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 СТАТИСТИЧНИХ МІР ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ НА ОСНОВІ КОМПОНЕНТНОГО АНАЛІЗУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-17-1

Хвостенко О.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні 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)

2021 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)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Хвостенко Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання статистичних мір для класифікації зображень на основі компонентного аналізу.

затверджена наказом по університету від «20» травня 2021 року № 663Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2021 р.
3. Вихідні дані до роботи Методи розпізнавання зображень, апарат машинного навчання, засоби детектування ключових точок на зображенні, методи класифікації об'єктів за допомогою детекторів ключових точок, вхідні бази зображень, метод класифікації з використанням статистичних характеристик.

4. Перелік питань, що потрібно опрацювати в роботі.
1. Моделювання статистичних мір для класифікації зображень.
2. Визначення дескрипторів ключових точок.
3. Моделі обробки ключових точок на зображенні.
4. Програмна реалізація методу та аналіз результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Актуальність проблеми класифікації зображень, постановка задачі, метод класифікації, тестові зображення, результат роботи програми, аналіз експериментальних результатів.

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	12.04.2021	виконано
2	Аналіз завдання, підбір літератури	13.04.21-14.04.21	виконано
3	Аналіз літератури з досліджуваної проблеми	14.04.21-19.04.21	виконано
4	Аналіз технічних засобів	19.04.21-23.04.21	виконано
5	Розробка методу	24.04.21-30.04.21	виконано
6	Програмна реалізація	01.05.21-10.05.21	виконано
7	Оформлення пояснювальної записки	10.05.21-21.05.21	виконано
8	Перевірка на плагіат	25.05.2021	
9	Рецензування	27.05.2021	
10	Підготовка презентації та доповіді	29.05.2021	
11	Занесення роботи в електронний архів	30.05.2021	
12	Попередній захист кваліфікаційної роботи	31.05.2021	

Дата видачі завдання 12.04.2021 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 5 табл., 28 рис., 1 дод., 41 джерело.

РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, КЛЮЧОВА ТОЧКА, КЛАСИФІКАТОР, ДЕТЕКТОР ORB, ДЕСКРИПТОР, СТАТИСТИЧНИЙ РОЗПОДІЛ.

Об’єктом роботи є методи класифікації зображень у системах комп’ютерного зору.

Метою роботи є побудова класифікатора зображень на основі компонентного аналізу даних – множини дескрипторів структурного опису із використанням статистичних мір для обчислення релевантності описів розпізнаваного об’єкту та еталонів.

Застосовано методи статистичного моделювання та компонентного аналізу векторних даних. Проведено дослідження результативності класифікаторів з використанням опису як множин дескрипторів ключових точок зображень.

У результаті роботи здійснена програмна реалізація методу для класифікації у базі еталонних зображень, а також роботу було апробовано у вигляді тез доповіді та статті.

IMAGE RECOGNITION, KEYPOINT, CLASSIFIER, ORB DETECTOR, DESCRIPTOR, STATISTICAL DISTRIBUTION.

The object of the research is the methods of image classification in computer vision systems.

The aim of the research is to build a classifier of images based on component data analysis – a set of descriptors of the structural description using statistical measures to calculate the relevance of the descriptions of the recognized object and etalons.

Methods of statistical modeling and component analysis of vector data are applied. A study of the effectiveness of classifiers using descriptions as sets of descriptors of key points of images.

As a result, the software implementation of the method for classification in the database of reference images is carried out, and also the work was tested in the form of abstracts collection and article.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – ключові точки

OpenCV – Open Source Computer Vision Library

FAST – Features from Accelerated Segment Test

BRIEF – Binary Robust Independent Elementary Features

rBRIEF – rotated Binary Robust Independent Elementary Features

ORB – Oriented FAST and Rotated BRIEF

BRISK – Binary Robust Invariant Scalable Keypoints

SURF – Speeded up Robust Features

SIFT – Scale Invariant Feature Transform

РКЛ – розходження Кульбака-Лейблера

СКВ – середньоквадратичне відхилення

ВСТУП

Статистичні розподіли останнього часу стали першорядним засобом інтелектуального аналізу даних у системах розпізнавання образів. Якщо опис розпізнаваного об’єкту подано множиною векторів (дескрипторів), статистичний апарат стає ключовим способом прийняття рішення про клас візуального об’єкту [1–6].

Ймовірнісні розподіли даних описів у складі системи блоків для дескрипторів КТ показали свою високу результативність у аспекті якості класифікації та швидкодії оброблення. Виникає нагальна необхідність впровадження апарату розподілів у загальному виді для системи багатовимірних даних дескрипторів опису за встановленими класами образів, що визначаються заданою базою еталонів [7–9].

Метою роботи є побудова класифікатора зображень на основі компонентного аналізу даних – множини дескрипторів структурного опису із використанням статистичних мір для обчислення релевантності описів розпізнаваного об’єкту та еталонів.

Для дослідження та впровадження класифікаторів пропонуються застосування засобів: детектору ORB для формування дескрипторів ключових точок, інтелектуального аналізу даних, математичної статистики, а також апарату визначення релевантності для множин векторів даних та програмне моделювання.

Задачами дослідження є побудова моделей класифікації у синтезованому просторі образів ймовірнісних розподілів, аналіз параметрів, що впливають на їх ефективність, експериментальне оцінювання результативності класифікаторів засобами програмного моделювання за наслідками оброблення експериментальної бази зображень.

1 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ДЕСКРИПТОРІВ КТ

1.1 Основні задачі розпізнавання образів

Методи класифікації зображень у системах комп'ютерного зору отримали свій подальший розвиток із застосуванням детекторів ключових точок [1, 2].

Розпізнавання образів (pattern recognition) – це розділ теорії штучного інтелекту (artificial intelligence), що вивчає методи класифікації об'єктів. За традицією об'єкт, що піддається класифікації, називається образом (pattern). Образом може бути цифрова фотографія (розпізнавання зображень), буква або цифра (розпізнавання символів), запис мови (розпізнавання мови) тощо. У межах теорії штучного інтелекту розпізнавання образів включається в більш широку наукову дисципліну – теорію машинного навчання (machine learning), метою якої є розробка методів чи алгоритмів, що здатні навчатися.

Навчання, як і інтелект, охоплює настільки широкий спектр процесів, що складно дати йому точне визначення. Словникові визначення включають такі фрази, як «отримувати розуміння, вміння або знання за допомогою інструктування, вивчення або досвіду» і «модифікація поведінкової тенденції шляхом використання досвіду» Психологи і зоологи досліджують здібність навчання тварин і людей. Існують паралелі між тваринним і машинним навчанням. Звичайно, достатньо велика кількість технік у машинному навчанні походить від зусилль психологів вдосконалити свої теорії людського і тваринного навчання через обчислювальні моделі. Можна сказати, що машини навчаються, змінюючи свою структуру, програму або дані (у відповідь на зовнішню інформацію) таким чином, що очікувана майбутня продуктивність машини покращується.

Розпізнавання образів – це одна з найфундаментальніших проблем теорії інтелектуальних систем. З іншого боку, задача розпізнавання образів має сильне практичне значення.

Поряд з терміном «розпізнавання» часто вживається інший – «класифікація». Ці два терміни зазвичай вважаються синонімами, але вони не є цілком взаємо-замінюваними. Кожний з них має свої сфери застосування, і інтерпретація обох термінів не рідко залежить від специфіки прикладної задачі. Часто вважають обидва терміни синонімами, якщо явно не обумовлене щось інше.

Як і кожна математична дисципліна, розпізнавання образів має власний математичний апарат, який включає математичну статистику, методи оптимізації, дискретну математику, алгебру і геометрію [10].

Метою машинного навчання є розробка методів побудови алгоритмів, що здатні навчатися. Існує два підходи до навчання: індуктивне і дедуктивне.

Індуктивне навчання, що також називають навчанням за прецедентами, засноване на виявленні загальних властивостей об’єктів на базі неповної інформації, отриманих емпіричним шляхом.

У дедуктивнному навчанні використовується формалізація знань експертів у вигляді баз знань (експертних систем тощо). Розпізнавання образів має широке застосування і використовується при створенні усіх комп’ютерних систем, на які покладаються інтелектуальні функції, тобто функції, пов’язані із прийняттям рішень замість людини: медична діагностика, криміналістична експертиза, пошук інформації та інтелектуальний аналіз даних тощо.

Одним із базових понять апарату розпізнавання є поняття множини. У комп’ютері множина представляється набором неповторюваних однотипних елементів. Слово «неповторюваних» означає, що якийсь елемент у множині або є, або його там немає. Універсальна множина включає всі можливі для розв’язуваної задачі елементи, порожня не містить жодного.

Образ – це класифікаційне концентрування в системі класифікації, що об’єднує (виділяє) певну групу об’єктів за певною ознакою. Образи володіють характерними властивостями, виявляючись в тому, що ознайомлення з кінцевим числом явищ з однієї і тієї ж множини надає змогу дізнаватися про як завгодно велике число його представників.

Образи мають характерні об’єктивні властивості в тому сенсі, що різні особи, що навчаються на різному матеріалі спостережень, здебільшого однаково й незалежно один від одного класифікують одні й ті ж об’єкти. У класичній постановці завдання розпізнавання універсальна множина ділиться на частини-образи. Кожне відображення розглядаємого об’єкта на сприймаючі органи системи, що розпізнає, незалежно від його положення щодо цих органів, прийнято називати зображенням об’єкта, а множини таких зображень, об’єднані загальними властивостями, являють собою образи.

Методика віднесення елемента до певного образу називається вирішальним правилом. Іншим важливим поняттям є метрика, що представляє собою спосіб визначення відстані між елементами універсальної множини. Чим менша ця відстань, тим більш схожими є об’єкти (символи, звуки та ін.) – те, що розпізнається. Зазвичай елементи задаються у вигляді набору чисел, а метрика – у вигляді функції. Насамперед, ефективність програми залежить від вибору представлення образів і реалізації метрики, тому що один і той самий алгоритм розпізнавання з різними метриками буде помилятися з різною частотою.

Навчання представляє собою процес вироблення в деякій системі тієї чи іншої реакції на групи зовнішніх ідентичних сигналів через багаторазовий вплив на систему зовнішнього коригування. Таке зовнішнє коригування у навчанні називають «заохоченнями» і «покараннями».

Прецедент – це об’єкт, що належить до заданого класу заздалегідь. Прикладом прецеденту може бути набір ознак пацієнта із відомим діагнозом, з яким порівнюється набір ознак людини, діагноз якої ще невідомий. Кожний образ – це набір чисел, які описують його властивості і називаються ознаками

(feature). Упорядкований набір ознак об’єкта є вектором ознак (feature vector). Вектор ознак – це точка в просторі ознак (feature space) [11, 12].

Навчання по прецедентах поділяють на три основних типи: контрольоване навчання, або навчання з учителем (supervised learning), неконтрольоване навчання (unsupervised learning), або навчання без учителя, і навчання з підкріпленням (reinforcement learning).

Метод контрольованого навчання застосовується у випадках, коли є великі обсяги даних, наприклад, тисячі фотографій домашніх тварин з маркерами (мітками, ярликами): це кішка, а це собака. Необхідно створити алгоритм, за допомогою якого машина могла б по фотографії визначити, хто на ній зображений: кішка або собака. У ролі «вчителя» в даному прикладі виступає людина, яка заздалегідь проставила маркери. Машина сама вибирає ознаки, за якими вона відрізняє кішок від собак. У подальшому знайдений нею алгоритм може бути швидко переналаштований на рішення іншої задачі, як варіант – на розпізнавання курей і качок. Комп’ютер сам виконає складну і копітку роботу по виділенню ознак, за якими буде розрізняти цих птахів. А нейромережа, яку навчили розпізнавати кішок, може швидко навчити обробляти результати комп’ютерної томографії.

Вже існує безліч маркованих даних, проте даних без маркерів все ж набагато більше. Це зображення без підписів, аудіо записи без коментарів, тексти без анотацій. Завданням машини при неконтрольованому навчанні є знаходження зв’язку між окремими даними, виявлення закономірності, підбір шаблонів, упорядкування даних або опис їх структури та виконання класифікації даних. Неконтрольоване навчання використовується, наприклад, в рекомендаційних системах, коли в інтернет–магазині базуючись на аналізі попередніх покупок, покупцеві пропонуються товари, що можуть зацікавити його з більшою ймовірністю, ніж інші. Або коли після перегляду якогось відеокліпу на порталі YouTube користувачеві у рекомендаціях пропонують десятки посилань на ролики, чимось схожі на переглянутий. Або коли Google у відповідь на один і той же запит ранжує посилання в результатах пошуку для

одного користувача інакше, ніж для іншого, оскільки враховує історію пошуків [11].

Навчання з підкріпленням – це окремий випадок контрольованого навчання, але вчителем в даному випадку є «середовище». Комп'ютер (його в цій ситуації часто називають «агент») не має попередньої інформації про середовище, але має змогу здійснювати в ній будь-які дії. Середовище реагує на ці дії й таким чином надає агенту дані, які дозволяють йому реагувати на них і вчитися. Фактично агент і середовище утворюють систему зі зворотним зв'язком. Навчання з підкріпленням, на відміну від навчання з учителем і без вчителя, використовується для вирішення більш складних завдань. Воно використовується, наприклад, в системах навігації для роботів, які навчаються уникати зіткнень з перешкодами шляхом набуття досвіду, отримуючи зворотний зв'язок при кожному зіткненні. Навчання з підкріпленням використовується також в логістиці, при складанні графіків і плануванні завдань, при навчанні машини логічним іграм (покер, нарди, го та ін.).

Класифікатор, або вирішальне правило (decision rule) – це функція, яка ставить у відповідність вектору ознак образу клас, до якого він належить. Задачу розпізнавання образів можна розділити на ряд підзадач.

1. Генерування ознак (feature generation) – вимірювання або обчислення числових ознак, що характеризують об'єкт.

2. Вибір ознак (feature selection) – визначення найбільш інформативних ознак для класифікації (в цей набір можуть входити не лише первинні ознаки, але й функції від них).

3. Побудова класифікатора (classifier construction) – конструювання вирішального правила, на підставі якого здійснюється класифікація.

4. Оцінка якості класифікації (classifier estimation) – обчислення показників правильності класифікації (точність, чутливість, специфічність, помилки першого та другого роду) [1, 10, 11].

1.2 Методи виділення ключових точок зображення

Останнім часом набули прикладного застосування такі детектори КТ, як ORB і BRISK [13–15], які обчислюють дескриптор КТ у вигляді бінарного вектора з розмірністю, кратною ступеню двійки. Бінарне представлення значно прискорює процес зіставлення дескрипторів за рахунок можливості застосування двійкових операцій і відповідно спрощує апаратну реалізацію системи розпізнавання. Крім того, бінарна арифметика дає потенцію застосувати ефективний апарат оброблення двійкових даних і синтезувати нові підходи для визначення подібності у просторі дескрипторів при побудові правил класифікації.

