Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

**МОДЕЛЮВАННЯ ШВИДКІСНОГО МЕТОДУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ІЗ ВПРОВАДЖЕННЯМ ОЦІНОК ВІДСТАНЕЙ ДЛЯ СКЛАДУ ОПИСУ**
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Дебре В.С.
(прізвище, ініціали)

Спеціальності 122 Компютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Дебре Віктору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання швидкісного методу класифікації зображень із впровадженням оцінок відстаней для складу опису.

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, середовище розробки Microsoft Visual Studio, платформа .NET, мова програмування C#, модульна бібліотека взаємодії OpenCV з платформою .NET та інтеграції Emgu.CV.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз швидкодії існуючих структурних методів розпізнавання зображень.
2. Розробка моделі для оцінювання відстані від дескрипторів до класів.
3. Реалізувати нові моделі класифікації зображень за правилом трикутника.
4. Порівняти результативність та продуктивність запропонованих методів класифікації з традиційними.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, оцінка швидкодії класифікаторів зображень, моделі класифікації зображень з використанням оцінювання, постановка задачі, тестові зображення, результати програмного моделювання.

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

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| Консультант з дотримання діючих стандартів та норм | Доцент Творошенко І.С. | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 10.04.2023 | |
| 2 | Аналіз завдання, підбір літератури | 11.04.23-17.04.23 | |
| 3 | Аналіз літератури з досліджуваної проблеми | 18.04.23-20.04.23 | |
| 4 | Аналіз методів аналізу зображень | 21.04.23-01.05.23 | |
| 5 | Розробка методів класифікації зображень | 02.05.23-14.05.23 | |
| 6 | Програмна реалізація | 15.05.23-23.05.23 | |
| 7 | Оформлення пояснювальної записки | 24.05.23-26.05.23 | |
| 8 | Перевірка на плагіат | 27.05.23 | |
| 9 | Рецензування | 28.05.23 | |
| 10 | Підготовка презентації та доповіді | 29.05.23-30.05.23 | |
| 11 | Занесення роботи в електронний архів | 31.05.23 | |
| 12 | Попередній захист кваліфікаційної роботи | 06.06.23 | |

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 59 с., 9 табл., 14 рис., 49 джерел.

КОМП'ЮТЕРНИЙ ЗІР, ШВИДКІСНИЙ КЛАСИФІКАТОР ЗОБРАЖЕНЬ, МОДЕЛЬ ДЛЯ ОЦІНКИ ВІДСТАНІ, ДЕСКРИПТОР, ОЦІНКА НАЛЕЖНОСТІ ДО КЛАСУ.

Об'єктом роботи є методи класифікації зображень на підставі опису із множини дескрипторів ключових точок.

Метою цієї роботи є створення моделі швидкісного класифікатора зображень на основі оцінювання відстані від дескриптора до опису еталону з використанням властивостей сторін трикутника. Класифікатор забезпечує швидку та точну класифікацію зображень у реальному часі.

Здійснено розробку та моделювання метрик оцінки зображень, заснованих на використанні правила трикутника для швидкої оцінки та віднесення вхідних зображень до класів за допомогою відстаней між дескрипторами та опису еталону.

На підставі провадження методу оцінки до попередньо обрахованих атомарних значень для кожного еталону для класифікації вдалося скороти витрати пам'яті та прискорити час виконання при збереженні достатнього рівня точності на навчальній вибірці даних.

COMPUTER VISION, FAST IMAGE CLASSIFIER, DISTANCE ESTIMATION MODEL, DESCRIPTOR, CLASS MEMBERSHIP ESTIMATES.

The object of the work is image classification methods based on a description from a set of descriptors of key points.

The aim of this work is to create a model of a high-speed image classifier based on the estimation of the distance from the descriptor to the reference description using the properties of the sides of the triangle. The classifier provides fast and accurate classification of images in real time.

Have been developed and modeled image evaluation metrics based on the use of the triangle rule for rapid evaluation and classification of input images using descriptor distances and benchmark description.

Based on the implementation of the evaluation method to the pre-calculated atomic values for each standard it was possible to reduce the memory consumption and speed up the execution time for the classification while maintaining a sufficient level of accuracy on the training data sample.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – ключові точки

ORB – Oriented FAST and Rotated BRIEF

BRIEF – Binary Robust Independent Elementary Features

BRISK – Binary Robust Invariant Scalable Keypoints

CPU – Central Processing Unit

NFT – Non Fungible Token

ШІ – штучний інтелект

# ВСТУП

Дослідження та розробка нових методів інтелектуального аналізу та оброблення багатовимірних даних є важливим напрямком розвитку комп’ютерного зору [1-8]. Особливу увагу приділяють застосуванню систем комп’ютерного зору в умовах з обмеженими ресурсами. Одним із викликів для дослідників є забезпечення швидкодії в режимі реального часу, при цьому зберігаючи якісні результати.

Потреба в високій швидкості обробки зображень стає все більш актуальною в сучасному світі. Застосунки, такі як безпілотні автомобілі, системи відео нагляду, робототехніка та розпізнавання облич, потребують миттєвого аналізу та відгуку на події. Сучасні методи обробки зображень та моделювання класифікаторів дають можливість досягти цієї швидкості, використовуючи високу обчислювальну потужність [5-14].

Традиційні методи класифікації зображень, такі як фільтри та порогова обробка, стають менш релевантними через їхню обмежену ефективність та точність. Вони можуть бути недостатньо точними для розв’язання складних задач, які вимагають від системи глибокого розуміння вмісту зображень та їх контексту. Відповідно, нові методи моделювання класифікаторів зображень, які враховують ці вимоги, стають все більш актуальними та доцільними.

Зростання обчислювальних потужностей відіграє важливу роль у розвитку сучасних методів обробки зображень та виявлення образів. Високопродуктивні обчислювальні ресурси, такі як графічні процесори та спеціалізовані апаратні пристрої для штучного інтелекту, дозволяють обробляти великі обсяги даних значно швидше, ніж раніше. Це забезпечує можливість використання більш потужних та ефективних моделей класифікаторів зображень, забезпечуючи кращу точність та швидкість оброблення.

# 1 АНАЛІЗ СУЧАСНИХ ДЕТЕКТОРІВ ДЛЯ ФОРМУВАННЯ ДЕСКРИПТОРІВ КЛЮЧОВИХ ТОЧОК

## 1.1 Методи формування ключових точок зображення

Методи формування ключових точок та їх дескрипторів призначені для виявлення особливих та стійких точок на зображенні, які можуть бути використані для розпізнавання об’єктів, порівняння зображень або стеження за рухом об’єктів [5, 6]. Проаналізуємо деякі популярні методи формування ключових точок.

SURF (Speeded Up Robust Features). Цей метод є прискореною версією SIFT та також виявляє масштабо-інваріантні ключові точки. Він використовує апроксимацію Гауссових пірамід інтегральними зображеннями та використовує вектори Хаара для опису ключових точок. SURF, в цілому, швидший, ніж SIFT, але може бути менш точним у деяких сценарі’ях.

FAST (Features from Accelerated Segment Test). Цей метод виявляє ключові точки, порівнюючи інтенсивність пікселів у деякому околі з інтенсивністю центрального пікселя. Якщо достатня кількість пікселів у околі суттєво відрізняється від центрального пікселя, точка вважається ключовою. FAST швидкий та ефективний, але не є масштабо-інваріантним або поворотно-інваріантним.

FAST-9 є покращеною версією алгоритму FAST, який використовується для виявлення ключових точок на зображенні. Основна відмінність між цими двома алгоритмами полягає в кількості пікселів, що використовуються для визначення, чи є піксель ключовим. Ось декілька пунктів, що демонструють переваги дескриптора FAST-9 порівняно з FAST [6]:

– вища точність: Основна перевага дескриптора FAST-9 полягає в тому, що він забезпечує вищу точність виявлення ключових точок порівняно з оригінальним алгоритмом FAST. Це досягається завдяки тому, що FAST-9

використовує більшу кількість пікселів для визначення, чи є піксель ключовим, тому він має менше помилок виявлення ключових точок на зображенні;

– швидкість обробки: Незважаючи на те, що FAST-9 використовує більшу кількість пікселів для визначення ключових точок, він все ще залишається швидким алгоритмом, що може обробляти зображення в режимі реального часу;

– вища стійкість до шуму: FAST-9 має вищу стійкість до шуму на зображенні, що може бути корисним при обробці зображень з низькою якістю або з великою кількістю шуму;

– простота в реалізації: FAST-9 може бути легко реалізований на різних мовах програмування, що робить його привабливим для застосування в різних проєктах та на різних платформах;

– розширені можливості налаштування: FAST-9 має більш широкі можливості налаштування, ніж оригінальний алгоритм FAST. Наприклад, можна змінити поріг детекції ключових точок або кількість пікселів, використовуваних для визначення ключових точок, щоб отримати кращий результат для конкретного проєкту або задачі.

Отже, використання дескриптора FAST-9 виявляється привабливим для застосування в різних проєктах, що потребують виявлення ключових точок на зображеннях. Його вища точність, швидкість та стійкість до шуму роблять його корисним інструментом для різних завдань, таких як відеоспостереження, медична діагностика та обробка зображень в режимі реального часу [6].

Для реалізації цього алгоритму на (рис. 1.1) спочатку визначаються ключові точки: для прийняття рішення про те, чи вважати задану точку $C$ особливою чи ні, у цьому методі розглядається яскравість пікселів на колі з центром у точці $C$ та радіусами 3 та 5 з формули (1.1)

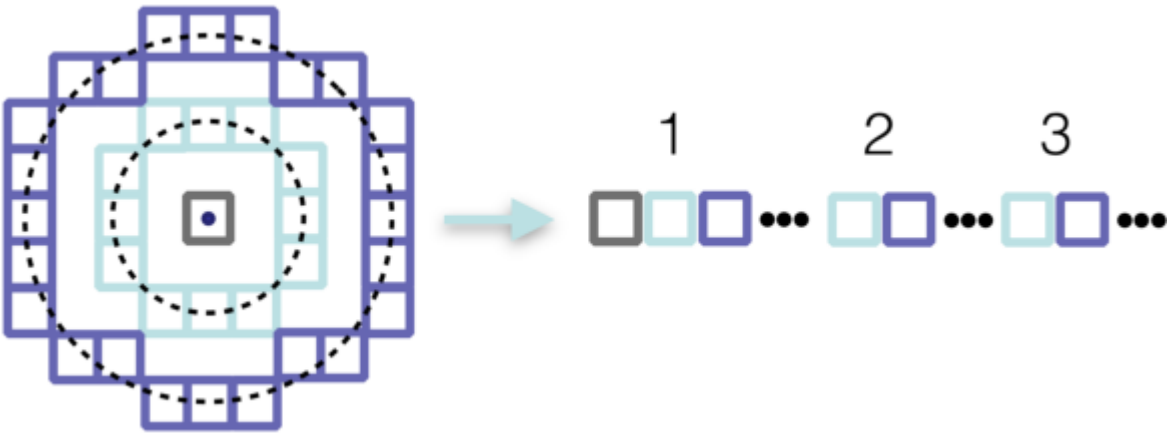



Рисунок 1.1 – Демонстрація роботи дескриптора FAST-9

Точка позначається як особлива, якщо на колі існує поспіль $n=9$ пікселів, які темніші, або 12 пікселів, які світліші, ніж центр.

$$I_C - t < I_p < I_C + t \quad I_C - t < I_p < I_C + t\,, \tag{1.1}$$

де $I$ – яскравість пікселів;

$t$ – деякий заздалегідь фіксований поріг яскравістю.

Далі створюється бінарне дерево на основі цих в точок в залежності від значень яскравості, чи воно є темнішим чи світлішим, тоді у листі дерева перебуває бінарне значення або особлива або не особлива точка, а інших вершинах дерева перебуває номер точки, яку треба аналізувати. Ентропія множини точок обчислюється як:

$$H = (c + \bar{c})\log_2(c + \bar{c}) - c\log_2 c - \bar{c}\log_2 \bar{c}\,, \tag{1.2}$$

де $c$ – число особливих точок;

$\bar{c}$ – число не особливих.

