726	125.50510	119.65374	99.52662
2	96.08303	125.76485	146.54780	144.66094	134.45388	118.42790	99.03574	95.74367
3	82.88380	123.86544	151.26026	138.51428	111.71485	95.41861	87.49831	83.52795
4	96.44666	125.10978	140.26862	127.62840	112.90223	107.72255	101.13728	98.04938
5	93.05994	127.63650	136.32773	114.24900	113.96932	137.81359	138.05184	112.64953
6	112.44837	97.42777	88.24323	106.19082	138.25820	140.30152	99.31734	65.01487
7	113.38885	114.43246	113.28881	117.12519	127.43919	125.25677	103.85852	90.24386
8	122.59141	121.12502	115.07244	109.29485	108.03989	105.62185	103.59067	115.45000
9	141.56294	129.58795	118.90487	122.33348	131.79149	127.13473	111.75194	110.03685
10	117.32679	107.88252	111.89202	123.15134	116.62098	102.38429	108.26939	124.48758
11	86.35585	121.36633	128.61975	99.53321	94.41397	132.24485	160.04051	145.69782
12	123.71345	140.76866	127.70526	101.26657	115.84763	147.93408	137.92680	106.58044
13	101.16951	123.74422	128.86671	123.11726	135.31217	141.42961	118.74466	108.32452
14	114.51281	113.42863	99.84735	100.43531	132.91550	149.85813	121.26791	97.97762
15	108.00865	110.73663	108.82803	107.68609	113.99736	115.81969	106.48007	105.63465
16	117.67517	134.38295	133.18191	110.84542	101.99236	118.13241	133.30305	131.43601
17	124.05836	129.40296	121.86599	109.28815	115.99920	134.54320	137.03625	124.38434
18	109.79337	111.87177	113.34157	110.71960	106.34682	106.92866	111.47265	112.96142
19	97.76400	108.92360	115.35725	105.17454	93.79630	104.45869	123.48707	118.15888

Рис. 3.28. Фрагмент відновленої матриці другого тестового зображення

	0	1	2	3	4	5	6	7
0	122.86468	133.15367	145.36809	145.47376	135.69683	132.82253	135.13424	130.23877
1	127.52882	123.09182	119.42533	115.67195	115.58384	123.98597	123.86922	103.75678
2	117.50272	111.43232	130.65065	152.13513	138.67938	119.64439	126.02616	128.44777
3	155.86076	134.91441	117.55089	123.80558	139.87663	134.87791	110.90091	104.37140
4	148.06897	144.99663	136.65384	129.12910	130.27194	132.11467	121.92780	109.85650
5	129.27791	139.04286	145.12620	138.83324	132.52628	139.40199	144.53509	130.41090
6	139.66941	152.65749	151.01854	129.06227	115.01280	125.30268	141.20445	144.16245
7	129.47924	137.62162	144.28552	142.52517	135.09961	129.90247	131.70892	137.85044
8	123.39939	134.48802	138.25555	136.57817	141.74876	144.42143	135.44558	130.61307
9	124.84045	144.28488	167.38652	160.76567	127.92864	117.38382	133.69556	130.32817
10	117.38118	134.32685	163.05569	169.91424	150.16006	147.33525	160.44885	146.87602
11	128.29249	141.81954	152.23179	150.69715	146.33151	147.44195	151.13389	157.61361
12	128.65772	141.47177	144.59154	139.66241	147.39753	161.19004	158.84200	148.98334
13	114.47095	112.16555	112.87917	122.18248	136.71387	145.42634	145.25288	144.14808
14	114.42198	121.43719	119.89111	116.08581	126.03499	140.35412	145.12847	148.13657
15	113.10079	115.80580	123.39977	138.79530	152.22379	148.95453	139.32712	145.98451
16	136.56371	126.78794	134.48678	159.91345	168.40965	150.85463	142.93044	160.27818
17	129.78021	130.52617	132.34020	141.85933	156.73503	162.78967	158.75610	153.43226
18	135.04413	147.47782	137.83140	115.16199	117.43180	139.00907	150.35446	152.15199
19	142.55714	141.42428	138.55874	132.28052	125.35873	124.99667	131.39291	136.32379

Рис. 3.29. Фрагмент відновленої матриці третього тестового зображення

3.2.3. Графіки результату досліджень

Розглянемо наступні графіки:

- Для першого зображення

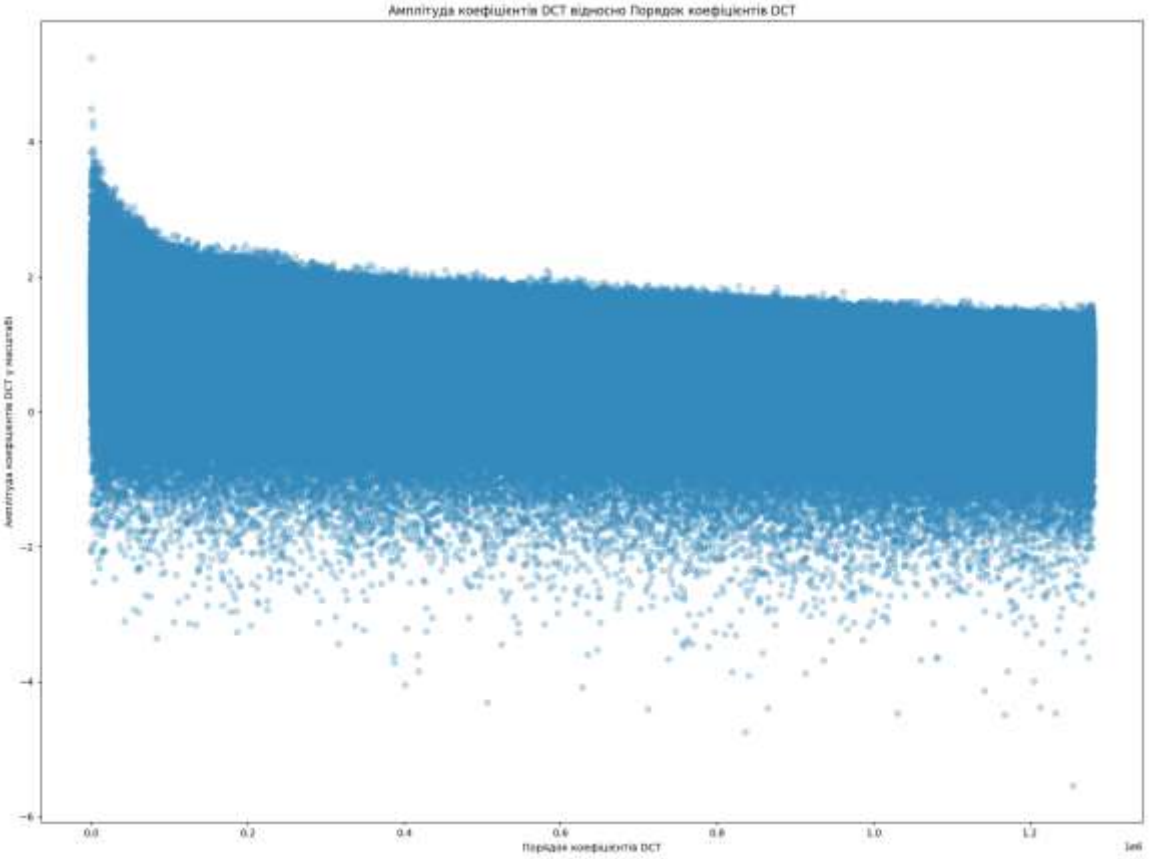

Рис. 3.30. Амплітуда коефіцієнтів *DCT* відносно Порядок коефіцієнтів *DCT*

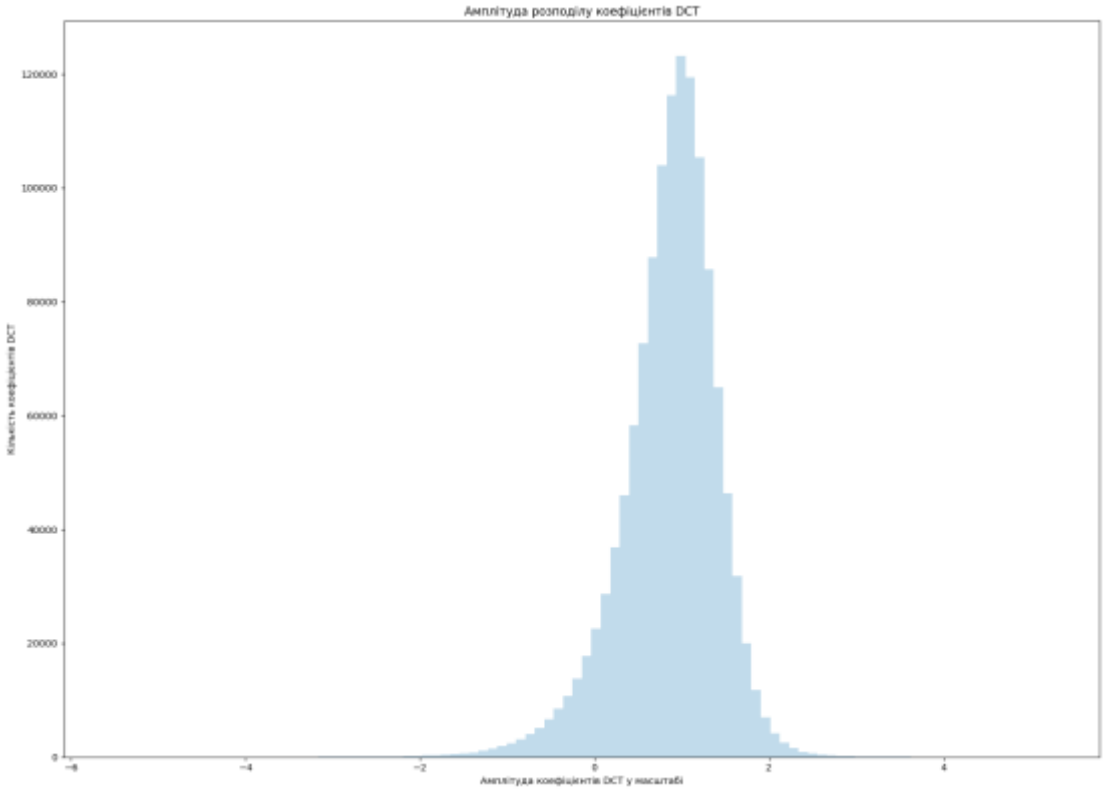

Рис. 3.31. Амплітуда розподілу коефіцієнтів *DCT*

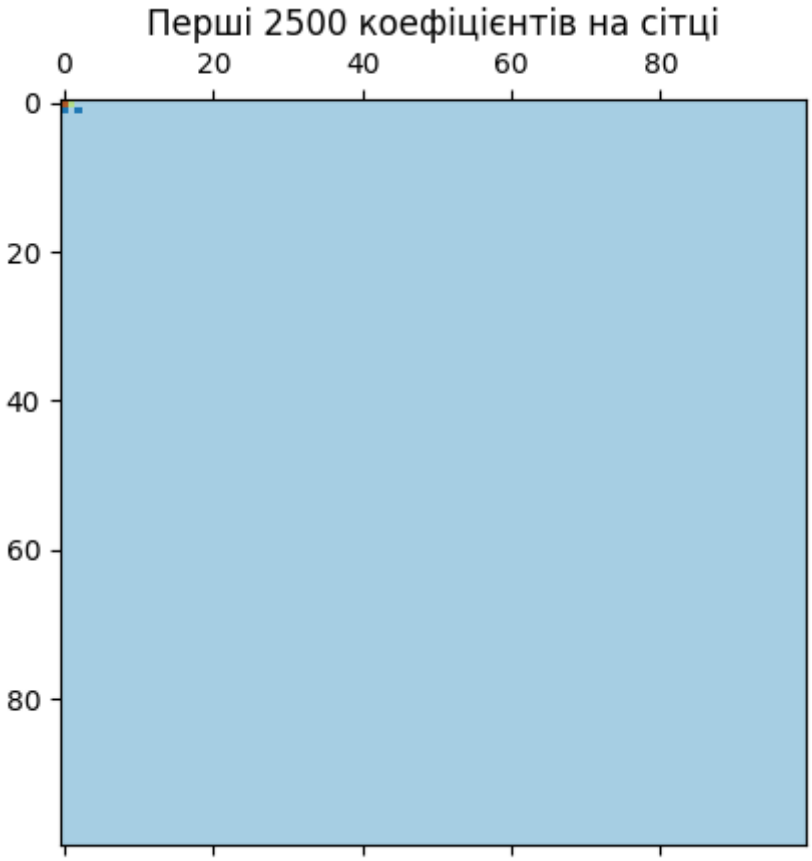

Рис. 3.32. Перші 2500 коефіцієнтів *DCT* на сітці

- Для другого зображення

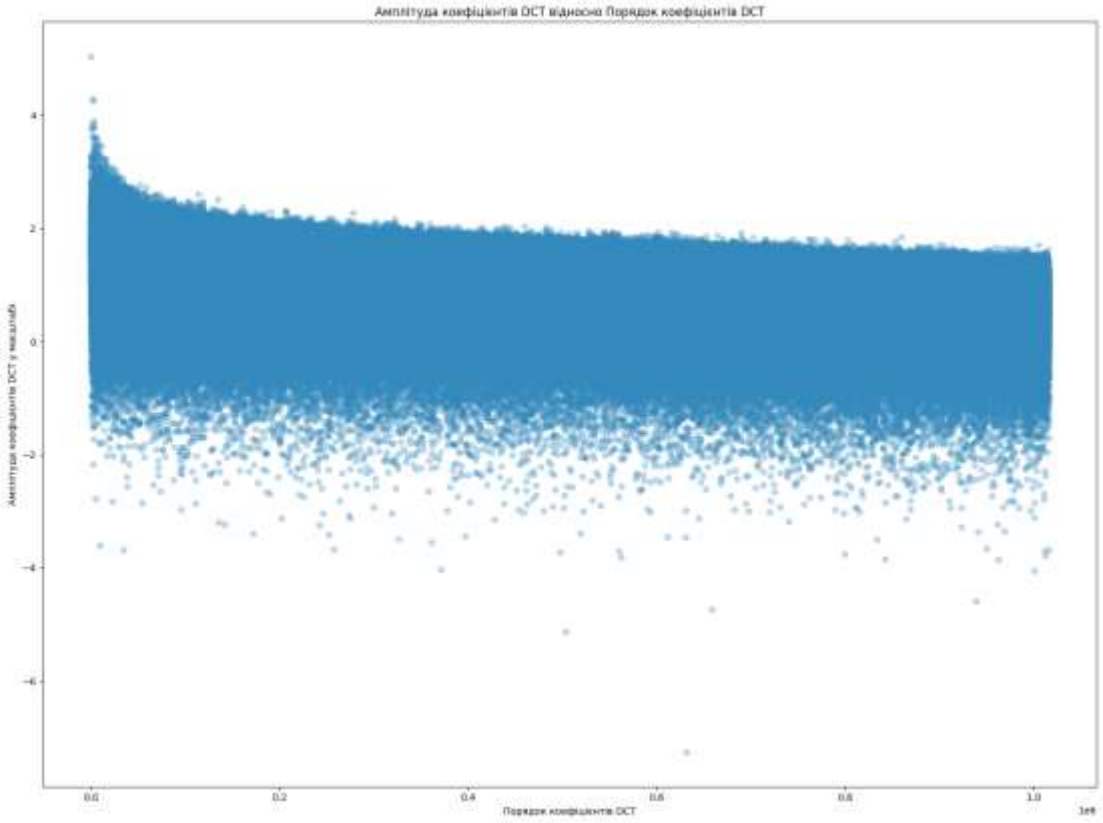

Рис. 3.33. Амплітуда коефіцієнтів *DCT* відносно Порядок коефіцієнтів *DCT*

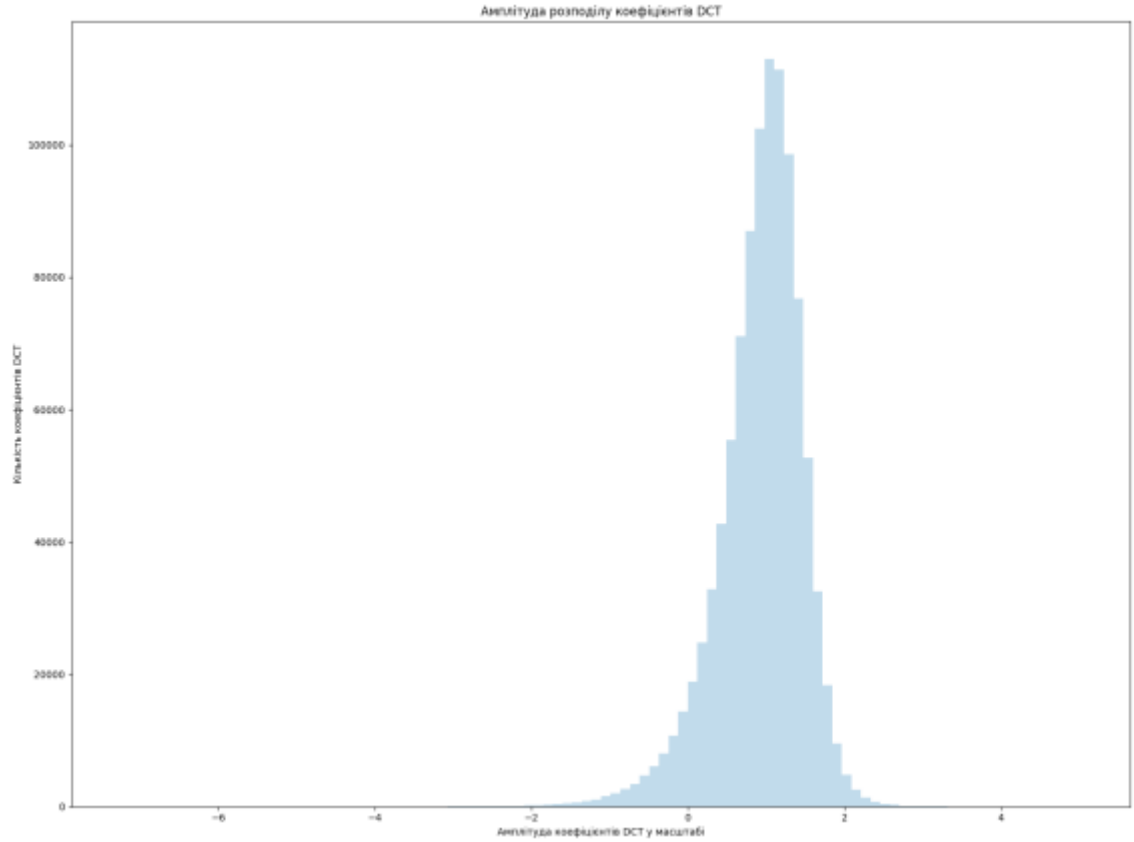

Рис. 3.34. Амплітуда розподілу коефіцієнтів *DCT*

Рис. 3.35. Перші 2500 коефіцієнтів *DCT* на сітці

- Для третього зображення

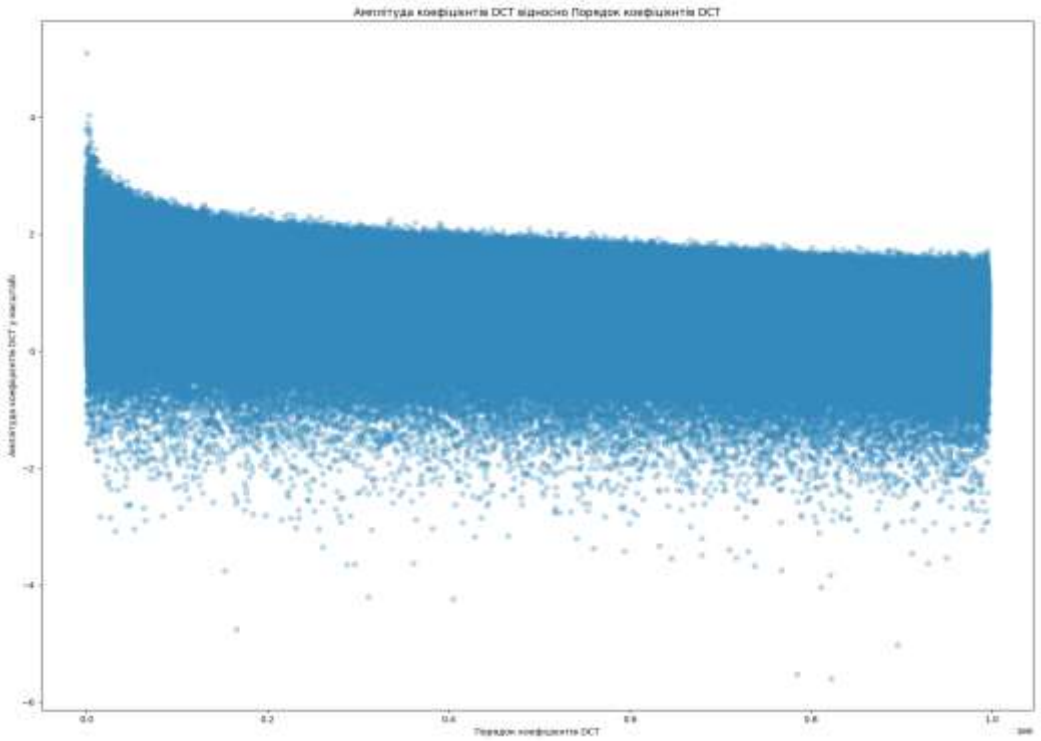

Рис. 3.36. Амплітуда коефіцієнтів *DCT* відносно Порядок коефіцієнтів *DCT*

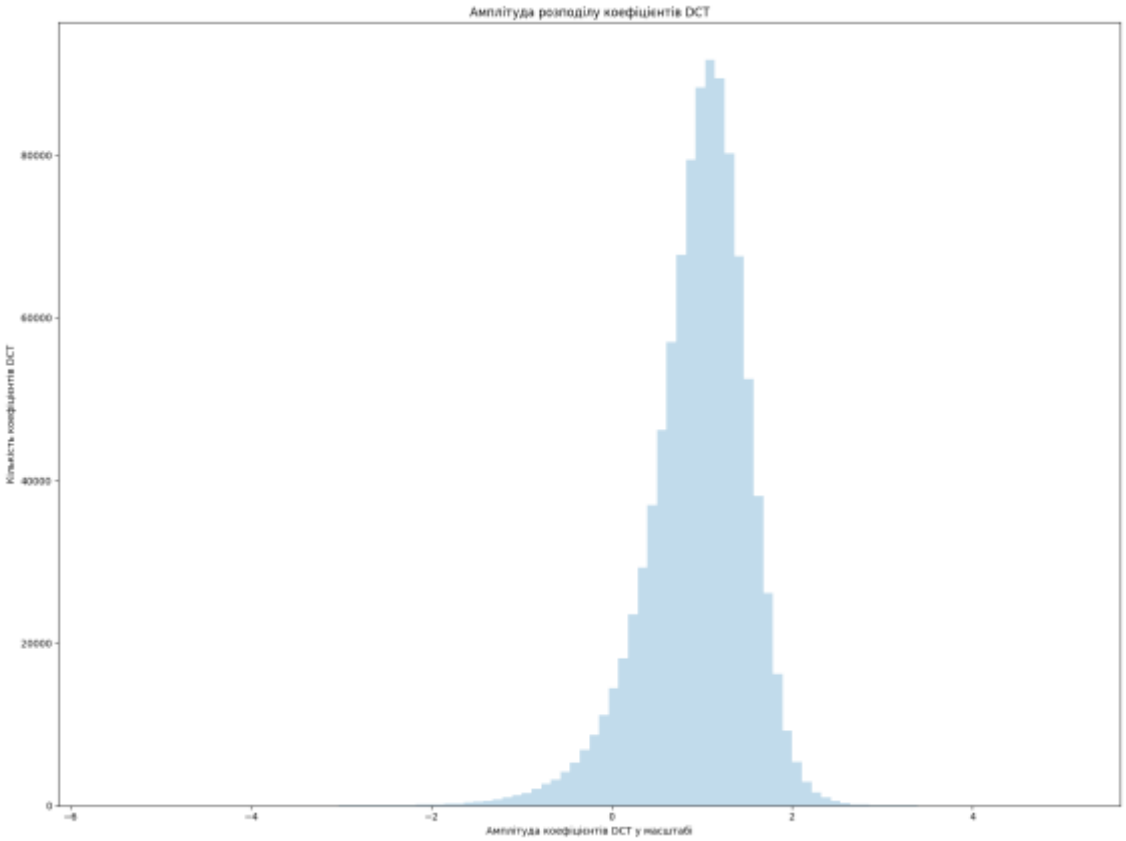

Рис. 3.37. Амплітуда розподілу коефіцієнтів *DCT*

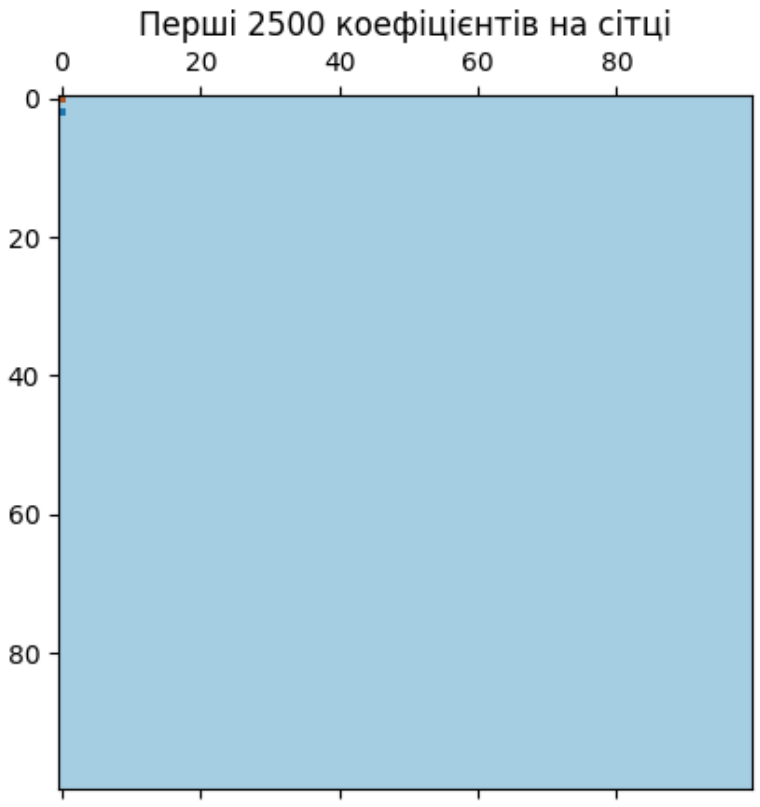

Рис. 3.38. Перші 2500 коефіцієнтів *DCT* на сітці

Ці графіки важливі у двох випадках. По-перше, це говорить про те, що великі коефіцієнти і дуже малі коефіцієнти є досить малими числами (зауважимо, що вісь *X* має лог-масштаб). Таким чином, компроміс між стисненням і якістю зображення зазвичай залежить від коефіцієнтів, які мають середні значення діапазону. Легко отримати дуже великі коефіцієнти і відхилити дуже малі коефіцієнти в реконструйованому зображенні, але не дуже легко включати або