Аналіз властивостей алгоритмів для реалізації детекторів ORB, BRISK показує, що вони менш вимогливі до обчислювальних ресурсів у порівнянні з іншими. Виграш у швидкості обчислень пояснюється спрощеною процедурою побудови дескрипторів, що дає за результатами тестування помітний виграш у швидкодії оброблення при порівняльній або кращій точності, ніж для традиційних детекторів SIFT і SURF [13, 16]. Відзначається більш висока точність детектування з використанням BRISK на окремих тестових зображеннях у порівнянні із застосуванням SURF-дескрипторів.

Встановлення відповідності для дескрипторів ORB, BRISK здійснюється обчисленням відстані Хемінга, тобто підрахунком числа бітів, відмінних у двох бінарних векторах. Відповідні дії зводяться до побітової логічної операції XOR, що обчислюється ефективно. Модульність побудови ORB, BRISK дає можливість комбінувати побудову детектора КТ в 15 поєднанні з довільним способом визначення його дескриптора і навпаки, оптимізуючи бажану продуктивність вирішуваних завдань [4].

Таким чином, основні переваги методів ORB і BRISK полягають у забезпеченні більш високої продуктивності за рахунок спрощення процесу оброблення шляхом використання дескрипторів бінарного типу.

Історично раніше розроблено детектори SIFT (Scale Invariant Feature Transform) и SURF (Speeded Up Robust Features) [13, 16]. Вони дають можливість сформувати як множину координат КТ, так і визначити їх дескриптори. Метод SURF при цьому набув порівняно із SIFT переважної популярності із-за поглибленого розроблення та забезпечує суттєво більшу швидкодію оброблення даних в процесі розпізнавання. Тривають цілеспрямовані спроби створення більш простих способів детектування КТ, обчислення та порівняння дескрипторів, що забезпечують як достатній рівень інваріантності до спотворень, так і перевагу в швидкодії оброблення.

З метою забезпечення високої якості ідентифікації або розпізнавання довільних об'єктів важливо, наскільки описи еталонів розрізняються в рамках застосовуваного методу. Чим значніша відмінність елементів описів або чим істотніше різниться їх склад, тим вище ймовірність правильного розпізнавання [17]. Результати досліджень показують, що показник швидкодії розпізнавання за рахунок трансформації системи ознак може бути поліпшений в десятки разів. Грануляція елементів множинного опису на підставі подібності його складових може забезпечити необхідне стиснення обсягу ознак, ефективну трансформацію простору, а також формування підмножин найбільш інформативних ознак з метою ефективного зниження обчислювальних витрат [13].

Метод SURF, застосовуваний для побудови описів, формує опис $Z \subset R^n$ як підмножину $R_1^n = z|z \in R^n, \left||z|\right| \approx 1$ n-вимірних дійсних векторів, евклідова норма яких дорівнює одиниці [16]. На практиці ця умова реалізується в наближеному виді $R_1^n = z|z \in R^n, \left||z|\right| \approx 1$. Опис SURF може містити сотні векторів, представлених у формі з плаваючою комою, що загалом істотно сповільнює оброблення. Завдання скорочення числа векторів передбачає побудову стисненого опису на основі відображення, варіантом якого є формування підмножини значно меншої потужності шляхом застосування процедури відбору (редукції) ознак із Z. У теорії розпізнавання

це називають формуванням підмножини найбільш значущих або інформативних ознак [13, 18].

Основним критерієм оцінювання подібності елементів є метрика ρ. Найчастіше в якості ρ використовують евклідову або манхетенську метрики із R^n. Критерієм еквівалентності двох дескрипторів КТ є значення ρ, що не перевищує апріорної величини порога δ_z. Два дескриптори вважаються еквівалентними, якщо виконано $\rho(z_1, z_2) \leq \delta_z$. Поріг δ_z визначають як відсоток (точність) від максимально можливого значення, він приймається рівним, наприклад, 1%, 3%, 5%, 10% [19]. В цілому вибір δ_z залежить також від застосовуваної процедури оброблення. Значення порогів можуть відрізнятися, наприклад, у процедурах навчання, при встановленні еквівалентності дескрипторів і при класифікації.

Здатність гранулювання інформації – важлива властивість інтелектуальних систем розпізнавання, що використовують семантику для формалізації рішення задач. Відомі чіткий і нечіткий підходи гранулювання [20]. Грануляцію структурного представлення об'єктів розуміємо у двох аспектах: уявлення опису у вигляді вектора через перехід до мультимножини шляхом класифікації його елементів, а також як угруповання подібних елементів на основі властивостей унікальності або кластерного аналізу [13]. Принципова відмінність першого підходу полягає в тому, що при цьому апріорно задається базова множина, яка фактично визначає класи для елементів опису.

Інформаційною гранулою називають підмножину універсуму, на якому визначено відношення подібності (нерозрізненості, еквівалентності), гранула представляє собою об'єднання атомарних елементів [13, 20]. У результаті універсум або фіксований опис можна зобразити як множину гранул. Розроблено основи теорії міри і відношень на гранулах. Міра гранули найчастіше обчислюється як сума (інтеграл) значень функції приналежності елементів $d(A) = \sum_{a \in A} \mu_A(a)$, де $\mu_A(a)$ – значення функції приналежності елемента гранулі, $\mu_A(a) \in [0,1]$. Дії над гранулами здійснюються за законами

теорії множин. Гранули можуть включати одна одну, утворювати ієрархії. Виконання грануляції на практиці реалізується деяким методом і залежить від ряду параметрів.

Зауважимо важливі для застосувань властивості детекторів ORB та BRISK. Так, метод ORB визначає значніші за розміром масиви КТ у порівнянні з SURF, однак, при цьому його дескриптори часто просторово скупчуються і не завжди відображають ключові особливості об’єкту [4].

У детекторі BRISK є можливість управління кількістю сформованих КТ. На число виявлених КТ впливають такі показники, як: поріг алгоритму FAST для різниці між інтенсивністю центрального пікселя та пікселями кола навколо нього; номер октави, що встановлює крок стиснення зображення; параметр масштабу для вибору околиці КТ; радіуси для BRIEF (у пікселях), де взято зразки навколо координат КТ; кількість точок для BRIEF при відборі зразків; мінімальний та максимальний пороги для пар точок при формуванні дескриптора та ін. [14, 15].

Метод ORB (Oriented FAST and Rotated BRIEF), який є модифікованою комбінацією методу FAST для виявлення ключових точок та визначенням дескриптора у вигляді бінарного рядка по методу BRIEF (Binary Robust Independent Elementary Features).

Метод FAST (Features from Accelerated Test) – один з найбільш поширених на практиці детекторів ключових точок, визначається тим, що будує прості дерева рішень для класифікації пікселів на ключові точки та решту.

Рисунок 1.1 демонструє роботу детектора ключових точок ORB.

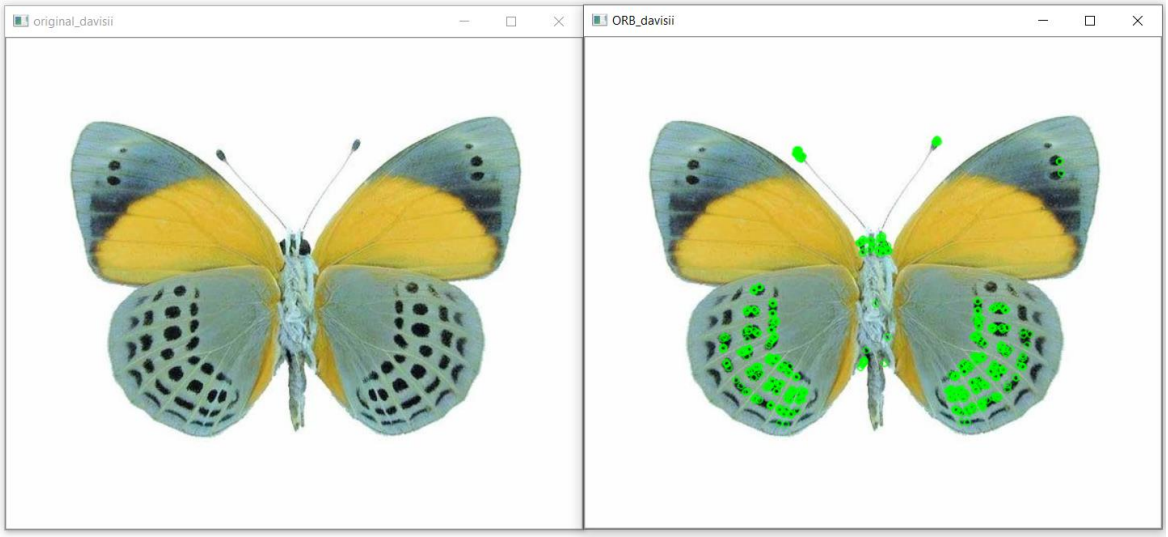

Рисунок 1.1 – Приклад роботи детектора ORB

Детектор ORB у порівнянні з SURF вважається швидшим у реалізації, але він є менш стійким до масштабування. SURF оперує дескрипторами розміром 64, ORB – 32 компонента, обидва детектора інваріантні до геометричних перетворювань зміщення, повороту, масштабу [8].

На першому етапі, детектор ORB використовує FAST для пошуку ключових точок, далі уточнює їх на підставі детектору кутів Harris. FAST (Features from Accelerated Segments Test) швидко вибирає ключові точки, порівнюючи рівні яскравості в певній області пікселів.

1. Обирається піксель p, для якого буде вирішуватись, чи є він ключовою точкою. Нехай значення яскравості (інтенсивності) пікселя буде позначатися як I_p, а t – порогове значення.

2. Навколо точки p розглядається окружність розміром 16 пікселів (рис. 1.2).

4. Точка p вважається ключовою, якщо серед усіх суміжних пікселів окружності існує n пікселів, кожен з яких яскравіший, ніж I_{p+t}, або темніший, ніж I_{p-t}.

5. Для того, щоб зробити алгоритм швидшим, розглядається інтенсивність пікселів 1, 5, 9, 13. Якщо хоча б для трьох зазначених точок виконуються умови з п.4, тоді p вважається кутом.

6. Далі проводиться перевірка для інших пікселів в околі.

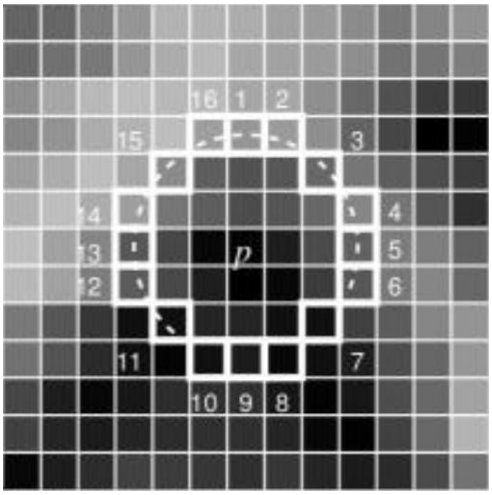

Рисунок 1.2 – Приклад обрання зони навколо точки *p*

У цього алгоритму існує декілька обмежень. Для n <12 алгоритм працює не достатньо гарно за різних умов, тому що при n <12 число виявлених КТ є дуже високим. Порядок, у якому 16 пікселей оброблюються, визначає швидкість роботи алгоритму.

Підхід до машинного навчання був доданий до алгоритму для вирішення цих проблем [21].

1. Обирається набір зображень для тренувань.

2. У кожному зображенні запускається алгоритм FAST для виявлення КТ, що приймають один піксель одночасно та оцінюють всі 16 пікселів у колі.

3. Для кожного пікселя *p* зберігають 16 пікселів навколо нього, як вектор (рис. 1.3).

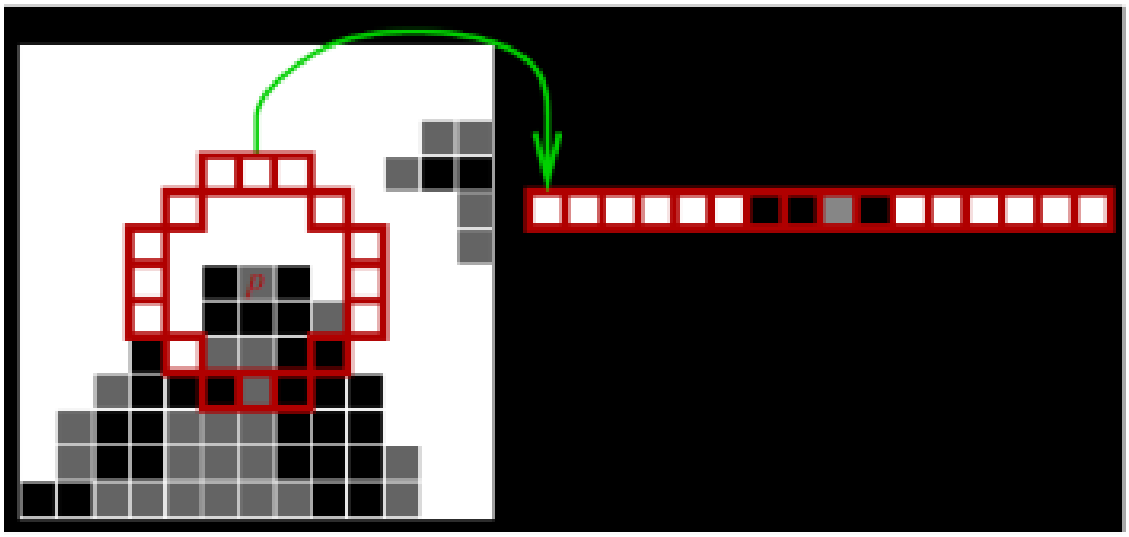

Рисунок 1.3 – 16 значень, що оточують піксель p зберігаються у векторній формі

4. Виконується повторення попередніх кроків для всіх пікселів у всіх зображеннях. Це вектор p, який містить всі дані для розпізнавання. (Зверніть увагу на різницю між p – пікселем та p – вектором).

5. Кожне значення (один з 16 пікселів, скажімо x) у векторі, може мати три стани. Темніший, ніж p, світліший за p або аналогічний p.

Математично,

$$S_{p->x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad (dar\ ker) \\ s, & I_p - t < I_{p\to x} < I_p + t \quad (similar) \\ b, & I_p + t \le I_{p\to x} \quad (brighter) \end{cases},$$

де $S_{p->x}$ – це стан;

$I_{p->x}$ – інтенсивність пікселя x;

t – поріг.