Останнім етапом алгоритму є операція не максимального придушення, щоб отримати з кількох розташованих точок одну. Для реалізації пропонується використовувати оригінальний спосіб на основі суми

абсолютних різниць між центральною точкою та точками кола в такому вигляді:

$$V = \max(\sum_{x \in S_{bright}} |I_x - I_p| - t, \sum_{x \in S_{dark}} |I_x - I_p| - t). \quad (1.3)$$

де $S_{bright}$ і $S_{dark}$ відповідно групи пікселів світліше і темніше;

$t$ – граничне значення яскравості;

$I_p$ – яскравість центрального пікселя;

$I_x$ – яскравість пікселі на колі.

ORB (Oriented FAST and Rotated BRIEF) – це метод виявлення ключових точок та дескрипторів, що базується на FAST для виявлення ключових точок і BRIEF для створення бінарних дескрипторів. ORB був розроблений для того, щоб створити ефективний, швидкий та робастний алгоритм, який може конкурувати з SIFT та SURF, але має менші обчислювальні вимоги.

KAZE – це також алгоритм виявлення ключових точок та опису. Він відрізняється від інших популярних алгоритмів, таких як SIFT та SURF, тим, що використовує нелінійну масштабно-просторову фільтрацію замість гаусової фільтрації. Та його модифікація AKAZE – це розширена версія алгоритму KAZE, яка пропонує прискорений та ефективний підхід до виявлення ключових точок та створення описувачів на зображеннях. AKAZE відрізняється від KAZE за своєю більшою швидкістю та нижчими обчислювальними вимогами. AKAZE виявляє ключові точки, використовуючи нелінійні масштабні простори, які забезпечують кращу адаптацію до геометричних та фотометричних перетворень. Він також використовує алгоритм MLDB (Modified Local Difference Binary) для створення бінарних описувачів, які забезпечують швидке порівняння та відповідність між ключовими точками.

BRISK (Binary Robust Invariant Scalable Keypoints) – це ефективний алгоритм виявлення ключових точок та створення описувачів, який базується на швидкому порівнянні бінарних рядків для знаходження відповідностей між ключовими точками. Цей алгоритм розроблений з метою забезпечення високої швидкості та точності при обмежених обчислювальних ресурсах. BRISK використовує масштабні простори для виявлення ключових точок, які стійкі до зміни масштабу. Орієнтація ключових точок обчислюється на основі градієнтів зображення, що забезпечує поворотно-інваріантність [8].

Для аналізу даних в роботі використовується алгоритм BRISK. Алгоритм BRISK працює в декілька етапів:

– побудова масштабно-просторової піраміди: вихідне зображення розбивається на декілька октав (групи рівнів), кожна з яких містить декілька рівнів зменшення роздільної здатності зображення;

– виявлення ключових точок: Виявлення ключових точок здійснюється шляхом пошуку максимальних та мінімальних значень в масштабно-просторовій піраміді, використовуючи FAST-9 детектор;

– застосовуються масштабно-інваріантні критерії відбору для виявлення точок, стійких до зміни масштабу;

– орієнтація ключових точок: Для кожної ключової точки визначається орієнтація, яка забезпечує інваріантність до повороту. Це робиться шляхом обчислення довготних та поперечних градієнтів зображення в околі ключової точки;

– обчислення бінарних описів: Відмінною особливістю BRISK є використання бінарних описів характеристик замість плаваючих чисел, що зменшує обсяг пам’яті та прискорює обчислення. В описах використовуються відносні інтенсивності пікселів в околі ключової точки з урахування.

Для опису ключових точок у методі Brisk використовуються бінарні дескриптори, які складаються з 512 бітів. Для створення цих дескрипторів метод використовує локальні патчі, які розбиваються на під-патчі (рис. 1.2).

Кожен під-патч відображається на біт дескриптора за допомогою лінійної комбінації значень пікселів [11].

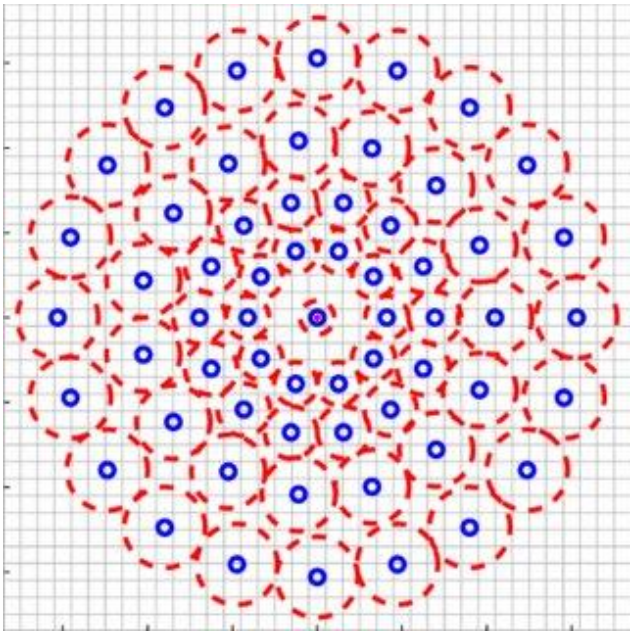


Рисунок 1.2 – Демонстрація роботи дескриптора BRISK

BRISK також може бути використаний для вирішення задач в комп’ютерному зорі, таких як визначення розміру та форми об’єктів, розпізнавання образів та зображень, класифікації даних та багатьох інших.

## 1.2 Аналіз застосування традиційних методів класифікації

Класичний метод класифікації та віднесення до еталонів – це метод розпізнавання об’єктів за допомогою порівняння їх характеристик з еталонними об’єктами, які відображають типові представники класів [14-20]. Цей метод використовується в різних підгалузях науки та техніки, таких як комп’ютерний зір, розпізнавання мови, біометричні системи тощо. Основні етапи класичного методу класифікації та віднесення до еталонів включають:

– збір даних: збір відповідних даних для представлення еталонних об’єктів та класів;

– виділення ознак: визначення набору ознак, які допоможуть представити та розрізнити об’єкти різних класів. Ознаки можуть бути простими, як кольори або форми, або складнішими, як текстури або спектральні характеристики;

– побудова еталонів: створення еталонів для кожного класу на основі обраних ознак. Еталони можуть бути статичними або динамічними;

– віднесення об’єктів до еталонів: порівняння нових об’єктів з еталонами за допомогою відстані або міри схожості. Об’єкти приписуються до класу з найбільш схожим еталоном;

– оцінка результатів: оцінка точності та ефективності класифікації за допомогою різних метрик, таких як точність, повнота, $F$-міра тощо. За потреби можна вдосконалити ознаки або еталони для покращення результатів класифікації.

Для збору та виділення ознак опису у роботі використовуються дескриптори BRISK, які знаходять ключові точки для зображення та масив дескрипторів за ними [18, 21-24]. Ці операції виконуються для всіх еталонів та зображення, яке буде оцінюватись. Для кожної ключової точки знаходиться дескриптори, які її описують, а потім між дескрипторами еталону та зображення для класифікації знаходиться хемінгова відстань, і якщо значення цієї відстані перевищує попередньо обраний поріг, тоді класифікатор додає голос до певного з еталонів. І після проходження порівнянь з усіма еталонами відбирається еталон за найбільшою кількістю голосів, і при перевищенні половини кількості дескрипторів можемо сказати про чітку класифікацію до цього еталону, при значенні голосів менше половини – класифікація є статистично нечіткою.

Для класифікації за правилом трикутника виконуються ті ж самі кроки, як і для класичного методу голосування за конкретні еталони, але для безпосередньо самого виконання голосування. Для того щоб визначити до якого класу віднести спочатку знаходяться медоїди кожного з еталонів. І для кожного медоїда знаходиться елемент з найбільшої і найменшою хемінговою

відстанню до них в кожному еталоні. Оскільки ці обрахунки виконуються тільки 1 раз для конкретних еталонів, час на їх виконання не враховується на сам час класифікації зображень. І в безпосередньо під класифікації дескриптори зображення порівнюються за хемінговою відстанню не з усіма дескрипторами еталонів, а з вибраними мінімумами для еталонів, або максимумами, залежно від навчальної вибірки та ефективності того чи іншого метода в конкретному випадку. Після чого так само за результатами голосування до еталону з найбільшої кількістю голосів відносять вхідне зображення.

Для виконання метода трикутника та порівняння за медоїдами спочатку необхідно знайти відповідно медоїди для еталонів. Для кожної із матриць порівнянь еталонів самих з собою знаходиться мінімальна сума хемінгових відстаней в рядку матриці, тобто знаходиться найближчий із медоїдів, що має найменшу відстань до найбільшої кількості елементів цього еталону.

Після отримання значення цієї найменшої суми по індексу з матриці береться дескриптор, який відповідає цьому медоїду, після чого для нього по тій самій матриці знаходиться дескриптор з найбільшою відстанню до нього та дескриптор з найменшою відстанню. Отримуємо індекси цих елементів та самі ці елементи. Вони будуть використані для метода трикутника при класифікації.

Ці медоїди з представляють собою вибрані точки даних з усього набору, що знаходиться найближче до центру вибраного класу, але і при цьому є одними з вхідних значення, на прикладі медоїдів для двомірних кластерів (рис. 1.3).

Але для відстаней буде використовуватись тільки медоїди для одномірного набору даних (рис. 1.4) з розподілом значень від 0 до 512 в рамках набору Хемінгових відстаней.

Метод оцінки за медоїдами набагато більш тривіальний та потребує в рази менше розрахунків, але і має дуже малу точність. Його суть полягає в тому що так само як і для методу трикутнику треба виконати всі кроки до

знаходження медоїдів, включно. Після чого сам етап оцінювання та віднесення до класу відбувається наступним чином: для зображення, над яким проводиться класифікація теж знаходиться медоїд, і знаходяться хемінгові відстані від цього медоїду до медоїдів кожного з еталонів, і класифікатор відносить зображення до еталону з найменшою відстанню до його медоїду.

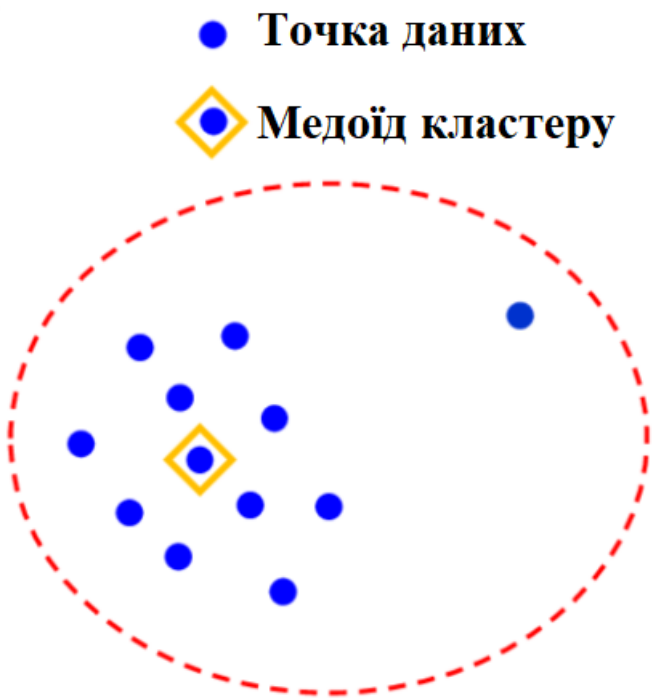



Рисунок 1.3 – Приклад медоїду на двомірному кластері даних

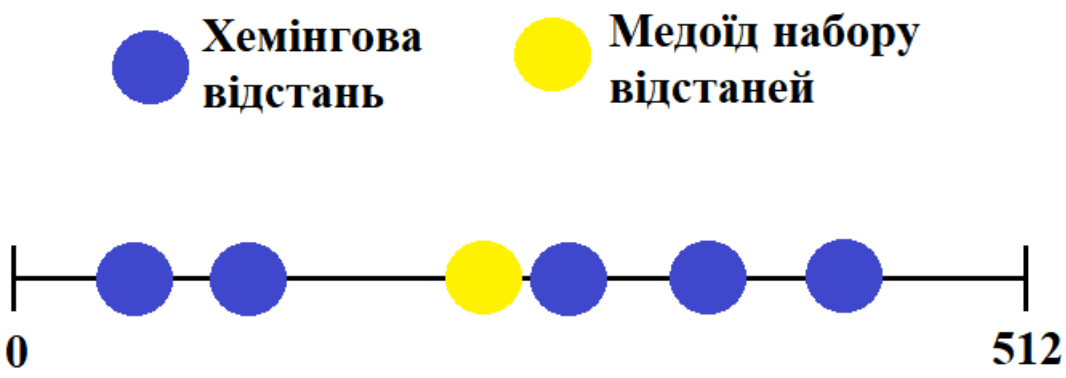



Рисунок 1.4 – Приклад медоїду на одновимірному кластері даних

## 1.3 Постановка задачі

Використання нових більш швидких та менш вибагливих методів класифікації зображень є актуальним завданням у сучасних застосуваннях комп’ютерного зору.

