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)

РОЗРОБЛЕННЯ МЕТОДУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ НА ПІДҐРУНТІ МАТРИЦІ ВІДСТАНЕЙ У ПРОСТОРІ ХЕШ-КОДІВ ДЛЯ ДАНИХ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Молодняк Д. А.
(прізвище, ініціали)

Спеціальність 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

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

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Молодняку Данилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення методу класифікації зображень на підґрунті матриці відстаней у просторі хеш-кодів для даних

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 20 червня 2026 р.
3. Вихідні дані до роботи: набори зображень для тестування методу класифікації, бібліотека комп'ютерного зору з відкритим кодом OpenCV, алгоритм виявлення та опису ключових точок AKAZE, метод локально чутливого хешування, медод випадкового хешування, метрики для порівняння дескрипторів, науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз методів класифікації зображень на основі множини дескрипторів та обґрунтування вибору підходу.
2. Алгоритм AKAZE для виявлення та опису ключових точок зображення.
3. Методу побудов матриці відстаней у просторі хеш-кодів.
4. Метод класифікації зображень із застосуванням матриці відстаней.
5. Експериментальне дослідження розробленого методу та аналіз результатів класифікації.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) постановка задачі, структурна схема методу класифікації зображень, діаграма значень матриці відстаней у просторі хеш-кодів, тестові зображення, результати класифікації, графіки точності розробленого методу.

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	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.05.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.05.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-01.05.26	
6	Програмна реалізація	24.04.26-18.05.26	
7	Аналіз отриманих результатів	19.05.26-28.05.26	
8	Оформлення пояснювальної записки	29.05.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 78 с., 5 табл., 9 рис., 1 дод., 36 джерел.

АЛГОРИТМ AKAZE, ДЕСКРИПТОР, КЛАСИФІКАЦІЯ, КЛЮЧОВА ТОЧКА ЗОБРАЖЕННЯ, КОМП'ЮТЕРНИЙ ЗІР, МАТРИЦЯ ВІДСТАНЕЙ, ХЕШУВАННЯ.

Об'єктом роботи є методи класифікації зображень за описом у формі множини дескрипторів ключових точок.

Метою роботи є розроблення методу класифікації зображень з використанням матриці відстаней у хешованому просторі даних.

Використано методи комп'ютерного зору, детектор ознак AKAZE, засоби хешування, методи лінійної алгебри та статистичне подання у формі гістограм.

Практична цінність полягає в автоматизації класифікації зображень та скороченні обчислювальних витрат завдяки векторному поданню опису.

Розроблено метод виділення ознак зображень, їх стиснення у 16-бітні хеш-коди та швидкої класифікації за значенням гістограми матриці відстаней опису.

Область застосування: системи комп'ютерного зору, візуальні пошукові рушії, інфраструктура візуального контролю якості, розумні периферійні пристрої.

AKAZE ALGORITHM, CLASSIFICATION, COMPUTER VISION, DESCRIPTOR, DISTANCE MATRIX, HASHING, IMAGE KEYPOINT.

The objective of this work is methods for classifying images based on a description consisting of a set of keypoint descriptors.

The goal of this work is to develop a method for classifying images using a distance matrix in a hashed data space.

The study utilized computer vision techniques, the AKAZE feature detector, hashing algorithms, linear algebra methods, and statistical representation in the form of histograms.

The practical value lies in the automation of image classification and a significant reduction in computational costs through vector representation of the description.

A method has been developed for extracting image features, compressing them into 16-bit hash codes, and rapid classification based on histogram values of the description distance matrix.

Applications: computer vision systems, visual search engines, visual quality control infrastructure, smart peripheral devices.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

мс –мілісекунди

КТ – ключова точка

AKAZE – Accelerated-KAZE (прискорений алгоритм KAZE)

BLAS – Basic Linear Algebra Subprograms (базові підпрограми лінійної алгебри)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

LSH – Locality-Sensitive Hashing (локально-чутливе хешування)

OpenCV – Open Source Computer Vision Library (бібліотека комп'ютерного зору з відкритим вихідним кодом)

SIFT – Scale-Invariant Feature Transform (масштабно-інваріантне перетворення ознак)

SIMD – Single Instruction Multiple Data (одна інструкція, множина даних)

SURF – Speeded Up Robust Features (прискорені стійкі ознаки)

XOR – Exclusive OR (логічна операція виключного або)

ВСТУП

Стрімкий розвиток інформаційних технологій та систем штучного інтелекту зумовлює необхідність швидкої обробки величезних масивів візуальних даних. Аналіз сучасних літературних джерел свідчить, що класичні структурні методи комп'ютерного зору забезпечують високу точність розпізнавання [1 – 3], проте вимагають значних обчислювальних ресурсів через квадратичну складність порівняння багатовимірних дескрипторів. З іншого боку, сучасні тенденції використання глибоких згорткових нейромереж дозволяють досягти високих результатів, але їхнє застосування є критично обмеженим на автономних пристроях через значне енергоспоживання та потребу в спеціалізованих графічних процесорах [4 – 6]. З огляду на це, обґрунтовується гостра необхідність розроблення нових методів класифікації, які б поєднували точність структурного аналізу з обчислювальною легкістю [7], притаманною оптимізованим алгоритмам, здатним працювати в умовах обмеженої оперативної пам'яті.

Обґрунтування основних проєктних рішень базується на необхідності подолання проблеми алгоритмічних затримок. Для виділення стабільних структурних ознак обрано алгоритм нелінійної дифузії AKAZE, який найкраще зберігає геометрію об'єктів. Головним напрямком роботи та оптимізації стало застосування математичного апарату локально-чутливого хешування (LSH) для агресивного стиснення 488-бітних дескрипторів у компактні 16-бітні хеш-коди. Для усунення «вузьких місць» традиційного зіставлення прийнято рішення про переведення розрахунків відстані геммінга у площину алгебраїчних векторизованих обчислень, а також здійснено перехід до порівняння глобальної топології об'єктів на основі інтегральних гістограм [8] за манхеттенською метрикою.

Завдання кваліфікаційної роботи – розроблення методу класифікації зображень на підґрунті матриці відстаней у просторі хеш-кодів для даних.

Під час роботи використано алгоритми комп’ютерного зору для детектування ознак, методи машинного навчання для зниження розмірності, апарати лінійної алгебри для векторизованих обчислень та непараметричну статистику для аналізу розподілів.

Галузі використання результатів кваліфікаційної роботи охоплюють широкий спектр сучасних інформаційних систем. Завдяки суттєвому скороченню обчислювальних витрат та стабілізації використання оперативної пам’яті, розроблений метод є ідеальним для інтеграції в системи комп’ютерного зору автономних роботів, інфраструктуру автоматизованого контролю якості, пошукові рушії електронної комерції та розумні периферійні пристрої (edge computing), де використання традиційних нейромереж є ресурсозатратним.

1 АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Аналіз предметної області та виявлення практичної проблеми

Стрімкий розвиток інформаційних технологій та систем штучного інтелекту зумовлює зростаючу потребу в ефективній автоматизованій обробці великих масивів візуальних даних. Щоденне генерування терабайтів графічної інформації в Інтернеті речей (IoT), системах відеоспостереження, безпілотних апаратах та промислових комплексах вимагає створення алгоритмів, здатних працювати в режимі реального часу. Перехід від концепції централізованих хмарних обчислень (cloud computing) до периферійних обчислень (edge computing) диктує нові правила: системи повинні аналізувати візуальний потік безпосередньо на місці зйомки, мінімізуючи затримки передачі даних та забезпечуючи конфіденційність. У цьому контексті задача класифікації зображень є фундаментальною для широкого кола сучасних прикладних систем: від автоматизованого контролю якості на виробничих лініях до візуальних пошукових рушіїв електронної комерції та інтелектуальних систем безпеки.

Класичні структурні методи комп’ютерного зору, як зазначено у роботах [1, 3, 5], базуються на виділенні локальних ознак зображення у вигляді ключових точок та дескрипторів. Такі підходи демонструють високу точність розпізнавання об’єктів, успішно справляючись із задачами масштабування, зміни ракурсу та часткового перекриття (оклюзії) об’єктів у кадрі. Структурний опис дозволяє системі «розуміти» об’єкт не як монолітну матрицю пікселів, а як набір взаємопов’язаних геометричних елементів. Однак їхнє масштабування на великі бази даних пов’язане з принциповою обчислювальною проблемою: операція крос-порівняння багатовимірних бінарних дескрипторів має квадратичну складність $O(N^2)$, що призводить до критичного зростання часових витрат при збільшенні розміру вибірки. У промислових масштабах такий підхід швидко вичерпує ліміти оперативної

пам’яті та створює так зване «вузьке місце» (bottleneck) на етапі пошуку найближчого сусіда, оскільки процесор змушений виконувати мільйони ітеративних порівнянь для кожної пари зображень.

З іншого боку, сучасні тенденції застосування глибоких згорткових нейронних мереж (CNN) дозволяють досягати видатних показників точності у задачах розпізнавання зображень [9, 10]. Нейромережі здатні автоматично формувати ієрархічні ознаки, абстрагуючись від локальних шумів, та вирішувати задачі комплексної семантичної сегментації. Проте застосування таких моделей критично обмежене на автономних пристроях та в умовах периферійних обчислень (edge computing) через низку факторів. По-перше, це значне енергоспоживання та вимога до наявності спеціалізованих графічних процесорів (GPU) або тензорних ядер (TPU) [11]. Для безпілотних апаратів, мобільних роботів або портативних смарт-камер, які живляться від акумуляторів, такі апаратні вимоги є неприйнятними. По-друге, нейромережі є надзвичайно залежними від обсягу та якості навчальної вибірки: додавання нового класу об’єктів часто вимагає повного або часткового перенавчання моделі (fine-tuning), що є негнучким рішенням для систем, які динамічно оновлюються. Таким чином, у сучасній науковій літературі зафіксовано чітку суперечність між точністю нейромережевих підходів та їхньою обчислювальною й енергетичною ефективністю, що визначає актуальність пошуку альтернативних структурних методів.