6. Залежно від стану весь вектор p буде підрозділятися на три підмножини P_d, P_s, P_b.

7. Визначається змінна K_p, яка є істиною, якщо p є КТ або помилковою, якщо p не є КТ.

8. Використовується алгоритм ID3 (класифікатор дерева рішень) для запиту кожного піднабору, використовуючи змінну K_p для знань про справжній клас.

9. Алгоритм ID3 працює за принципом мінімізації ентропії. Обробляється 16 пікселів таким чином, щоб знайти справжній клас (э точка КТ чи ні) з мінімальною кількістю запитів. Обирається піксель x, який містить найбільшу інформацію про піксель p. Ентропія для вектора p може бути математично представлена як

$$H(P) = (c + \overline{c}) - c\, log_2\, c - \overline{c}\, log_2\, \overline{c}$$

$$where \quad c = \left|\{\, p \mid K_p \text{istrue} \,\}\right| \quad (number of corners)$$

$$and \quad \overline{c} = \left|\{\, p \mid K_p \text{isfalse} \,\}\right| \quad (number of noncorners) \;.$$

10. Рекурсивно застосовується мінімізація ентропії до всіх трьох підмножин.

11. Процес припиняється, коли ентропія підмножини дорівнює нулю.

12. Цей порядок запитів, який вивчається деревом рішень, можна також використовувати для швидкого виявлення в інших зображеннях.

Виявлення кількох суміжних КТ є однією з проблем початкової версії алгоритму. Цього можна уникнути, застосовуючи немаксимальне придушення після виявлення КТ.

Алгоритм полягає в наступному.

1. Обчислюється функція оцінки *V* для кожної з виявлених точок. Функція оцінки визначається як сума абсолютної різниці між пікселями в сусідній дузі та центрального пікселю.

2. Розглядаються дві сусідні КТ й порівнюються їх значення *V*.

3. КТ, що має нижче значення *V* відкидується.

Весь процес можна узагальнити математично наступним чином:

$$\mathrm{V} = max \begin{cases} \sum(pixel\ values\ - \rho)\ if\ (value\ - \rho) > t \\ \sum(\rho\ - pixel\ values)\ if\ (\rho - value\ ) > t \end{cases},$$

де p – центральний піксель;

t – поріг виявлення, а значення пікселів відповідають N суміжних пікселів у колі.

Функцію оцінки можна також визначити альтернативними способами. Ключовим моментом тут є визначення функції евристики, яка може порівняти два сусідні виявлені кути та усунути ті, що є незначними для порівняння [21].

На другому етапі детектор ORB використовує BRIEF, що бере всі КТ, знайдені алгоритмом FAST, і перетворює їх у бінарний вектор КТ, щоб разом вони могли представляти об’єкт. Бінарний вектор КТ також відомий як двійковий дескриптор КТ – це вектор КТ, який містить лише 1 і 0. За алгоритмом BRIEF, кожна ключова точка описується вектором КТ, який становить 128 – 512 біт.

Алгоритм BRIEF спочатку згладжує зображення за допомогою ядра Гауса (Gaussian kernel), щоб запобігти чутливості дескриптора до високочастотних шумів. Потім BRIEF обирає випадкову пару пікселів у визначеній області навколо цієї ключової точки. Визначена область навколо пікселя відома як патч, що являє собою квадрат певної ширини та висоти пікселя. Перший піксель у випадковій парі береться з розподілу Гауса (Gaussian distribution), зосередженого навколо КТ зі стандартним відхиленням або сігмою. Другий піксель у випадковій парі береться з гауссового розподілу, центрованого навколо першого пікселя зі стандартним відхиленням. Тепер, якщо перший піксель яскравіший за другий, він присвоює значення 1 відповідному біту, в іншому випадку – 0.

Далі, алгоритм BRIEF знову обирає випадкову пару і призначає значення. Наприклад, для 128-бітного вектора цей процес повторюється 128 разів для КТ. BRIEF створює такий вектор для кожної ключової точки на зображенні. Однак BRIEF також не є інваріантом щодо обертання, тому ORB

використовує rBRIEF (BRIEF з урахуванням обертання). ORB намагається додати цю функціональність, не втрачаючи швидкості BRIEF.

Нехай патч згладженого зображення – *p*. Бінарний тест τ визначається:

$$Where(p; x, y)\ is\ defined\ as:$$
$$\tau(p; x, y) = \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) \geq p(y) \end{cases}$$
$$p(x)\ is\ the\ intensity\ value\ at\ pixel\ x,$$

де *p (x)* – інтенсивність *p* в точці *x*.

КТ визначається як вектор *n* двійкових тестів:

$$f(n) = \sum_{1<i<n} 2^{i-1}\tau(p; x_i, y_i).$$

Ефективність відповідності BRIEF різко падає при обертанні в площині більше ніж на кілька градусів. ORB пропонує метод керування BRIEF відповідно до орієнтації ключових точок. Для кожного набору КТ з *n* двійкових тестів у точці (x_i, y_i) потрібна матриця $2 \times n$:

$$S = \begin{pmatrix} x_1, \ldots x_n \\ y_1, \ldots x_n \end{pmatrix}.$$

Вона використовує орієнтацію патча θ та відповідну матрицю обертання R_θ і створює керовану версію S_θ:

$$S_\theta = R_\theta S.$$

У результаті, керований оператор BRIEF має вид

$$g_n(p, \theta) = f_n(p) | (x_i, y_i) \in S_\theta.$$

Керований оператор дискретизує кут з кроком $2\pi / 30$ (12 градусів) і створює таблицю пошуку з попередньо обчислених паттернів BRIEF. До тих пір, поки орієнтація ключової точки θ буде послідовною у поданнях, для обчислення її дескриптора буде використовуватися правильний набір точок S_θ.

ORB визначає алгоритм rBRIEF наступним чином:

а) запустіть кожен тест на всіх навчальних патчах;

б) упорядкуйте тести за їх відстанню від середнього значення 0,5, утворюючи вектор *T*;

в) жадібний пошук:

1) помістіть перший тест у вектор результату *R* і видаліть його з *T*;
2) візьміть наступний тест з *T* і порівняйте його з усіма тестами в *P*. Якщо його абсолютна кореляція перевищує поріг, відкиньте його, у зворотньому випадку – додайте його до *R*;
3) Повторюйте попередній крок, поки в *R* не буде 256 тестів. Якщо їх меньше 256, підніміть поріг і повторіть спробу.

rBRIEF показує значне поліпшення дисперсії та кореляції порівняно з керованим BRIEF.

Алгоритм ORB використовує піраміду багатомасштабних зображень. Піраміда зображення (image pyramid) – це багатомасштабне зображення одного зображення, яке складається з послідовностей зображень, якими є версіями усіх зображення з різною роздільною здатністю. Кожен рівень піраміди містить зменшену версію зображення, ніж попередній рівень. Таким чином, ORB є частковим інваріантом щодо зміни масштабу зображення.

Дескриптори ORB мають вид бінарного вектора розміром 256. Дескриптори кодуються у Open CV у вигляді типу uchar (unsigned char – тип даних в C++, використовують для зберігання символів, об’єм 8 біт, значення 0…255), а не в бітах. Вони зберігаються в матриці, де кількість рядків дорівнює числу виявлених дескрипторів, а число стовпців

дорівнює 32 (256 бітів дескриптора трансформуються у 32 uchar). Дескриптор ORB (два приклади) має вигляд, зображений на рисунку 1.4, де одиниця позначена чорним кольором, а нуль – білим [1, 4].

Рисунок 1.4 – Гістограми для дескриптора ORB

1.3 Постановка задачі

Впровадження та моделювання статистичних мір для класифікації зображень на основі компонентного аналізу складу структурного опису є актуальним завданням в задачах обробки і розпізнавання образів.

Об’єктом роботи є методи класифікації зображень у системах комп’ютерного зору.

Метою роботи є побудова класифікатора зображень на основі компонентного аналізу даних – множини дескрипторів структурного опису із використанням статистичних мір для обчислення релевантності описів розпізнаваного об’єкту та еталонів.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів розпізнавання та класифікації зображень;

– реалізувати комп’ютерну модель для класифікації з використанням статистичних мір релевантності компонентів даних у базі еталонних зображень;

– провести дослідження результативності класифікаторів з використанням опису як множин дескрипторів ключових точок зображень;

– оцінити експериментально завадостійкість розроблених методів в умовах дії адитивних завад.

2 ЗАСТОСУВАННЯ ЗАСОБІВ СТАТИСТИКИ ДЛЯ ПОБУДОВИ КЛАСИФІКАТОРІВ

2.1 Формалізація задачі класифікації

Перспективною ідеєю в аспекті скорочення обсягу обчислювальних витрат для процесу класифікації візуальних об’єктів, які містить аналізоване зображення, є побудова деякої множини опорних точок у просторі дескрипторів КТ, які формують будь-який структурний опис [13, 22–25]. Комбінацією числа КТ, близьких до опорних точок у просторі їх значень (дескрипторів КТ), можна універсально отримати опис довільного образу об’єкта. Схожі можливості також забезпечує апарат мультимножин.

Розвиненням ідеї кластерного уявлення для структурних описів у межах прикладної бази зображень є безпосередня побудова правила класифікації, яке за значенням дескриптора буде відносити КТ до фіксованого еталонного класу. У результаті аналізу накопичується певна кількість голосів КТ, за якою згодом визначається номер класу для аналізованого об’єкта. Для близьких за змістом зображень таке правило із високою ефективністю важко реалізувати, тому що схожі зображення часто містять і чимало однакових дескрипторів опису. Дослідження науковців показують, що для достатнього різноманіття баз зображень синтез класифікаційного правила щодо окремих дескрипторів КТ все-таки може дати непогані прикладні результати [13]. Викликає прикладний інтерес застосування інтелектуальних технологій навчання у просторі описів як множин КТ для досягнення значимих показників класифікації [18, 25].

По своїй природі оброблювані дані у комп’ютерному зорі мають багатовимірний ймовірнісний розподіл. Але іноді їх можна подати у вигляді системи розподілів одновимірних величин, що значно спрощує класифікацію та суттєво знижує необхідні обчислювальні затрати [26].

Аналіз багатовимірних сигналів шляхом представлення їх як множини фрагментів досить популярний у методах оброблення зображень, так як часто найважливіша інформація про образи візуальних об’єктів зосереджується у окремих деталях.

Бітова природа дескрипторів КТ у просторі бінарних векторів дає можливість впровадити подання та аналіз окремого дескриптора як ланцюжка елементів (наприклад, байтів чи набору бітів), діапазон значень яких відомий. Це дає змогу здійснювати статистичний аналіз даних з урахуванням внутрішнього змісту наявного дескрипторного опису об’єкту. З точки зору важливості інформації всі елементи ланцюжка рівноцінні, але місце їх розміщення у складі дескриптора фіксоване, тому є можливість аналізувати чи обробляти упорядковані послідовності елементів. Ланцюжкова структура допускає застосування статистичних підходів інтелектуального аналізу, заснованих на ймовірнісних оцінках наявних значень даних, щоб прийняти рішення про віднесення об’єкту з описом у вигляді множини дескрипторів до відповідного класу [13].

Статистичне подання є одним із найбільш популярних інструментів у сучасному інтелектуальному аналізі даних задля виявлення закономірностей чи встановлення системи знань, що містять дані. Будь-які процедури голосування структурних ознак зображень базуються на статистичному аналізі даних [25, 27]. Методи статистичного аналізу ґрунтуються на виявленні у змісті аналізованої інформації суттєво значимих характеристик, вартість яких оцінюють значенням параметру частоти зустрічальності як оцінки ймовірності аналізованої події чи закономірності. Процедури навчання та самонавчання в інтелектуальних системах в основному теж базуються на визначенні найбільш вживаних компонентів даних [25, 28]. На підставі аналізу скінченного об’єму наявних даних здійснюють апроксимацію ймовірнісного розподілу, що дає змогу розпізнавати їх образи. Важливим аспектом, що досить точно відображає узагальнені властивості чи 8 характеристики даних,

є безпосереднє використання значень розподілів у методах розпізнавання [14, 16].

Один із можливих підходів побудови класифікатора засновано на використанні таких статистичних характеристик, як математичне очікування, дисперсія, оцінки медіани та ін. [17, 29]. Але більш інформативним є все-таки використання безпосередньо самих значень розподілів, як це впроваджено у методах інтелектуального аналізу. Це дає змогу суттєвіше врахувати відмінності та особливості значень даних, що відображають розрізнювальні властивості розпізнаваних класів для зображень.

Задача класифікації у векторному просторі може бути вирішена потужними засобами математичної статистики, призначення якої як раз і полягає у добуванні якомога більш повної інформації про аналізовані явища, виходячи зі обмеженого обсягу даних спостережень. Існує два принципово різних підходи статистичного оцінювання – баєсовський і частотний (зокрема, метричний) [2, 7, 26]. Перший інтерпретує випадковість як міру незнання, що закладено в ідеї переходу від апріорних знань до апостеріорних з урахуванням здійснених спостережень. Другий розглядає випадковість як об'єктивну невизначеність і спирається виключно на наявні спостереження, як правило, на частоти появи досліджуваних подій, 41 отже, не пов'язаний ні з якими апріорними припущеннями щодо розподілу досліджуваних величин.

Кожний із зазначених підходів може бути підвалиною для побудови мір релевантності для векторних описів об'єктів [6, 30, 31]. Ці міри для класифікації зображень мають певні особливості, зокрема, як переваги, так і окремі недоліки, що потребує осмислення і докладного аналізу із тестуванням.

Обчислення апостеріорних ймовірностей віднесення опису об'єкта до множини еталонів є підставою для безпосереднього здійснення класифікації. Основна ідея апарату Баєса полягає у ідентифікації аналізованого об'єкту з еталоном, що має найбільше значення апостеріорної ймовірності. Виходячи з кластерних описів еталонів $Z = \{Z^j\}_{j=1}^{J}$ і об'єкта $O = \{o_l\}$, за формулою Баєса

обчислюємо апостеріорні ймовірності $P(Z^j/O)$ належності об'єкта до кожного з описів Z^j:

$$P(Z^j/O) = \frac{P(O/Z^j)\cdot P(Z^j)}{\sum_{j=1}^{J} P(O/Z^j)\cdot P(Z^j)}, \quad (2.1)$$

де $P(O / Z^j)$ – апріорна ймовірність належності об'єкта до еталону Z^j;

$P(Z^j)$ – апріорна ймовірність появи еталону Z^j (вважаємо $P(Z^j) = 1/J$).