Ставиться завдання розроблення швидкісних методів та алгоритму для реалізації засобів оцінювання відстані від дескриптора до опису еталону з використанням властивості трикутника. На практиці необхідно застосувати дескриптор ключових точок BRISK на базі програмної бібліотеки OpenCV.

Об’єктом роботи є методи класифікації зображень на підставі опису із множини дескрипторів ключових точок.

Метою цієї роботи є створення моделі швидкісного класифікатора зображень на основі оцінювання відстані від дескриптора до опису еталону з використанням властивостей сторін трикутника. Класифікатор забезпечує швидку та точну класифікацію зображень у реальному часі.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих структурних методів розпізнавання зображень за критеріями точності і швидкодії;

– розробити модель для оцінювання відстані від дескриптора до класу за правилом трикутника;

– застосувати розроблену модель у програмному застосунку для класифікації у базі зображень;

– проаналізувати результати моделювання різних моделей та провести порівняння їх результативності та продуктивності.

# 2 МОДЕЛІ АНАЛІЗУ ДАНИХ ДЛЯ ШВИДКІСНОЇ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

З огляду на великі обсяги багатовимірних даних, які аналізуються і оброблюються в сучасних системах комп'ютерного зору, є важливим ефективне використання обчислювальних ресурсів. Для досягнення цієї мети потрібно зменшити обчислювальні витрати, які потрібні для класифікації чи розпізнавання візуальних об'єктів. Зазвичай, нові методи розпізнавання базуються на оптимальних метричних чи статистичних процедурах для пошуку відповідності між вхідними та еталонними даними, з використанням лінійного пошуку як на множині класів, так і в межах опису класу [19, 25].

Аналізовані підходи базуються на повній апріорній інформації про описи з бази еталонів, яка використовується для навчання класифікатора та визначення трансформованого простору даних, що сприяє зменшенню обчислювальних витрат.

Відомий підхід до визначення належності даних до класу за допомогою наближених оцінок, які базуються на статистичному аналізі відстаней між елементами у еталонному описі класу [20, 21, 26, 27]. Цей метод дає можливість оцінювати відстані від елемента до множини та використовувати нерівність трикутника у числовому метричному просторі. Головна ідея полягає у використанні попереднього статистичного аналізу для бази еталонних описів задля отримання параметрів для оцінювання належності даних до певного класу.

## 2.1 Схема класифікатора зображення з використанням оцінювання

Для формування складу опису та використання запропонованого метода необхідно виконати наступні кроки у відповідності до схеми (рис. 2.1).

У формалізмі задачі прискорення класифікації через оцінювання використовується підмножина об’єктів з мітками класів, що називається підмножиною навчання. За допомогою цієї підмножини будується базова модель, яка дає оцінки належності кожного об’єкту до кожного класу. Ці оцінки використовуються для зменшення кількості обчислень, необхідних для визначення класу кожного об’єкту [20, 24, 27].


Рисунок 2.1 – Алгоритм виконання класифікації

Для формування складу опису та використання запропонованого метода необхідно виконати наступні кроки:

Виконання аналізу реалізується наступними кроками.

Крок 1. На вхід подається зображення в відповідному форматі.

Крок 2. За допомогою детектора, такого як BRISK, визначити ключові точки та дескриптори на їх основі.

Крок 3. Обчислення медоїдів для описів еталонів.

Крок 4. Знаходження Хемінгової відстані до еталонів за класичним методом голосування або запропонованим методом за правилом трикутника.

Крок 5. Класифікація на підставі зіставлення з базою еталонів.

Проведення аналізу опису та подальшої класифікації в відповідності еталонів передбачає формалізм задачі прискорення класифікації через оцінювання, який відноситься до машинного навчання та полягає у побудові апроксимації моделі класифікації з метою її прискорення [22-27]. Задача полягає в тому, щоб зменшити час, необхідний для визначення класу для кожного об’єкту, зберігаючи при цьому точність класифікації.

У формалізмі задачі прискорення класифікації через оцінювання використовується підмножина об’єктів з мітками класів, що називається

підмножиною навчання. За допомогою цієї підмножини будується базова модель, яка дає оцінки належності кожного об’єкту до кожного класу. Ці оцінки використовуються для зменшення кількості обчислень, необхідних для визначення класу кожного об’єкту.

Один з можливих підходів полягає в обиранні того класу, для якого оцінка належності є максимальною [24-29]. Таким чином, задача прискорення класифікації через оцінювання є важливою задачею в області машинного навчання, оскільки дає можливість зменшити час, необхідний для обробки великих обсягів даних.

У структурних методах класифікації зображень опис $Z$ візуального об’єкта подається у вигляді скінченної множини (2.1) із $s$ дескрипторів ключових точок (КТ). Дескриптор $z_v$ – це числовий вектор розмірності $n$.

$$Z = \{z_v\}_{v=1}^{s}. \tag{2.1}$$

Опис об’єкту та еталонів – це скінченна множина багатовимірних векторів. Нехай задано множину (2.2), де база еталонів є об’єднанням із $N$ описів, що конструюють набір із $N$ розпізнаваних класів.

$$E = \bigcup_{i=1}^{N} E_i . \tag{2.2}$$

Фактично $E$ – це сукупна множина векторів-дескрипторів складу усіх еталонів (2.3)

$$E = \{E_i\}_{i=1}^{N} = \{\{e_v(i)\}_{v=1}^{s}\}_{i=1}^{N}, \tag{2.3}$$

де $i$ – номер класу;

$v$ – поточний номер елементу всередині класу;

$s$ – фіксоване число дескрипторів у кожному з еталонних описів [25, 26].

Традиційна постановка задачі класифікації зображень за описом у вигляді множини дескрипторів ключових точок зводиться до визначення ступеня релевантності двох множин багатовимірних векторів та оптимізації цього критерію на наявній множині еталонів.

При цьому одним із найбільш дієвих практичних підходів є спосіб класифікації «дескриптор об'єкта – еталон», який ґрунтується на визначенні відповідності класу для дескрипторів ключових точок [27]. Для цього обчислюється відстань: $p(z, E_i)$ від кожного дескриптора $z \in Z$ аналізованого об'єкта $Z$ до кожної із $N$ множин дескрипторів еталонів $E_i$, $i \in \{1,2,...,N\}$. На підставі мінімізації отриманої відстані визначається клас еталону, до якого найбільш ймовірно відноситься кожний із дескрипторів об'єкту. Наступними етапами є підрахунок акумульованого числа голосів для наборів дескрипторів, віднесених до кожного із еталонів $E$, на підставі чого здійснюється оптимізація числа голосів на множині класів $i \in \{1,2,...,N\}$ і визначається клас аналізованого об'єкту.

У такому підході ключовим з точки зору критерію швидкодії обчислення є впровадження правила класифікації:

$$z \to [1,2,...,N], \tag{2.4}$$

довільного вектора $z \in Z$ до одного із $N$ класів шляхом обчислення відстані $\rho$ від елемента $z \in Z$ об'єкта до класу $E_i$.

$$\rho(z, E_i) = \min_{v=1,...,s} (\rho(z, e_v(i))). \tag{2.5}$$

Зважаючи на те, що число s дескрипторів у описі обчислюється кількома сотнями ( $s$ = 500 і більше), а послідовний аналіз усієї сукупності еталонів мультиплікативно збільшує необхідний об'єм обчислень, важливою прикладною задачею є впровадження засобів скорочення обсягу обчислень

при реалізації (2.5), наприклад, шляхом оцінювання відстані (2.5) з повноцінним використанням наявної вхідної інформації та умов класифікації, за яких описи еталонів $E_i$ є множинами числових векторів і вважаються апріорно заданими. Модель (2.5) реалізує пошук за принципом «найближчого сусіда», коли шукається об'єкт, найближчий за відстанню до запиту [30].

Запровадимо для прискорення обчислення (2.5) всередині еталонного опису спосіб, що ґрунтується на визначенні оцінки $\tilde{\rho}$ для відстані (2.5) у вигляді.

$$\tilde{\rho}(z, E_i) = F(z, E_i, \rho), \tag{2.6}$$

де $F$ – деяке правило формування оцінки $\rho$, результат застосування якого параметрично залежить від аналізованого дескриптора, множини дескрипторів опису і виду відстані $\rho$.

Оцінка відстані між дескрипторами використовується для порівняння та сумісності між ключовими точками різних зображень. Різні метрики відстані можуть бути використані для різних типів дескрипторів. Часто використовується Хемінгова відстань – це метрика, яка вимірює відстань між двома рядками або бітовими послідовностями однакової довжини [30]. Щоб знайти Хемінгову відстань між двома рядками, потрібно порівняти кожну позицію в одному рядку з відповідною позицією в іншому рядку. Якщо символи в обох позиціях відрізняються, то Хемінгова відстань збільшується на 1. Якщо символи в обох позиціях співпадають, то Хемінгова відстань не змінюється [31].

Наприклад, рядок «10101» має Хемінгову відстань 2 від рядка «11100», тому що символи в позиціях 2 та 4 відрізняються.

Формально, якщо ми маємо два рядки $A$ та $B$ довжини $n$, то Хемінгова відстань між ними обчислюється як:

$$d_h(A,B) = sum(A_i \neq B_i), \quad (2.7)$$

де $A_i$ та $B_i$ – символи відповідної позиції в рядках $A$ та $B$,

$sum(A_i \neq B_i)$ – сума, що береться по всім позиціям в рядках $A$ та $B$.

## 2.2 Правило трикутника і його застосування для оцінювання

Для виконання безпосередньо класифікації уже існує класичний метод та запропонований метод сформований за правилом трикутника. Для нього розглянемо формальне подання правила $F$ з (2.6), скориставшись властивостями сторін трикутника на площині. Нехай $a$, $b$, $c$ – довжини сторін (рис. 2.2). Тоді для кожної із них, наприклад, для a , дійсні наступні умови, що випливають із нерівності трикутника:

$$c - b \leq a \leq a + b. \quad (2.8)$$

Застосуємо оцінку (2.8) у метричному просторі багатовимірних векторів (дескрипторів КТ). Нехай $e^*(i) \in E_i$ – деяка фіксована точка множини $E_i$ розглянемо точку $d(i) \in E_i^*$, що належить множині $E_i^* = E_i \setminus e^*(i)$ з виключенням $e^*(i)$. Тоді відстань $a_i = \rho(z, d(i))$ від точки $z$ об'єкта до точки $d(i)$ множини можна оцінити через обчислену відстань $b_i = \rho(z, e^*(i))$ та попередньо визначену відстань $c_i = \rho(d(i), e^*(i))$ (рис. 2.2).

$$\rho(d(i), e^*(i)) - \rho(z, e^*(i)) \leq \rho(z, d(i)) \leq \rho(d(i), e^*(i)) + \rho(z, e^*(i)). \quad (2.9)$$

Введемо позначення: $c_{i,\min} = \min\limits_{d(i)\in \mathrm{E}_i^*} \rho(d(i), e^*(\mathrm{i}))$,

$c_{i,\max} = \max\limits_{d(i)\in \mathrm{E}_i^*} \rho(d(i), e^*(\mathrm{i}))$ з рисунку 2.2.

Зважаючи на необхідність отримання найбільш точної оцінки, що визначається найкоротшим розміром інтервалу у нерівності (2.9), цілеспрямовано обираємо найбільше значення лівої частини (2.9) серед точок множини і найменше значення її правої частини [32].