відкидати середні значення, виходячи виключно з їх амплітуд. У цих коефіцієнтах потрібно дивитися на частоти, до яких вони належать, якщо вони знаходяться в якомусь високочастотному діапазоні, то вони будуть відкинуті, тоді як, якщо вони належать до нижчого діапазону частот, це може ввести в сигнал помітні і великі артефакти. Замість порівняння лише з середньоквадратичною похибкою, щоб дізнатися, де зупинитися в коефіцієнтах, можна перевірити кращі показники, які розглядають візуальну вірність або навіть сприйману якість, щоб знайти найкраще місце між ступенем стиснення та якістю зображення. У реалізаціях, щонайменше, для *JPEG*, це здійснюється за допомогою заздалегідь визначеного числа. Тому, якщо зображення має різні розподіли пікселів, ніж більшість природних зображень, або має дуже високі частотні складові, то можна побачити дуже помітні артефакти навколо країв зображень.

На наступних трьох рисунках буде зображено перші 64 зображення з поетапного відновлення зображення для двох дослідів. Для того, щоб проаналізувати схожість зображень будуть використані алгоритми з дескрипторами для зіставлення зображень. Також варто зауважити, що для різних цілей об’єм даних має визначатися на основі потреб, від об’єму даних залежить і якість зображення.

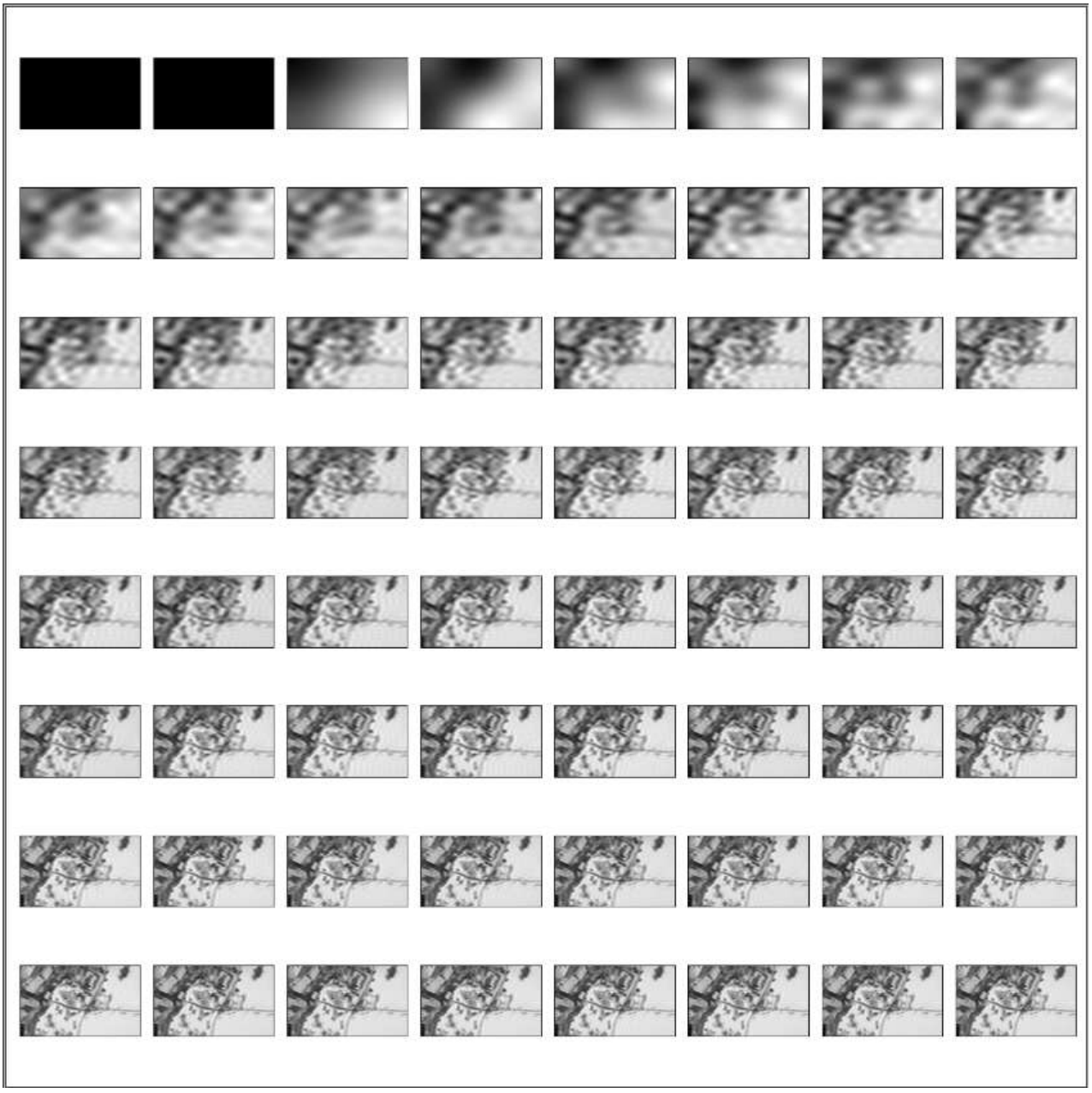

Рис. 3.39. Перші 64 зображення з поетапного відновлення матриці для першого тестового зображення