Дискретна природа задачі призводить до обчислення набору ймовірностей узагальненого гіпергеометричного розподілу [13]:

$$P(O/Z^j) = \prod_{i=1}^{k} C_{h_i[Z^J]}^{h_i[O]} / C_H^s, \quad (2.2)$$

де $s = \sum_{i=1}^{k} h_i[O]$ – потужність множини елементів опису об'єкту.

Формула (2.2) має сенс лише для невід'ємних цілих значень параметрів, з умовами

$$h_i[O] \le h_i[Z^j], s \le H, i = 1,\dots,k,\ j = 1,\dots,J. \quad (2.3)$$

Обмеження (2.3) потребують коригування даних, яке можна здійснити пропорційним змінюванням характеристик об'єкта. Коректування об'єкта при невиконанні обмежень (2.3) пропонуємо здійснювати з округленням результату до цілого числа за допомогою коефіцієнта $\lambda_{\text{кор}}^j$ за формулою:

$$h_i^{`}[O] = \lambda_{\text{кор}}^j \cdot h_i[O], i = 1,\dots,k,\ j = 1,\dots,J\ , \quad (2.4)$$

що є реалізацією оптимізаційної процедури загального вигляду

$$v = arg\ \ opt_{j=1,\dots,J}\ \Lambda(h[O], h[Z^j]). \quad (2.5)$$

Наведений метод є оптимальним у сенсі найменшої в середньому ймовірності помилки класифікації [13].

Частотні підходи нерідко називають метричними, навіть якщо міра Λ в (2.5) не задовольняє аксіомам метрики. Метричний класифікатор відносить досліджуваний об’єкт до еталону, міра подібності якого виявилася оптимальною. При цьому додатково здійснюють перевірку значущості отриманого мінімуму релевантності відповідно до встановленого порогу [27].

Найпоширенішою є евклідова метрика, яка ефективно використовується у випадках достатньої відмінності еталонів між собою, і часто демонструє нечутливість до варіативності даних у інших ситуаціях.

Останнього часу дослідники методів розпізнавання зображень зосереджують увагу на мірі «розходження Кульбака-Лейблера» (РКЛ). Цю міру для нашої задачі можна інтерпретувати як значення середньої інформації щодо відмінності об’єкту О від еталону Z^j [13]:

$$D_{KL}(O||Z^j) = \sum_{i=1}^{k} h_i^*[O]\, ln \frac{h_i^*[O]}{h_i^*[Z^J]}, \tag{2.6}$$

де $h_i^*[O]$, $h_i^*[Z^J]$ – елементи нормованих описів (2.1).

Сутність міри (2.6) пояснимо теоретичними міркуваннями, що обґрунтовують її застосування для класифікації. Обчислимо апостеріорні ймовірності $P(Z_j/O)$ за формулою (2.1), зокрема, для j = 1, j = 2 маємо:

$$\frac{\mathrm{P}(\mathrm{Z}^1/\mathrm{O})}{\mathrm{P}(\mathrm{Z}^2/\mathrm{O})} = \frac{\mathrm{P}(\mathrm{O}/\mathrm{Z}^1)\cdot\mathrm{P}(\mathrm{Z}^1)}{\mathrm{P}(\mathrm{O}/\mathrm{Z}^2)\cdot\mathrm{P}(\mathrm{Z}^2)}. \tag{2.7}$$

Після логарифмування (2.7)

$$\ln\frac{P(O/Z^1)}{P(O/Z^2)} = \ln\frac{P(Z^1/O)}{P(Z^2/O)} - \ln\frac{P(Z^1)}{P(Z^2)}. \tag{2.8}$$

Інтерпретація (2.8) при порівнянні двох еталонів [13]:

а) це різниця між логарифмами шансів належності об'єкту до Z^1 та Z^2 до та після спостережень на користь еталону Z^1, її можна інтерпретувати як інформацію, отриману в результаті спостереження;

б) це логарифм відношення правдоподібності, що трактується як інформація щодо об'єкту *O* для його розрізнення на користь Z^1 проти Z^2;

в) це критерій прийняття рішення, який полягає в обранні еталону, для якого логарифм відношення апостеріорної ймовірності до апріорної є більшим.

Випадкова величина, що приймає значення (2.8), виходячи з нормованих кластерних розподілів. Ліва частина (2.8) є $\ln\frac{h_i^*[O]}{h_i^*[Z^J]}$, $I = 1...k$, що можна розуміти як інформацію щодо об'єкту О на предмет його відмінності від еталону Z^1. Вважаємо, що така величина приймає вказані значення з ймовірностями (частотою) $h_i^*[O]$, $I = 1...k$. Обчислення її математичного сподівання приводить до виразу РКЛ (2.6) і інтерпретується як середня інформація щодо розрізнення об'єкту від еталону, або як середня величина втрат інформації при віднесенні О до еталону Z^1.

Міра (2.6) не є метрикою, оскільки не задовольняє аксіомам симетричності та нерівності трикутника, та є невід'ємною, причому рівність нулю має місце тоді і лише тоді, коли $h_i^*[O] = h_i^*[Z^J]$.

Міра (2.6) відповідає базовому методу математичної статистики, що дає оцінку параметрів розподілу, – методу максимальної правдоподібності, та приводить до вибору розподілу (еталону), що знаходиться на мінімальній відстані (2.6) від невідомого розподілу об'єкта [13, 32].

Узагальненням міри РКЛ є розходження Рен'ї, яке для нашої задачі можна розглядати як інформацію порядку $\alpha(\alpha > 0, \alpha \neq 1)$ щодо відмінності об'єкту О від еталону Z^1 [32]:

$$D_\alpha(O||Z^j) = \frac{1}{\alpha-1}\ln\sum_{i=1}^{k}\frac{(h_i^*[O])^\alpha}{(h_i^*[Z^J])^{\alpha-1}}\,. \qquad (2.9)$$

Розходження Рен'ї також не є метрикою, однак є на сьогодні прикладним інструментарієм для порівняння зображень чи їх описів за рахунок ряду властивостей, що забезпечують зрозуміле з практичної точки зору і ефективне з теоретичного погляду розпізнавання образів [32]. Міра (2.9) приймає невід'ємні значення, а нульове значення досягається лише при $h_i^*[O] = h_i^*[Z^J]$, I = 1…k, і є не спадною функцією аргументу α . При α →1 вираз (2.9) приймає вигляд (2.6) і відповідає важливому окремому випадку розходження Рен'ї – мірі РКЛ, тобто, $D_1(O||Z^j) = D_{KL}(O||Z^j)$.

Особливим можна також вважати значення параметра α = 0,5, що приводить міру (2.9) до відстані Бхаттачарія [33]:

$$D_{0,5}(O||Z^j) = -2\ln\sum_{i=1}^{k}\sqrt{h_i^*[O]\cdot h_i^*[Z^j]}, \qquad (2.10)$$

яка є єдиним випадком симетричності розходження Рен'ї відносно обох описів O і Z^j, а також є метрикою. Доведено [33], що оптимальним значенням параметра α ∈[0;1] є значення α = 0,5. При цьому оптимальним вважається значення параметра, при якому ненормована величина розходження Рен'ї $(\alpha-1)\cdot D_\alpha(O||Z^j)$ приймає найбільше значення, що відповідає нижній границі ймовірності помилки розпізнавання [13].

Таким чином, у випадку значної близькості об'єктів (за їх структурними описами) найкращим для їх розрізнення значенням параметру можна вважати α = 0,5, яке у певному сенсі мінімізує величину розходження Рен'ї. Окремий інтерес представляє також значення α = 2 , яке приводить міру (2.9) до вигляду логарифму математичного сподівання відношення частот $\frac{h_i^*[O]}{h_i^*[Z^J]}$:

$$D_2(O||Z^j) = ln\sum_{i=1}^{k}\frac{(h_i^*[O])^2}{h_i^*[Z^J]}\,. \qquad (2.11)$$

Наведена міра з точністю до монотонного перетворення еквівалентна відстані $\chi^2(O, Z^j)$:

$$D_2(O||Z^j) = ln\big(1 + \chi^2(O, Z^j)\big), \tag{2.12}$$

де $\chi^2(O, Z^j) = \sum_{i=1}^{k} \frac{(h_i^*[O] - h_i^*[Z^J])^2}{h_i^*[Z^J]}$ – відстань хі-квадрат, що відповідає класичному у математичній статистиці критерію χ^2 Пірсона, за яким перевіряється статистична гіпотеза про відсутність відмінностей двох структурних описів.

Беззаперечно, що основою сучасного апарату data science є статистичні методи, які теоретично та практично розвиваються вже протягом довгого часу. Їх перевагами є строге теоретичне обґрунтування та наявність розроблених програмних середовищ для практичного впровадження. Одним із найбільш застосованих у задачах комп'ютерного зору є програмні засоби бібліотеки OpenCV [34]. Бібліотека OpenCV містить понад 2500 оптимізованих класичних та сучасних алгоритмів аналізу зображень і машинного навчання. Можливості таких сучасних програмних бібліотек комп'ютерного зору, як Open CV, забезпечують вирішення ряду нагальних практичних задач: аналіз вмісту зображень, пошук та розпізнавання заданих об'єктів, виявлення тексту, відстеження рухів об'єктів, виявлення спільних елементів на порівнюваних зображеннях, реалізація методів навчання для прикладних баз відеоданих тощо [13].

Розглянемо багатовимірний простір B^{n} усяких бінарних векторів розмірністю n, де будемо конструювати образи об'єкту і еталонів. Зафіксуємо мультимножину векторів $E_i \subseteq \mathrm{B}^n$ як опис візуального об'єкту (зображення) у просторі множин дескрипторів КТ, $E_i = \{e_v(i)\}_{v=1}^{s}$, $s = card\ E_i$ – число дескрипторів у множині [1, 4]. Ознаки – це вектори $e_k \in \mathrm{B}^n$, скінченна множина яких створює опис об'єкту [6].

Задамо $\forall(e_k, e_\tau), e_k \in \mathrm{B}^n, e_\tau \in \mathrm{B}^n$ відстань ρ: $B^n \times B^n \to [0, \infty]$ у векторному просторі B^n. Прикладом є Хемінгова метрика, для бінарних даних діапазон значень цієї метрики фіксований – $[0, n]$. Відстань є фундаментальним критерієм еквівалентності на множині $\{e_k\}$, так як віддзеркалює візуальну схожість піксельних околів КТ для функції яскравості зображення, яку відображає дескриптор. Еквівалентність $e_{k\sim}e_\tau$ для двох дескрипторів e_k, e_τ визначаємо на підставі порогу δ_ρ для величини метрики:

$$\mathrm{e_k}\sim\mathrm{e_\tau} : \rho(\mathrm{e_{k\sim}e_\tau}) \leq \delta_\rho.$$

Класифікація передбачає наявність деякої бази Е описів еталонних зображень розмірністю N: $E = \{E_1, E_2, \ldots, E_N\}$. Кожен еталонний опис E_i репрезентує для класифікатора окремий клас та має вид скінченної множини дескрипторів КТ із B^n [6].

Задачею є побудова класифікатора K: $B^n \to [1,2,\ldots,N]$ на основі конструювання ймовірнісної системи ознак за результатом навчання на матеріалі бази еталонів $E = \{E_1, E_2, \ldots, E_N\}$ [1, 13].

Провідна ідея побудови класифікатора: для кожного дескриптора об’єкту чи еталонів встановити ступінь належності до встановлених класів у вигляді статистичного розподілу, а потім на підставі сформованої системи компонентних розподілів побудувати інтегровану ансамблеву міру релевантності щодо опису аналізованого об’єкту, а далі застосувати її у класифікаторі шляхом оптимізації значення релевантності у системі класів.

На підставі наявної бази описів еталонів шляхом навчання створюємо новий простір образів компонентних даних класифікації у складі значень їх ймовірнісної міри належності до класів. Впровадження такого підходу із використанням рішень ансамблю компонентів забезпечує універсальність і вагому результативність класифікації [6].

2.2 Побудова ансамблю розподілів для компонентів

Трансформуємо опис еталону $E_i = \{e_v(i)\}_{v=1}^{s}$ із бази E у n-мірному векторному просторі у деякий «центр опису» – агрегований вектор (i – номер еталону) [4, 30]

$$\alpha(i) = (\alpha_1(i), \alpha_2(i), \ldots, \alpha_n(i)),$$

який обчислюємо на підставі множини E_i. Центр опису α можна визначити, наприклад, шляхом обчислення середнього значення чи медіани для фіксованої множини векторів [6, 7, 26]. Обчислимо вектори $\alpha(i)$ як статистичні характеристики для кожного із еталонів і покладемо їх в підґрунтя класифікації.

Окремий вектор $e_v(i) \in E_i$ еталону, як і будь-який дескриптор об'єкту, можна формально розглядати як елемент ансамблю n-компонентних векторів

$$e_v(i) = (e_{v,1}(i), e_{v,2}(i), \ldots, e_{v,n}(i)). \quad (2.13)$$

Розглянемо спектр аналізованих даних у аспекті віднесення складового елементу опису до системи еталонних класів на підставі визначення деякої функції належності зі значеннями із діапазону 0…1 [8, 35, 36]:

$$\mu: B^n \to [0,1], \mu(e_v(i)) \in [0,1], \quad (2.14)$$

аргументами функції μ є дескриптор опису і номер класу.

Функцію належності μ визначимо на підставі основоположної характеристики у data science – співвідношення значень мір, що виражають число сприятливих випадків (подібність до конкретного класу) та загального числа N випадків (сума подібностей до усіх класів) [3, 6, 30]

$$\mu(e_v(i)) = \frac{\eta(e_v,i)}{\sum_{i=1}^{N}\eta(e_v,i)}. \quad (2.15)$$

Загальне число випадків задається кількістю N класів. Міра подібності $\eta(e_v,i)$ елемента до класу може бути задана через відстань ρ у векторному просторі до центра класу, наприклад, через манхетенську метрику

$$\rho(e_v,i) = \sum_{k=1}^{n} | e_{v,k}(i) - \alpha_k(i) |. \quad (2.16)$$

Для випадку, якщо $\alpha(i) \in B^n$, $e_v(i) \in B^n$ замість (2.16) можна застосувати відстань $\chi(e_v,i)$ Хемінга (число не співпадаючих бітів) у просторі B^n, тоді подібність $\eta(e_v,i)$ буде визначена як $\eta(e_v,i) = n - \chi(e_v,i)$.

Для кожного елемента $e_v(i)$ за виразом (2.15) обчислимо значення вектора d його статистичного розподілу за множиною N класів

$$d = (d(1), d(2), \ldots, d(N)),\ d(i) = \mu(e_v(i)),\ \sum_{i=1}^{N} d(i) = 1. \quad (2.17)$$

Зі статистичної точки зору вектор d для довільного дескриптору еталону (об’єкту) виражає ступінь близькості дескриптора до класу.