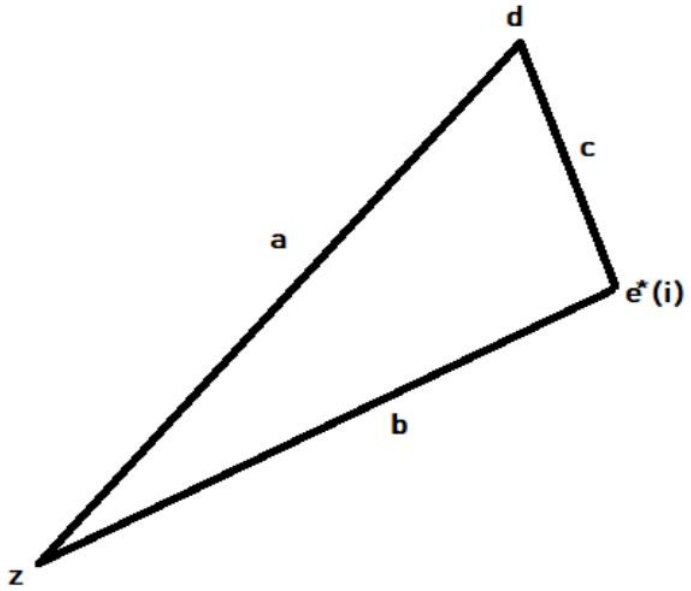



Рисунок 2.2 – Демонстрація геометричної інтерпретації нерівності трикутника (2.9)

У результаті приходимо до оцінки:

$$c_{i,\max} - b_i \le \rho(\mathrm{z}, \mathrm{E}_i) \le c_{i,\min} + b_i . \qquad (2.10)$$

На рисунку 2.2 наведено геометричну інтерпретацію нерівності (2.10) для спрощеної ситуації: двовимірний простір, а також типовий випадок взаємного розташування аналізованих елементів. Тут $M$, $O$, $D$, $Dm$, $DM$ – точки на площині, які задають відповідно розташування медоїда, окремого

дескриптора об’єкта $z$, окремого дескриптора еталону $d$ (відмінного від медоїда), дескриптора еталону, найближчого до, дескриптора еталону, найвіддаленішого до (рис. 2.3) (для одного окремо взятого еталону) [33-39].

Нехай також: $D_1$ – така точка, що $MD_1 = c_{\min}$ , $OD_1 = a$; $D_2$ – така точка, що $MD_2 = c_{\max}$, $OD_2 = a$.

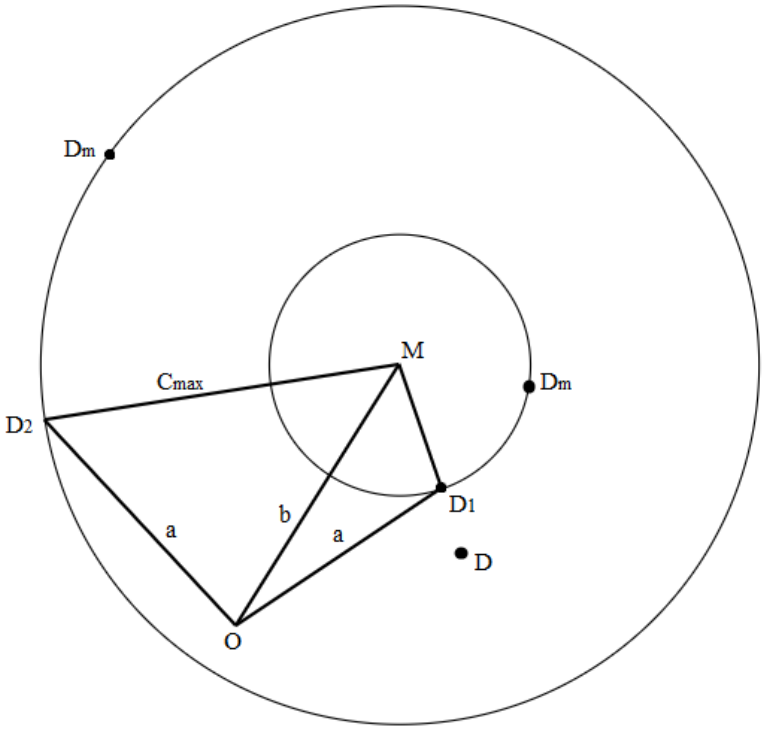



Рисунок 2.3 – Геометрична інтерпретація оцінки

Тоді за нерівністю трикутника (2.8) маємо -

– $\triangle OMD_1$: $a \le c_{\min} + b$;

– $\triangle OMD_2$: $a \ge c_{\max} - b$.

Зауважимо, що результат оцінки (2.10) параметрично залежить як від складу множини $E_i$, так і від обраної точки $e^*(i) \in E_i$ всередині множини $E_i$. Результат оцінювання на підставі (2.10) – це деяке числове значення $\rho_i$ для відстані (2.6) від дескриптора до класу з номером $i$.

Складові $c_{i,\min}$, $c_{i,\max}$ оцінки (2.10) безпосередньо можна отримати на етапі попереднього аналізу наявних еталонних даних, тому на обсяг обчислень у процесі класифікації її визначення не впливає. Для отримання поточної оцінки $\rho_i$ необхідно тільки обчислити відстань $b_i$ від запиту до точки $e^*$(i) множини $E_i$.

## 2.3 Класифікація на підставі оцінювання

За результатом застосування моделі оцінювання (2.10) для кожного дескриптора об'єкта $Z$ класифікатор здійснює деяку процедуру аналізу відповідно до деякої моделі $A$

$$k = \underset{i=1,\ldots,N}{\text{opt}} \; \text{A}\,\{c_{i,\max} - b_i \le \rho(z, E_i) \le c_{i,\min} + b_i\}, \qquad (2.11)$$

що узагальнює значення оцінок, отриманих для бази еталонів $\{E_i\}$, і визначає еталон з найкращою оцінкою [34-37].

Впровадимо модель $A$ у вигляді оптимального вибору класу дескриптора як визначення мінімуму серед отриманих оцінок зверху як (2.12)

$$k = \arg\min_{i=1,\ldots,N} \left(c_{i,\min} + b_i\right). \qquad (2.12)$$

У випадку неоднозначного визначення $k$ за формулою (2.12), якщо рівноцінний мінімум досягнуто одразу для кількох класів, здійснимо вибір класу дескриптора шляхом визначення максимуму серед отриманих оцінок знизу (ліва частина нерівності (2.10)):

$$k = \arg\max_{i=1,\ldots,N} \left(c_{i,\max} - b_i\right). \qquad (2.13)$$

Як показує досвід, для об’ємного векторного масиву даних, яким є дескрипторний опис, малоймовірна ситуація, коли оцінки (2.12), і навіть оцінка (2.13) співпадуть між собою. Але у цьому випадку формально можна застосувати більш складні варіанти аналізу двох і більше точок множини $E_i$ або двох і більше найближчих до неї точок множини $E_i$.

Взагалі, такі методи допомагають збільшити точність оцінювання, знижуючи при цьому швидкість. Крім того, у випадку неоднозначної оцінки (2.12), можна скористатися простим практичним способом віднесення дескриптора до одного з неоднозначно оцінених класів або до всього набору класів, для яких отримано однаковий мінімум [35].

Формально будь-яка точка $e^*(i) \in E_i$ множини може слугувати точкою оцінювання. Проте, детальний аналіз показує, що для забезпечення достатньої точності оцінювання необхідно вибрати «внутрішню» точку множини [38, 39]. До таких точок можна віднести медоїд як вимір «центру множини», геометричний центр, центр ваги (середнє значення компонентів), середню точку від діаметру множини, центр описаного багатовимірного шару навколо точок множини тощо.

Важливо що параметр медоїду, що є точкою множини з мінімальною сумарною відстанню до решти точок. Медоїд обчислюється наступним чином: він є елементом множини, і його можна універсально застосувати при довільній вимірності даних. На основі цього медоїду результативно можна будувати нові класифікаційні ознаки для сукупності точок як структурного опису об’єкта.

Якщо множина дескрипторів складається з однакових точок (векторів), то для оцінки відстаней до еталонів можна безпосередньо використовувати самі відстані до будь-якої точки множини. Таким чином, використання оцінки виду (2.10) з її моделями використання (2.11), (2.12), (2.13) у процесі класифікації дозволяє уникнути складної процедури об’ємного лінійного пошуку при обчисленні (2.5) і суттєво (пропорційно об’єму опису) скоротити

обчислювальні витрати. Для визначення обраної точки множини можна використати, наприклад, матрицю внутрішніх відстаней, яка містить усі попарні відстані між елементами множини.

Для визначення класу цілісного об'єкту $Z$ на підґрунтя його складу із $s$ інгредієнтів введемо вектор $\{h_i\}_{i=1}^{N}$ з цілими значеннями, де будемо накопичувати отримані номери (голоси) класів для кожного компонента $z \in \mathrm{Z}$. На підставі впровадження локального класифікатора для кожного $z \in \mathrm{Z}$ відповідно до (2.11) визначимо тепер номер класу $k$ та інкрементуємо акумулятор $h_k = h_k + 1$ для відповідного номеру класу.

За результатом оброблення усього опису $Z$ об'єкту накопичуємо вектор $\{h_i\}_{i=1}^{N}$. Клас об'єкту визначимо традиційно як аргумент від максимуму числа голосів:

$$k = \arg \max_{i=1,\ldots,N} h_i \,||\, \mathrm{h}_k \geq \delta_h, \qquad (2.14)$$

де $\delta_h$ – деякий поріг для граничного мінімального числа голосів, що встановлюється експериментально для заданої бази. Якщо нерівність $h_k \geq \delta_h$ у (2.14) не виконана, клас об'єкту вважається не встановленим.

Перевага в часі від впровадженого нововведення (2.11) замість схеми лінійного пошуку на множині із $s$ компонентів пропорційний зростанню значення $s$, яке може досягати кількості дескрипторів при визначенні зображення. Витрати полягають у витраченні ресурсів на визначення оцінки (2.11), а накопичення голосів і віднесення до класу відповідно до (2.14) подібне як для традиційного, так і для модифікованого підходів.

Оцінювання (2.6) і результат його застосування у вигляді (2.12) може бути узагальнено шляхом впровадження таких способів:

– використання замість однієї ключової точки для множини двох і більше точок множини;

– використання для однієї точки двох і більше найближчих до неї точок множини;

– застосування додаткових логічних правил для підвищення достовірності оцінки при віднесенні дескриптора об’єкту до еталону. Такі правила можуть базуватися на інших параметрах множини, таких як діаметр, найбільш віддалена точка та ін.

Запропоновані підходи націлені на розширення засад із точок множини, за якими здійснюється оцінювання [40, 41]. Однак, їх застосування загалом ускладнює оброблення і зменшує виграш у часі у порівнянні з оцінюванням (2.11), (2.12), (2.13).

Для класифікації зображення в роботі використовується наступні методи: класичний метод голосування, альтернативний метод оцінювання за правилом трикутника, який використовує в тому числі класичний метод голосування, а також метод за медоїдами на основі оцінювання відповідності їх значень для еталонів.

## 2.4 Аналіз швидкодії методів

Для теоретичної оцінки швидкодії виконання методів голосування можна застосувати метрику Big O-нотація. Це спосіб оцінки складності алгоритму за часом, який він потребує для виконання у найгіршому випадку. Вона вимірює асимптотичну складність алгоритму, фокусуючись на зростанні кількості операцій, які алгоритм виконує відносно збільшення розміру вхідних даних [42].

Big O-нотація виражає складність алгоритму у вигляді функції:

$$t = O(f(n)), \tag{2.15}$$

де $f(n)$ є функція, яка описує кількість операцій;

$n$ – розмір вхідних даних.

Основні приклади Big O-нотацій включають:

– $O(1)$ – константна складність: час виконання алгоритму не залежить від розміру вхідних даних;

– $O(\log n)$ – логарифмічна складність: час виконання алгоритму зростає логарифмічно відносно розміру вхідних даних;

– $O(n)$ – лінійна складність: час виконання алгоритму зростає лінійно відносно розміру вхідних даних;

– $O(n \cdot \log n)$ – лінійно-логарифмічна складність: час виконання алгоритму зростає пропорційно добутку розміру вхідних даних та логарифму розміру вхідних даних;

– $O(n^2)$ – квадратична складність: час виконання алгоритму зростає пропорційно квадрату розміру вхідних даних.

Big O-нотація допомагає проаналізувати, наскільки ефективно алгоритм працює для різних обсягів вхідних даних і дозволяє порівняти різні алгоритми за їх потенційною продуктивністю [43].