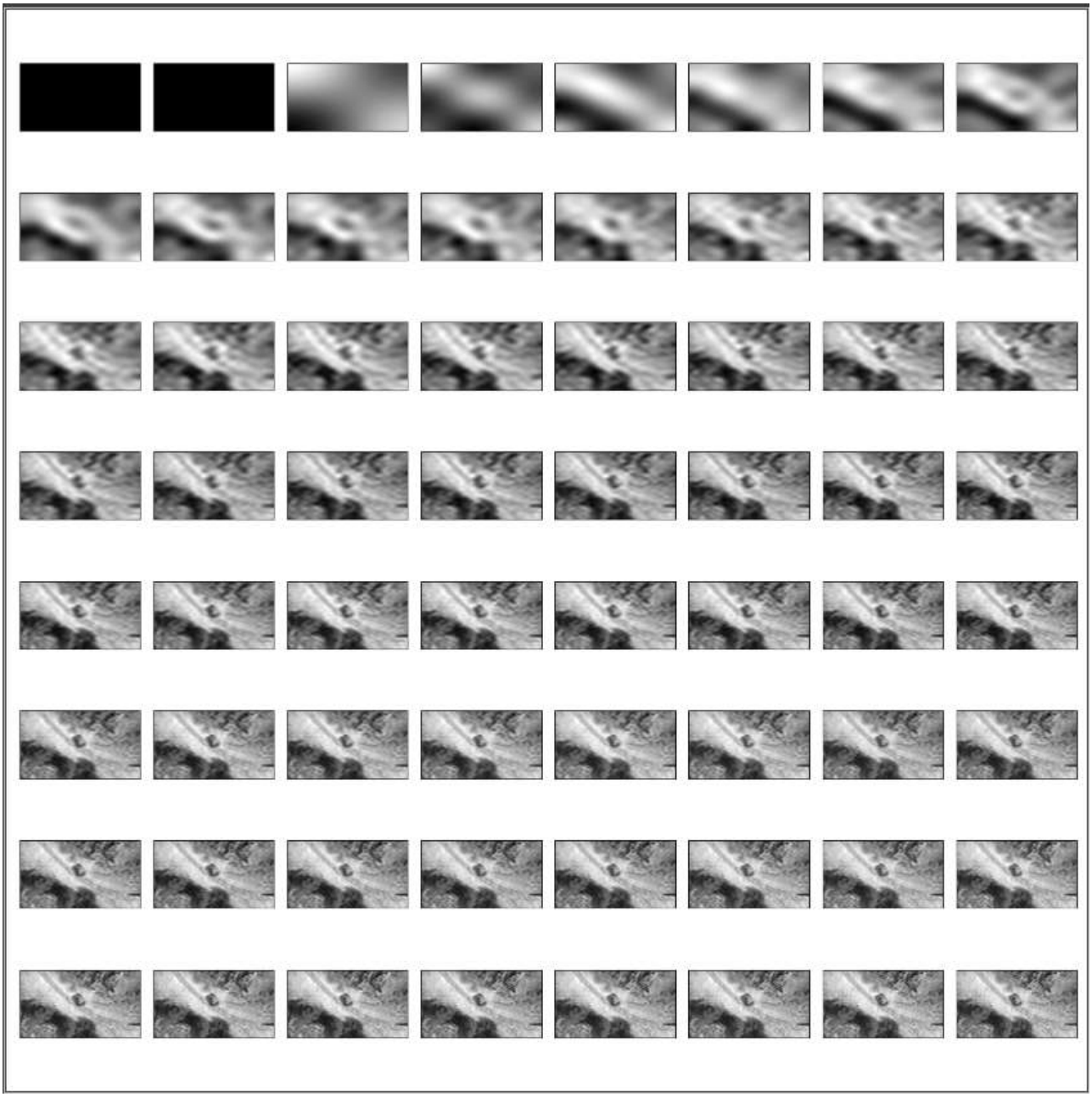

Рис. 3.40. Перші 64 зображення з поетапного відновлення матриці для другого тестового зображення

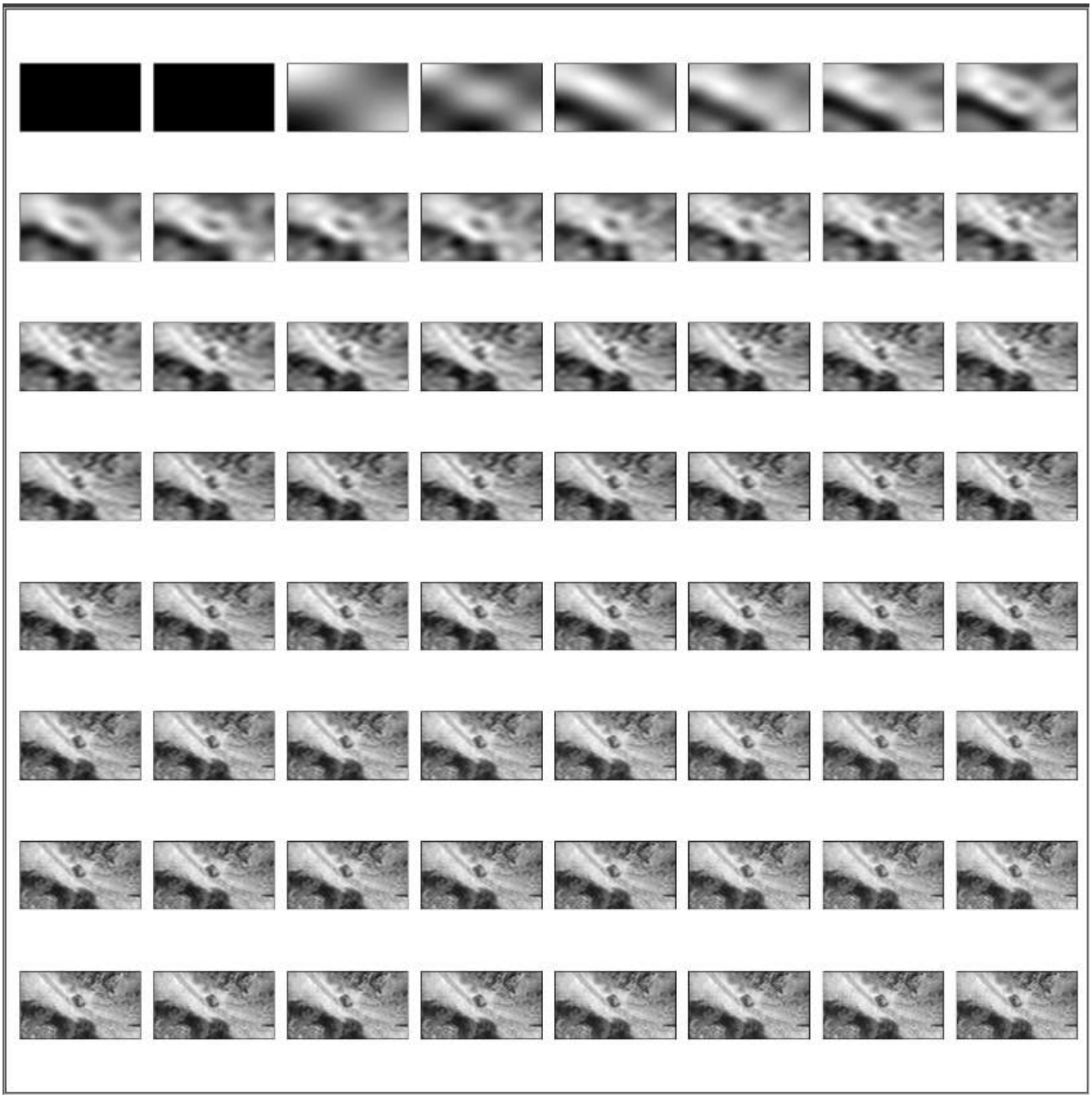

Рис. 3.41. Перші 64 зображення з поетапного відновлення матриці для третього тестового зображення

У моїй попередній дипломній роботі (степінь “Бакалавр”) оцінка візуальної вірності визначалася тільки на основі людського аналізу за допомогою ока. Дослідивши відносно нову тему, таку як *Computer Vision*, було вирішено проаналізувати існуючі алгоритми оцінки схожості зображень.

3.2.3. Дослідження алгоритмів оцінки схожості зображень для тестових зображень

Для дослідження було обрано 3 алгоритми:

- *Brute-Force Matching with ORB Descriptors*
- *Brute-Force Matching with SIFT Descriptors*
- *FLANN based Matching with SIFT Descriptors*

Результати досліджень першого зображення:

Рис. 3.42. 500-те відновленне зображення з 870

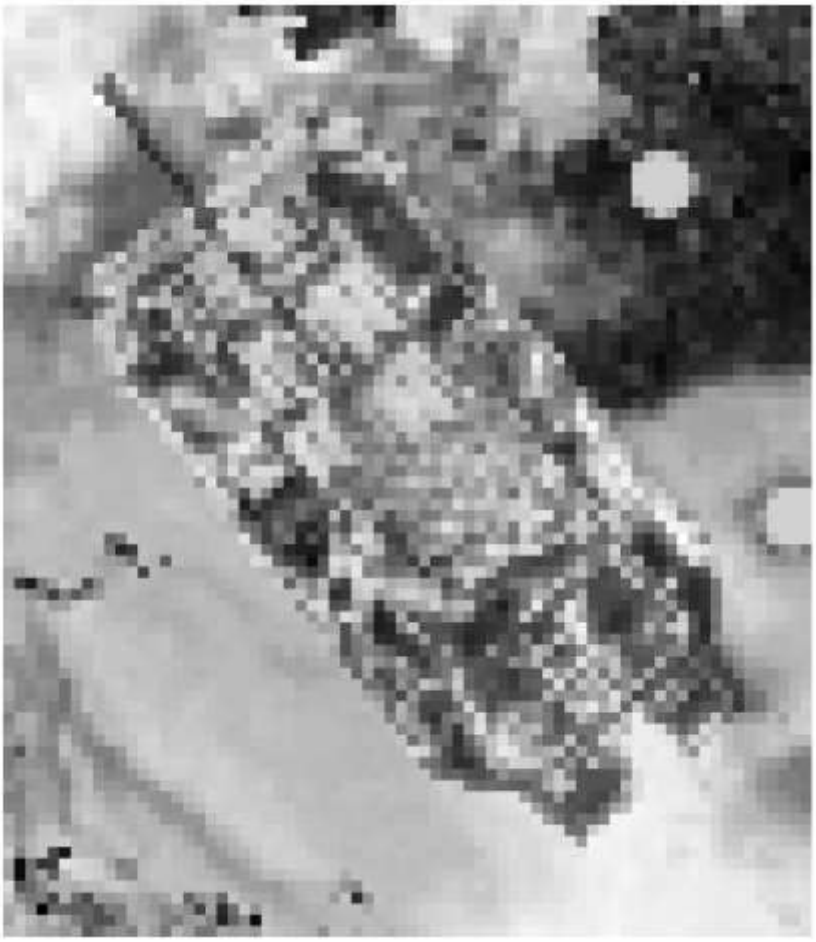

Рис. 3.43. Об'єкт для розпізнавання

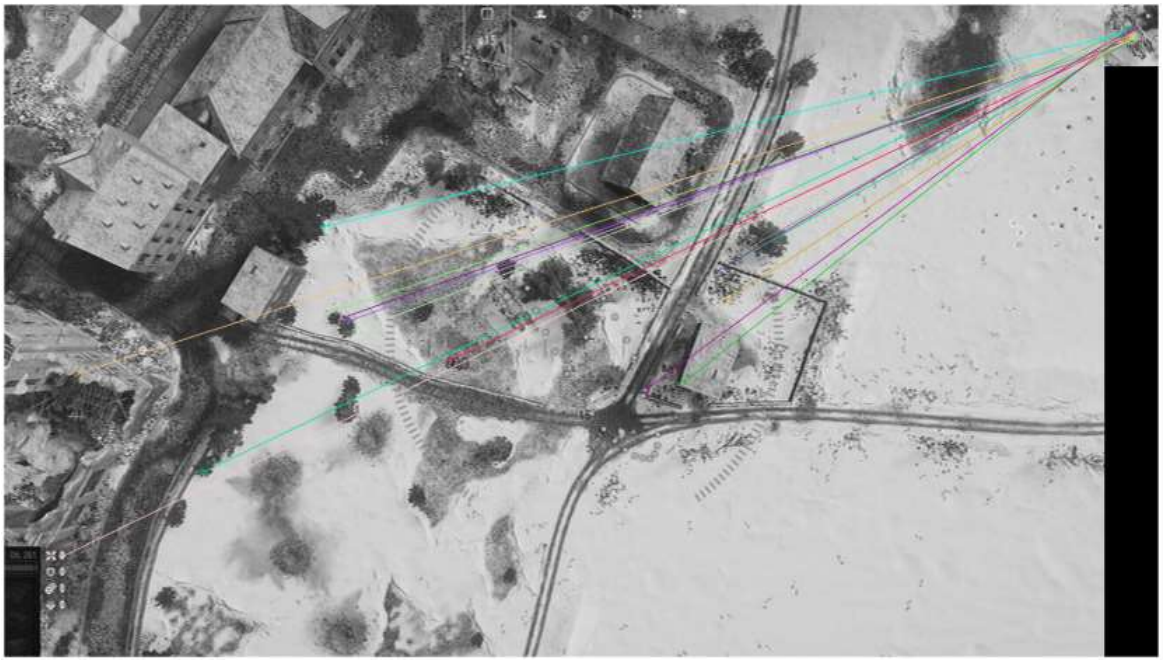

Рис. 3.44. Результат роботи алгоритму *Brute-Force Matching with ORB Descriptors*