Будемо розглядати матрицю $D = \{\{d_k(i)\}_{k=1}^{s}\}_{i=1}^{N}$, що аналогічно нечіткому поданню задає значення міри належності (2.15) для всіх компонентів аналізованого опису. Фактично D визначає розподіл сукупностей даних за визначеними апріорі класами.

Значення матриці D дають можливість запровадити логічне оброблення вхідних даних на предмет видалення можливих завад (тобто хибних дескрипторів) шляхом аналізу значень відстаней (2.16) чи значень (2.17) з використанням порогу. У той же час цей аналіз може бути безпосередньо впроваджено на етапі класифікації [6].

2.3 Моделі класифікатора

На основі матриці D побудуємо класифікатор K, який для структурного опису довільного об’єкта реалізує відображення

$$K: D \to [1,2,\ldots,N]$$

із множини розподілів компонентів даних у множину класів.

На першому етапі збудуємо розподіли (2.17) за класами даних для множини еталонів бази. Зрозуміло, що для кожного представника $E_i \in E$ класифікатор K повинен отримати номер відповідного еталону, опис якого поступає на вхід класифікатора. Це є першочерговим принципом адекватності функціонування класифікатора, який повинен впевнено розрізняти описи із множини еталонів. Наприклад, у розподілі даних для 1-го еталону (1-й стовпець матриці D) перша компонента повинна домінувати над іншими. Аналогічно для 2-го еталону домінуючим елементом розподілу повинен бути 2-й і т.д.

Проаналізуємо можливі способи побудови класифікатора.

1. Визначення стовпця матриці D з максимальною сумою елементів

$$K: \text{j}=\arg\max_{i} \sum_{v=1}^{s} d_v(i), \tag{2.18}$$

що встановлює клас j об’єкту через агрегацію розподілів кожного із класів (окремий стовпець) за всією множиною складових опису. Класифікація (18) відповідає найбільш правдоподібному рішенню, так як побудована на додаванні значень однотипних розподілів [1, 2, 7, 26].

2. Обчислення максимального значення для кожного рядка матриці D

$$c_v = arg \max_{i=1,\ldots,N} \{d_v(i)\}, \tag{2.19}$$

тобто шляхом визначення для кожного дескриптора опису найбільш вагомого класу за вектором розподілів, що відповідає параметру моди [2, 35, 36]. За результатом (2.19) для всієї множини дескрипторів опису отримуємо вектор голосів

$$h = (h_1, h_2, \ldots, h_N),\ h_b = \sum_{v=1}^{s} c_v,\ b = \overline{1, N}, \tag{2.20}$$

на підставі якого визначимо номер класу

$$r = arg \ \max_{b} \mathrm{h}_b, \tag{2.21}$$

що набрав максимальне значення серед голосів дескрипторів об’єкту. Це метод голосування на множині дескрипторів, де клас визначається на підґрунті моди розподілу [7, 37].

За результатами попереднього оброблення еталонної інформації на підставі матриці розподілів можна оцінити результативність застосування запропонованого підходу для множини еталонних описів.

Наприклад, точність prec класифікації можна оцінити відношенням гуртового числа TP дескрипторів бази еталонів, для яких правильно визначено клас, до загального їх числа у відповідності до моделі [2].

$$Prec = TP/(TP+FP). \tag{2.22}$$

Показник повноти compl (частота істинно позитивних результатів) оцінюється як доля правильно розпізнаних дескрипторів еталону.

$$Compl = TP/(TP+FN). \tag{2.23}$$

Наведені показники (2.22, 2.23) якості розпізнавання є нечутливими до істинно негативних результатів, що можна вважати їх перевагою, оскільки в реальних прикладах таких результатів може бути значна кількість [2, 38].

Розглянуті варіанти (2.18), (2.19)–(2.21) побудови класифікатора природно можна трактувати в рамках теорії ансамблевих моделей, де за рахунок створення та агрегування відгуків компонентних класифікаторів (локальних рішень) синтезується «сильний» класифікатор із гарантовано вищою результативністю прийняття рішень. Найбільше розглянуті підходи відповідають моделі бустінгу [13, 30, 39].

За результатами досліджень ансамбль класифікаторів у більшості випадків забезпечує кращу точність аналізу даних чи навчання, однак, викликає необхідність вирішування ряду проблем, таких як суттєве збільшення часових та обчислювальних витрат, складність інтерпретації результатів, обґрунтування та вибір способів комбінування локальних рішень [1, 8, 13, 30]. У нашому випадку гурт локальних класифікаторів складається із базових моделей одного типу, тобто є однорідним. Загалом ансамблеве класифікаційне рішення Θ можна подати у виді комбінування скінченної множини локальних рішень θ_v.

$$\text{K: } j = \arg\operatorname*{opt}_{i} \Theta\left[\{\theta_v(d_v(i))\}_{v=1}^{s}\right]. \qquad (2.24)$$

У ансамблевих моделях аналізу зображень з метою врахування тільки значимих локальних рішень часто застосовують систему параметрів порогів [1, 4, 7], яка забезпечує відділення завад і загалом підвищує надійність. Наприклад, у модифікації класифікатора (2.19)–(2.21) клас $d_m = \max\limits_{i=1,\dots,N}\{d_v(i)\}$ для локального рішення визначається тільки у випадку, якщо виконується умова, що порівнює значення оптимуму d_m з порогом δ_d або з найближчим до нього локальним оптимумом d_{m-1} (λ – числовий коефіцієнт):

$$d_m > \delta_d \text{ або } d_m > \lambda d_{m-1}. \quad (2.25)$$

Зауважимо, що поріг δ_d можна встановити за результатами навчання із вчителем на множині дескрипторів для еталонів, тобто результативне значення моди має бути не меншим від його значення для «свого» еталону. Цей аналіз на етапі навчання класифікатора треба проробити для усіх еталонів і у якості порогу δ_d обрати серед усіх отриманих мод найбільше [6, 30].

Один із засобів підвищення результативності ансамблевих методів класифікації шляхом адаптації до наявних даних полягає в тому, щоб на підставі введення логічних процедур відібрати деяку концентровану (або найбільш інформативну) підмножину елементів, які стануть підґрунтям для класифікаційного рішення [6, 8, 30]. Стосовно опису як множини дескрипторів КТ така процедура може полягати у відборі підмножини елементів, що знаходяться в межах заданої відстані від центру, або фіксованого числа елементів, найближчих до центру опису. Нехай $Z \subset B^n$ – опис, $s = card\ Z$ – його потужність, тоді введемо процедуру

$$\mathrm{L(Z)} \to \mathrm{Z^*},\ \mathrm{Z^*} \subset \mathrm{B^n},\ \mathrm{card\ Z^* = s^*},\ \mathrm{s^* < s}. \quad (2.26)$$

Застосування (2.26) не тільки значно скорочує час обчислень але й часто сприяє покращенню показників класифікації [30]. Процедура *L* реалізується на етапі попереднього оброблення, тому на час здійснення класифікації не впливає.

Для перевірки завадостійкості розробленого класифікатора застосовано адитивний гаусовий шум. До кожного пікселя зображення додається значення шуму з відповідним математичним очікуванням (як правило нульовим) та СКВ нормального розподілу. Функція щільності розподілу визначається за формулою:

$$\rho_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \quad (2.27)$$

де μ – математичне очікування;

σ – СКВ.

3 РЕЗУЛЬТАТИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

3.1 Обґрунтування вибору програмного середовища

У ході реалізації методу дослідження результативності класифікаторів зображень було застосовано мову програмування Python, а також бібліотеку відкритого доступу OpenCV [34].

OpenCV (Open Source Computer Vision Library) – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C / C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних і комерційних цілях – поширюється в умовах ліцензії BSD.

У бібліотеку входять понад 2500 алгоритмів, в яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації людських дій у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок зі стереокамер, зшивання зображень для отримання високої роздільної здатності зображення цілої сцени, знайти подібні зображення з бази даних зображень, видалити червоні очі із зображень, зроблених за допомогою спалаху, стежити за рухами очей, розпізнавати декорації та встановлювати маркери, щоб накласти їх на доповнену реальність тощо. OpenCV має понад 47 тисяч користувачів користувачів спільноти та передбачає кількість завантажень, що перевищує 18 мільйонів. Бібліотека широко використовується у компаніях, дослідницьких групах та державних органах [34].

Поряд з такими відомими компаніями, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, існує безліч стартапів, таких як Applied Minds, VideoSurf та Zeitera, що широко

використовують OpenCV [34]. Розгорнуте використання OpenCV – від зшивання зображень вуличного перегляду, виявлення вторгнень у відеоспостереження в Ізраїлі, моніторингу шахтного обладнання в Китаї, допомоги роботам в орієнтації та підхопленні об’єктів у Willow Garage, виявлення аварій при потопленні басейну в Європі, запуску інтерактивного мистецтва в Іспанія та Нью-Йорк, перевіряючи злітно-посадкові смуги на наявність сміття в Туреччині, перевіряючи ярлики на продуктах на заводах по всьому світу для швидкого виявлення обличчя в Японії [40, 41].

Бібліотека містить інтерфейси C++, Python, Java та MATLAB та підтримує Windows, Linux, Android та Mac OS. OpenCV схиляється здебільшого до програм зору в реальному часі та використовує переваги інструкцій MMX та SSE, коли вони доступні. Зараз розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написаний мовою C++ і має шаблонний інтерфейс, який безперебійно працює з контейнерами STL.

У Python є багато можливостей, деякі з яких можна виділити такі:

– дуже зручна для розробників мова, що означає, що кожен і кожен може навчитися кодувати її за пару годин або днів. У порівнянні з іншими об’єктно-орієнтованими мовами програмування, такими як Java, C, C++ та C#, Python є однією з найпростіших у вивченні;

– мова програмування з відкритим кодом, що означає, що кожен може створювати та сприяти її розвитку. У Python є інтернет-форум, на якому щодня збираються тисячі кодерів для подальшого вдосконалення цієї мови. Разом з цим Python можна безкоштовно завантажувати та використовувати в будь-якій операційній системі, будь то Windows, Mac або Linux;

– графічний інтерфейс або графічний інтерфейс користувача є одним із ключових аспектів будь-якої мови програмування, оскільки він має можливість додати чуття коду та зробити результати більш наочними. Python підтримує широкий спектр графічних інтерфейсів, які можна легко

імпортувати в інтерпретатор, що робить це однією з найулюбленіших мов для розробників;

– одним з ключових аспектів Python є його об’єктно–орієнтований підхід. Це в основному означає, що Python визнає концепцію інкапсуляції класів та об’єктів, що дозволяє програмам бути ефективними в довгостроковій перспективі;

– Python був розроблений як мова програмування високого рівня, а це означає, що коли ви кодуєте на Python, вам не потрібно знати про структуру кодування, архітектуру, а також управління пам’яттю;

– інтегрована за своєю природою мова. Це означає, що інтерпретатор python виконує коди по одному рядку за раз. На відміну від інших об’єктно-орієнтованих мов програмування, нам не потрібно компілювати код Python, що робить процес налагодження набагато простішим та ефективним. Ще однією перевагою цього є те, що після виконання код Python негайно перетворюється в проміжну форму, також відому як байт-код, що полегшує виконання, а також економить час роботи в довгостроковій перспективі [40, 41].

3.2 Особливості програмної реалізації

Для реалізації методу застосовано базу зображень, що містить три еталони, на основі яких будуються центри класів, а також базу зображень, що порівнюються з центрами класів еталонних зображень. Із застосування бібліотеки OpenCV будуються дескриптори ORB для кожного зображення (обрано значення = 500), після чого розраховуються центри.

Центри класів було вирішено знайти двома способами для досягнення більш точних результатів.

Для розрахування за байтовим поданням було створено функцію findCentersByBytes. У якості параметру функція отримує вектор дескрипторів. Байти кожного дескриптора додаються за стовпцями, потім діляться на 500 за

кількістю дескрипторів. Результатом роботи функції буде вектор із 32 чисел-байтів (рис. 3.1).

```
def findCentersByBytes(descriptors):
    centers = []
    for i in range(len(descriptors[0])):
        centers.append(sumColumn(descriptors, i) / 500)
    return centers
```

Рисунок 3.1 – Програмна реалізація функції findCentersByBytes

Для обчислення за бітовим представленням працюємо у бінарній арифметиці, додаємо відповідні біти усіх дескрипторів, у результаті чого отримано число від 0 до 500, нормуємо на 500. Отримуємо вектор з 256 чисел з плаваючою комою (рис. 3.2).

```
def findCentersByBits(descriptors):
    numbersListOfLists = []
    binaryVectors = descriptorsToBinaryStrLists(descriptors)

    for i in range(len(descriptors[0])):
        numbersListOfLists.append(sumBitsColumn(binaryVectors, i))
    flat_list = [item for sublist in numbersListOfLists for item in
sublist]
    center = [x / 500 for x in flat_list]
    return center
```

Рисунок 3.2 – Програмна реалізація функції findCentersByBits

Кожен дескриптор порівнюється з усіма центрами за допомогою розрахунку відстані по Хемінгу й відноситися до найближчого за відстанню (рис 3.3).

```
def hamming2(s1, s2):
    """Calculate the Hamming distance between two bit strings"""
    assert len(s1) == len(s2)
    return sum(c1 != c2 for c1, c2 in zip(s1, s2))
```

Рисунок 3.3 – Програмна реалізація функції hamming

Необхідно визначити належність кожного дескриптора розглядаємого зображення до кожного центру класів. Для цього функція functionOfAttachment приймає у якості параметрів розраховані відстані Хемінга, а також розмір досліджуваних дескрипторів (256 біти). Функція

functionOfAttachment застосовує функцію measuresOfLikeness для розрахунку міри подібності (рис. 3.4).

```
def functionOfAttachment(distances, descriptor_size):
    allMeasures = measuresOfLikeness(distances, descriptor_size)
    attachments = []

    for measures in allMeasures:
        attachment = []
        sumOfMeasures = sum(measures)
        for measure in measures:
            attachment.append(measure / sumOfMeasures)
        attachments.append(attachment)

    return attachments

def measuresOfLikeness(allDistances, descriptor_size):
    allMeasures = []
    for distances in allDistances:
        measures = []
        for distance in distances:
            measures.append(descriptor_size - distance)
        allMeasures.append(measures)
    return allMeasures
```

Рисунок 3.4 – Програмна реалізація функцій functionOfAttachment й measuresOfLikeness

Для того, щоб оцінити правильність розпізнавання еталонів, будується функція getAggregateNumbersForAttachmentEtalone, що приймає у якості параметру вектори значень функції належності, що були розраховані на попередньому кроці. Метод підсумовує стовпці розподілів для кожного з еталонів й повертає вектор агрегованих значень (рис. 3.5).

```
def getAggregateNumbersForAttachmentEtalone(attachments):
    aggregateNumbers = [sum(x) for x in zip(*attachments)]
    return aggregateNumbers
```

Рисунок 3.5 – Програмна реалізація функції getAggregateNumbersForAttachmentEtalone

Щоб виконати розподіл дескрипторів на еталони та отримати вектори значень функції належності, обирається максимальне число й відповідно до номеру класу оновлюється значення елементу словника класифікаторів.