Предметна область даної роботи охоплює системи структурного аналізу та класифікації зображень об’єктів із складною геометрією – зокрема, таких об’єктів як холодна зброя (ножі), що характеризуються гострими кутами, різноманітними текстурами поверхонь та наявністю інтенсивних відблисків на металевих ділянках. Візуальний аналіз подібних предметів є вкрай нетривіальною задачею комп’ютерного зору. Фізичні властивості металу призводять до того, що гладке лезо часто взагалі не містить власних текстурних ознак і працює як дзеркало, відбиваючи навколишнє середовище, тоді як руків’я (особливо з насічками, паракордом або композитними

матеріалами) може бути перенасичене дрібними високочастотними деталями. Зміна кута освітлення навіть на кілька градусів може повністю змінити картину градієнтів на поверхні леза, генеруючи так звані псевдо-ознаки (штучні кути, утворені світлом). Саме така специфіка робить ці об’єкти ідеальним та надзвичайно складним тестовим середовищем для апробації нових алгоритмічних рішень у галузі комп’ютерного зору, вимагаючи від детектора максимальної стійкості до оптичних аномалій.

Практична проблема, що вирішується у роботі, полягає в тому, що існуючі алгоритми структурного зіставлення (зокрема, метод пошуку найближчого сусіда на основі побітової відстані геммінга між повнорозмірними дескрипторами) є непридатними для потокової обробки великих баз зображень через надмірні обчислювальні витрати, а також нездатні виконувати коректне семантичне ранжування візуально подібних, але фізично різних об’єктів одного класу. Класичний попіксельний матчинг занадто жорстко відкидає точки, що не збігаються ідеально, працюючи за логікою «все або нічого». Вирішення цієї проблеми потребує розроблення принципово нового методу, що поєднував би точність і глибину структурного аналізу з обчислювальною ефективністю та здатністю до семантичного узагальнення [6].

1.2 Методи виділення ознак та дескрипторів ключових точок

Центральним елементом структурних методів класифікації зображень є якість, повторюваність та стійкість виділених локальних ознак. У науковій літературі добре описані детектори та дескриптори ключових точок, які формують так звану класичну школу комп’ютерного зору. Серед них найбільш фундаментальними та відомими є SIFT (Scale-Invariant Feature Transform) та SURF (Speeded Up Robust Features) [12]. Ці алгоритми будують багатомасштабне подання зображення (scale space) на основі лінійних фільтрів

гаусса, що забезпечує інваріантність дескрипторів до масштабування та повороту. Однак лінійна (ізотропна) природа цих фільтрів має критичну ваду: розмиття застосовується рівномірно до всіх пікселів зображення. Це неминуче призводить до деградації та згладжування високочастотних деталей зображення – саме тих контурів, кутів та різких переходів контрасту, що є визначальними для ідентифікації об’єктів зі складним рельєфом поверхні (наприклад, контурів леза).

Спробою подолати обчислювальну надмірність SIFT та SURF стали алгоритми ORB (Oriented FAST and Rotated BRIEF) та BRIEF (Binary Robust Independent Elementary Features). ORB поєднує детектор кутових точок FAST із бінарним дескриптором BRIEF, забезпечуючи суттєво вищу швидкість обробки. Проте інваріантність ORB до масштабування є обмеженою, а якість виділених ознак на зображеннях із різким перепадом яскравості або складною текстурою помітно поступається більш досконалим методам. Зокрема, через використання лінійного розмиття на етапі побудови піраміди масштабів ORB, як і SIFT, схильний до розмивання значущих контурів об’єктів.

Сучасним та високоефективним рішенням цієї проблеми є алгоритм AKAZE (Accelerated-KAZE), детально описаний у фундаментальній роботі [13]. Головна перевага AKAZE над його попередниками полягає у використанні нелінійного простору масштабів, що будується на основі математичного апарату рівнянь нелінійної експліцитної дифузії (fast explicit diffusion). Диференціальна природа нелінійної дифузії дозволяє системі локально адаптувати ступінь згладжування зображення залежно від наявності різких градієнтів (контурів). Функція провідності в AKAZE діє як інтелектуальний бар’єр: вона ефективно пригнічує візуальний шум на однорідних ділянках (наприклад, фоні або гладких частинах об’єкта), але блокує розмиття на межах об’єктів, зберігаючи їх абсолютно чіткими навіть на вищих рівнях піраміди масштабів. Порівняльний аналіз дескрипторів KAZE, AKAZE, SURF та ORB для задач розпізнавання, проведений у дослідженні [14], переконливо підтверджує, що AKAZE забезпечує

найкращий баланс між точністю виділення ознак, стійкістю до зміни освітлення та обчислювальною швидкодією, особливо при аналізі геометрично складних контурів.

Кожна ключова точка, знайдена алгоритмом AKAZE, описується спеціалізованим бінарним дескриптором M-LDB (Modified Local Difference Binary) розміром 61 байт, що еквівалентно 488 бітам неперервної інформації. Формування такого опису відбувається шляхом парного порівняння середніх інтенсивностей пікселів у чітко визначеному та орієнтованому за головним градієнтом локальному околі навколо точки. Бінарний формат дескриптора є особливо привабливим з точки зору апаратної ефективності: на відміну від SIFT чи SURF, які оперують масивами чисел із рухомою комою (float), порівняння бінарних дескрипторів виконується через елементарну логічну операцію XOR (виключне АБО). Ця операція є однією з найшвидших інструкцій АЛП (арифметико-логічного пристрою) сучасних центральних процесорів, здатних виконуватися за один машинний такт [8, 15].

Проте, як показано у роботах [16, 17], попри феноменальну ефективність бінарного формату на мікрорівні, масове крос-порівняння 488-бітних дескрипторів на макрорівні (наприклад, для формування матриці зіставлення 500×500 точок для кожної пари зображень) залишається обчислювально надмірною задачею. Таке навантаження провокує відоме в інформатиці «прокляття розмірності». Робота з такими довгими масивами створює величезну кількість промахів процесорного кешу (cache misses), оскільки дані не поміщаються у швидку пам’ять (*L1/L2* кеш) і процесор змушений постійно очікувати їх підвантаження з повільної оперативної пам’яті (RAM). Ця обставина унеможливлює потокову обробку відео або баз даних на стандартному обладнанні без застосування спеціалізованих апаратних прискорювачів і гостро обґрунтовує необхідність розроблення методів ефективного агресивного стиснення простору ознак перед етапом їхнього метричного порівняння.

1.3 Аналіз методів зниження розмірності та хешування даних

Одним із найбільш перспективних і математично обґрунтованих підходів до скорочення розмірності простору бінарних дескрипторів є метод локально чутливого хешування (Locality-Sensitive Hashing, LSH). Важливо розуміти, що LSH докорінно відрізняється від традиційних криптографічних хеш функцій (таких як MD5 або SHA-256). Якщо завдання криптографічного хешування зводиться до лавинної зміни результату навіть при зміні одного біта на вході (для забезпечення безпеки та унікальності), то LSH розроблено з абсолютно протилежною метою. Його головне завдання – гарантувати, що просторово подібні (близькі) об’єкти у вихідному багатовимірному просторі отримають однакові або дуже близькі хеш-коди після трансформації. У роботі [18] детально описано застосування LSH на основі методу випадкових проєкцій (random projection) для задач масового інформаційного пошуку візуальних даних.

Теоретичним фундаментом цього підходу слугує лема Джонсона-Лінденштрауса (Johnson-Lindenstrauss lemma), яка математично гарантує: множину точок у багатовимірному евклідовому просторі можна ізометрично відобразити у простір значно меншої розмірності таким чином, що відносні попарні відстані (кути) між точками зберігатимуться із наперед заданою і контрольованою малою похибкою [19, 20]. Ця лема доводить, що інформаційна надмірність високорозмірних даних може бути безпечно зрізана без руйнування внутрішньої топології кластерів.

В контексті роботи з бінарними дескрипторами метод випадкових проєкцій реалізується шляхом множення нормалізованого і центрованого вектора ознак на попередньо згенеровану матрицю псевдовипадкових значень (що підпорядковуються нормальному розподілу) з наступною пороговою бінаризацією результату. Геометрично цей процес означає проведення випадкових ортогональних гіперплощин через багатовимірний простір ознак. Ці площини розсікають простір і розділяють дескриптори на позитивні (1) та

негативні (0) півпростори залежно від того, по який бік від площини опинився вектор. Таким чином, громіздкий 488-бітний дескриптор миттєво стискається до компактного 16-бітного хеш-коду, що скорочує фізичний обсяг даних майже у 96,7%. При цьому відносне розташування семантичних кластерів у просторі надійно зберігається, що є абсолютно необхідною умовою для коректної класифікації візуальної інформації та запобігання втраті критичних геометричних ознак [21, 22].

Застосування LSH для задач пошуку зображень також глибоко досліджено у роботі [21], де автори поєднують принципи локально чутливого хешування з механізмами уваги (attention) глибоких нейронних мереж. Порівняльний аналіз результатів у цій та подібних працях підтверджує, що компактне хешове подання суттєво скорочує час пошуку (retrieval time) у великих базах даних без критичної втрати якості ранжування за подібністю [23]. Відмова від багатовимірного перебору на користь компактних 16-бітних структур, які ідеально поміщаються в апаратні регістри процесора, відкриває шлях до побудови надшвидких пошукових індексів та систем класифікації реального часу.

1.4 Метрики для оцінювання подбності описів

Після виділення інваріантних ознак та екстремального стиснення їхньої розмірності ключовим архітектурним питанням є вибір відповідної математичної метрики для оцінювання ступеня подібності між об’єктами. На мікрорівні (при порівнянні окремих точок) для бінарних векторів галузевим стандартом є відстань геммінга (Hamming distance), яка показує абсолютну кількість позицій з відмінними бітами [24]. Однак, як переконливо показано у роботах [9, 11], традиційний ітеративний підхід до обчислення матриць відстаней геммінга через класичну побітову логічну операцію XOR у програмних циклах є вкрай малоефективним для задач масового порівняння.