Для класичного алгоритму порівняння дескрипторів передбачає перебір всіх відстаней від дескрипторів шуканого зображення, до кожного дескриптора, кожного з еталонів, що задає наступну складність по часу:

$$t = O(\sum_{i=1}^{k} m_i n), \tag{2.16}$$

де $n$ – кількість дескрипторів в оцінюваному зображенні;

$m_i$ – кількість дескрипторів в $i$-тому еталоні;

$k$ – кількість еталонів.

В той час як для методу оцінки за методом трикутника за рахунок попередніх прорахунків та знаходження медоїдів для еталонів зображення сам процес можна прискорити

$$O(\sum_{i=1}^{k} m_i n) \to O(k \cdot n). \qquad (2.17)$$

І для (2.17) позначення для такого ж як (2.16), з цього можна виокремити, що для запропонованого методу є значна перевага в часі, бо в даному методі не потрібно буде знаходити Хемінгову відстань до кожного дескриптора кожного еталону, а лише до одного медоїда в кожному з еталонів.

# 3 КОМП'ЮТЕРНА МОДЕЛЬ ШВИДКІСНОЇ КЛАСИФІКАЦІЇ. РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

## 3.1 Обґрунтування вибору інструментарію програмної реалізації

У рамках кваліфікаційної роботи був розроблений алгоритм для класифікації зображень за методом трикутника та оцінки складу опису. Цей метод має перевагу в швидкості у порівнянні з класичним методом. Для реалізації застосунку і оцінювання зменшення часу новим методом було обрано мову програмування C# та фреймворк розробки .NET 6 та розширення для відкритої бібліотеки OpenCV для цієї платформи під назвою Emgu.Cv, застосунок було розробленно на платформі операційної системи Windows 10 та у IDE Visual Studio 2022.

Для реалізації було обрано платформу .NET. Платформа .NET від Microsoft має ряд переваг при розробці програмного застосунку для розпізнавання зображень та детального складу опису. Ось деякі з них:

– на практиці C# має високу швидкість розробки, тому що .NET має потужні бібліотеки і фреймворки, такі як ML.NET, Emgu.Cv та ін., що сприяють швидкій розробці алгоритмів розпізнавання зображень та обробки даних;

– вона має кросплатформеність: тобто завдяки платформі .NET 6 можливо створювати та розгортати застосунки на різних платформах, таких як Windows, Linux і macOS, без зміни коду;

– платформа має високу продуктивність: .NET надає відмінну продуктивність завдяки оптимізованому управлінню пам'яттю, паралельному виконанню коду та Just-In-Time (JIT) компіляції;

– забезпечення безпеки типів та коду. Платформа .NET має вбудовані механізми безпеки, такі як авторизація, аутентифікація та захист від атак на рівні системи, що забезпечують захист даних та коду. Також в ній є

вбудований збірник сміття з 3-ма поколіннями, що забезпечує правильне очищення та перезапис даних;

– також .NET має велику спільноту розробників, яка надає підтримку та ресурси для навчання та вирішення проблем;

– багато компонентів .NET є відкритими за ліцензією MIT, що дозволяє розробникам долучатися до розвитку платформи, вносити свої внески та отримувати доступ до останніх оновлень та виправлень;

– інтеграція з різними базами даних: .NET підтримує велику кількість баз даних, таких як SQL Server, PostgreSQL, MySQL, Oracle, SQLite та ін., що дозволяє легко зберігати та обробляти дані, пов’язані з розпізнаванням зображень та детальним описом;

– платформа .NET легка в тестуванні та має потужні інструменти для Unit testing, інтеграційного тестування та автоматизованого тестування, що допомагає забезпечити високу якість коду та стабільність застосунку;

– візуальні інструменти, такі як Visual Studio, спрощують процес розробки, надаючи графічний інтерфейс для створення, редагування та налагодження коду.

З урахуванням всіх цих переваг, .NET є потужною платформою для розробки програмного застосунку по розпізнаванню зображень та детального складу опису, надаючи розробникам гнучкість, продуктивність, безпеку та широкі можливості інтеграції.

В якості середовища програмної розробки для платформи було обрано IDE Visual Studio 2022.

Visual Studio 2022 є останньою версією популярного інтегрованого середовища розробки (IDE) від Microsoft, яке має ряд переваг для розробників:

– має 64-бітну архітектура: Visual Studio 2022 має 64-бітну архітектуру, що дозволяє IDE використовувати більше пам’яті, покращуючи продуктивність та швидкість роботи, особливо при роботі з великими проєктами;

– гнучкість налаштувань: Visual Studio 2022 надає розширений набір опцій налаштувань для персоналізації робочого середовища згідно з потребами та вподобаннями розробника;

– інтелектуальне кодування: Visual Studio 2022 включає удосконалення IntelliSense, що спрощує написання коду завдяки автодоповненню, переходу до визначень, підказкам, навігації та іншим розумним функціям;

– вбудовані інструменти розв'язання проблем: Visual Studio 2022 має набір вбудованих інструментів для налагодження коду, профілювання, тестування та аналізу продуктивності, що допомагає знайти та виправити помилки та проблеми з продуктивністю.

В якості системи контролю версій для підтримки проєкту було обрано Git, бо ця система дозволяє зберігати всю історію змін, що забезпечує безпеку від того, що програма може зламати весь функціонал, оскільки можна повернутись до попередньої версії для безпеки.

В якості віддаленого репозиторію для зберігання коду було обрано Github оскільки, це сайт має інтеграцію з системою Git та завантажує всю історію Git на свої репозиторії, крім того в ньому можна налаштувати правильний доступ, та це сховище є безпечним, для того щоб з нього не можна було зламати результати досліджень.

На наступному рисунку 3.1 можна побачити історію змін в системі Git.

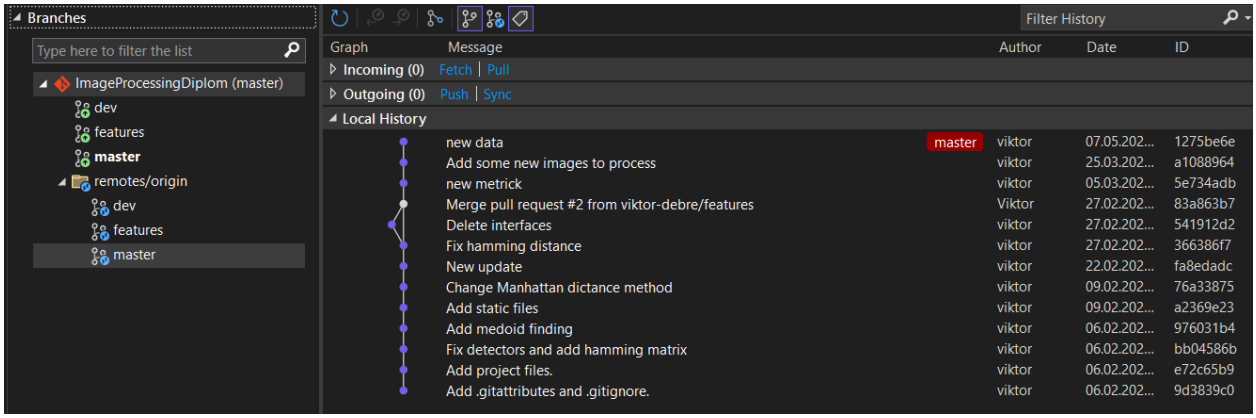



Рисунок 3.1 – Історія в системі контролю версій Git

Працювати з системою Git та віддаленим репозиторієм Github можна через термінал або інтерфейс, вбудований в IDE Visual Studio 2022, який надає можливість управляти історією змін, надавати доступ іншим користувачам та об’єднувати зроблені зміни між собою за допомогою механізму гілок.

Для обробки зображень в ході хоботи було обрано бібліотеку OpenCV (Open Source Computer Vision Library) – це відкрита бібліотека, розроблена спеціально для обробки зображень та комп’ютерного зору. Вона має ряд переваг при розпізнаванні образів:

– має відкритий код: OpenCV має відкритий код і розповсюджується за ліцензією BSD, що дозволяє розробникам вільно використовувати, модифікувати та розповсюджувати код;

– підтримка мов програмування: OpenCV підтримує ряд мов програмування, таких як C++, Python, Java та C#, що дозволяє розробникам використовувати свої улюблені мови для реалізації алгоритмів розпізнавання образів;

– багато алгоритмів: OpenCV містить велику кількість алгоритмів обробки зображень та комп’ютерного зору, таких як фільтрація, перетворення, сегментація, відстеження об’єктів, розпізнавання облич, аналіз тексту на зображеннях та багато іншого;

– висока продуктивність: OpenCV оптимізована для високої продуктивності та низької затримки завдяки використанню власних оптимізацій та апаратних прискорювачів, таких як CUDA та OpenCL;

– легкість інтеграції: OpenCV може бути легко інтегрована з іншими бібліотеками, фреймворками та інструментами, що дозволяє розробникам побудувати складні системи розпізнавання образів;

– активна спільнота: OpenCV має активну спільноту розробників та користувачів, яка надає підтримку, ресурси та навчальні матеріали для допомоги розробникам у розумінні та використанні бібліотеки;

– документація та навчальні матеріали: OpenCV має хорошу документацію, включаючи посібники, приклади коду, статті та відео, які допомагають розробникам швидко освоїти та застосувати алгоритми розпізнавання образів;

– відтворюваність результатів: завдяки відкритому коду та активному розвитку, OpenCV є відмінною основою для наукових досліджень та експериментів, дозволяючи відтворити результати та порівняти різні методи розпізнавання образів;

– ураховуючи всі ці переваги, OpenCV є відмінним вибором для розробки програмного забезпечення та систем розпізнавання образів, дозволяючи розробникам створювати ефективні, надійні та гнучкі рішення у сфері комп'ютерного зору.

Для реалізації на .NET бібліотеки OpenCV використано NuGet package, який реалізує основні функції розпізнавання зображень. Emgu.CV – це кросплатформенна обгортка .NET для OpenCV, що дозволяє розробникам використовувати функціональність OpenCV у своїх програмах [43]. Ця бібліотека має наступні переваги:

– спрощує взаємодію з OpenCV через використання об'єктно-орієнтованого програмування;

– легка інтеграція з іншими бібліотеками та фреймворками .NET;

– підтримка різних версій OpenCV, включаючи останні релізи;

– дозволяє розширити можливості .NET-застосунків у сфері комп'ютерного зору та обробки зображень.

Але для запуску Emgu.CV на .NET 6 необхідна ще додатковий NuGet package для відтворення його саме на операційній системі windows 10 Emgu.CV.runtime.windows та версії цих бібліотек 4.0.6 (рис. 3.2).

Для забезпечення розробки атомарністю та розбиттям на сервіси та відповідності стандартам, програмна реалізація розбита на різні сервіси і для вбудування сервісів та використання їх в усіх модулях програми використано

пакет NuGet package Microsoft.Extensions.DependencyInjection, для впровадження незалежності модулей Dependency між собою.

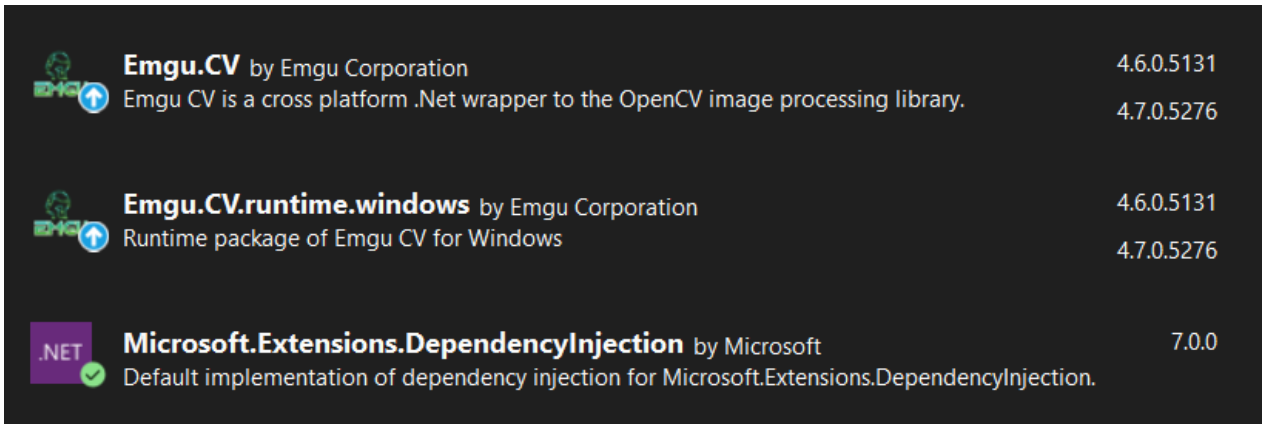



Рисунок 3.2 – Набір NuGet package для розробки

Також ця бібліотека дозволить відповідати патернам функціонального програмування та швидко виконувати написаний код та правильно звільняти ресурси пам’яті під час безпосередньо виконання програми.