Словники у мові програмування Python – невпорядковані колекції довільних об’єктів з доступом по ключу. Їх іноді ще називають асоціативними масивами або хеш-таблицями.

Функція getClassifications повертає список класифікаторів (рис. 3.6).

```
def getClassifications(attachments):
    listsOfClassifications = []
    for row in attachments:
        classifications = {1: 0, 2: 0, 3: 0}
        ind = 1

        for number in row:
            if number == max(row):
                classifications.update({ind: 1})
                break
            ind = ind + 1
        listsOfClassifications.append(classifications)

    return listsOfClassifications
```

Рисунок 3.6 – Програмна реалізація функції getClassifications

Функція getClassificationCounts будується для оброблення результатів методу getClassifications. Проводиться підрахунок кількості значень, що належать до певного еталону. Результатом роботи функції буде словник, який зберігає номери еталонних класів з відповідною кількістю дескрипторів, що належать до них (рис. 3.7).

```
def getClassificationCounts(listsOfClassifications):
    res = {1: 0, 2: 0, 3: 0}
    for dictionary in listsOfClassifications:
        res = {x: res.get(x, 0) + dictionary.get(x, 0) for x in
set(res).union(dictionary)}

    return res
```

Рисунок 3.7 – Програмна реалізація функції getClassificationCounts

Для подальшої оцінки досліджуваної класифікації необхідно реалізувати функцію, що розраховує критерій повноти. Вхідні параметри включають розраховані раніше словники класифікацій, а також номер еталонного класу. Проводиться підрахування правильно визначених до класу еталону дескрипторів, а також неправильно визначених до інших класів з

інкрементацією відповідних змінних й подальшим калькулюванням результату (рис. 3.8).

```
def getRecall(classifications, numberOfClass):
    countCorrectClassification = 0
    countFalseClassification = 0

    for dictionary in classifications:
        if dictionary.get(numberOfClass) == 1:
            countCorrectClassification = countCorrectClassification + 1
        else:
            countFalseClassification = countFalseClassification + 1
    result = countCorrectClassification / (countCorrectClassification +
countFalseClassification)

    return result
```

Рисунок 3.8 – Програмна реалізація функції getRecall

Друга функція, що необхідна для оцінювання результативності класифікації за назвою getPrecision, повертає значення критерію точності. Вона приймає у якості вхідних параметрів словники класифікацій класу еталону, номер класу еталону, а також словники класифікацій усіх інших еталонів. Проводиться інкрементація змінної, що відповідає за кількість правильно визначених до елементів класу еталону, а також змінної, що містить кількість дескрипторів інших класів, які невірно визначені до аналізованого класу. У результаті, функція повертає остаточне значення критерію точності, що було визначено калькулюванням зазначених вище змінних (рис. 3.9).

```
def getPrecision(classificationsOfCurrent, numberOfClass,
classificationsOfSecond,classificationsOfThird):
    countFalseClassificationOfOthers = 0
    for dictionary in classificationsOfCurrent:
        if dictionary.get(numberOfClass) == 1:
            countCorrectClassification = countCorrectClassification+1
    for dictionary in classificationsOfSecond:
        if dictionary.get(numberOfClass) == 1:
            countFalseClassificationOfOthers =
countFalseClassificationOfOthers + 1
    for dictionary in classificationsOfThird:
        if dictionary.get(numberOfClass) == 1:
            countFalseClassificationOfOthers =
countFalseClassificationOfOthers + 1
    result = countCorrectClassification / (countCorrectClassification
+ countFalseClassificationOfOthers)
    return result
```

Рисунок 3.9 – Програмна реалізація функції getPrecision

3.3 Інструкція користувача

Розроблена програма представляє собою набір методів, що сумісно дозволяють виконати послідовне моделювання статистичних мір для класифікації зображень на основі компонентного аналізу, тому її буде розглянуто у якості бібліотеки. Для її використання спочатку необхідно підключити бібліотеку відкритого доступу OpenCV. Це продемонстровано наступній послідовністю дій:

Крок 1. Завантаження дистрибутиву Anaconda для мови Python.

На офіційному сайті https://www.anaconda.com/*, у нашому випадку, завантажте Anaconda для ОС Windows 64-бітної версії для Python 3.8.

Відкривши сторінку сайту, буде відображено меню, продемонстроване на рисунку 3.10.

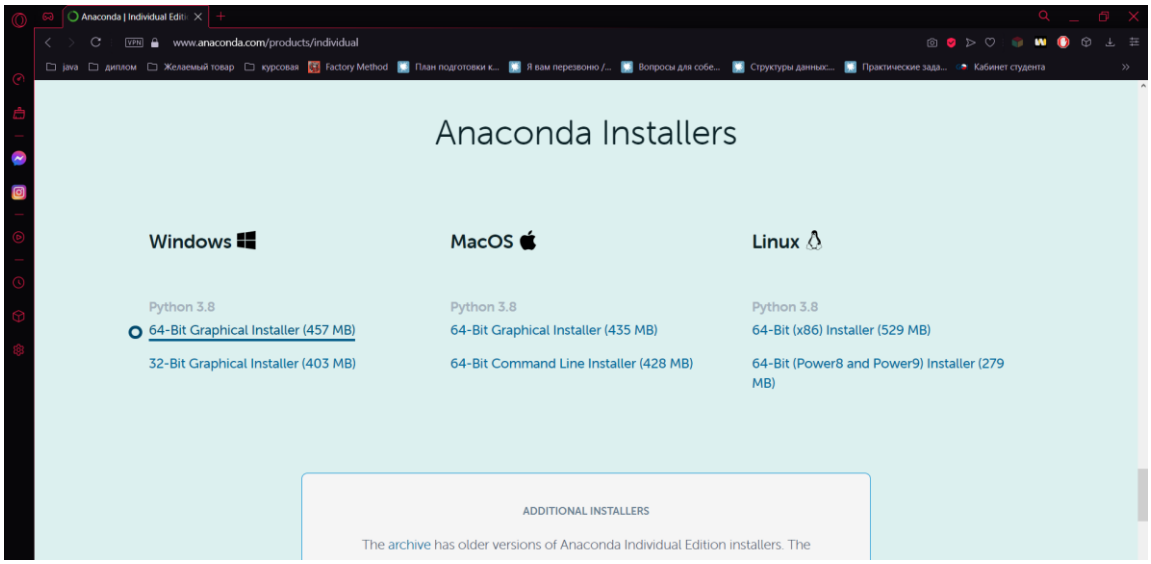

Рисунок 3.10 – Сторінка офіційного сайту Anaconda

Відкривши інсталятор Anaconda й дійшовши до пункту Advanced Options, обов'язково додайте Anaconda3 до PATH змінних оточення й зареєструйте Anaconda3 у якості об'єкту за замовченням, що продемонстровано на рисунку 3.11.

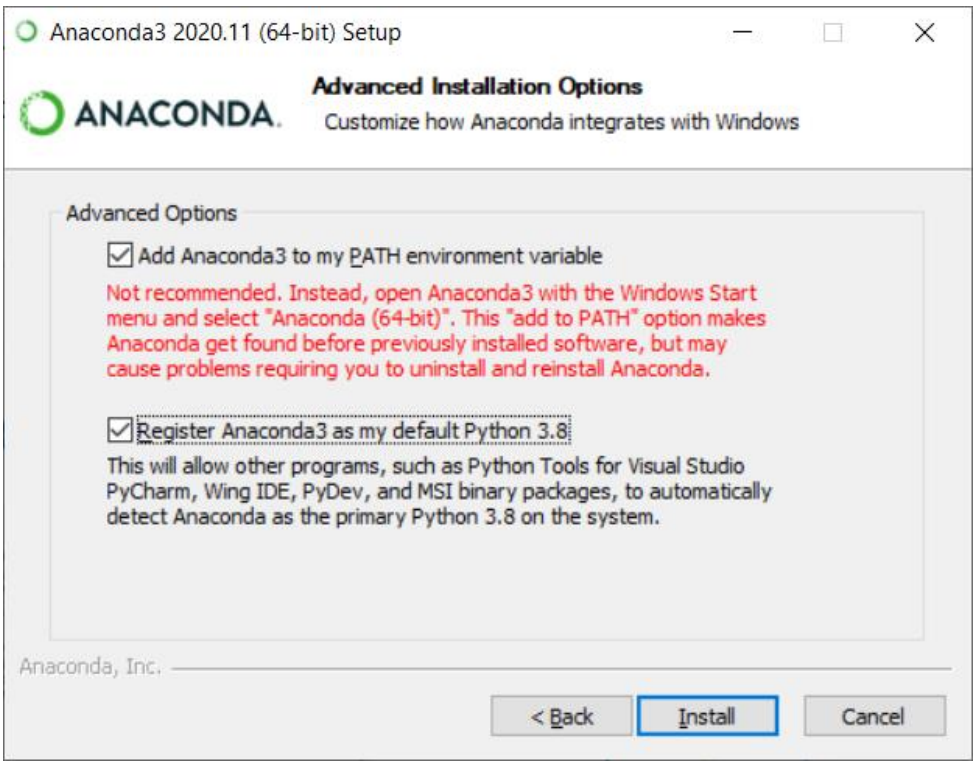

Рисунок 3.11 – Сторінка інсталятору Anaconda

Крок 2. Створіть віртуальне оточення.

Відкрийте командний рядок й пропишіть команду, що зазначена на рисунку 3.12.

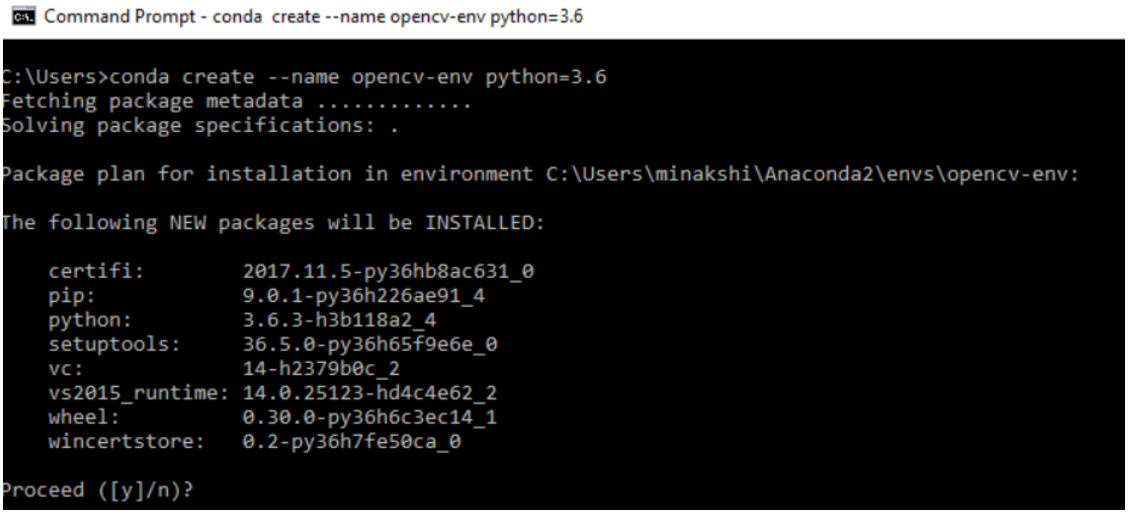

Рисунок 3.12 – Результат введення команди створення оточення

Після чого, натисніть на Enter, ви маєте отримати наступний результат, що продемонстрований на рисунку 3.13.

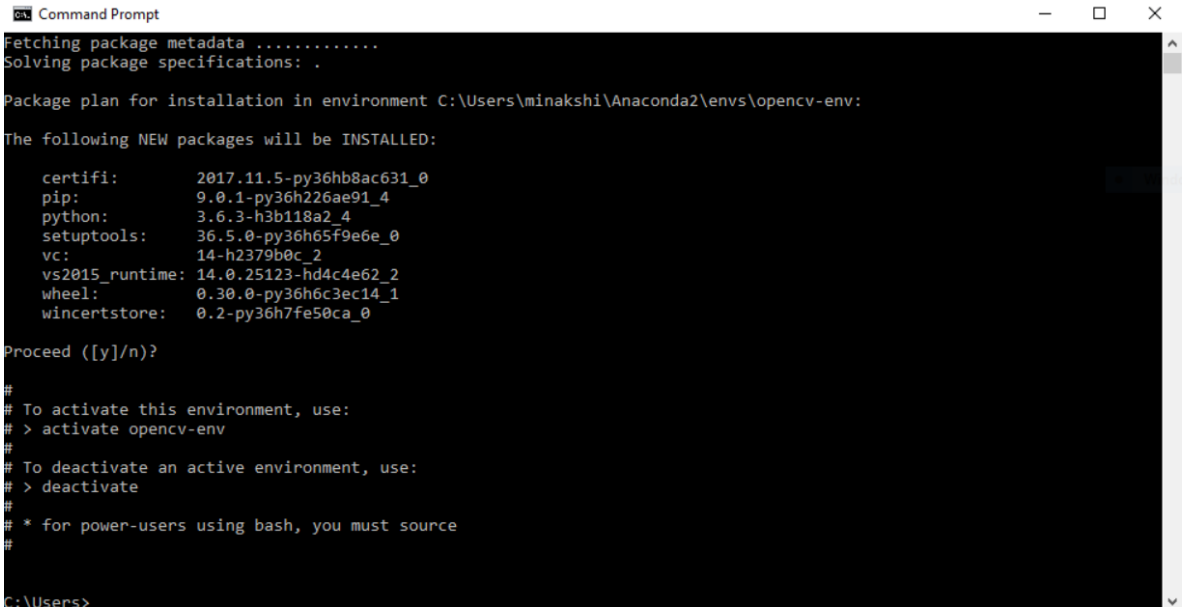

Рисунок 3.13 – Результат інсталювання віртуального оточення

Крок 3. Завантаження відкритої бібліотеки OpenCV для Python.

Щоб активувати оточення для OpenCV, у командному рядку пропишіть команди, що продемонстровані на рисунку 3.14.