Рис. 3.45. Результат роботи алгоритму *Brute-Force Matching with SIFT Descriptors*

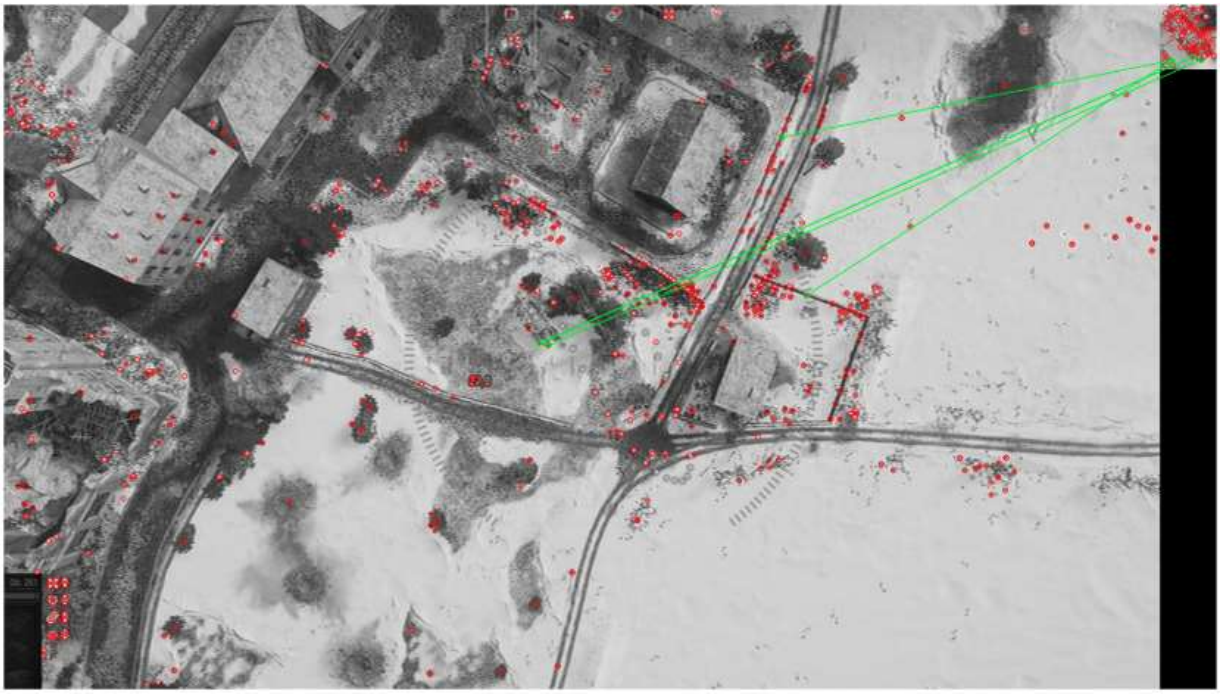

Рис. 3.46. Результат роботи алгоритму *FLANN based Matching with SIFT Descriptors*

Результати досліджень другого зображення:

Рис. 3.47. 500-те відновленне зображення з 757

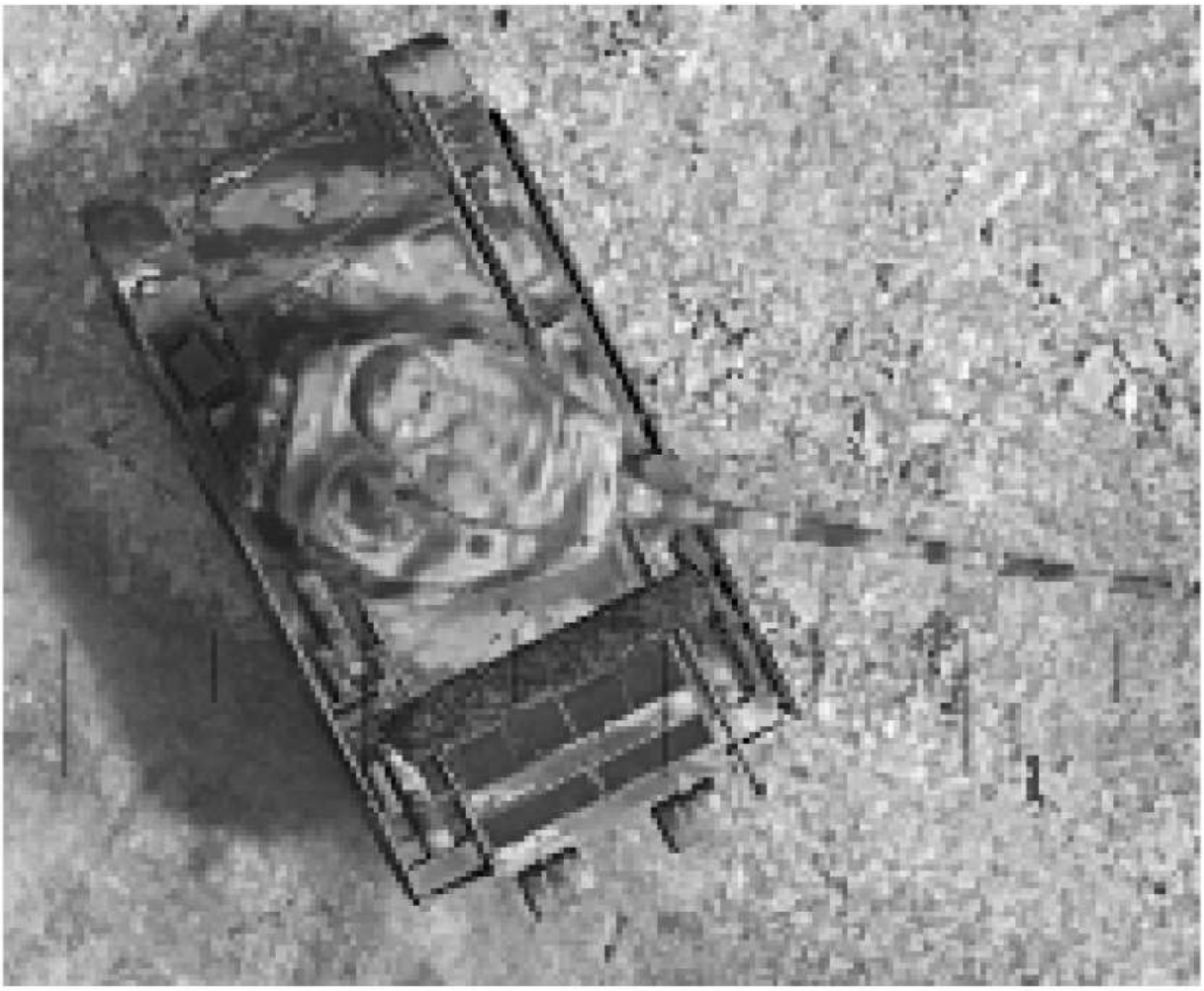

Рис. 3.48. Об'єкт для розпізнавання

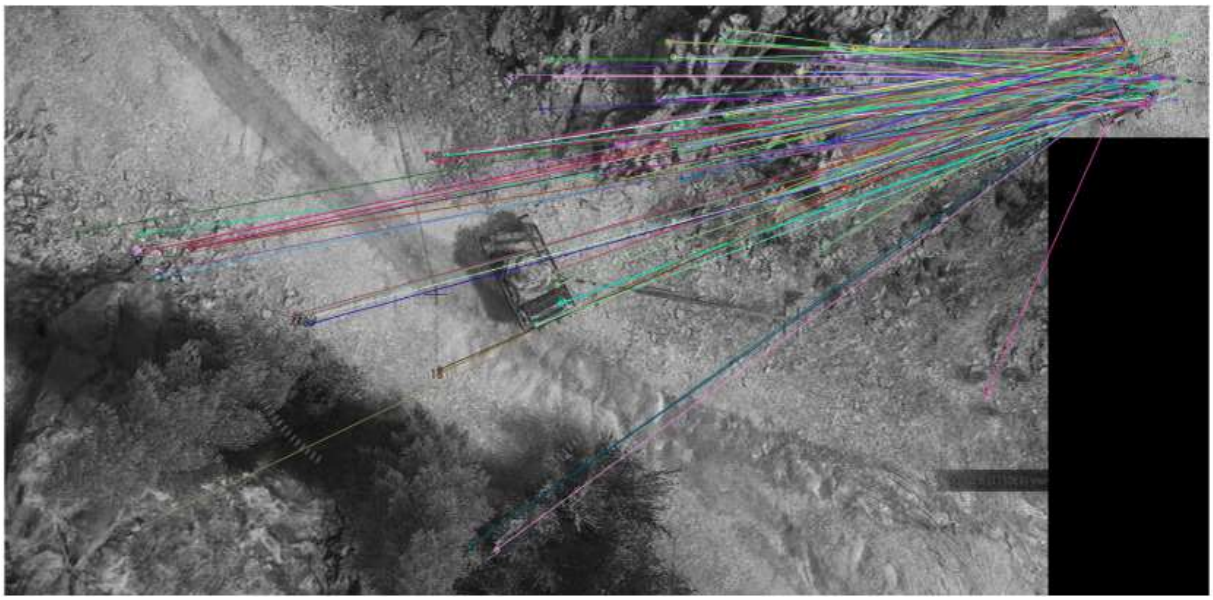

Рис. 3.49. Результат роботи алгоритму *Brute-Force Matching with ORB Descriptors*