## 3.2 Результати моделювання

Для вхідної вибірки еталонів було вибрано NFT зображення (рис. 3.3), для обробки зображення перетворено на монохромне для спрощення алгоритму (рис. 3.4).

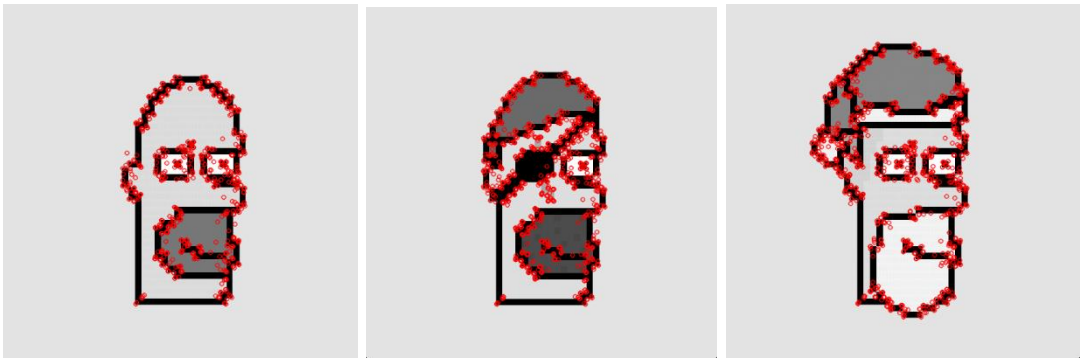
Далі на зображенні було проставлено ключові точки, знайдені за допомогою детектора BRISK, їх виділено для наочності червоними колами (рис. 3.4).

Для вибірки було обрано 3 еталони для класифікації та по 500 дескрипторів в еталонах, знайдених по ключовим точкам. Оскільки дескриптори знайдені за допомогою детектору BRISK і мають максимальне значення в 8 бітів, їх кількість для однієї ключової точки становить 64 дескриптори, тобто сумарно їх описує значення в 512 бітів. Але вони

зберігаються в форматі цілих чисел, перетворених з двійкової системи в десяткову як масив значень з 64 елементів. Кожен з них приймає значення від 0 до 255, адже спочатку складається з 8 бітів і може приймати будь яке значення в діапазоні від 0 до $2^8 - 1$ тобто 1 байт інформації. Для подальших обрахунків ці дескриптори потім знову перетворюються на бітовий формат.

а) б) в)

Рисунок 3.3 – Приклад вхідних NFT зображень:
а) перше зображення; б) друге зображення; в) третє зображення



а) б) в)

Рисунок 3.4 – Приклад ключових точок, виділених на монохромному зображенні:
а) перший еталон; б) другий еталон; в) третій еталон

В програмній реалізації далі за алгоритмом відповідно були реалізовані такі функції, які дозволять оцінювати хемінгову відстань та проводити голосування.

Для методу оцінки відстані за допомогою оператора xor порівнювалися всі біти 2 дескриптора між собою, і потім за допомогою побітового зсуву відшукувалась кількість одиничних результатів операції xor. Ця сума і представляє собою хемінгову відстань між дескрипторами.

Для кожного еталону розраховано матрицю відстаней кожного дескриптора з кожним, тобто для 1 еталону відстані дескрипторів між собою з 1,2,3 і для інших так само, тобто сумарно 9 матриць відстаней.

Але при цьому ці матриці будуть дзеркальними, і порівняння однакових еталонів, але в різному порядку, тобто 1 та 2 еталону або 2 та 1 будуть однаковими. Отже, можна прораховувати лише кількість відстаней, рівну наступному, в загальному випадку це сума від 1 до кількості еталонів.

Для реалізації класичного методу класифікації необхідно розробити методи порівняння Хемінгової відстані кожного дескриптора із еталонів з кожним дескриптором зображення, яке потрібно класифікувати. Коли відстань знайдено, вона порівнюється з порогом, який попередньо визначається, цей поріг відбиратиме тільки значущі дескриптори [43, 44]. А ті, що не мають достатнього наближення до еталонів, оцінюватися не будуть. Для оцінки використовувались пороги відстані в 64 та 128, тобто $\frac{1}{8}$ та $\frac{1}{4}$ максимальної відстані в 512 одиниць. Далі для мінімуму відстані, що пройшли поріг з 3 еталонів зарахувати голос до 1 з класів. Коли по всім 500 дескрипторам з еталонів знайдено голоси, за найбільшою кількістю голосів призначається зображенню призначається клас еталону.

Відповідно для цих операцій відбувається прохід по циклу масиву дескрипторів довжиною 500 елементів. Далі реалізується вкладений цикл для кожного кроку попередньому, в якому порівнюється 1500 дескрипторів з еталонів, та знаходиться мінімальний елемент та проводиться голосування [45].

Разом з результатами відслідковується час для виконання алгоритму для подальшого порівняння методів та оцінки їх ефективності. Також результати записуються в файл для збору статистики відпрацювання.

Для перевірки правильної роботи методу проведено голосування при віднесенні зображення з 1 еталону до цих 3 класів на рисунку 3.5.

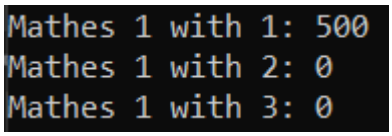



Рисунок 3.5 – Приклад результатів виконання програми класичним методом

Як видно з цих результатів, класифікатор відпрацював правильно. Визначено 500 дескрипторів до 1 еталону, усі віднесено до 1 класу.

Для метода трикутника, так само як для і для класичного методу, ті самі методи для знаходження відстані, але в якості порівняння береться елемент з мінімально або максимальної відстанню від медоїду еталонів.

Для перевірки аналогічно класичному методу для методу трикутника здійснено порівняння 1 зображення з еталонами на рисунку 3.6. Як бачимо з результатів методу – класифікація теж виконана правильно, бо 1 еталон набрав більше половини голосів, і це означає чітке віднесення до 1 класу.

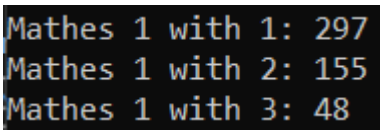



Рисунок. 3.6 – Приклад результатів виконання програми методом оцінки до медоїдів

Для методу медоїдів оцінка проводиться окремо, додатково знаходиться медоїд зображення, який порівнюється з медоїдами еталонів за Хемінговою відстанню, і при найменшому значенні цієї відстані зображення відноситься до цього класу [46-48]. Також було використано подібний до

попереднього метод оцінки, при якому знаходиться медоїд для зображення, яке буде класифікуватись. Після чого оцінюються відстані від цього медоїду до інших медоїдів еталонів, і Хемінгова відстань, що має найменше значення, вибирається як еталон. І метод відносить задане зображення до цього класу. Але недоліком цього методу буде розрахунок медоїду в зображенні для класифікації, тому він виконуватиметься повільніше, ніж попередні методи, але швидше, ніж класичне голосування. Адже на сам процес голосування витрачається лише кількість операцій, що рівна кількості еталонів.

Для реалізації всіх описаних методів використовувались 2 підходи програмування, а саме: ООП та функціональне програмування. А також для відповідності сучасній архітектурі програмних застосунків – програма відповідає принципам SOLID.

Об’єктно-орієнтоване програмування (ООП) – це парадигма програмування, яка використовує «об’єкти» дані, структури і процедури разом, як основні будівельні блоки програм. Вона заснована на ідеї, що програма може бути складена зі множини об’єктів, що взаємодіють між собою.

ООП базується на чотирьох основних принципах:

– інкапсуляція означає, що стан об’єкта захищений від прямого доступу. Всі внутрішні дані об’єкта та його поведінка (методи) сховані від зовнішнього світу. Доступ до них можливий тільки через спеціальні методи: гетери і сетери;

– наслідування дозволяє створювати нові класи на основі вже існуючих, переймаючи їх властивості та поведінку. Це дозволяє уникнути повторення коду та спрощує розширення та підтримку програм;

– поліморфізм означає, що один і той же метод може мати різну реалізацію в різних класах. Це дозволяє програмі працювати з об’єктами різних класів через єдиний інтерфейс, що значно спрощує код та збільшує його гнучкість;

– абстракція дозволяє розглядати об’єкт відокремлено від його внутрішньої структури та поведінки. Це допомагає сконцентруватися на тому, що об’єкт робить, а не на тому, як він це робить.

Функціональне програмування – це парадигма, що зосереджена на обчисленнях як результаті застосування функцій до вхідних даних.

Основні характеристики функціонального програмування включають незмінність даних, функції вищого порядку, чисті функції, рекурсію та відкладені обчислення. Незмінність даних означає, що дані не можуть бути змінені після створення. Функції вищого порядку можуть приймати інші функції, як аргументи, або повертати їх як результат. Чисті функції завжди повертають однаковий результат для одних і тих самих аргументів і не мають побічних ефектів. Рекурсія часто використовується замість циклів для повторення операцій. Відкладене обчислення дозволяє уникнути непотребних обчислень шляхом виконання обчислень лише тоді, коли результат дійсно потребується.

Функціональне програмування може бути корисним при розв’язанні задач, які можна розбити на незалежні підзадачі, оскільки це полегшує розподіл обчислень та паралельну обробку. Воно також полегшує написання безпечного та передбачуваного коду, оскільки немає спільного стану, який може бути змінений.

Використання об’єктно-орієнтовного підходу в цій програмній реалізації дозволить перейти до абстракцій та прискорить швидкість розробки, а також допоможе автоматично керувати пам’яттю, що значно знизить складність написаного коду та зробить його набагато зрозумілішим. Але швидкодія програми в нашому випадку є теж дуже важливою. Тому в частині реалізації, де безпосередньо використовуються алгоритми та обчислення, будемо користуватися функціональним підходом, що дозволить максимально детально керувати будь-яким із алгоритмів класифікації та швидко модифікуючи його.

Тобто, дана програмна реалізація використовує переваги цих двох підходів програмування та їх використання, і оскільки дана програма є тільки прототипом. Такий підхід дасть можливість швидко переписати цю програму на будь яку іншу мову програмування, яка підтримує ООП принципи та в якої є підтримка бібліотеки OpenCV, або навіть декомпозувати на елементи з низьким рівнем абстракції, такі як C++ для виконання швидкими модулями.

Для підтримання правильної та гнучкої архітектури розроблений код також відповідає принципам проєктування SOLID. SOLID – це акронім, що використовується для позначення п’яти основних принципів об’єктно-орієнтованого програмування і дизайну архітектури, які сприяють розробці програмного забезпечення, яке є легким для розуміння, підтримки та розширення [49]. Вони включають:

– Single Responsibility Principle (SRP): Кожен клас або модуль повинен мати лише одну причину для зміни. Це означає, що кожен клас або модуль повинен мати лише один обов’язок;

– Open-Closed Principle (OCP): Класи або модулі повинні бути відкритими для розширення, але закритими для модифікації. Це означає, що поведінку класу можна змінити без зміни його вихідного коду;

– Liskov Substitution Principle (LSP): Об’єкти в програмі повинні бути замінюваними на екземпляри їхніх підтипів без зміни правильності цього програмного забезпечення. Це означає, що класи, які наслідуються від батьківського класу, повинні зберігати поведінку, визначену в батьківському класі;

– Interface Segregation Principle (ISP): Клієнти не повинні залежати від інтерфейсів, які вони не використовують. Це означає, що великі інтерфейси слід розбивати на менші та більш конкретні, щоб клієнти використовували тільки ті інтерфейси, які їм потрібні;

– Dependency Inversion Principle (DIP): Програма повинна залежати від абстракцій, а не від конкретних реалізацій.

Як представлено на (рис. 3.7) реалізація представлена в вигляді окремих класів які через Dependency Injection впроваджуються в програму як незалежні модулі. Кожен за класів виконує своє призначення та має атомарне призначення для відповідності Single Responsibility Principle (SRP).