Він провокує високі накладні витрати середовища виконання (особливо в інтерпретованих мовах на кшталт Python) і практично не допускає повноцінної апаратної векторизації. Альтернативним і значно прогресивнішим підходом є алгебраїчна модель обчислення через скалярний добуток матриць. Розкладання логічного XOR у суму векторів мінус їх подвоєний скалярний добуток дозволяє делегувати обчислення оптимізованим бібліотекам лінійної алгебри (BLAS) із застосуванням SIMD – інструкцій багатоядерного процесора.

На макрорівні класифікації зображень (оцінка подібності цілих об’єктів) сучасною тенденцією є відмова від локального зіставлення точок на користь використання інтегральних гістограмних моделей подання. У роботах [1, 3, 5] детально досліджено використання матриць відстаней і статистичних дескрипторів у вигляді одновимірних гістограм для структурних методів класифікації. Зазначені роботи демонструють, що перехід від прямого, жорсткого порівняння окремих локальних ознак до аналізу глобальних статистичних розподілів (частот зустрічальності різних відстаней всередині об’єкта) дозволяє системі «бачити» предмет цілком. Цей підхід дозволяє системі успішно абстрагуватися від випадкових оптичних шумів, мікротекстурних відмінностей або часткових перекриттів, зосереджуючись на загальній семантичній та геометричній подібності об’єктів. Гістограма виступає унікальним інтегральним «паспортом» структурної організації кадру.

У роботах [25 – 27] детально обґрунтовано перевагу лінійної манхеттенської відстані (*L1*-норми) над традиційною квадратичною Евклідовою відстанню (*L2*-нормою) при порівнянні гістограмних розподілів у задачах комп’ютерного зору. Проблема *L2*-норми полягає в тому, що вона, зводячи різницю координат у квадрат, математично гіперболізує (непропорційно збільшує) вплив статистичних викидів та локальних аномалій. Наприклад, один яскравий відблиск на лезі ножа може створити сплеск в одному відсіку гістограми, і Евклідова метрика зведе цей сплеск у квадрат,

штучно класифікуючи об’єкт як абсолютно інший предмет. Такий підхід є абсолютно неприйнятним при аналізі металевих об’єктів зі складною формою. Натомість, манхеттенська метрика штрафує відхилення виключно лінійно. Це забезпечує системі фундаментальну властивість високої стійкості (робастності) до поодиноких оптичних артефактів, дозволяючи коректно і плавно ранжувати зображення за їхньою реальною семантичною близькістю.

У дослідженні [9] запропоновано набір інструментів для швидкого метричного пошуку у структурних методах класифікації, що емпірично підтверджує практичну ефективність поєднання бінарних дескрипторів з алгебраїчними метриками відстані геммінга та гістограмного порівняння. Авторами доведено, що правильно спроєктований алгоритмічний конвеєр здатен суттєво прискорити пошук у великих колекціях зображень, зберігаючи високу точність без необхідності застосування енергоємних нейромережевих архітектур.

Традиційні алгоритми класифікації, як правило, вирішують задачу так званої закритої множини (closed set recognition): система безальтернативно відносить будь-який вхідний об’єкт до одного з відомих класів еталонної бази даних, незалежно від того, чи є такий об’єкт взагалі представленим у базі еталонів. Якщо системі, яка «знає» лише різні моделі ножів, показати фотографію побутового інструменту, вона гарантовано класифікує його як один із ножів. У реальних умовах промислової експлуатації або в автоматизованих системах безпеки (де ціна помилки є критичною) така поведінка алгоритму неминуче призводить до критичних помилок першого роду (false positives), коли сторонній, нерелевантний предмет хибно ідентифікується як цільовий об’єкт.

Задача розпізнавання у відкритій множині (open set recognition) є значно складнішою і вирішується шляхом введення інтелектуального порогового механізму відсікання, заснованого на аналізі обчислених метричних відстаней. Теоретичні аспекти прийняття статистичних рішень та методи мінімізації помилок другого роду в системах розпізнавання детально розглянуто у

фундаментальній роботі [7]. Обґрунтування порогових констант (thresholds) не може бути інтуїтивним; воно вимагає емпіричного аналізу мінімальних міжкласових відстаней. Це є ключовим елементом надійної роботи в умовах відкритої множини і вимагає проведення масштабного крос-порівняльного експерименту на базі наявних еталонних даних для визначення «коридору толерантності» алгоритму.

Математичний зв’язок між розподілом відстаней геммінга для незалежних випадкових бінарних векторів і законом великих чисел, що детально описано у роботі [9], є потужною теоретичною основою для оцінювання статистичної ізоляції розподілів відстаней для «своїх» (цільових) та «чужих» (аномальних) об’єктів. Зокрема, відоме в криптографії та теорії інформації положення про те, що математичне сподівання відстані геммінга між випадковими, некорельованими бінарними векторами довжиною N дорівнює $N/2$, дозволяє аналітично оцінити очікуваний «коридор» відстаней для різних класів об’єктів. Цей закон гарантує, що випадковий шум (невідомий об’єкт) згенерує гістограму, пік якої буде зосереджений суворо посередині спектру відстаней. Саме це математичне сподівання дозволяє інженерам обґрунтовано встановити порогові значення для системи класифікації, гарантуючи, що випадкові оптичні завади або сторонні об’єкти фізично не зможуть перетнути межу допуску і будуть коректно відхилені системою як невідомі.

1.5 Аналіз програмних засобів

Стандартним, загальновизнаним інструментом для роботи з дескрипторами ключових точок у сучасних задачах комп’ютерного зору є бібліотека OpenCV (Open Source Computer Vision Library). Для обчислення матриць відстаней геммінга та пошуку збігів ця бібліотека надає вбудований клас BFMatcher (Brute-Force Matcher) та функцію пакетного порівняння

cv2.batchDistance. Однак, як зазначено у дослідженні [15] та підтверджено практикою, ці стандартні C++ засоби часто демонструють вкрай нестабільну поведінку при масовому крос-порівнянні великих бінарних масивів у середовищі Python. Це пов’язано з особливостями управління пам’яттю, необхідністю постійного копіювання даних між віртуальною машиною Python та C++ ядром OpenCV, а також фрагментацією пам’яті на низькому рівні. Крім того, стандартний BFMatcher не підтримує повноцінну агресивну апаратну векторизацію через спеціалізовані BLAS – бібліотеки, що створює штучне програмне обмеження його продуктивності, не дозволяючи розкрити потенціал сучасних багатоядерних процесорів.

Наукові роботи [19, 20] аналізують прогресивні технології структурної класифікації на основі статистичного аналізу множини дескрипторів та пропонують моделі прискорення класифікації на основі оцінювання відстані дескриптора безпосередньо до класу (а не до кожної окремої точки). Запропоновані у них підходи емпірично підтверджують перспективність відмови від жорсткого побітового порівняння (яке страждає від шумів) на користь статистичного аналізу розподілів. Однак зазначені роботи фокусуються на окремих математичних аспектах і не розглядають побудови повноцінного, замкнутого обчислювального конвеєра: від попереднього агресивного стиснення дескрипторів через випадкові проєкції LSH до інтегральної гістограмної класифікації із захисним механізмом розпізнавання у відкритій множині (open set).

У монографіях [2, 3] систематизовано теоретичні основи та практичні методи інтелектуального аналізу та оброблення даних у задачах класифікації багатовимірних зображень. Загальний та критичний аналіз представлених у світовій літературі підходів свідчить про фактичну відсутність готових, «коробкових» програмних рішень, що гармонійно та оптимізовано поєднують детектор нелінійної дифузії AKAZE, LSH-стиснення дескрипторів, алгебраїчну векторизовану модель обчислення матриць відстаней геммінга та інтегральну гістограмну класифікацію з підтримкою Open Set Recognition в

єдиному монолітному програмному конвеєрі. Саме ця прогалина в існуючому програмному забезпеченні комп'ютерного зору обґрунтовує високу практичну необхідність та наукову актуальність розробки, представленої у даній кваліфікаційній роботі.

У роботі [28] докладно розглянуто потужні можливості бібліотеки NumPy та сучасних засобів векторизованих обчислень мови Python для ефективної реалізації математичних алгоритмів обробки великих даних (data science). Застосування векторизованих функцій NumPy дозволяє повністю делегувати ресурсомісткі обчислення (такі як скалярні добутки багатовимірних матриць) оптимізованим прекомпільованим C-бібліотекам з апаратним прискоренням (SIMD). Це дозволяє розробникам повністю уникати повільних ітеративних циклів інтерпретатора Python (які обмежуються Global Interpreter Lock - GIL), що є принципово важливим інженерним рішенням для досягнення необхідної високої швидкодії при потоковій обробці великих масивів візуальної інформації.

1.6 Постановка задачі

Автоматична класифікація зображень об'єктів зі складною геометрією (зокрема, з відблисками та високочастотними текстурами) є надзвичайно актуальним завданням у контексті розвитку систем комп'ютерного зору для автономних, мобільних і ресурсообмежених пристроїв (edge computing). Проведений критичний аналіз наукової літератури та існуючих програмних рішень переконливо показав, що традиційні структурні методи, засновані на крос-порівнянні повнорозмірних бінарних дескрипторів, мають квадратичну обчислювальну складність $0(N^2)$, перевантажують оперативну пам'ять і не здатні виконувати коректне семантичне ранжування візуально подібних об'єктів. З іншого боку, передові нейромережеві підходи є абсолютно непридатними для Edge-середовищ через свою екстремальну ресурсоємність

та потребу в GPU. У зв’язку з цим ставиться завдання розробити оптимізований структурний метод класифікації зображень, що поєднує точність нелінійної дифузії з математичною оптимізацією через алгебраїчну векторизацію та компактне хешове подання ознак.

Об’єктом роботи є методи класифікації зображень за описом у формі множини дескрипторів ключових точок.

Метою роботи є розроблення методу класифікації зображень з використанням матриці відстаней у хешованому просторі даних.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати сучасні підходи до класифікації візуальних даних та виявити обмеження існуючих структурних методів;

− розробити модель стиснення структурних дескрипторів на основі локально чутливого хешування (LSH) з використанням методу випадкових проєкцій;

− розробити алгоритм оцінювання подібності зображень на базі гістограм розподілу відстаней геммінга із застосуванням манхеттенської метрики;

− здійснити програмну реалізацію запропонованого методу та провести порівняльне тестування з традиційними підходами;

− емпірично обґрунтувати порогові константи для реалізації механізму розпізнавання у відкритій множині (open set recognition).