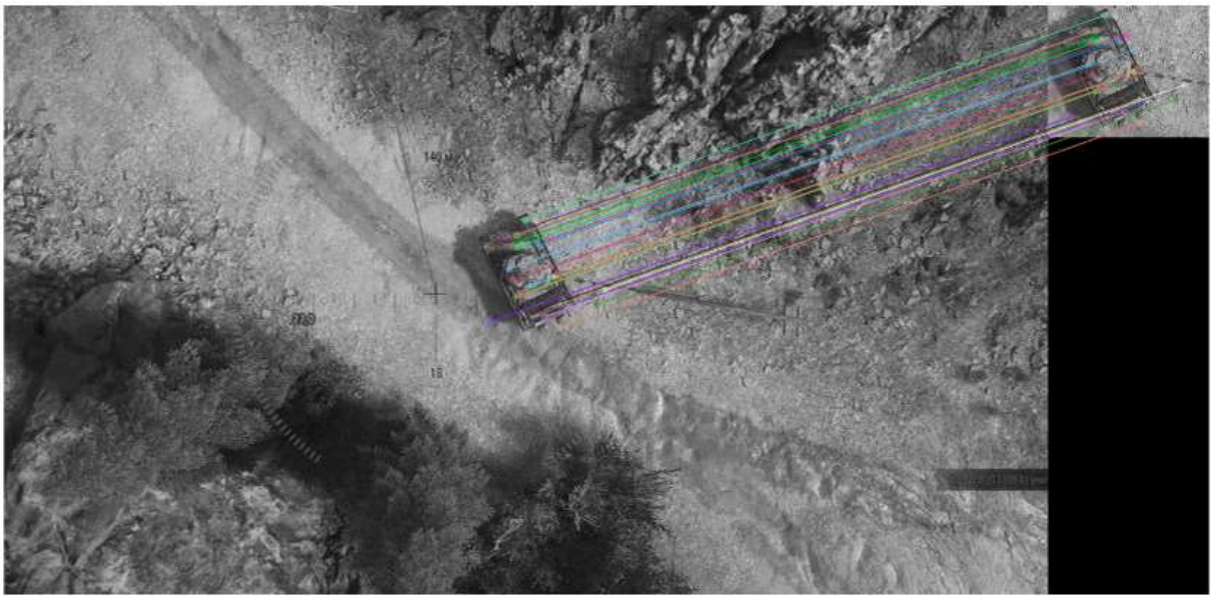

Рис. 3.50. Результат роботи алгоритму *Brute-Force Matching with SIFT Descriptors*

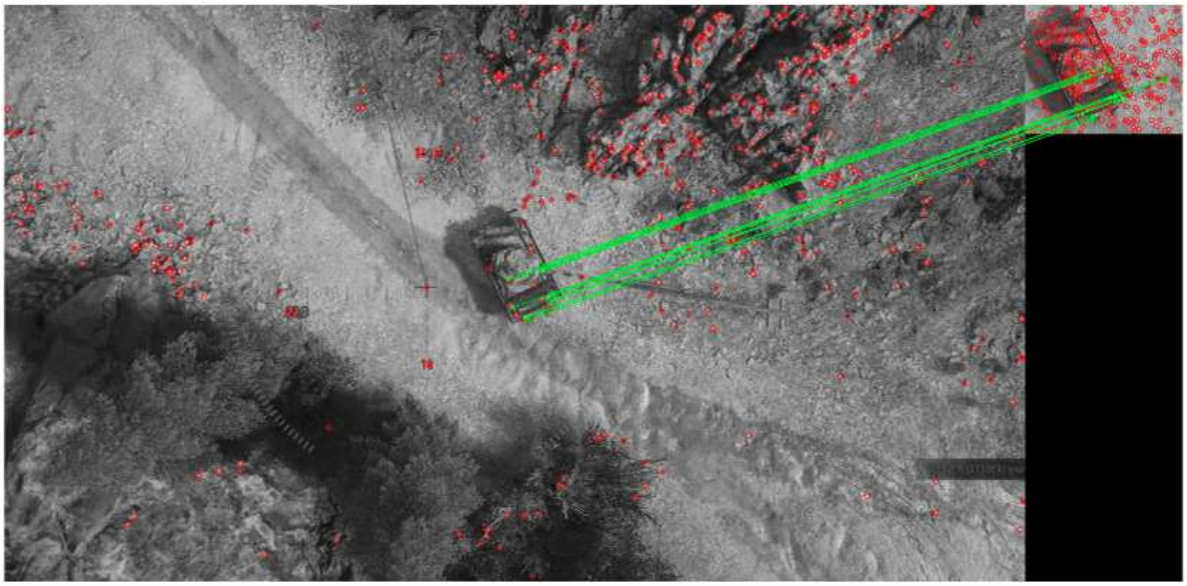

Рис. 3.51. Результат роботи алгоритму *FLANN based Matching with SIFT Descriptors*

Результати досліджень третього зображення:

Рис. 3.52. 500-те відновленне зображення з 763

Рис. 3.53. Об'єкт для розпізнавання

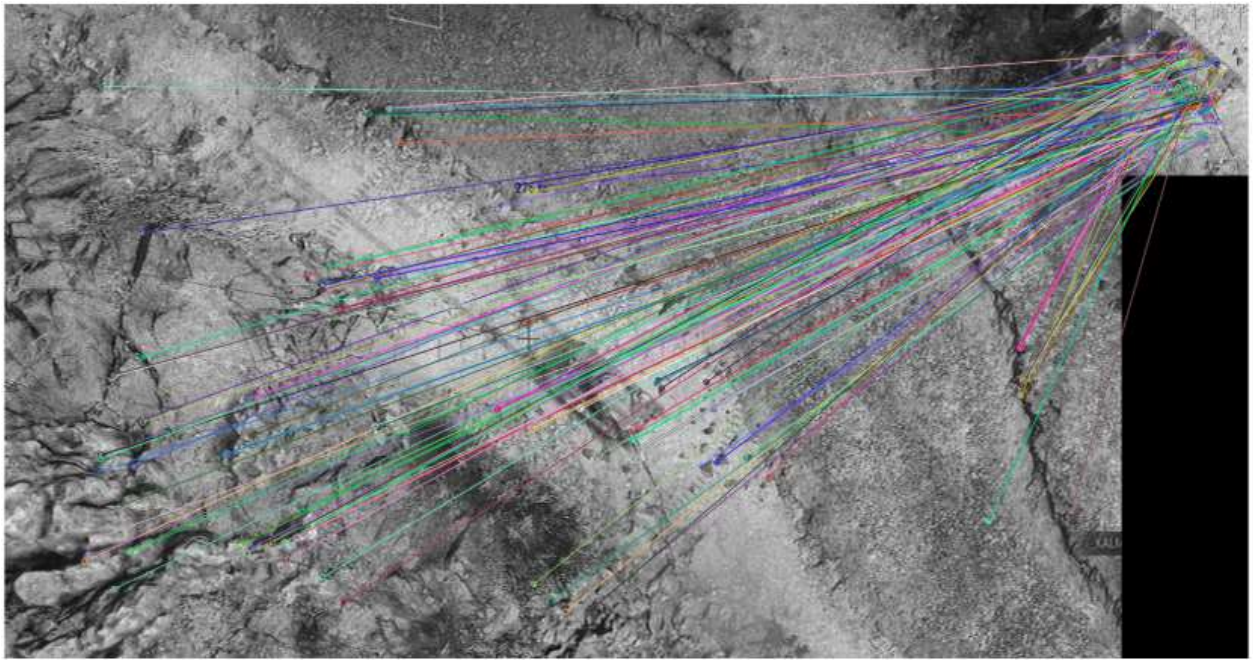

Рис. 3.54. Результат роботи алгоритму *Brute-Force Matching with ORB Descriptors*

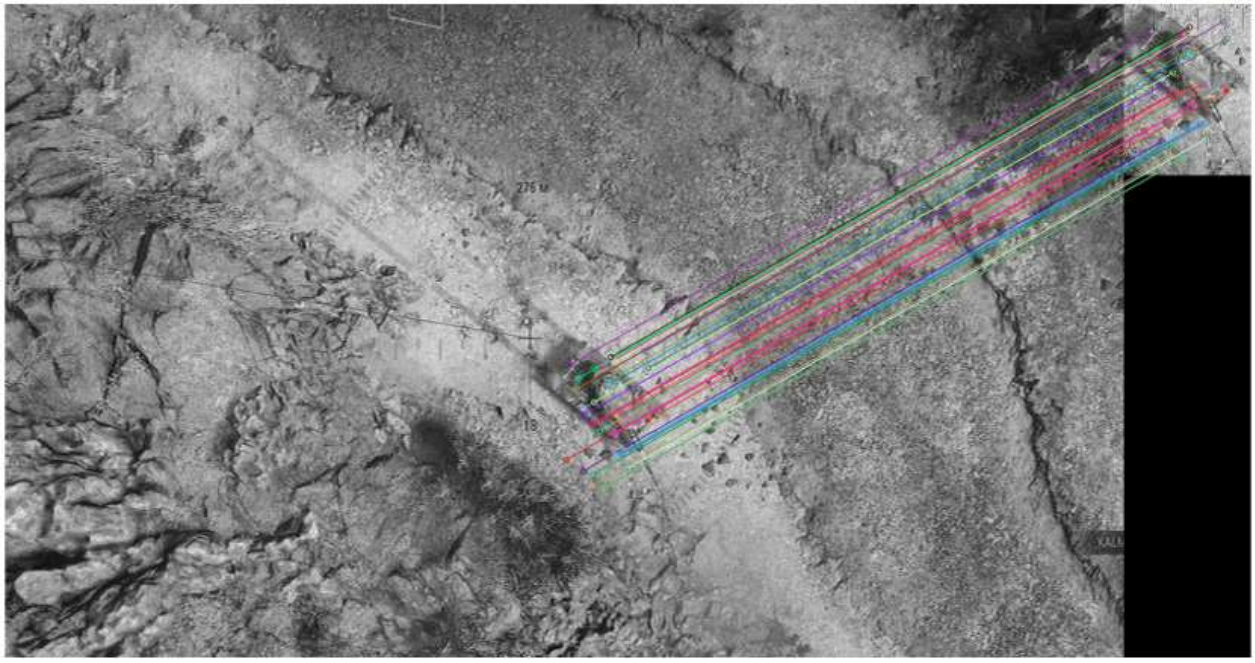

Рис. 3.55. Результат роботи алгоритму *Brute-Force Matching with SIFT Descriptors*