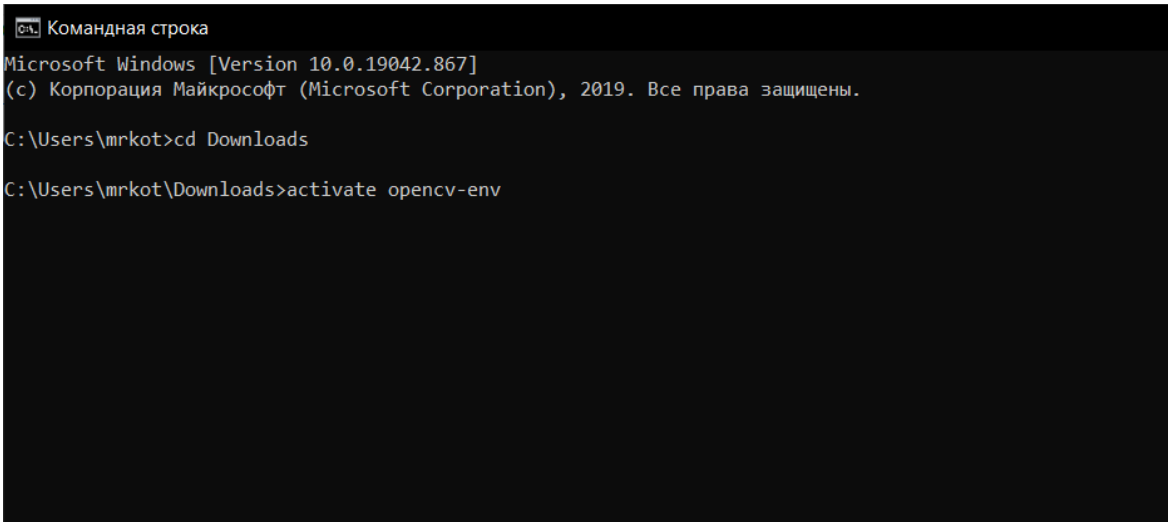

Рисунок 3.14 – Демонстрація введення команд для активування оточення

Далі, у командному рядку Python, введіть наступні команди, що продемонстровані на рисунку 3.15.

```
pip install numpy scipy matplotlib scikit-learn jupyter
pip install opencv-contrib-python
pip install dlib
```

Рисунок 3.15 – Команди, що інсталюють бібліотеки

Щоб імпортувати cv2 й dlib й остаточно підготувати бібліотеку OpenCV до застосування, введіть наступні команди у командному рядку Python, продемонстровані на рисунку 3.16.

```
Python 3.6.3 |Anaconda, Inc.| (default, Nov  8 2017, 15:10:56) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.3.1'
>>> import dlib
>>> dlib.__version__
'19.7.0'
>>>
```

Рисунок 3.16 – Демонстрація введення команд для імпортування cv2 й dlib

Крок 4. Застосування бібліотеки OpenCV.

Початкові дані, необхідні для подальшої побудови дескрипторів, знаходяться у папці проекту за назвою images. За допомогою вбудованого методу бібліотеки cv2 imread, збережіть зображення у змінні. Потім, виконайте створення змінної orb, визвавши метод ORB_create й вказавши потрібну вам кількість дескрипторів, що будуть розраховані. Метод detectAndCompute здійснить обчислення, результатом чого будуть координати КТ та набір розрахованих дескрипторів (рис. 3.17).

```
import cv2
import centers

img_or_chimaera = cv2.imread("images/butterfly_chimaera.jpg",
cv2.IMREAD_ANYCOLOR)
img_or_machaon = cv2.imread("images/butterfly_machaon.jpg",
cv2.IMREAD_ANYCOLOR)
img_or_davisii = cv2.imread("images/butterfly_davisii.jpg",
cv2.IMREAD_ANYCOLOR)
orb = cv2.ORB_create(nfeatures=500)
keypoints_orb_chimaera, descriptors_chimaera =
orb.detectAndCompute(img_or_chimaera, None)
keypoints_orb_machaon, descriptors_machaon =
orb.detectAndCompute(img_or_machaon, None)
```

Рисунок 3.17 – Зчитування зображень й побудова дескрипторів ORB

Метод drawkeypoints здійснить побудову дескрипторів за координатами, що зберегаються у змінній keypoints_orb, після чого виконайте завантаження результату за допомогою imwrite, вказавши шлях до папки. Завершуючим кроком буде виконання методу imshow для відображення результату роботи ORB детектора (рис. 3.18).

```
img_ORB_chimaera = cv2.drawKeypoints(img_or_chimaera,
keypoints_orb_chimaera, None, (0, 255, 0))

img_ORB_machaon = cv2.drawKeypoints(img_or_machaon,
keypoints_orb_machaon, None, (0, 255, 0))

img_ORB_davisii = cv2.drawKeypoints(img_or_davisii,
keypoints_orb_davisii, None, (0, 255, 0))

cv2.imwrite("images/butterfly_chimaera_ORB.jpg", img_ORB_chimaera)

cv2.imwrite("images/butterfly_machaon_ORB.jpg", img_ORB_machaon)

cv2.imwrite("images/butterfly_davisii_ORB.jpg", img_ORB_davisii)

cv2.imshow("original_chimaera", img_or_chimaera)

cv2.imshow("original_machaon", img_or_machaon)

cv2.imshow("original_davisii", img_or_davisii)

cv2.imshow("ORB_chimaera", img_ORB_chimaera)

cv2.imshow("ORB_machaon", img_ORB_machaon)


cv2.imshow("ORB_davisii", img_ORB_davisii)
```

Рисунок 3.18 – Запис зображень з побудованими дескрипторами ORB й відображення користувачу

3.4 Аналіз експериментальних результатів

У ході дослідження сформовано базу еталонних зображень, що містить три зображення метеликів, на кожному з яких знайдено 500 детекторів ключових точок із використанням методу ORB (рис. 3.19) [11].

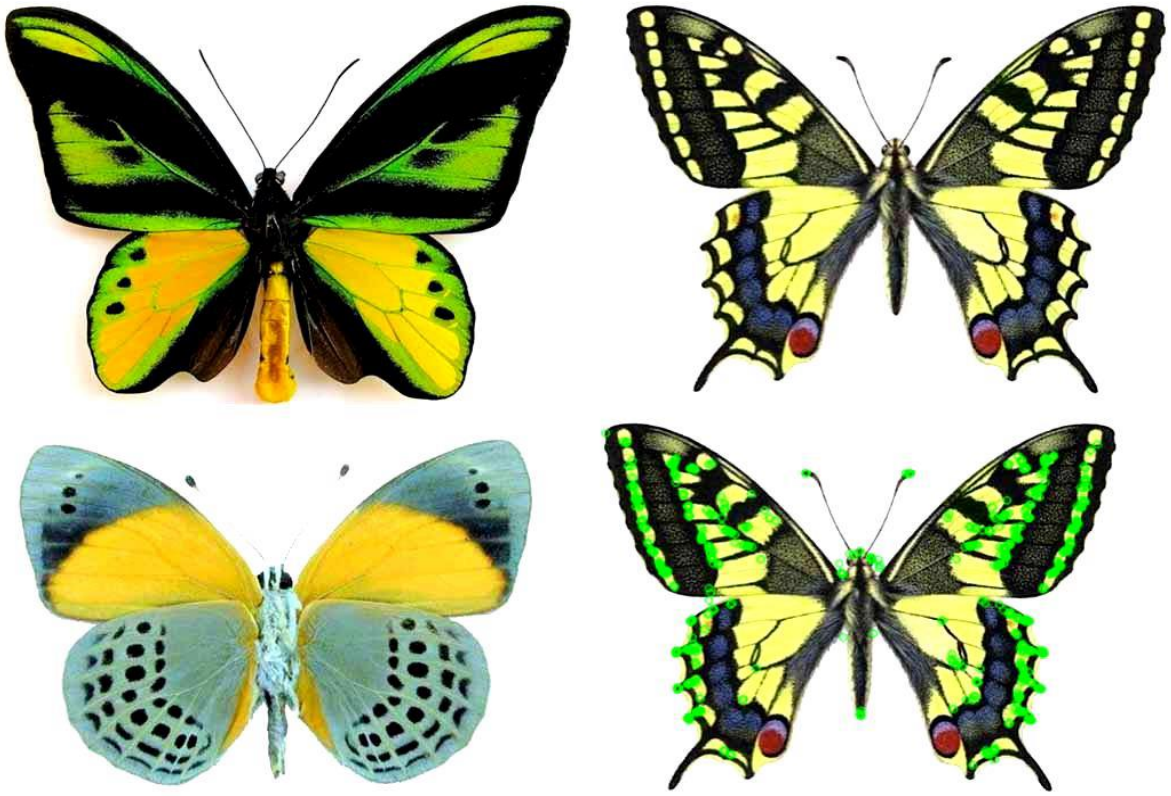

Рисунок 3.19 – Еталонні зображення метеликів Химера, Махаон, Давісій та виділені координати КТ

Застосовано два способи пошуку центру класу для кожного із описів як множини векторів за байтовим поданням, а також за бітовим поданням.

За байтовим поданням: байти кожного дескриптора просумовано за стовпцями, потім поділено на 500 за кількістю дескрипторів. Отримано вектор із 32 чисел-байтів, що продемонстровано на рисунку 3.20.

```
[109.252, 124.77, 142.224, 176.94, 139.494, 128.834, 148.888, 142.504, 109.9, 137.078, 130.358,
 137.484, 117.068, 138.39, 143.822, 136.742, 150.5, 141.798, 122.688, 140.004, 145.866,
126.912, 110.538, 184.148, 122.478, 125.226, 157.35, 150.062, 127.438, 145.614, 135.756,
139.718]
```

Рисунок 3.20 – Приклад множини векторів за байтовим поданням

За бітовим представленням: працюємо у бінарній арифметиці, додаємо відповідні біти усіх дескрипторів, у результаті чого отримано число від 0 до 500, нормуємо на 500. Отримано вектор з 256 чисел з плаваючою комою, що продемонстровано на рисунку 3.21.

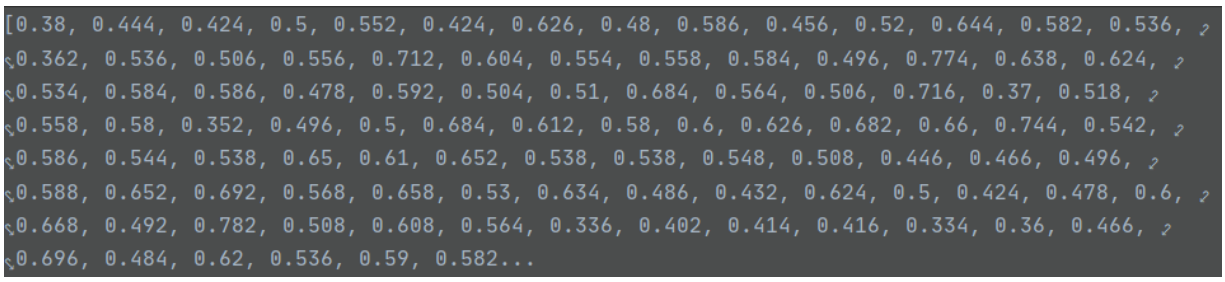

Рисунок 3.21 – Приклад множини векторів за бітовим поданням

Застосовано метрику Хемінга для знаходження відстані між дескрипторами та центрами класів, так як ведеться робота з бінарними даними. Обчислено функцію належності для кожного дескриптору об’єкта до центрів усіх еталонів за формулою 2.15 з розмірністю бінарного дескриптору 256.

Для побудови класифікатора застосовано пошук максимуму належності серед еталонних значень розподілу для множини дескрипторів (мода розподілу) за формулою 2.19 з подальшим визначенням максимуму числа голосів за кожний з еталонів [1, 2, 6, 12].

На підставі результатів класифікації для експериментальної бази зображень, наведених у таблиці 3.1, здійснено оцінювання результативності за показниками Prec (критерій точності) і Compl (критерій повноти) за таблицями 3.2 й 3.3.

Отже, значення отриманих показників не є достатньо високими (вони не є близькими до 1), що пояснюється значною подібністю досліджуваних об’єктів. Однак, ті самі показники для істинно негативних результатів є значно більшими (табл. 3.3), що може свідчити на користь якості розпізнавання за множиною дескрипторів. У той же час уся множина еталонів (рис. 3.20 – 3.21) класифікується правильно і досить впевнено, так як різниця між максимумом голосів та його найближчим конкурентом перевищує 42%, а для окремих еталонів (наприклад, 1 та 2) досягає 60% [12].

Таблиця 3.1 – Результати класифікації (число дескрипторів)

	Еталон 1			Еталон 2			Еталон 3		
	Віднесених до еталону 1	Віднесених до еталону 2,3	∑	Віднесених до еталону 2	Віднесених до еталону 1,3	∑	Віднесених до еталону 3	Віднесених до еталону 1,2	∑
Кількість дескрипторів	281	219	500	249	251	500	274	226	500
Кількість дескрипторів решти еталонів	144 + 140 = 284	356 + 360 = 716	1000	111 + 86 = 197	389 + 414 = 803	1000	107 + 108 = 215	393 + 392 = 785	1000
∑	565	935	1500	446	1054	1500	489	1011	1500

Таблиця 3.2 – Значення показників результативності істинно позитивних результатів

	Еталон 1	Еталон 2	Еталон 3	Середнє значення	Мінімальне значення
Prec	0,497	0,55	0,56	0,536	0,497
Compl	0,562	0,498	0,548	0,536	0,498

Таблиця 3.3 – Значення показників результативності для істинно негативних результатів

	Еталон 1	Еталон 2	Еталон 3	Середнє значення	Мінімальне значення
Prec	0,766	0,762	0,776	0,768	0,762
Compl	0,716	0,803	0,785	0,768	0,716

Оцінимо дію класифікатора в умовах дії адитивних завад. Проведемо дослідження з дією шуму Гаусу (2.27) із заданим математичним очікуванням, що дорівнює нулю, а також СКВ = 10, результат продемонстровано на рисунку 3.22.