2 МОДЕЛІ ТРАНСФОРМАЦІЇ ПРОСТОРУ ОЗНАК І ПРИЙНЯТТЯ КЛАСИФІКАЦІЙНИХ РІШЕНЬ

2.1 Аналіз методів трансформації простору ознак

Процес розпізнавання та класифікації візуальної інформації фундаментально залежить від якості та стійкості вилучених із зображення ключових ознак. Класичні методи комп'ютерного зору, такі як SIFT (Scale-Invariant Feature Transform) або SURF, для побудови багатомасштабного подання (scale space) використовують лінійні фільтри гаусса [24]. Однак лінійне розмиття призводить до неминучої втрати високочастотної інформації: разом із візуальним шумом сенсора незворотно розмиваються і важливі контури об'єктів (краї, кути, переходи контрасту). У контексті предметної області даної роботи це є критичним недоліком, оскільки при класифікації таких об'єктів, як ножі, саме гострі грані, форма вістря та дрібні деталі рельєфу руків'я є визначальними геометричними ідентифікаторами. Для усунення цього недоліку в розробленій системі математичним базисом для виявлення ознак обрано сучасний алгоритм AKAZE (Accelerated-KAZE) [8].

Фундаментальна відмінність алгоритму AKAZE полягає у використанні простору нелінійних масштабів. Математична модель алгоритму базується на рівняннях нелінійної дифузії в частинних похідних, що дозволяє локально адаптувати ступінь розмиття зображення залежно від наявності чітких контурів. Динаміка цього процесу дифузії описується класичним диференціальним рівнянням:

$$\frac{\partial L}{\partial t} = div(c(x, y, t) \times \nabla L), \tag{2.1}$$

де $L(x, y, t)$ – яскравість зображення у просторових координатах x, y на параметричному масштабі t;

∇L – векторний градієнт яскравості;

div – оператор дивергенції;

c(x, y, t) – диференційна функція провідності.

Ця функція виступає своєрідним математичним бар'єром: вона алгоритмічно блокує поширення дифузії на різких переходах яскравості (межах об'єкта), тим самим ефективно пригнічуючи шум на однорідних ділянках, але зберігаючи краї об'єктів абсолютно чіткими незалежно від масштабу. Для забезпечення високої обчислювальної ефективності розв'язання цього складного диференціального рівняння AKAZE використовує схеми швидкої експліцитної дифузії (Fast Explicit Diffusion, FED), що робить його придатним для роботи в режимі реального часу.

Після формування нелінійного простору масштабів здійснюється пошук просторових екстремумів. Для кожної оброблюваної точки обчислюється матриця Гессе (Hessian matrix), яка відображає міру локальної кривизни функції яскравості:

$$H = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{pmatrix}. \quad (2.2)$$

Алгоритм розраховує визначник цієї матриці (детермінант), що слугує головною метрикою відгуку (response). Точки, в яких визначник досягає локального максимуму одночасно в просторових координатах та на шкалі масштабів, фіксуються як ключові. Цей підхід дозволяє виділяти локальні структури типу «пляма» (blob) або кут, які відзначаються найвищою стабільністю до афінних перетворень, зміни ракурсу зйомки чи масштабування.

Далі для кожної знайденої ключової точки генерується бінарний дескриптор M-LDB (Modified Local Difference Binary). Цей дескриптор формується шляхом попарного порівняння середніх значень інтенсивності в чітко орієнтованих локальних фрагментах навколо точки. Вибір саме

бінарного формату подання зумовлений його екстремальною апаратною ефективністю: порівняння таких дескрипторів виконується на рівні найшвидших процесорних інструкцій через логічне XOR (виключне АБО). У межах даної системи стандартний розмір генерованого M-LDB дескриптора становить 61 байт (або 488 бітів) неперервної бінарної інформації [8, 16].

Хоча 488-бітні дескриптори описують кожну точку з вичерпною точністю, їхня значна розмірність робить операцію масового крос-порівняння матриць (наприклад, 500×500 точок для кожної пари зображень) обчислювально надмірною задачею. Таке навантаження провокує «прокляття розмірності», швидко переповнюючи кеш-пам’ять процесора та унеможливлюючи потокову класифікацію на стандартному обладнанні. Для розв’язання цієї проблеми розроблено програмний алгоритм агресивного стиснення даних на базі математичної моделі локально чутливого хешування (Locality-Sensitive Hashing, LSH) із застосуванням методу випадкових проєкцій (random projection).

Фундаментом цього підходу виступає лема Джонсона-Лінденштрауса (Johnson-Lindenstrauss lemma) [29, 30]. Вона математично доводить феномен того, що множину точок у багатовимірному евклідовому просторі можна лінійно відобразити у простір значно меншої розмірності таким чином, що попарні відносні відстані (кути) між точками зберігатимуться із наперед задано малою похибкою. Це гарантує, що відносне розташування семантичних кластерів (топологія даних) залишається незмінним, дозволяючи системі відрізняти класи один від одного навіть після втрати понад 96% початкового обсягу даних. Організація такого ізометричного відображення здійснюється шляхом перемноження багатовимірного вектора ознак на попередньо згенеровану ортогоналізовану матрицю випадкових значень.

Практичне інженерне моделювання цього процесу трансформації для кожного локального дескриптора реалізується у вигляді строго визначеного математичного та обчислювального конвеєра [31]:

– оригінальний 61-байтний дескриптор D алгоритмічно розгортається у вихідний одномірний бітовий вектор X, де $X \in \{0,1\}^{488}$. На цьому етапі дані представлені суто в додатному діапазоні у вигляді суцільного лінійного масиву, що забезпечує ідеальну локалізацію даних у кеші процесора для подальших паралельних операцій;

– для забезпечення симетрії та уникнення математичного зміщення (bias) під час майбутніх скалярних множень, вектор X піддається процедурі центрування. Кожен елемент вектора синхронно зміщується на постійну величину:

$$x_i' = x_i - 0{,}5. \tag{2.3}$$

У результаті цього перетворення початковий бінарний діапазон {0, 1} переходить у збалансований діапазон {-0,5, 0,5}. Це переносить центр мас вектора точно в початок координат. Без виконання цього кроку більшість проєкцій могли б згрупуватися виключно по один бік від гіперплощини, що різко знизило б інформаційну ентропію та корисність фінального хеш-коду;

– для запобігання непередбачуваним коливанням абсолютної довжини вектора обчислюється його евклідова норма (*L2*-норма), після чого вектор X' стандартизується:

$$X_{norm} = \frac{X'}{\|X'\|_2}, \tag{2.4}$$

де $\|X'\|_2 = \sqrt{\sum (x_i')^2}$.

Ця операція відіграє критичну геометричну роль: вона гарантує, що всі вектори проєктуються на поверхню гіперсфери одиничного радіуса. Завдяки цьому вплив абсолютної амплітуди вектора нівелюється, і подальше матричне множення стає чистою мірою кутової подібності (косинусної відстані) між ознаками;

– головним актом зниження розмірності є обчислення скалярного добутку між нормалізованим вектором X_{norm} та глобальною статичною матрицею проєкцій $W \in \mathbb{R}^{488 \cdot 16}$. Елементи матриці W згенеровані згідно зі стандартним нормальним (гауссовим) розподілом $\mathcal{N}(0,1)$. Математично лінійне перетворення описується як:

$$Y = X_{norm} \cdot W. \tag{2.5}$$

Геометрично кожен із 16 стовпців цієї матриці задає вектор нормалі до випадкової гіперплощини, яка розсікає багатовимірний простір даних. У результаті множення генерується безперервний вектор $Y \in \mathbb{R}^{16}$ координати якого показують величину проєкції вихідного дескриптора на кожну з цих 16 осей;

– на фінальному етапі безперервні значення вектора Y необхідно трансформувати назад у вкрай економний дискретний простір. Для цього застосовується простий умовний оператор порівняння з нулем, який алгоритмічно відповідає математичній функції Хевісайда (одиничного стрибка):

$$H_i = \begin{cases} 1, & y_i > 0 \\ 0, & y_i \leq 0 \end{cases}. \tag{2.6}$$

Цей крок геометрично визначає, по який саме бік (додатний чи від’ємний) від кожної проведеної випадкової гіперплощини опинився вектор. Такий підхід забезпечує виняткову стійкість до локальних шумів: дрібні флуктуації вихідних даних змінять лише величину проєкції, але їх не вистачить, щоб «перекинути» вектор на протилежний бік площини, а отже, біт хешу залишиться незмінним. У результаті застосування цього математичного конвеєра складний 488-бітний дескриптор надійно та блискавично конвертується у надкомпактний 16-бітний LSH-хеш.

2.2 Моделі оцінювання подібності

Після успішного виділення локальних ознак та зниження їхньої розмірності шляхом локально чутливого хешування (LSH) виникає необхідність кількісного оцінювання ступеня схожості між об’єктами. У контексті розробленої інформаційної технології ця задача розв’язується на двох ієрархічних рівнях: на мікрорівні (оцінювання відстаней між окремими локальними точками дескрипторами) та на макрорівні (оцінювання семантичної подібності цілісних об’єктів на основі їхніх інтегральних розподілів). Для кожного з цих рівнів необхідно формалізувати стійку математичну модель, яка б забезпечувала баланс між точністю порівняння та можливістю глибокої апаратної оптимізації обчислень.

На мікрорівні базовою метрикою для порівняння бінарних векторів (як вихідних 488-бітних дескрипторів, так і 16-бітних LSH-хешів) виступає відстань геммінга (hamming distance) [9]. Класичне математичне та алгоритмічне визначення цієї відстані полягає у підрахунку кількості позицій, у яких відповідні символи двох векторів однакової довжини відрізняються один від одного. В обчислювальній техніці ця метрика традиційно реалізується за допомогою побітової логічної операції виключного АБО (XOR) із подальшим підрахунком встановлених одиничних бітів (popcount).