З представлених класів (рис. 3.7) видно, що кожен клас включає інформацію в вигляді властивостей, в яких зберігається інформація про самих себе. Наприклад для детектору зберігається знайдені ключові точки та дескриптори і для кожного зображення створюється свій екземпляр цього класу. Для результатів голосування створено структуру даних VoteResult для зручності представлення результатів. В класі HammingProvider наведено методи знаходження хемінгової відстані, а відповідно в інших методи для пошуку медоїду та безпосередньо голосування.

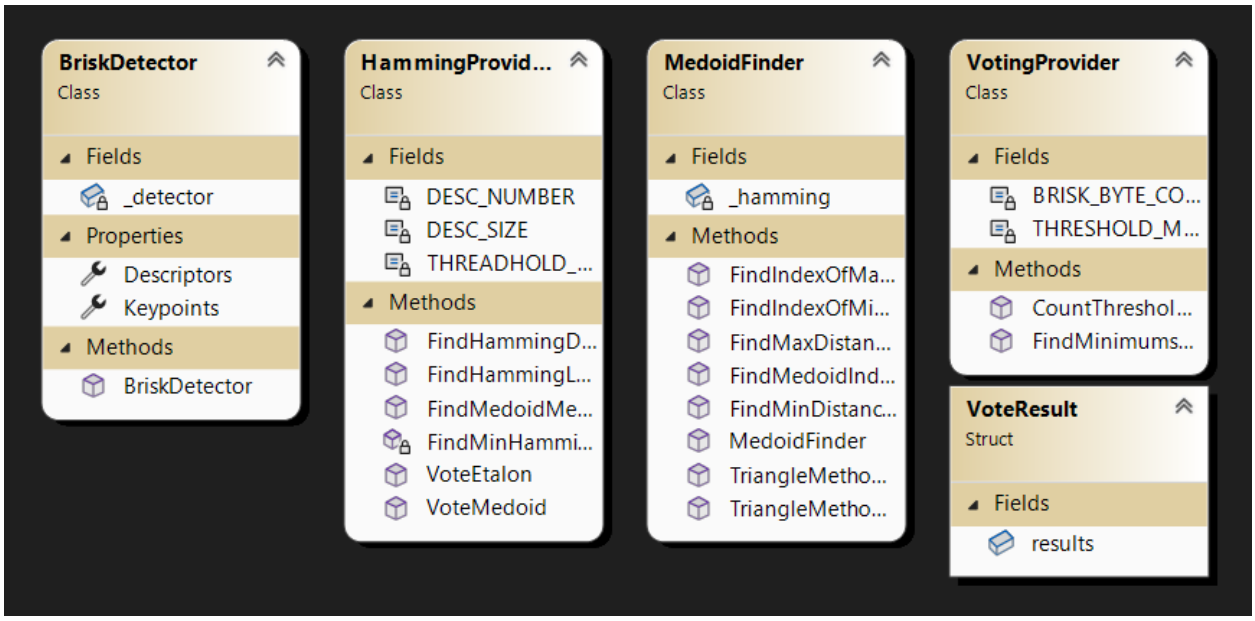



Рисунок 3.7 – Класи програми

Після розробки програмного забезпечення для класифікацію вдалося виокремити наступні результати для різних методів і зібрати статистику.

У таблиці 3.1 приведені результати виконання класичного методу голосування з різними порогами оцінки.

Таблиця 3.1 – Результати голосування класичним методом з порогами в відстань 64 та 128 одиниць

| **Зображення** | **Еталони** | **Метод лінійного пошуку (поріг 64)** | **Метод лінійного пошуку (поріг 128)** |
|---|---|---|---|
| 1 | 1 | 500 | 500 |
| | 2 | 0 | 0 |
| | 3 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| | 2 | 500 | 500 |
| | 3 | 0 | 0 |
| 3 | 1 | 0 | 0 |
| | 2 | 0 | 0 |
| | 3 | 500 | 500 |
| 4 | 1 | 41 | 148 |
| | 2 | 45 | 118 |
| | 3 | 78 | 167 |
| 5 | 1 | 98 | 148 |
| | 2 | 175 | 316 |
| | 3 | 18 | 35 |

Час, витрачений на цей метод, зібрано у таблиці 3.2.

Таблиця 3.2 – Витрачений час на класичний метод

| **Середній витрачений час у мс** | **Метод лінійного пошуку (поріг 64)** | **Метод лінійного пошуку (поріг 128)** |
|---|---|---|
| | 4566 | 4532 |

Запуск програми виконувався локально і в результатах представлено уже середнє значення часу із 10 запусків програми, для детальнішої оцінки. Час виконання програми залежить від багатьох факторів: від загруженості процесору, об’єму Garbage Collector в даний момент, які ресурси закешовані, та які звільняться. Але при розгляді середнього часу вплив цих факторів можна значно зменшити.

Результати виконання  методу за правилом трикутника приведено в таблиці 3.3.

Таблиця 3.3 – Результати голосування методом за правилом трикутника

| **Зображення** | **Еталони** | **Метод трикутника (максимуми)** | **Метод трикутника (мінімуми)** |
|---|---|---|---|
| 1 | 1 | 292 | 297 |
| | 2 | 158 | 155 |
| | 3 | 50 | 48 |
| 2 | 1 | 92 | 176 |
| | 2 | 372 | 276 |
| | 3 | 36 | 48 |
| 3 | 1 | 32 | 51 |
| | 2 | 154 | 188 |
| | 3 | 314 | 261 |
| 4 | 1 | 347 | 249 |
| | 2 | 107 | 160 |
| | 3 | 46 | 91 |
| 5 | 1 | 79 | 17 |
| | 2 | 396 | 438 |
| | 3 | 25 | 45 |

В таблиці 3.4 наведено час виконання для цих методів.

Таблиця 3.4 – Витрачений час на метод за правилом трикутником

| **Середній витрачений час у мс** | **Метод трикутника (максимуми)** | **Метод трикутника (мінімуми)** |
|---|---|---|
| | 13 | 14 |

Також для методу трикутника для вирішення ступеня класифікації можна скористатися наступною метрикою: при високому значенні відношення значення голосів 1 еталону до еталону 2-го по кількості голосів можна сказати, що класифікація є вдалою. Результати цієї оцінки приведено на таблиці 3.5.

Таблиця 3.5 – Результати голосування методом за правилом трикутника

| **Зображення** | **Метод трикутника (максимуми)** | **Метод трикутника (мінімуми)** |
|---|---|---|
| 1 | 1,85 | 1,92 |
| 2 | 4,04 | 1,57 |
| 3 | 2,04 | 1,39 |
| 4 | 3,24 | 1,56 |
| 5 | 5,01 | 9,73 |

Як видно з цієї таблиці всі зображення мають значення співвідношення оцінок більше ніж в 1,5 разів, тому можна казати про чітку класифікацію оцінюваних зображень і те, що метод за правилом трикутника відпрацював вірно, і відніс еталони до своїх класів. А зображення для оцінки дало такі самі результати, як і класичний метод голосування за лінійним пошуком.

Результати виконання методу оцінку до медоїдів та методу виокремлення медоїду зображення, яке буде класифікуватись приведено в таблиці 3.6.

Специфікою методу медоїду зображення від інших таблиць буде те, що оцінка відбувається на основу тільки 1 відстані, отже при віднесені до класу враховується тільки цей голос, і оцінка є завжди однозначною. Та в будь-якому випадку класифікатор віднесе результату до 1 з еталонів. При класифікації з тим самим зображенням, що і є еталоном класифікатору, автоматично віднесе зображення до відповідного класу через те, що медоїди однакові і відстань рівна 0.

Тому для доцільності оцінки ще додатково на таблиці 3.7 зображення саму цю відстань для 4, і 5 зображення, адже зрозуміло що для 1, 2 та 3 еталонів відстань до самих себе буде 0, і це не дасть нової інформації. Після чого необхідно буде оцінити, по аналогії з методом трикутника, але для мінімальності значення, яке значення буде приймати відношення в порівнянні 1-го значення при класифікації та 2-го при сортуванні за

зростанням. І якщо по цій метриці значення відношення буде перевищувати хоча б в 1,5 разів – тоді можна сказати про чітку класифікацію. Ці результати приведено на таблиці 3.8, але з 1 по 3 для методу медоїду зображення обрахунок значення не має сенсу, адже там завжди чітка класифікація і відстань приймає значення 0 до еталону.

Таблиця 3.6 – Результати голосування методом за правилом трикутника

| **Зображення** | **Еталони** | **Метод оцінки до медоїдів** | **Метод медоїду зображення** |
|---|---|---|---|
| 1 | 1 | 176 | 1 |
| | 2 | 120 | 0 |
| | 3 | 204 | 0 |
| 2 | 1 | 107 | 0 |
| | 2 | 252 | 1 |
| | 3 | 141 | 0 |
| 3 | 1 | 171 | 0 |
| | 2 | 109 | 0 |
| | 3 | 220 | 1 |
| 4 | 1 | 153 | 1 |
| | 2 | 160 | 0 |
| | 3 | 187 | 0 |
| 5 | 1 | 89 | 0 |
| | 2 | 284 | 1 |
| | 3 | 127 | 0 |

І з результатів цих відстаней 4 зображення з вибірки віднесло до 1 класу через те, що 141 найменша відстань, але інші еталони мають наближене значення до цього того класифікація є нечіткою. Так само і для 5 зображення його було віднесено до 2 класу за найменшою відстанню 194 одиниці, але різниця між відстанями є незначною і класифікація є нечіткою.

Таблиця 3.7 – Результати відстаней до медоїдів методом оцінки відстані до медоїду

| **Зображення** | **Еталони** | **Метод медоїду зображення значення хемінгової відстані** |
|---|---|---|
| 4 | 1 | 141 |
| | 2 | 163 |
| | 3 | 146 |
| 5 | 1 | 208 |
| | 2 | 194 |
| | 3 | 205 |

Таблиця 3.8 – Результати оцінки результативності класифікації методів за медоїдами

| **Зображення** | **Метод оцінки до медоїдів** | **Метод медоїду зображення значення хемінгової відстані** |
|---|---|---|
| 1 | 1,16 | – |
| 2 | 1,79 | – |
| 3 | 1,29 | – |
| 4 | 1,17 | 1,12 |
| 5 | 2,24 | 1,01 |

Результати витраченого часу цих 2-ох методів представлено в таблиці 3.9.

Таблиця 3.9 – Витрачений час на метод за методами з медоїдами

| **Середній витрачений час у мс** | **Метод оцінки до медоїдів** | **Метод медоїду зображення** |
|---|---|---|
| | 15 | 2673 |

Варто зазначити що метод медоїду зображення дуже сповільнюється через те, що в, безпосередньо, процес класифікації включені оцінки медоїду самого зображення що має складність квадрату кількості дескрипторів цього зображення, або в Big O анотації:

$$O(n^2 + m) \to O(n^2), \tag{3.1}$$

де $n$ – відповідно кількість дескрипторів в зображенні,

$m$ – кількість еталонів.

Через те, що порядок $n^2$ значно перевищує порядок $m$, як видно на (3.1) операції пов'язані з еталонами можна ігнорувати, адже для конкретної задачі їх завжди задана кількість рівна кількість – рівна кількості класів, і має лінійну складність. В той час як операції з дескрипторами має квадратичну складність.

## 3.3 Аналіз результатів оцінювання методів класифікації

Із експериментів, проведених у ході роботи, можна сказати, що методи класифікації за правилом трикутника, а також оцінювання за медоїдами є досить перспективними, адже мають перевагу у швидкості в порівнянні з класичним методом голосування для класифікації. А точність для цих методів залишається на достатньому рівні. Тому є вкрай необхідні подальші дослідження цих методів на більших вибірках даних та з іншими вхідними параметрами та порогами при оцінці значущості. В тому числі, безпосередньо, для наведених методів для підтвердження або спростування гіпотези підвищення ефективності їх застосування.

З результатів наведених в таблиці 3.1 можна сказати про високу ефективність класифікації класичним способом за лінійним пошуком, але

кількість операцій на виконання віднесення до класів складає квадратичну складність.

Метод трикутника виявився значно швидшим у порівнянні з класичним методом оцінки при обробці зображень. Обробка першого зображення методом трикутника займає всього 13 мілісекунд (мс), в той час як класичний метод вимагає 4566 мс, що робить метод трикутника в 351 разів швидшим.