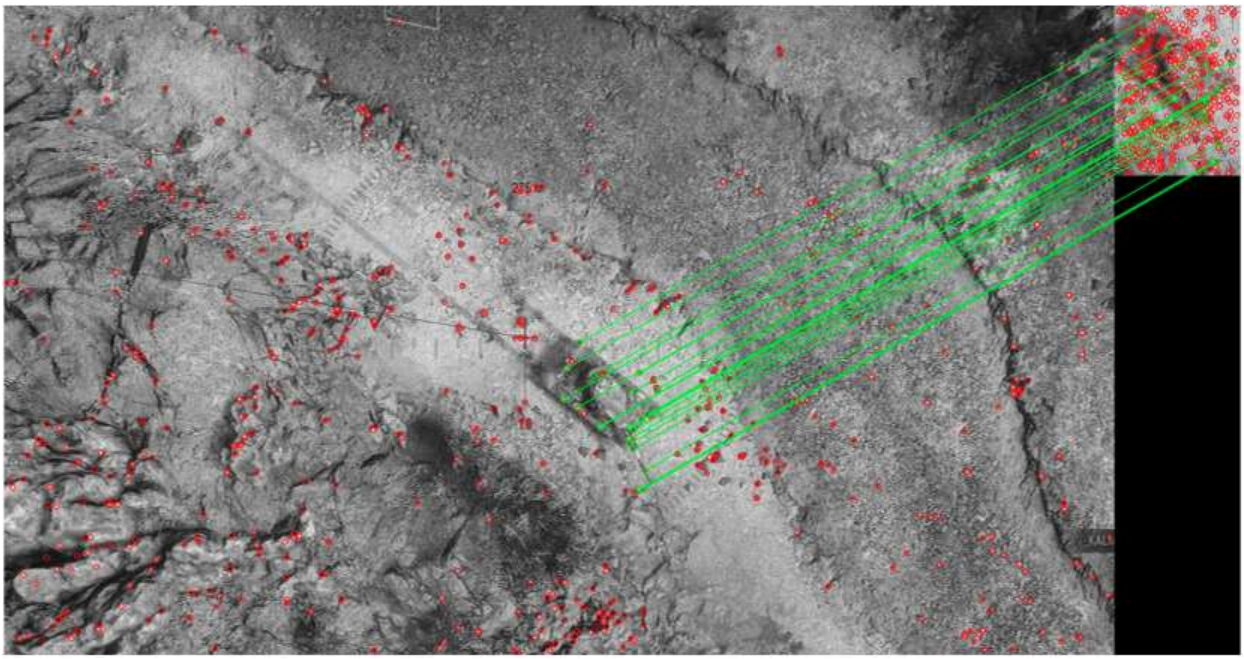

Рис. 3.56. Результат роботи алгоритму *FLANN based Matching with SIFT Descriptors*

З проведених дослідів найкращим алгоритмом для оцінки схожості зображень виявився алгоритм *Brute-Force Matching with SIFT Descriptors*. Виходячи з цього було вирішено провести додаткове дослідження використовуючи цей алгоритм, але знизити якість зображень обравши 200-те відновлене зображення для кожного з тестових зображень.

Результат додаткового досліду з використанням *Brute-Force Matching with SIFT Descriptors*:

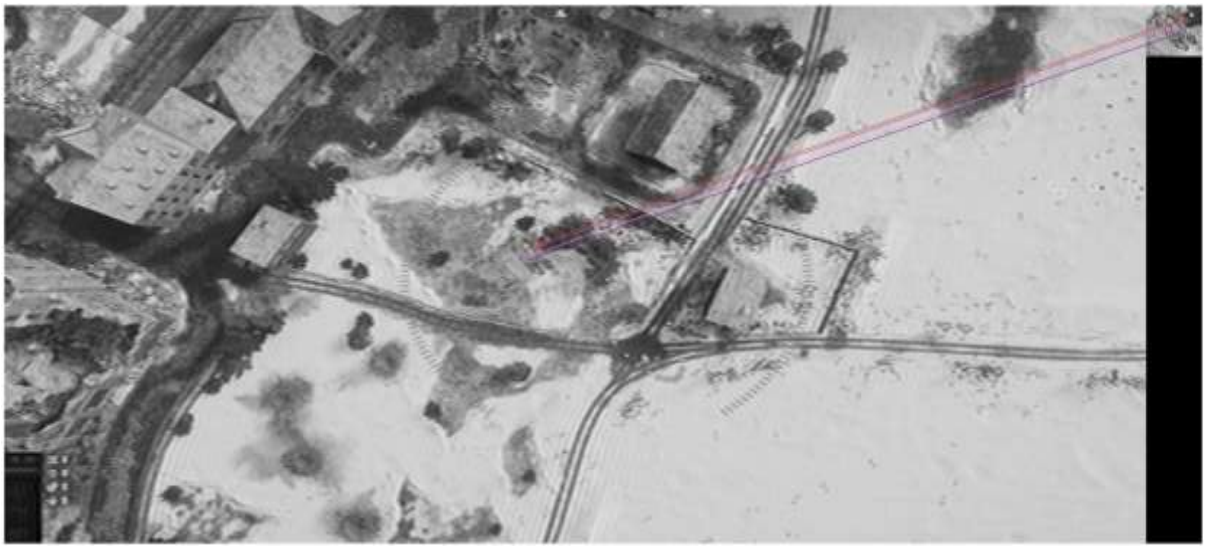

Рис. 3.57. Результат для першого зображення

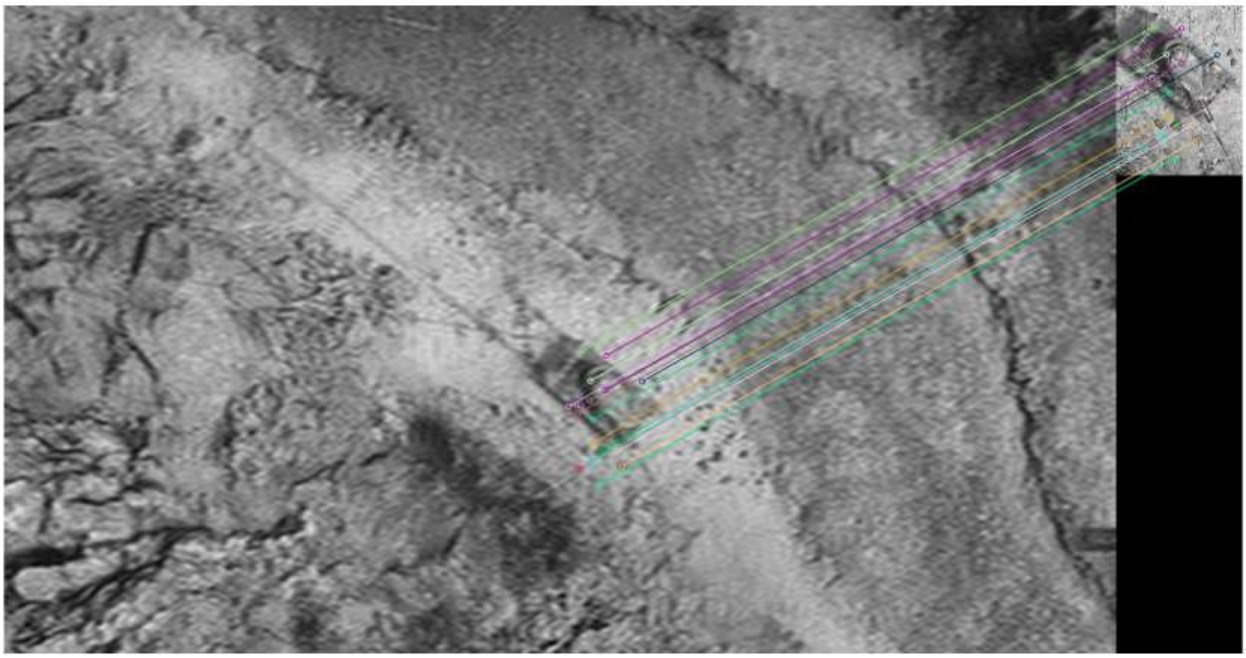

Рис. 3.58. Результат для другого зображення

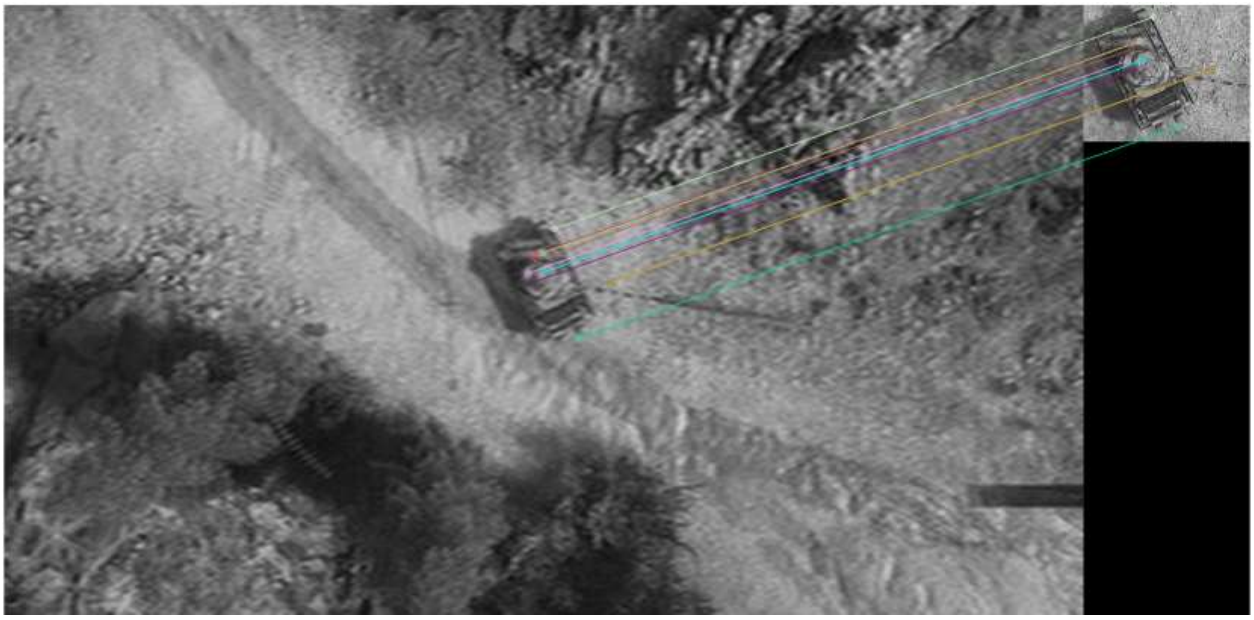

Рис. 3.59. Результат для третього зображення

Висновки до розділу

У даному розділі керуючись структурною схемою алгоритму були проведені тестові дослідження. Для оцінки візуальної вірності (схожості, зіставлення) зображень були досліджені та проаналізовані резуьтати роботи трьох алгоритмів - *Brute-Force Matching with ORB Descriptors*, *Brute-Force Matching with SIFT Descriptors*, *FLANN based Matching with SIFT Descriptors*. Після першого етапу дослідження було виявлено, що найкращий результат має алгоритм - *Brute-Force Matching with SIFT Descriptors.* Після цього було проведене додаткове дослідження зображень за допомогою цього алгоритму, але знизивши якість зображення більш ніж на половину. За результатом додаткового дослідження можна сказати, що цей алгоритм дозволяє автоматично без людського ока визначати схожість зображень та якість візуального сприйняття, а також істотно зменшити об’єм даних зображень. Для прикладу наведено об’єм даних вихідних і результуючих зображень відповідно: перше зображення - 2,5 *MB*, 105,4 *kB*; друге зображення - 2,3 *MB*, 99,7 *kB*; третє зображення - 2,3 *MB*, 109,7 *kB*.