Однак у задачах масового комп’ютерного зору, де необхідно виконувати крос-порівняння величезних масивів точок (генерація матриць відстаней $N \times M$), ітеративне застосування логічної операції XOR стає критичним «пляшковим горлом» (bottleneck) системи. Традиційні цикли перешкоджають використанню матричних співпроцесорів та сучасних бібліотек лінійної алгебри. Для подолання цього обмеження традиційну логічну операцію було переведено у площину лінійної алгебри шляхом побудови еквівалентної алгебраїчної моделі обчислення відстані геммінга.

Нехай задано два бінарні вектори рядки *A* та *B* довжиною *K*, де елементи $a_i, b_i \in \{0,1\}$. Логічна операція XOR для кожної координати може бути аналітично виражена через базові арифметичні операції:

$$a_i \oplus b_i = a_i + b_i - 2 \cdot a_i \cdot b_i.$$

Перевірка цієї алгебраїчної рівності підтверджує її істинність для всіх можливих станів бінарної системи:

– якщо $a_i = 1, b_i = 1$ то 1+1-2(1·1)=0;

– якщо $a_i = 1, b_i = 0$ то 1+0-2(1·0)=0;

– якщо $a_i = 0, b_i = 0$ то 0+0-2(0·0)=0.

Екстраполюючи цю рівність на весь багатовимірний простір для пошуку сумарної відстані геммінга D_H між векторами *A* та *B*, отримуємо наступне матричне рівняння:

$$D_H(A, B) = \sum_{i=1}^{K} A_i + \sum_{i=1}^{K} B_i - 2(A \cdot B^T), \quad (2.7)$$

де $A \cdot B^T$ – скалярний добуток двох векторів.

Ця математична трансформація має фундаментальне архітектурне значення. Заміна побітового порівняння на алгебраїчну суму з відніманням скалярного добутку дозволила системі повністю відмовитися від дорогих операцій логічного розгалуження (branching). Обчислення скалярного добутку матриць легко делегується низькорівневим математичним бібліотекам (наприклад, BLAS), які використовують векторні SIMD-інструкції сучасних процесорів (AVX2, AVX-512). Це забезпечує паралельну обробку масивів на апаратному рівні, гарантуючи високу обчислювальну швидкодію.

На макрорівні класифікації, після того як попарні відстані геммінга зведені у загальну квадратну матрицю, здійснюється перехід до аналізу глобальної структури об’єкта. Для цього матриця відстаней агрегується в

одновимірну інтегральну гістограму частот. Гістограма у цьому контексті є статистичним відображенням кількості пар ключових точок, що знаходяться на певній алгоритмічній відстані одна від одної.

Для оцінювання ступеня візуальної спорідненості двох різних зображень необхідно порівняти їхні згенеровані статистичні розподіли (гістограми). Вибір оптимальної метрики для такого порівняння є критично важливим етапом математичного моделювання. У межах розробленої технології як основний інструмент порівняння була обґрунтована та застосована манхеттенська відстань (*L1*-норма) [32].

Манхеттенська відстань D_{L1} між двома векторами гістограмами V_1 та V_2 розмірністю n (де n=16 для хешованого простору) обчислюється як сума абсолютних різниць їхніх відповідних координат (відсіків):

$$D_{L1}(V_1, V_2) = \sum_{i=1}^{n} |v_{1i} - v_{2i}| . \tag{2.8}$$

Вибір саме *L1*-норми, на противагу класичній Евклідовій відстані (*L2*-нормі), зумовлений глибоким математичним та прикладним обґрунтуванням. Евклідова відстань передбачає зведення різниці координат у квадрат:

$$D_{L2}(V_1, V_2) = \sqrt{\sum_{i=1}^{n} (v_{1i} - v_{2i})^2} . \tag{2.9}$$

Зведення у квадрат у *L2*-нормі створює ефект математичної гіперболізації [33]: метрика стає експоненційно чутливою до статистичних викидів (outliers) та локальних аномалій. У задачах комп'ютерного зору, і зокрема при аналізі металевих об'єктів складної форми (таких як леза ножів), такі аномалії виникають постійно. Наприклад, інтенсивний локальний відблиск від джерела світла на гладкій поверхні або випадкова глибока тінь

від руків’я можуть спричинити різкий сплеск частоти знайдених точок лише в одному-єдиному відсіку гістограми. Евклідова метрика зведе це аномальне відхилення у квадрат, що непропорційно збільшить загальну відстань між зображеннями і призведе до хибного висновку про те, що предмети належать до різних семантичних класів.

Манхеттенська відстань (*L1*-норма), натомість, штрафує кожне відхилення виключно за лінійним законом. Це забезпечує системі властивість високої робастності (стійкості). Метрика здатна ефективно абсорбувати та згладжувати одиничні локальні шуми чи оптичні артефакти, зосереджуючись на загальній (глобальній) концептуальній подібності статистичних розподілів. Гістограма відстаней геммінга за своєю природою лінійно акумулює інформацію про відмінності в бітах, тому застосування лінійної манхеттенської відстані для їх порівняння є природним математичним продовженням, яке повністю зберігає початкові метричні пропорції простору ознак.

2.3 Моделі прийняття рішень у методах класифікації

Математичні моделі простору ознак та метрик подібності, формалізовані у попередніх підрозділах, складають теоретичний фундамент системи. Однак для їхньої практичної реалізації у вигляді функціонуючого програмного забезпечення необхідно здійснити алгоритмічне моделювання – побудувати чіткі, логічно несуперечливі послідовності обчислювальних кроків (конвеєри обробки даних). У межах даної роботи було розроблено та змодельовано два концептуально різні алгоритмічні підходи до задачі класифікації: традиційний алгоритм локального зіставлення ознак та оптимізований алгоритм на основі глобальних статистичних розподілів (гістограм) [19].

Перший алгоритмічний підхід базується на класичній парадигмі пошуку найближчого сусіда (nearest neighbor search) у просторі нестиснутих бінарних

дескрипторів [20]. Основна ідея цього алгоритму полягає у встановленні жорстких взаємозв'язків (зіставлень) між кожною окремою ключовою точкою тестового зображення та найближчою до неї точкою на еталонному зображенні.

Алгоритмічний конвеєр цього методу, графічно представлений у вигляді блок-схеми на рисунку 2.1, складається з таких послідовних завдань:

– ініціалізація: завантаження тестового зображення та виділення N ключових точок (дескрипторів). Завантаження бази еталонних дескрипторі;

– крос-порівняння: для кожної ключової точки і з тестового зображення обчислюється відстань геммінга до всіх точок j поточного еталонного зображення;

– фільтрація мінімумів: алгоритм виконує сортування масиву відстаней та знаходить локальний мінімум – точку j_{min}, відстань до якої є найменшою;

– поріг відсікання (lowe's ratio test / thresholding): якщо знайдена мінімальна відстань перевищує допустимий ліміт (наприклад, алгоритм не знайшов достатньо схожої точки через зміну ракурсу), зв'язок відхиляється. Якщо відстань у межах норми, зв'язок фіксується як «успішний збіг»;

– голосування (voting): після перебору всіх точок алгоритм підраховує загальну кількість успішних збігів для поточного еталона. Цей процес повторюється для всіх еталонів у базі даних;

– прийняття рішення: система ранжує еталони за кількістю голосів. Еталон, що набрав максимальну кількість успішних збігів, визнається класом тестового об'єкта.

Незважаючи на інтуїтивну зрозумілість, алгоритмічне моделювання виявляє критичну вразливість цього підходу: крок пошуку мінімумів і сортування вимагає асимптотичної обчислювальної складності $O(N \log N)$ для кожної точки, що робить алгоритм непридатним для потокової обробки великих баз даних і надзвичайно чутливим до просторових деформацій (коли локальний збіг неможливий через зміну кута освітлення).

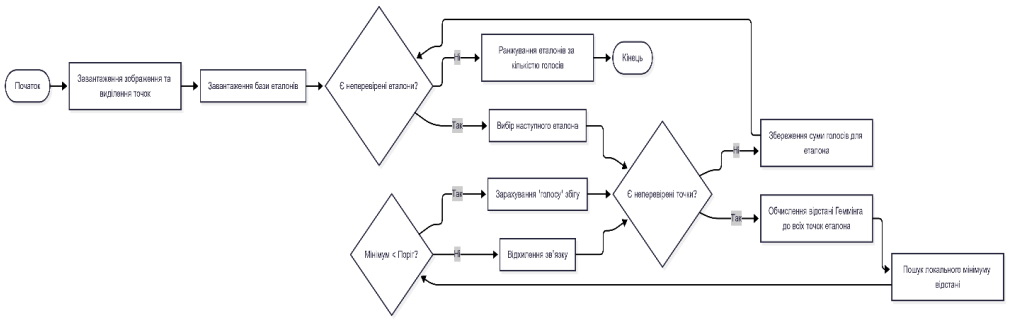

Рисунок 2.1 – Блок-схема алгоритму традиційного зіставлення локальних ознак (метод 1)

Алгоритм класифікації на основі інтегральних гістограм (методи 2 та 3).

Для подолання алгоритмічних обмежень локального зіставлення було змодельовано альтернативний конвеєр обробки, який абстрагується від пошуку одиничних збігів і переходить до аналізу глобальної топології об’єкта. Цей підхід реалізований у двох варіаціях: для повнорозмірних (метод 2) та хешованих (метод 3) просторів ознак. Найбільшу практичну цінність має метод 3, оскільки він інтегрує в алгоритм етап LSH-стиснення.