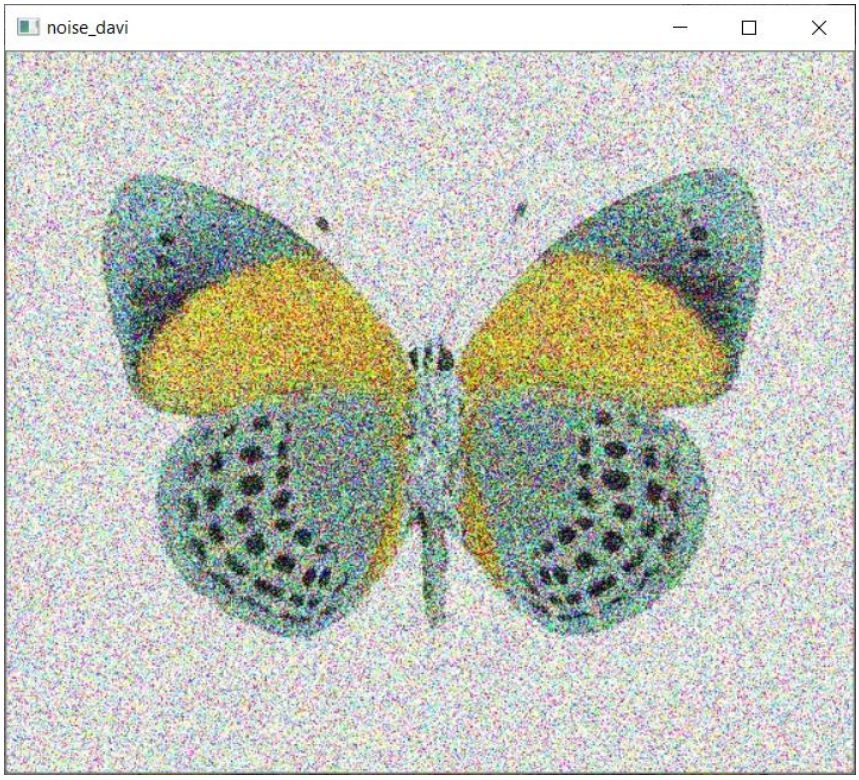

Рисунок 3.22 – Зображення Давісій із шумом з СКВ = 10

Для першого еталону точність зменшилась з 0,49 до 0,46, а повнота зменшилась з 0,56 до 0,49.

Для другого еталону: точність зменшилась з 0,55 до 0,48, а повнота зменшилась з 0,49 до 0,41.

Для третього еталону: точність зменшилась з 0,56 до 0,53, а повнота збільшилась з 0,54 до 0,58.

Отримавши розрахунки за таблицями 3.4 й 3.5, можна зробити висновки, що при значенні СКВ, що дорівнює 10, точність розпізнавання погіршується, повнота переважно зменшується, проте усі еталони класифіковано вірно, що підтверджує завадостійкість розробленого методу. Для поліпшення якості дослідження було проведено 30 експериментів.

Таблиця 3.4 – Результати класифікації (число дескрипторів) при СКВ = 10

	Еталон 1			Еталон 2			Еталон 3		
	Віднесених до еталону 1	Віднесених до еталону 2,3	∑	Віднесених до еталону 2	Віднесених до еталону 1,3	∑	Віднесених до еталону 3	Віднесених до еталону 1,2	∑
Кількість дескрипторів	245	255	500	207	293	500	291	209	500
Кількість дескрипторів решти еталонів	284	716	1000	223	777	1000	250	750	1000
∑	529	971	1500	430	1070	1500	541	959	1500

Таблиця 3.5 – Значення показників результативності істинно позитивних результатів при СКВ = 10

	Еталон 1	Еталон 2	Еталон 3	Середнє значення показника	Мінімальне значення показника
Prec	0,46	0,48	0,53	0,49	0,46
Compl	0,49	0,414	0,582	0,49	0,414

Посилимо дію завад. Для цього підвищимо значення параметру СКВ до 20 й 30, щоб оцінити вплив шуму Гаусу значного рівня на результати класифікації зображень, рисунки 3.23 й 3.24 демонструють результат.

Рисунок 3.23 – Зображення Давісій із шумом з СКВ = 20

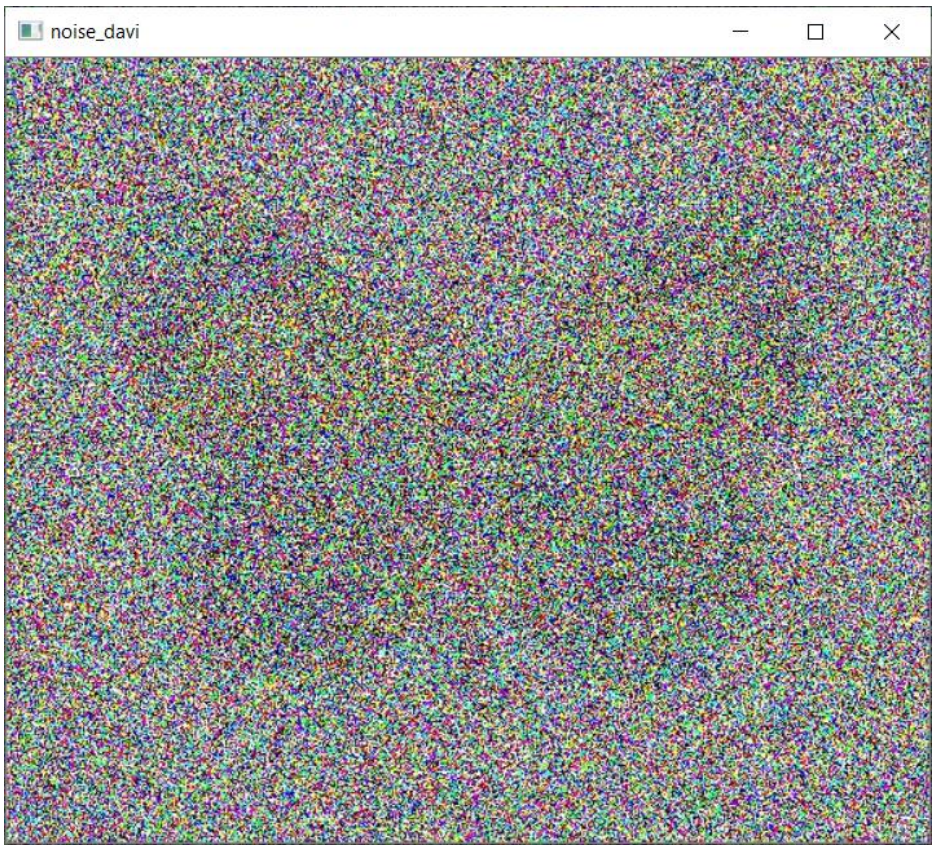

Рисунок 3.24 – Зображення Давісій із заданим шумом з СКВ = 30

Для першого еталону: точність зменшилась з 0,49 до 0,42, а повнота зменшилась з 0,56 до 0,48.

Для другого еталону: точність зменшилась з 0,55 до 0,41, а повнота зменшилась з 0,49 до 0,42.

Для третього еталону: точність зменшилась з 0,56 до 0,45, а повнота зменшилась з 0,54 до 0,38.

Наведені розрахунки з таблиць 3.6 та 3.7 підтверджують, що підвищення значення СКВ з 10 до 30 переважно погіршує результати розпізнавання зображень в плані зниження показників точності і повноти для множини дескрипторів, хоча усі еталони все ще класифіковано вірно. Цей факт підтверджує високу завадостійкість розробленого методу класифікації. Навіть при значному рівні шуму СКВ=30 класифікатор працює вірно, хоча візуально на рисунку 3.24 об’єкт майже не виділяється.

Таблиця 3.6 – Результати класифікації (число дескрипторів) при СКВ = 30

	Еталон 1			Еталон 2			Еталон 3		
	Віднесених до еталону 1	Віднесених до еталону 2,3	∑	Віднесених до еталону 2	Віднесених до еталону 1,3	∑	Віднесених до еталону 3	Віднесених до еталону 1,2	∑
Кількість дескрипторів	243	257	500	214	286	500	192	308	500
Кількість дескрипторів решти еталонів	325	675	1000	296	704	1000	230	770	1000
∑	568	932	1500	510	990	1500	422	1078	1500

Таблиця 3.7 – Значення показників результативності істинно позитивних результатів при СКВ = 30

	Еталон 1	Еталон 2	Еталон 3	Середнє значення	Мінімальне значення
Prec	0,42	0,41	0,45	0,42	0,41
Compl	0,48	0,42	0,38	0,42	0,38

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи запропоновано та програмно змодельовано ефективний метод побудови класифікатору зображень за статистичними розподілами для компонентів структурного опису із застосуванням детектору ORB.

Розроблений метод класифікації за результатом моделювання підтверджує свою працездатність та ефективність для класифікації зображень. Результативність методу може бути посилена введенням різноманіття видів метрик та мір подібності між центрами та дескрипторами, вибором способу формування центрів для еталонних описів, введенням логічного оброблення та стиснення структурного опису. Найкращі результати класифікації показала модель з використанням найбільш вагомого класу за вектором розподілів для кожного дескриптора, що відповідає параметру моди. Використання концентрованої частки даних опису дає можливість покращити його розрізнення з іншими описами.

Досліджено завадостійкість розробленого класифікатора шляхом моделювання впливу адитивного шуму Гауса. Проведено ряд експериментів з різним рівнем завад, доведено високу стійкість класифікатора до впливу шуму.

Класифікатор реалізовано у варіантах зіставлення інтегрального подання розподілів за класами і на підставі аналізу моди для розподілів окремих компонент.

Практична значущість роботи – побудова моделей класифікації у видозміненому просторі даних, підтвердження працездатності запропонованих модифікацій аналізу даних на прикладах зображень, розроблення програмних моделей для впровадження запропонованих методів класифікації у системах комп’ютерного зору.

У результаті роботи здійснена програмна реалізація методу для класифікації у базі еталонних зображень, а також роботу було апробовано у вигляді тез доповіді та статті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гороховатський, В. О., & Гадецька, С. В. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень.

2. 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.

3. Kohonen, T. (2001). Self-organizing Maps.-Springer Series in Information Sciences, V. 30, Springer.

4. Гороховатський, В. О., Пупченко, Д. В., & Солодченко, К. Г. (2018). Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення.

5. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). Ieee.

6. Гороховатський, В. О., Гадецька, С. В., Жадан, О. В., & Хвостенко, О. О. (2021). Дослідження результативності класифікаторів зображень за статистичними розподілами для компонентів структурного опису.

7. Флах, П. (2019). *Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных*. Litres.

8. Kim, S., & Kweon, I. S. (2006, January). Biologically motivated perceptual feature: Generalized robust invariant feature. In *Asian Conference on Computer Vision* (pp. 305-314). Springer, Berlin, Heidelberg.

9. Szeliski R. Computer Vision: Algorithms and Applications, London, Springer, 2010, 979 p.

10. Duda, R. O. (2000). Pattern classification/Duda RO, Hart PE, Stork DG.

11. Data base of butterflies and moths. – [Електронний ресурс]. – Режим доступу: https://www.nhm.ac.uk/our-science/data/butmoth/

12. Хвостенко, О. В. (2021). Дослідження класифікаторів зображень із використанням методу ORB.

13. Gorokhovatskyi, V.O, Gadetska, S.V., Styahlyk, N.I. & Vlasenko, N.V. (2020), “Classification of images based on an ensemble of statistical distributions by standard classes for structural description components”, Radio Electronics, Computer Science, Control, No. 4, pp. 85–94.

14. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). Ieee.

15. Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011, November). BRISK: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision* (pp. 2548-2555). Ieee.

16. Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.

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

18. Gorokhovatskiy, A. V., Gorokhovatskiy, V. A., Vlasenko, A. N., & Vlasenko, N. V. (2014). Quality Criteria for Multidimensional Object Recognition Based Upon Distance Matrices. *Telecommunications and Radio Engineering*, *73*(18).

19. Путятин, Е. П. (1990). *Обработка изображений в робототехнике*. Москва.

20. Zadeh, L. A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and systems*, *90*(2), 111-127.

21. Rosten, E., Porter, R., & Drummond, T. (2008). Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, *32*(1), 105-119.

22. Гороховатский, В. А., Путятин, Е. П., & Столяров, В. С. (2017). Исследование результативности структурных методов классификации изображений с применением кластерной модели данных. *Радіоелектроніка, інформатика, управління*, (3), 78-85.

23. Gorokhovatskyi, V. A., Gorokhovatskiy, A. V., & Peredrii, Y. O. (2018). Hashing of structural descriptions at building of the class image descriptor, computing of relevance and classification of the visual objects. *Telecommunications and Radio Engineering*, *77*(13).

24. Gorokhovatskyi, O., Gorokhovatskyi, V., & Peredrii, O. (2018). Analysis of Application of Cluster Descriptions in Space of Characteristic Image Features. *Data*, *3*(4), 52.

25. Gorokhovatskiy, V. A., Gorokhovatskiy, A. V., & Berestovsky, A. Y. (2016). Intellectual Data Processing and Self-Organization of Structural Features at Recognition of Visual Objects. *Telecommunications and Radio Engineering*, *75*(2).

26. Gorokhovatskyi, V., Gadetska, S., & Ponomarenko, R. (2019). Логічний аналіз та оброблення даних задля класифікації зображень на підставі формування статистичного центру опису. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *4*(56), 43-48.

27. Гороховатский, В. А. (2014). Структурный анализ и интеллектуальная обработка данных в компьютерном зрении.

28. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. *Telecommunications and Radio Engineering*, *79*(1).

29. Гороховатский, В. А. (2017). Методы определения релевантности изображений на основе медианной обработки структурных описаний. *Радіоелектроніка, інформатика, управління*, (1), 100-106.

30. Гороховатський, В. О., & Пономаренко, Р. П. (2020). Класифікація зображень на підставі формування незалежної системи кластерів у складі структурних описів бази еталонів.

31. Gorokhovatskyi, V., Gorokhovatskyi, O., Yevgenyi, P., & Olena, P. (2018, August). Quantization of the space of structural image features as a way to increase recognition performance. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 464-467). IEEE.

32. Кульбак, С. (1967). *Теория информации и статистика*. Наука. Гл. ред. физ.-мат. лит..

33. Hero, A. O., Ma, B., Michel, O., & Gorman, J. (2001). Alpha-divergence for classification, indexing and retrieval. In *University of Michigan*.

34. OpenCV foundation. (2017). OpenCV: Image Thresholding. Retrieved from https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html

35. Tvoroshenko, I. S., & Gorokhovatskyi, V. O. (2019). Intelligent classification of biophysical system states using fuzzy interval logic. *Telecommunications and Radio Engineering*, *78*(14).

36. Porter, F. C. (2008). Testing consistency of two histograms. *arXiv preprint arXiv:0804.0380*.

37. Wang, J., Markert, K., & Everingham, M. (2009, September). Learning Models for Object Recognition from Natural Language Descriptions. In *BMVC* (Vol. 1, No. 2009, p. 2).

38. Фукунага, К. (1979). *Введение в статистическую теорию распознавания образов: Пер. с англ*. Наука.

39. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic. *IEEE Access*, *9*, 13417-13428.

40. Python’s documentation, tutorials, and guides are constantly evolving. – [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/

41. Саммерфилд, М. (2009). Программирование на Python 3. Подробное руководство. *СПб.: Символ-Плюс*.