ВИСНОВКИ

На рівні представлення може здійснюватися стиснення, розпакування інформації, або ж її шифрування та дешифрування, а також запити, які адресовані іншому мережевому ресурсу перенаправляються на адрес цього ресурсу, якщо ці запити не можна обробити локально. Рівень представлення керує не тільки перетворенням даних у необхідний формат та їх поданням, він також займається самою структурою даних, які використовуються іншими додатками та застосунками. Тому такими маніпуляціями, які були описані вище, цей рівень забезпечує організацію даних при їх передачі по мережі. На цьому рівні існують і інші підпрограми, які можуть стискати тексти до необхідного розміру і перетворюють зображення у потік бітів, для того щоб ці дані могли передаватися у мережі. Стандарти які властиві рівню представлення визначають способи, якими можна представляти будь-які графічні зображення. Одним із стандартів рівня представлення, який може використовуватися для графічних зображень, є стандарт, розроблений Об'єднаної експертною групою по фотографії (*Joint Photographic Expert Group*), такий формат має абревіатуру *JPEG*. Саме на рівні представлення для перетворення графічних зображень у формат *JPEG* має застосування двомірне *DCT*, це перетворення є одним із ключових етапів алгоритму стиснення *JPEG*. У другому розділі було розглянуто показники ефективності інструментарію.

Такими показниками є: конструкція проходження, конструкція розгалуження, конструкція циклу, трудомісткість та наглядність розрахунків. В моїй дипломній роботі найбільш придатними інструментами для дослідження є: алгебри (в тому числі булева), теорія графів, математичне програмування, теорія ймовірностей, теорія рядів, математична статистика, теорія масового обслуговування, імітаційне моделювання та фізичне моделювання. В якості показника ефективності інструментарію визначено трудомісткість та наглядність розрахунків. Тож в результаті аналізу в якості інструментарію дослідження обрано теорію ймовірностей, математичну статистику та теорію рядів.

Якщо поглянути на перші перші 2500 коефіцієнтів у сітці 50 на 50, то ми можемо побачити, що багато коефіцієнтів насправді дуже малі. Це не тільки забезпечує хороше ущільнення зображення (менше коефіцієнтів означає високу швидкість ущільнення), але також забезпечує хороший компроміс між стисненням і якістю зображення. Як правило, дуже низькі частоти мають більш високе відношення порядків величин і схожі з дуже високими частотами.

Причина того, що *DCT* є настільки успішним, можна побачити з цих зображень, як можна було також побачити, що перші 40-50 коефіцієнтів з даних могли б захопити більшу частину зображення. Якщо ми хочемо використати ці коефіцієнти для реконструкції зображення, ми можемо отримати високу компресію, де ми не втрачаємо належної якості зображення. Компресії з втратами зазвичай використовують цей тип схеми для отримання ступеня стиснення. Вони відкидають частину високочастотної інформації (ребра і зображення швидкого переходу, де вони зберігають повільні змінні і постійні кольори). Оскільки більшість сигналів носять низьку частоту за природою, цей підхід працює на практиці досить добре. Було видалено високочастотні коефіцієнти і передані "важливі" коефіцієнти (коефіцієнти, які мають найбільшу інформацію про сигнал).

Середньоквадратичну похибку встановити не вдалося, тобто спосіб стиснення зображення *DCT*, який є основою алгоритму стиснення *JPEG*, при досліджуваних даних працює відмінно. У ході дослідження було виявлено, що для пришвидшення передачі даних, а саме зображення, можна передавати не повний об’єм даних, для того щоб встановити, що зображено на фотографії. Це допоможе пришвидшити передачу даних – чим менше об’єм даних, що передаються, тим швидше їх отримає адресат. Подальшою роботою над цією темою може бути робота з коефіцієнтами та параметрами алгоритму, вдосконалення та кастомізація алгоритму для конкретної потреби, а також додаткові дослідження вдосконаленого алгоритму та знаходження оптимального зображення для передачі з бортового комп’ютера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сипко Р.В. Двомірне дискретно — косинусне перетворення для передачі інформації з бортового комп'ютера: диплом. проект. НАУ, Київ, 2019.
2. Капітонова Ю.В., Кривий С.Л., Летічевський О. М., Луцький Г.М., Печурін М.К. Основи дискретної математики. Підручник. – К.: Наукова думка, 2002. – 580с.
3. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посіб. – 2-е вид., доопрац. – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2009. – 360 с.
4. *Jones C.A., Drake F.L. Python & XML // BHV.* –2014. –807 *c.*
5. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/Wi-Fi*, вільний.
6. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов
7. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/Bluetooth*, вільний.
8. ГОСТ 19.701–90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения
9. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу*: https://uk.wikipedia.org/wiki/*Теорія_графів, вільний.
10. ГОСТ 7.11–2004 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках»
11. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/*Фізичне_моделювання, вільний.
12. ДСТУ ГОСТ 7.1:2006 «Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання»
13. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/*Імітаційн_моделювання, вільний.

14. ДСТУ 3582: 2013 «Бібліографічний опис скорочення слів і словосполучень в українській мові»

15. ДСТУ 8302-2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання»

16. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/*Математична_статистика, вільний.

17. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/*Мат_програмування, вільний.

18. *Web*-сайт: *https://uk.wikipedia.org*;

Режим доступу: *https://uk.wikipedia.org/wiki/*Теорія_масового_обслуговування, вільний.

19. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. –63 с.

20. Гамма Э., Холм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2006. –366 с.

21. *David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints*. – 2004. – 28 *с*.

22. Кулик М.С., Полухін А.В Положення про дипломні роботи (проекти) випускників Національного авіаційного університету: Положення про дип.роботи. – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2011. – 42 с.

ДОДАТОК А

Код програми для *DCT* на мові програмування *Python*:

```
import numpy as np
import matplotlib.pyplot as plt

from math import sqrt
from PIL import Image
from scipy import fftpack
from sklearn.metrics import mean_squared_error

from check_similarity import CheckSimilarity


def get_mean_squared_error(actual):
    dct = get_2d_dct(pixels)
    idct = get_2d_idct(dct)
    rms = sqrt(mean_squared_error(actual, idct))
    return rms


def get_image():
    image = Image.open('test_images/7_full.png')
    img_grey = image.convert('L')
    img = np.array(img_grey, dtype=np.float)
    return img


def get_2d_dct(img):
    """ Get 2D Cosine Transform of Image
```

```
    """
    return fftpack.dct(fftpack.dct(img.T, norm='ortho').T, norm='ortho')


def get_2d_idct(coefficients):
    """ Get 2D Inverse Cosine Transform of Image
    """
    return fftpack.idct(fftpack.idct(coefficients.T, norm='ortho').T, norm='ortho')


def get_reconstructed_image(raw):
    img = raw.clip(0, 255)
    img = img.astype('uint8')
    img = Image.fromarray(img)
    return img


pixels = get_image()
dct_size = pixels.shape[0]
dct = get_2d_dct(pixels)
reconstructed_images = []

for ii in range(dct_size):
    print(ii)
    dct_copy = dct.copy()
    dct_copy[ii:, :] = 0
    dct_copy[:, ii:] = 0

    # Reconstructed image
    r_img = get_2d_idct(dct_copy)
    reconstructed_image = get_reconstructed_image(r_img)
```

```
    # Create a list of images
    reconstructed_images.append(reconstructed_image)

plt.figure(figsize=(16, 12))
plt.scatter(range(dct.ravel().size), np.log10(np.abs(dct.ravel())), c='#348ABD', alpha=.3)
plt.title('Амплітуда коефіцієнтів DCT відносно Порядок коефіцієнтів DCT')
plt.xlabel('Порядок коефіцієнтів DCT')
plt.ylabel('Амплітуда коефіцієнтів DCT у масштабі')
plt.show()

plt.figure(figsize=(16, 12))
plt.hist(np.log10(np.abs(dct.ravel())), bins=100, color='#348ABD', alpha=.3, histtype='stepfilled')
plt.xlabel('Амплітуда коефіцієнтів DCT у масштабі')
plt.ylabel('Кількість коефіцієнтів DCT')
plt.title('Амплітуда розподілу коефіцієнтів DCT')
plt.show()


plt.matshow(np.abs(dct[:100, :100]), cmap=plt.cm.Paired)
plt.title('Перші 2500 коефіцієнтів на сітці')
plt.show()

fig = plt.figure(figsize=(16, 16))
counter = 1
for ii in range(64):
    plt.subplot(8, 8, counter)
    plt.imshow(reconstructed_images[ii], cmap='gray')
    reconstructed_images[ii].save(f'for_1_full_images/image_{ii}.jpeg')
```

```
    reconstructed_images[ii+1].save(f'images/image_{ii+1}.jpeg')
    cs = CheckSimilarity(f'for_1_full_images/image_{ii}.jpeg', f'test_images/1_full.png', f"Image {ii} vs. Original")
    print(cs.compare_images())
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    counter += 1

plt.show()
```

Код програми для визначення схожості зображень на мові програмування *Python*:

```
import cv2
import matplotlib.pyplot as plt


def load_and_show_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    plt.figure(figsize=(12, 12))
    plt.imshow(image, cmap=plt.cm.gray)
    plt.axis("off")
    plt.show()

    return image


def brute_force_matching_with_orb_descriptors(train_image, test_image):
    orb = cv2.ORB_create()

    kp1, des1 = orb.detectAndCompute(train_image, None)
    kp2, des2 = orb.detectAndCompute(test_image, None)

    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)

    img3 = cv2.drawMatches(train_image, kp1, test_image, kp2, matches, None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.figure(figsize=(15, 15))
```

```
    plt.imshow(img3, cmap=plt.cm.gray)
    plt.axis("off")
    plt.show()


def brute_force_matching_with_sift_descriptors(train_image, test_image):
    orb = cv2.SIFT_create()

    kp1, des1 = orb.detectAndCompute(train_image, None)
    kp2, des2 = orb.detectAndCompute(test_image, None)

    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des1, des2, k=2)

    good = [[m] for m, n in matches if m.distance < 0.3 * n.distance]

    img3 = cv2.drawMatchesKnn(train_image, kp1, test_image, kp2, good, None, flags=2)
    plt.figure(figsize=(15, 15))
    plt.imshow(img3, cmap=plt.cm.gray)
    plt.axis("off")
    plt.show()


def flann_matching_with_sift_descriptors(train_image, test_image):
    sift = cv2.SIFT_create(700)

    kp1, des1 = sift.detectAndCompute(train_image, None)
    kp2, des2 = sift.detectAndCompute(test_image, None)

    FLANN_INDEX_KDTREE = 1
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
```

```
    search_params = dict(checks=50)  # or pass empty dictionary
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(des1, des2, k=2)

    matches_mask = [[0, 0] for i in range(len(matches))]
    for i, (m, n) in enumerate(matches):
        if m.distance < 0.7 * n.distance:
            matches_mask[i] = [1, 0]

    draw_params = dict(matchColor=(0, 255, 0),
                       singlePointColor=(255, 0, 0),
                       matchesMask=matches_mask,
                       flags=0)
    img3 = cv2.drawMatchesKnn(train_image, kp1, test_image, kp2, matches, None, **draw_params)
    plt.figure(figsize=(15, 15))
    plt.imshow(img3, cmap=plt.cm.gray)
    plt.axis("off")
    plt.show()

def main():
    train_image = load_and_show_image('for_7_full_images/image_200.jpeg')
    test_image = load_and_show_image('test_images/7_cropped.png')

    brute_force_matching_with_orb_descriptors(train_image, test_image)
    brute_force_matching_with_sift_descriptors(train_image, test_image)
    flann_matching_with_sift_descriptors(train_image, test_image)

if __name__ == '__main__':
```