Алгоритмічна послідовність гістограмної класифікації з використанням LSH-трансформації, блок-схема якої наведена на рисунку 2.2, виглядає наступним чином:

– виділення та нормалізація: система виділяє N ключових точок тестового зображення і жорстко обмежує вибірку до фіксованого значення (наприклад, 500 найкращих точок за метрикою відгуку) для стабілізації подальших матричних обчислень;

– LSH-трансформація: масив 488-бітних дескрипторів проходить векторизовану трансформацію через множення на глобальну матрицю проєкцій. Результатом є надкомпактна матриця 16-бітних хеш-кодів;

– побудова внутрішньої матриці відстаней: алгоритм генерує симетричну матрицю розмірністю $N \times N$, де кожен елемент є відстанню геммінга між хешами самого тестового об’єкта. Замість пошуку найближчих сусідів з іншим зображенням, об’єкт порівнюється сам із собою для виявлення своєї внутрішньої структури;

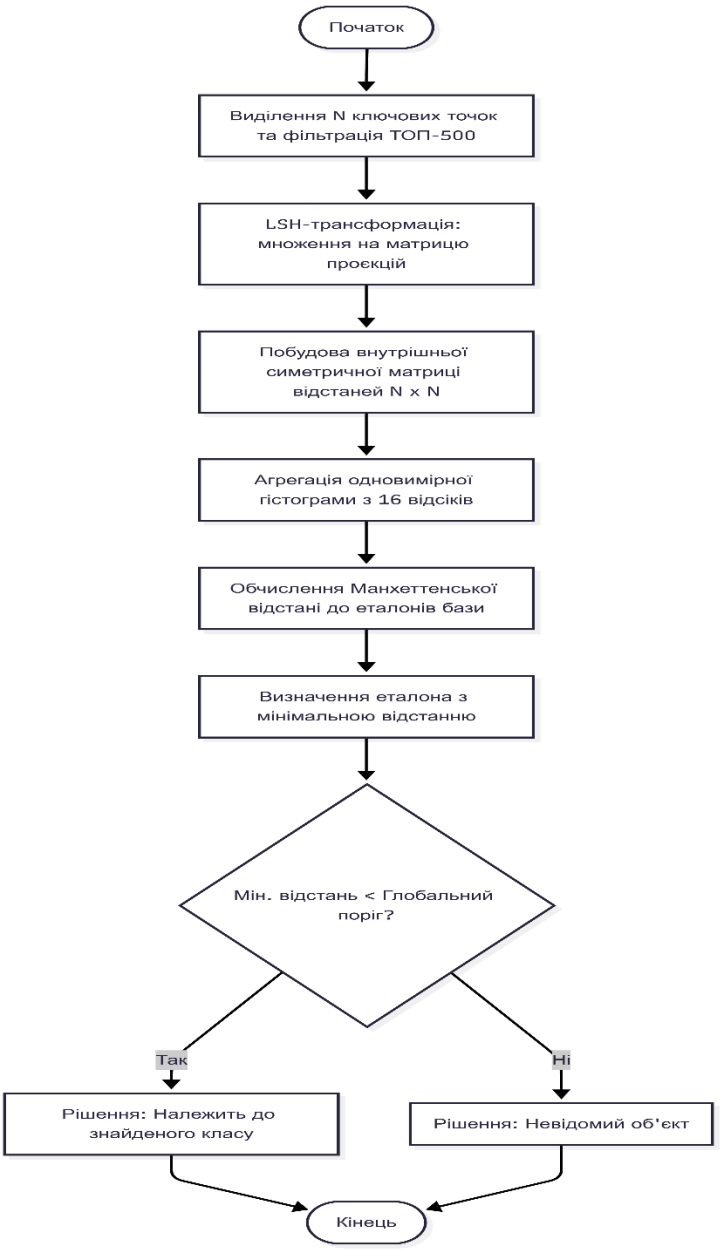

Рисунок 2.2 – Блок-схема алгоритму класифікації на основі інтегральних хеш-гістограм (метод 3)

– агрегація гістограми: квадратна матриця розгортається в одновимірний масив. За допомогою алгоритмічної функції підрахунку частот (аналог np.bincount) генерується гістограма з 16 відсіків, яка стає семантичною сигнатурою зображення;

– метричне порівняння: система витягує з бази даних заздалегідь обчислені гістограми еталонів. Для кожної пари «тест – еталон» обчислюється манхеттенська відстань (сума модулів різниць відповідних відсіків);

– порогове відсікання (open set validation): якщо мінімальна знайдена манхеттенська відстань перевищує глобально заданий алгоритмічний поріг (наприклад, 3000), об’єкт ідентифікується як «Невідомий клас». Інакше система видає назву найближчого еталона.

Алгоритмічне моделювання оптимізованого конвеєра доводить його абсолютну архітектурну перевагу. Усунення етапу перебору та сортування зводить обчислювальну складність порівняння двох зображень до *O(1)* (оскільки розмір гістограм завжди фіксований – 16 або 488 елементів). Акумуляція внутрішніх попарних відстаней дозволяє системі створювати робастний (стійкий) «відбиток» топології предмета, ігноруючи при цьому одиничні оптичні аномалії, які гарантовано призвели б до збою у традиційному методі 1.

2.4 Моделювання етапів прийняття класифікаційного рішення

Перехід від теоретичних математичних моделей та абстрактних алгоритмів до практичної розробки програмного забезпечення вимагає стандартизованого проєктування архітектури. Для вирішення цього завдання в межах даної роботи було застосовано методологію об’єктно орієнтованого аналізу та проєктування з використанням уніфікованої мови моделювання UML (Unified Modeling Language) [7]. Такий підхід дозволяє деталізувати функціональні вимоги, візуалізувати потік управління даними та спроєктувати статичну структуру класів майбутнього програмного застосунку.

Моделювання функціональних вимог здійснюється за допомогою діаграми прецедентів (use case diagram), поданої на рисунку 2.3, яка ілюструє взаємодію зовнішніх акторів (користувачів) із програмною системою. Головним актором у розробленій системі виступає «Користувач» (дослідник або оператор), який ініціює процес класифікації.

Основні прецеденти системи включають:

– вибір директорії еталонів – користувач за допомогою графічного інтерфейсу вказує шлях до папки з базою даних зображень;

– вибір базового зображення – користувач обирає конкретний файл, який необхідно класифікувати;

– автоматичний пакетний аналіз – базовий прецедент, який запускається системою самостійно після отримання шляхів до файлів. Він включає в себе (відношення <<include>>) приховані математичні підпроцеси: виділення ознак, LSH-хешування та паралельний розрахунок одразу всіма трьома методами класифікації;

– перегляд результатів (термінал) – система виводить користувачу остаточний рейтинг схожості, спрацювання порогів відсікання та системні таймери обчислень;

– отримання файлових звітів – система автоматично формує та зберігає підсумкові звіти у форматі Excel та візуальні зображення з ключовими точками у виділену директорію OUTPUT.

Рисунок 2.3 – Діаграма прецедентів (use case diagram) програмної системи

Для візуалізації внутрішньої логіки та послідовності обчислювальних процесів у часі було побудовано діаграму діяльності (activity diagram), яку зображено на рисунку 2.4. Вона деталізує реальний потік управління розробленого скрипта від моменту ініціалізації до видачі остаточного рішення.

Діаграма демонструє двофазний характер алгоритмічного конвеєра. На першій фазі система ізольовано обробляє базове зображення (виділення ознак, LSH-трансформація, генерація базових гістограм). На другій фазі запускається ітеративний цикл обробки бази даних, де кожне еталонне зображення проходить аналогічний конвеєр та одразу порівнюється з базовим. Важливим фінальним елементом динамічної моделі є вузол прийняття рішень (decision node), у якому манхеттенські відстані перевіряються на відповідність глобальним порогам THRESHOLD_488 та THRESHOLD_16 для ідентифікації невідомих об’єктів.

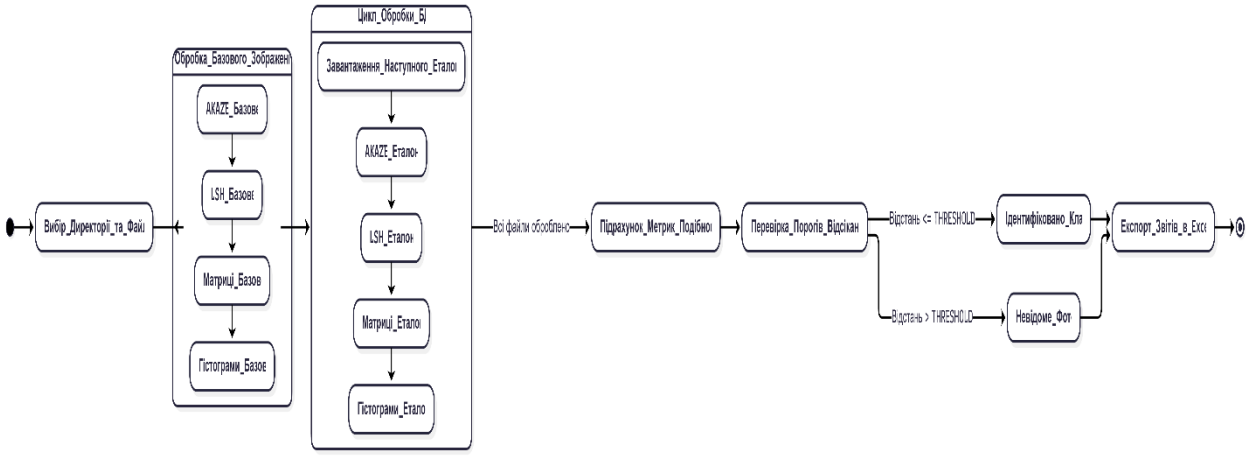

Рисунок 2.4 – Діаграма діяльності (activity diagram) процесу класифікації

Для реалізації розроблених алгоритмів принципами модульного та об’єктно орієнтованого програмування було спроєктовано статичну структуру системи у вигляді діаграми класів (class diagram), наведеної на рисунку 2.5. Архітектура застосунку базується на принципі єдиної відповідальності (single responsibility principle), де кожна група математичних операцій інкапсульована у відповідний логічний модуль.