Хоча теоретичний приріст швидкості і становить 500 разів для заданої вибірки, але у зв’язку з обмеженістю програмного забезпечення та відсутністю повної ізольованості від зовнішніх факторів на системних приладах на яких проводилась оцінка, попри це приріст в швидкості залишився досить значним, і при більшій кількості дескрипторів, він буде наближатися до теоретичного максимуму, тобто в прирості часу в кількість разів рівній кількості дескрипторів.

Окрім того, метод трикутника виявився досить надійним у класифікації зображень, що дозволяє вважати його не лише швидким, але і точним інструментом для такого роду задач.

В результаті метод трикутника виявився більш ефективним у порівнянні з класичним методом оцінки, що демонструє його переваги як по швидкості обробки, так і по надійності класифікації зображень.

З таблиці 3.3 та таблиці 3.5 можемо зробити висновки, що результати оцінки метода трикутника задовільні, всі зображення мають чітку класифікацію.

Для метода за максимумами значення відношення в середньому є більшим ніж для мінімумів, тому використання саме методу максимумів від медоїда за правилом трикутника має більшу точність при класифікації. Але метод трикутника за мінімумами, попри це, показує достатню точність, тому необхідно провести додаткові дослідження та порівняння цих методів.

За значеннями таблиць 3.6-3.8 можна зробити висновки, що методи класифікації за медоїдами можуть неправильно віднести зображення до якихось класів. Також навіть у випадках правильного віднесення до класу

точність класифікації має дуже малий показник, при цьому для методу оцінки за медоїдом зображення необхідно додатково розраховувати сам медоїд цього зображення, що дуже сповільнює процес класифікації і має також квадратичну складність, так само як і для класичного методу лінійного пошуку. Отже, результативність та виграш у часі в порівнянні з іншими методами має метод за правилом трикутника з використанням максимумів.

# ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод класифікації зображень за правилом трикутника при оцінюванні відстані. Розглянутий спосіб оцінювання універсально може бути використано у будь-яких застосуваннях для прискорення обчислення значень метрики чи іншої міри релевантності (на базі метрики) щодо довільних множин даних.

Запропонована інновація у класифікаторах в прикладному аспекті може бути результативнішою за кластерування, хешування чи зіставлення за центрами із-за гнучкішого використання апріорних знань про значення відстаней всередині множини, параметрів чи просторових границь множини, можливості управління ступенем інтегральності при оцінюванні.

Виграш у часі класифікації пропорційно зростає із збільшенням числа дескрипторів у описі. Експериментально для опису у 500 дескрипторів досягнуто виграш у швидкодії більш ніж у 400 разів.

Впровадження оцінювання суттєво спростило і прискорило процес визначення класу зображення без значного зниження показнику результативності.

Практична значущість роботи полягає у побудові прикладних моделей класифікації із використанням оцінювання, підтвердженні працездатності, високої швидкодії та класифікаційної результативності запропонованих модифікацій на прикладах зображень, створенні програмних застосунків для впровадження розроблених класифікаторів у системах комп’ютерного зору.

Перспективи можуть бути пов’язані із опрацюванням схем оцінювання на масштабній множині класів, де є можливість здійснити попередній метричний аналіз даних для описів бази зображень.

Результати роботи апробовано у вигляді тез доповідей під час XXVII Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [32].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Верес, О. М., Кісь, Я. П., Кугівчак, В. А., & Рішняк, І. В. (2018). Вибір методів для пошуку однакових або схожих зображень. Вісник Національного університету “Львівська політехніка”. Серія: Інформаційні системи та мережі, (887), 43-50.

2. Jiang, X., Ma, J., Xiao, G., Shao, Z., & Guo, X. (2021). A review of multimodal image matching: Methods and applications. *Information Fusion*, *73*, 22-71.

3. Zhang, W., Zhou, T., Xu, C., & Liu, M. (2021). A SIFT-like feature detector and descriptor for multibeam sonar imaging. *Journal of Sensors*, *2021*, 1-14.

4. Mery, D. (2014). Inspection of complex objects using multiple-X-ray views. *IEEE/ASME Transactions on Mechatronics*, *20*(1), 338-347.

5. Oliveira, A. J., Ferreira, B. M., & Cruz, N. A. (2021). A performance analysis of feature extraction algorithms for acoustic image-based underwater navigation. Journal of Marine Science and Engineering, *9*(4), 361.

6. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification. *IEEE Access*, *10*, 124738-124746. p. 6-10.

7. Daradkeh. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Cluster representation of the structural description of images for effective classification. p. 11-15

8. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент. С. 124.

9. Gadetska, S., Gorokhovatskyi, V., Stiahlyk, N., & Vlasenko, N. (2022). Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods. p. 9-11

10. Гороховатський, В. О., Стяглик, Н. І., & Жадан, О. В. (2022). Застосування багатокомпонентної моделі даних для описів класів у задачі класифікації зображень. С. 7

11. Gadetska, S. V., Gorokhovatskyi, V. O., Stiahlyk, N. I., & Vlasenko, N. V. (2021). Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points. Radio Electronics, Computer Science, Control, (4), 58-68.

12. Gorokhovatsky, V., Gadetska, S., Zhadan, O., & Khvostenko, O. (2021). Дослідження результативності класифікаторів зображень за статистичними розподілами для компонентів структурного опису. Advanced Information Systems, 5(1), 5-11.

13. Leskovec, J., Rajaraman, A., & Ullman, J. D. (2020). Mining of massive data sets. Cambridge university press. P. 511.

14. Flach, P. (2012). Machine learning: the art and science of algorithms that make sense of data. Cambridge university press. P. 409

15. Schütze, H., Manning, C. D., & Raghavan, P. (2008). Introduction to information retrieval (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.

16. Mashtalir S., Mashtalir V. (2020) Spatio-Temporal Video Segmentation. In: Mashtalir V., Ruban I., Levashenko V. (eds) Advances in Spatio-Temporal Segmentation of Visual Data. Studies in Computational Intelligence, vol 876. Springer, Cham. pp. 161-210 https.://doi.org/10.1007/978-3-030-35480- 0_4, https://link.springer.com/chapter/10.1007/978-3-030-35480-0_4

17. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In *Explainable Deep Learning AI* (pp. 125-146). Academic Press.

18. Gorokhovatskyi, V., Stiahlyk, N., Tsarevska, V. (2021). Combination method of accelerated metric data search in image classification problems. Advanced Information Systems, 5 (3), pp. 5–12.

19. Kinoshenko, D., Mashtalir, V., Yegorova, E., & Vinarsky, V. (2005). Hierarchical partitions for content image retrieval from large-scale database. In *Machine Learning and Data Mining in Pattern Recognition: 4th International Conference, MLDM 2005, Leipzig, Germany, July 9-11, 2005. Proceedings 4* (pp. 445-455). Springer Berlin Heidelberg.

20. Liu, Y., Zhang, D., Lu, G., & Ma, W. Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern recognition*, *40*(1), 262-282.

21. Berman, A. P., & Shapiro, L. G. (1999). A flexible image database system for content-based retrieval. *Computer Vision and Image Understanding*, *75*(1-2), 175-195.

22. Celik, C., & Bilge, H. S. (2017). Content based image retrieval with sparse representations and local feature descriptors: a comparative study. *Pattern Recognition*, *68*, 1-13.

23. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), pp. 10-16.

24. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень. С. 16

25. Mashtalir, S., & Mikhnova, O. (2017). Detecting significant changes in image sequences. *Multimedia Forensics and Security: Foundations, Innovations, and Applications*, 161-191.

26. Guo, X., Yang, H., Wei, M., Ye, X., & Zhang, Y. (2023). Few-shot object detection via class encoding and multi-target decoding. *IET Cyber-Systems and Robotics*, *5*(2), e12088. pp. 23-29

27. Vedaldi, A., & Fulkerson, B. (2010, October). VLFeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia* (pp. 1469-1472).

28. Kuchuk, H., Kovalenko, A., Ruban, I., & Ibrahim, B. F. (2019). Adaptive compression method for video information. *International Journal of Advanced Trends in Computer Science and Engineering*, *8*(1), 66-69.

29. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. *IEEE Access*, *9*, 92964-92973.

30. Yakovleva, O., & Nikolaieva, K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. *Advanced Information Systems*, *4*(4), 89-101.

31. Pop, C. M., Mogan, G. L., & Boboc, R. G. (2017). Real-Time Object Detection and Recognition System Using OpenCV via SURF Algorithm in Emgu CV for Robotic Handling in Libraries. *International Journal of Modeling and Optimization*, *7*(5). pp. 6-19

32. Дебре В. (2023) Впровадження оцінок відстаней у класифікації зображень. 27-ий міжнародний молодіжний форум «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ», т. 7, с. 44-45.

33. Gorokhovatskyi, V. O., & Gadetska, S. V. (2019). Determination of Relevance of Visual Object Images by Application of Statistical Analysis of Regarding Fragment Representation of their Descriptions. *Telecommunications and Radio Engineering*, *78*(3). pp. 211–220.

34. Gorokhovatskyi, V. A. (2018). Image classification methods in the space of descriptions in the form of a set of the key point descriptors. *Telecommunications and Radio Engineering*, *77*(9), 787-797.

35. Gorokhovatskiy, V. A. (2011). Compression of descriptions in the structural image recognition. *Telecommunications and Radio Engineering*, *70*(15). pp. 1363–1371.

36. Gadetska, S. V., & Gorokhovatskyi, V. O. (2018). Statistical measures for computation of the image relevance of visual objects in the structural image

classification methods. *Telecommunications and Radio Engineering*, *77*(12). pp. 1041–1053.

37. Gorokhovatskiy, V. A. (2016). Efficient Estimation of Visual Object Relevance during Recognition through their Vector Descriptions. *Telecommunications and Radio Engineering*, *75*(14). P. 1271–1283.

38. Гороховатський, В. О., Гадецька, С. В., Стяглик, Н. І., & Власенко, Н. В. (2020). Класифікація зображень на підставі ансамблю статистичних розподілів за класами еталонів для компонентів структурного опису., С. 85-94.

39. Gorokhovatskyi, V., Gadetska, S., & Ponomarenko, R. (2020). Recognition of visual objects based on statistical distributions for blocks of structural description of image. In *Lecture Notes in Computational Intelligence and Decision Making: Proceedings of the XV International Scientific Conference "Intellectual Systems of Decision Making and Problems of Computational Intelligence"(ISDMCI'2019), Ukraine, May 21–25, 2019 15* (pp. 501-512). Springer International Publishing.

40. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for Visual Objects by Request in the Form of a Cluster Representation for the Structural Image Description. *Advances in Electrical and Electronic Engineering*, *21*(1), 19-27.

41. Gorokhovatskyi, V., Tvoroshenko, I., & Chmutov, Y. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Advanced Information Systems*, *6*(3), С. 5-12.

42. Rubinstein-Salzedo, S. (2018). Big o notation and algorithm efficiency. In *Cryptography* (pp. 75-83). Cham: Springer International Publishing.

43. Гороховатський, Ст А., & Передрій, Є. О. (2009). Кореляційні методи розпізнавання зображення шляхом голосування систем фрагментів. Радіоелектроніка, інформатика, управління, (1(20)), 74-81.

44. Гороховатскій, В. А., Путятін, Y. P., & Столяров, V. S. (2017). Дослідження результативності структурних методів класифікації зображень

із застосуванням кластерної моделі даних. Radio Electronics, Computer Science, Control, (3), 78-85. С. 78-85.

45. Гороховатський, Ст А., & Путятін, Є. П. (2008). Структурне розпізнавання зображень з урахуванням моделей голосування ознак характерних точок. Реєстрація, зберігання та обробка даних. С.75-85.

46. Гороховатський, Ст А. (2014). Структурний аналіз та інтелектуальна обробка даних у комп'ютерному зорі. С. 316.

47. Гороховатський, В. О., Передрій, О. О., Творошенко, І. С., & Марков, Т. Є. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень. С. 5-13.

48. Гороховатський, В. О., Творошенко, І. С., & Сидоренко, Д. (2021). Класифікація зображень із використанням кластерного подання. С. 44-45.

49. Singh, H., & Hassan, S. I. (2015). Effect of solid design principles on quality of software: An empirical assessment. *International Journal of Scientific & Engineering Research*, *6*(4), 1321-1324.