Основні класи системи (що відображають функціональні блоки коду):

– MainController – головний модуль (main()), який керує потоком виконання, обробляє діалогові вікна (tkinter) та координує роботу таймерів;

– FeatureExtractor – модуль взаємодії з комп’ютерним зором. Реалізує методи детектування ознак (detect_akaze) та жорсткої фільтрації точок за метрикою відгуку (select_top);

– LSHasher – математичний модуль зниження розмірності. Зберігає глобальну згенеровану матрицю проєкцій PROJECTION_MATRIX і містить алгоритм лінійної трансформації дескрипторів (transform_descriptor);

– MetricCalculator – обчислювальне ядро системи на базі NumPy. Реалізує векторизовані методи обчислення відстаней геммінга (compute_hamming_matrix) без використання стандартних засобів OpenCV, а також розрахунок манхеттенської відстані між гістограмами;

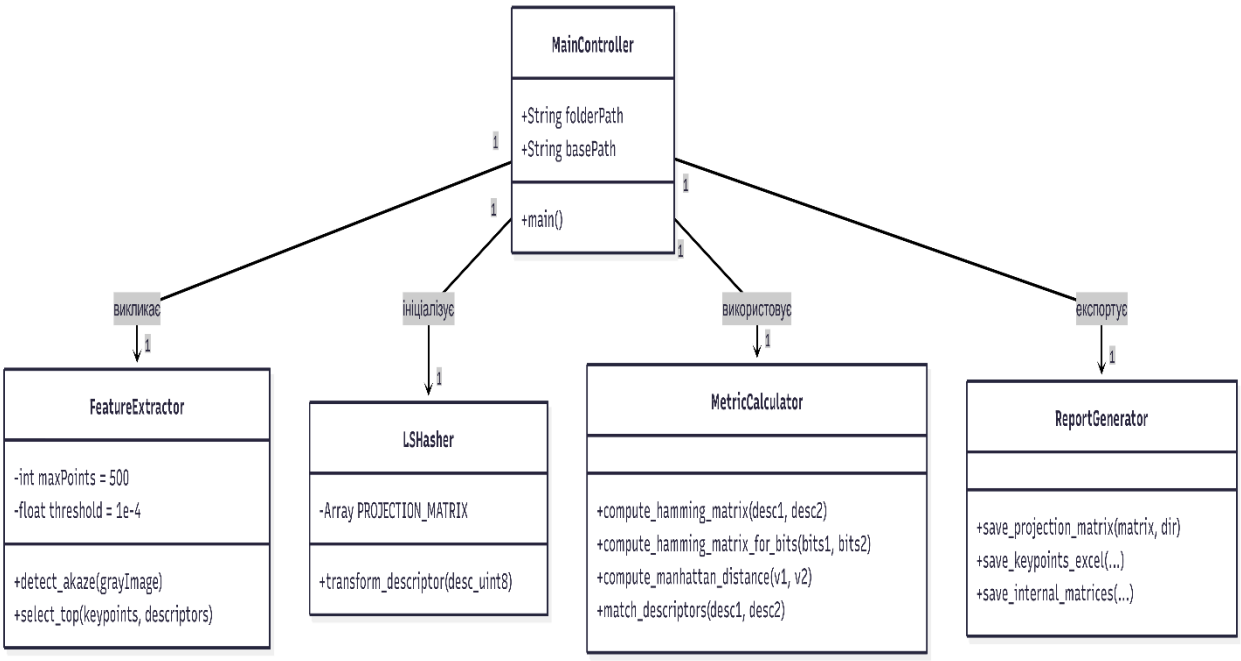

Рисунок 2.5 – Діаграма класів (class diagram) архітектури програмного застосунку

– ReportGenerator – файловий менеджер, який відповідає за вивантаження результатів. Об’єднує функції збереження зображень та генерації Excel таблиць за допомогою бібліотеки Pandas (save_keypoints_excel, save_internal_matrices).

Така декомпозиція системи на незалежні логічні модулі не лише точно відображає реальну архітектурну структуру розробленого Python скрипта, але й закладає міцний фундамент для його майбутнього масштабування та модернізації. Розподіл відповідальності між вузькоспеціалізованими класами (контролер, екстрактор, хешер, обчислювач та генератор звітів) повністю відповідає фундаментальним принципам програмної інженерії, зокрема принципу відкритості/закритості (open/closed principle) з класичної архітектурної тріади SOLID. На практиці це означає, що програмний код є відкритим для розширення функціоналу, але закритим для модифікації вже наявних перевірених механізмів. Наприклад, якщо в майбутньому виникне наукова необхідність замінити базовий алгоритм детектування AKAZE на більш сучасну глибинну нейромережеву архітектуру (таку як SuperPoint або SIFT-сумісні CNN), розробнику достатньо буде створити новий клас, що реалізує інтерфейс екстрактора, не вносячи при цьому жодних руйнівних змін у математичне ядро системи хешування чи логіку генерації Excel звітів.

Аналогічним чином модульність забезпечує безболісну заміну математичних метрик: замість манхеттенської відстані можна легко інтегрувати алгоритм обчислення Евклідової відстані або косинусної подібності шляхом простого додавання нового методу в обчислювальний клас MetricCalculator. Крім того, така архітектура суттєво полегшує подальшу технічну підтримку застосунку та імплементацію процесів автоматизованого тестування (зокрема unit тестування). Кожен логічний модуль можна ізольовано перевірити на наявність обчислювальних помилок, симулюючи синтетичні вхідні дані. У стратегічній перспективі, у разі розширення бази даних до десятків чи сотень тисяч зображень, інкапсульована логіка управління ресурсами дозволить з мінімальними зусиллями інтегрувати багатопотоковість (multithreading), паралельні обчислення (multiprocessing) або асинхронну обробку на рівні головного контролера MainController, залишаючи низькорівневі математичні класи цілком недоторканими та стабільними.

Окрім моделювання загального потоку управління (діаграма діяльності) та статичної архітектури (діаграма класів), для повноцінного об’єктно орієнтованого проєктування необхідно візуалізувати життєвий цикл передачі повідомлень між об’єктами. Для цього було розроблено діаграму послідовностей (sequence diagram), представлену на рисунку 2.6.

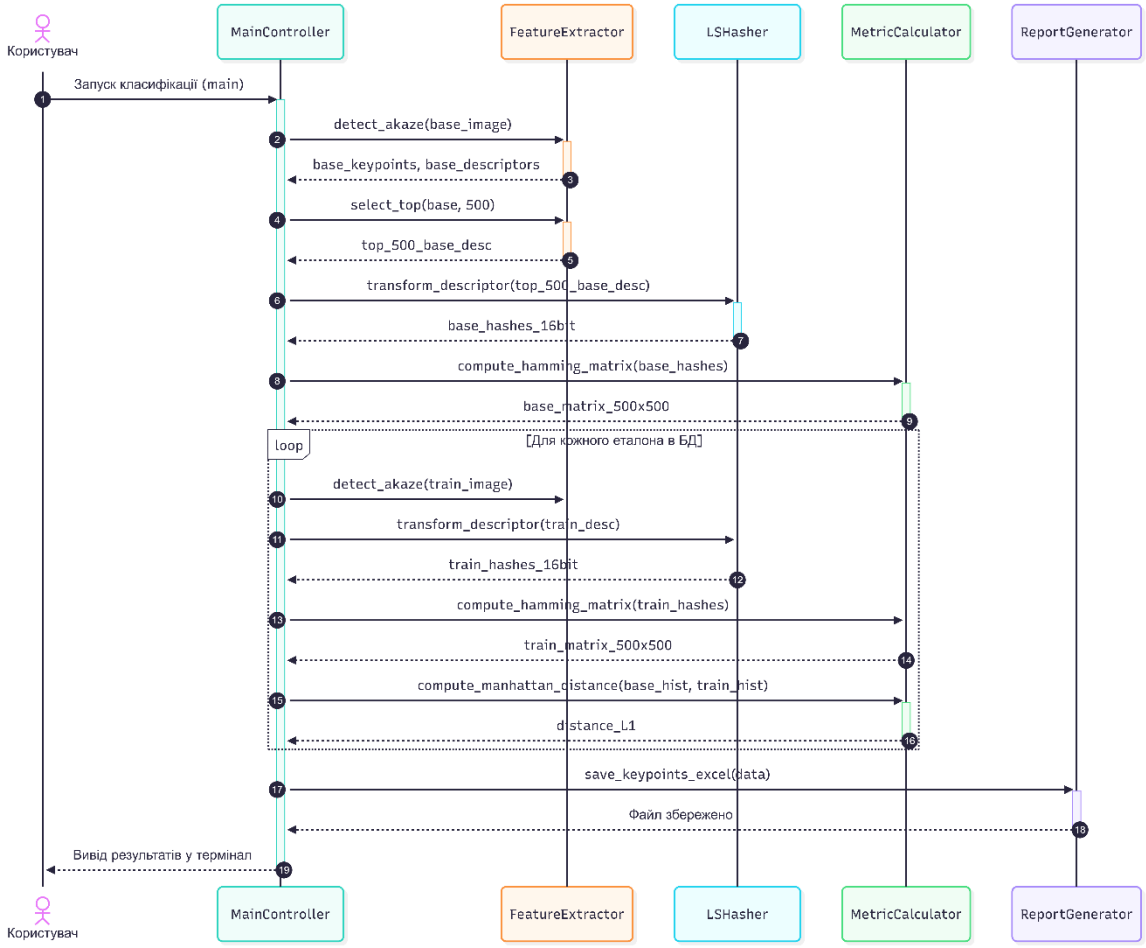

Рисунок 2.6 – Діаграма послідовностей (sequence diagram) взаємодії об’єктів системи

Ця діаграма відображає хронологічний порядок викликів методів та повернення результатів під час виконання оптимізованого алгоритму класифікації (метод 3). Цей вид UML моделювання є критично важливим для розуміння динаміки програмного застосунку, оскільки він дозволяє відстежити не лише статичні зв’язки між класами, але й реальний життєвий

цикл кожного обчислювального процесу у часі. Діаграма наочно фіксує моменти активації та деактивації об’єктів (так звані життєві лінії або lifelines), що дає змогу оцінити ефективність управління оперативною пам’яттю та тривалість існування масивних структур даних під час їхньої ітеративної обробки.

Модель демонструє, як головний контролер (MainController) виступає оркестратором системи. Роль оркестратора полягає у суворому управлінні потоком виконання (control flow) та координації передачі масивів даних між ізольованими логічними модулями. Він послідовно звертається до FeatureExtractor для отримання матриць дескрипторів, потім передає ці масиви до LSHasher для стиснення. Важливо зазначити, що всі зображені виклики є синхронними: головний потік управління блокується і очікує повного завершення математичних перетворень на кожному етапі, перш ніж ініціювати наступний крок, що гарантує абсолютну консистентність даних та унеможливлює виникнення стану гонитви (race condition).

Після отримання 16-бітних хешів, контролер ініціює роботу MetricCalculator для обчислення квадратних матриць відстаней геммінга та генерації інтегральних гістограм. У цей момент система делегує виконання найскладніших обчислень оптимізованим функціям NumPy, утримуючи проміжні результати виключно в межах локальної області видимості методів. Завершується цикл викликом ReportGenerator, який записує результати на жорсткий диск. Експорт даних у формат Excel та збереження візуалізацій здійснюються лише після успішного завершення всіх обчислень. Це архітектурне рішення надійно ізолює алгоритмічний час роботи процесора (CPU time) від повільних операцій вводу-виводу (I/O operations).

3 АНАЛІЗ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ ПРОГРАМНОЇ МОДЕЛІ

3.1 Обґрунтування вибору інструментальних засобів для виконання роботи

Процес розроблення програмно алгоритмічного комплексу для аналізу цифрових зображень вимагає ретельного підбору технології. Успішність реалізації математичних моделей, зокрема алгоритмів локально чутливого хешування (Locality-Sensitive Hashing, LSH) та роботи з великими масивами дескрипторів безпосередньо залежить від обчислювальної ефективності, надійності та стабільності обраних інструментальних засобів.

Основним критерієм вибору мови програмування була необхідність забезпечення високої швидкодії прототипування алгоритмів із можливістю їхньої подальшої інтеграції в складніші системи, у тому числі ті, що використовують технології штучного інтелекту та машинного навчання. З огляду на це, як базову мову програмування було обрано Python. Ця високорівнева мова загального призначення характеризується мультипарадигмальністю (підтримує об’єктно орієнтований, імперативний та функціональний підходи), кросплатформністю та лаконічним синтаксисом. Попри те, що Python є інтерпретованою мовою, її екосистема дозволяє делегувати ресурсомісткі обчислення оптимізованим модулям, написаним мовами C/C++, що нівелює проблему продуктивності під час обробки цифрових зображень [34].

Для вирішення специфічних завдань у межах дипломної роботи було сформовано комплексний набір спеціалізованих бібліотек:

а) бібліотека комп’ютерного зору OpenCV (cv2). Вона є фундаментальним інструментом для роботи з візуальними даними. OpenCV (Open Source Computer Vision Library) має відкритий вихідний код і містить понад 2500 оптимізованих алгоритмів [35]. У контексті даної роботи ця бібліотека використовується для розв’язання базових завдань попередньої

обробки: завантаження графічних файлів із файлової системи, перетворення їх у градації сірого (grayscale) для зменшення обчислювального навантаження, а також для виявлення ключових точок. Ключовим аргументом на користь OpenCV стала наявність вбудованої, оптимізованої та стабільної реалізації алгоритму AKAZE (Accelerated-KAZE). На відміну від застарілих або ліцензійно обмежених методів (наприклад, SIFT чи SURF), AKAZE в OpenCV забезпечує високу інваріантність до масштабування та обертання, що є критично важливим для надійної класифікації [36];

б) бібліотека для наукових обчислень NumPy [29]. Оскільки вдосконалення методу порівняння зображень передбачає роботу з багатовимірними векторами ознак та матрицями відстаней (зокрема, розмірністю 500×500), використання стандартних типів даних Python (наприклад, списків) призвело б до критичного падіння продуктивності. NumPy (Numerical Python) розв’язує цю проблему завдяки підтримці багатовимірних масивів і матриць, а також широкому набору високорівневих математичних функцій. У розробленому застосунку NumPy відіграє ключову роль у:

1) генерації випадкової матриці проєкцій для алгоритму LSH;

2) обчисленні евклідової норми (*L2*-норми) для стандартизації стовпців матриці;

3) здійсненні швидких побітових операцій (перетворення масивів байтів у масиви бітів за допомогою np.unpackbits);

4) обчисленні матриці відстаней геммінга. Власна реалізація розрахунку через скалярний добуток матриць та операції сум у NumPy дозволила обійти відомі апаратні помилки та нестабільність вбудованої функції cv2.batchDistance під час роботи з великими бінарними дескрипторами;

в) бібліотека для аналізу даних Pandas [28]: для того щоб обґрунтувати ефективність запропонованих алгоритмічних рішень, виникла потреба у структуруванні великих обсягів експериментальних даних. Pandas забезпечує

зручну абстракцію у вигляді DataFrame (двовимірних табличних структур). У межах програмної реалізації Pandas використовується для збору, агрегації та подальшого збереження результатів тестування (координат виявлених точок, значень кутів, масштабу, відгуків, оригінальних та хешованих дескрипторів). Здатність Pandas безперешкодно інтегруватися з форматом Excel (pd.ExcelWriter) дозволила автоматизувати генерацію звітів (match_summary.xlsx, histograms_summary.xlsx), що є необхідною умовою для глибокого аналітичного порівняння традиційних і розроблених методів;

г) засоби візуалізації (Matplotlib): для наочного представлення отриманих результатів, оцінювання якості роботи системи та підготовки графічних матеріалів застосовується бібліотека Matplotlib. Хоча OpenCV має вбудовані засоби для малювання на зображеннях (наприклад, побудова ліній збігів між ключовими точками), Matplotlib необхідний для побудови аналітичних графіків. Зокрема, вона дозволяє візуалізувати гістограми розподілу відстаней, порівнювати манхеттенські відстані для векторів розмірністю 488 та 16 бітів, і наочно демонструвати ефективність стиснення інформації;

д) інструментарій графічного інтерфейсу Tkinter: з метою спрощення процесу тестування та апробації програмного продукту, було прийнято рішення інтегрувати базовий графічний інтерфейс користувача (ГІК). Оскільки застосунок розробляється передусім для дослідницьких цілей, використання великих фреймворків (на кшталт PyQt або Kivy) було визнано недоцільним через їхню надмірність. Стандартна бібліотека Tkinter забезпечує весь необхідний функціонал для виклику системних діалогових вікон (filedialog). Це дає змогу користувачеві динамічно обирати директорії з наборами даних та вказувати базове еталонне зображення під час виконання програми, уникаючи необхідності редагування початкового коду («hardcoding» шляхів до файлів).

Незважаючи на те, що бібліотека OpenCV є стандартом де-факто у сфері комп’ютерного зору, під час практичної реалізації системи було виявлено ряд

архітектурних обмежень її вбудованих функцій, зокрема cv2.batchDistance та BFMatcher. Ці функції, розроблені переважно для традиційного зіставлення невеликих масивів дескрипторів, демонструють нестабільність при роботі з великими бінарними матрицями (наприклад, 500×500 точок), що супроводжується непередбачуваними стрибками споживання оперативної пам’яті (memory leaks) та періодичними помилками на рівні C++ серверної частини (segmentation fault). Для усунення цих вразливостей архітектуру застосунку було концептуально змінено: функцію розрахунку відстані геммінга було перенесено з OpenCV у площину суто математичних операцій бібліотеки NumPy. Таке рішення обґрунтовується глибокою оптимізацією NumPy для векторизованих обчислень. Замість ітеративного побітового порівняння, яке використовує OpenCV, розроблений алгоритм застосовує математичний трюк – розкладання логічної операції XOR через скалярний добуток матриць та суми векторів. Бібліотека NumPy, спираючись на низькорівневі математичні бібліотеки BLAS (Basic Linear Algebra Subprograms), виконує ці операції паралельно, використовуючи SIMD-інструкції сучасних процесорів. Це не лише повністю усунуло проблему нестабільності пам’яті, але й забезпечило рівні та прозорі умови для подальшого часового тесту всіх трьох методів класифікації, оскільки обчислення тепер відбуваються в єдиному оптимізованому середовищі.

Обраний комплекс інструментальних засобів повністю відповідає поставленим завданням. Поєднання OpenCV для безпосередньої роботи з пікселями, оптимізованого математичного апарату NumPy та засобів аналізу Pandas створює надійне підґрунтя для програмної реалізації, тестування та подальшого масштабування розроблених рішень у сфері обробки візуальної інформації.

3.2 Етапи розроблення та програмної реалізації запропонованих методів

Процес розроблення програмної складової роботи охоплює низку послідовних етапів: ініціалізація глобальних параметрів системи, виділення ознак із цифрових зображень, математична трансформація дескрипторів із застосуванням методів зниження розмірності, оптимізоване обчислення матриць відстаней та класифікація об’єктів за гістограмами розподілу. Усі етапи детально обґрунтовуються та наочно супроводжуються за допомогою відповідних рисунків і таблиць.

У цьому підрозділі покроково висвітлено етапи розроблення, а також наведено фрагменти власного програмного коду, які є концептуальним удосконаленням загальновідомих методів аналізу візуальної інформації.

Першим етапом роботи алгоритму є завантаження еталонного (базового) зображення та набору даних (зображень із бази) для порівняння. Оскільки кольорова інформація не є критично важливою для виявлення структурних ключових точок, усі зображення проходять етап препроцесингу: конвертацію в одноканальний простір (градації сірого).

Для знаходження спільних ознак застосовується алгоритм AKAZE [8, 16]. У розробленому застосунку ініціалізація детектора відбувається за допомогою функції cv2.AKAZE_create зі спеціально підібраними параметрами:

– descriptor_type=cv2.AKAZE_DESCRIPTOR_MLDB – використання бінарних дескрипторів Modified Local Difference Binary, які є ефективними для розрахунку відстані геммінга;

– threshold=1e-4 – навмисно занижений поріг чутливості алгоритму. Цей крок є критично важливим для роботи, оскільки він дозволяє алгоритму знаходити надлишкову кількість ключових точок (часто тисячі на одне зображення).

Оскільки обробка абсолютно всіх знайдених точок призвела б до експоненційного зростання обчислювального часу, було реалізовано алгоритм фільтрації select_top. Він сортує масив виявлених точок за метрикою response (відгук, що характеризує «силу» або контрастність знайденої точки) у порядку спадання та відбирає суворо визначену кількість – ТОП-500 точок. Таке обмеження гарантує стабільний розмір матриць порівняння (500×500) і дозволяє об’єктивно оцінювати часові витрати алгоритмів за рівних умов.

Застосування жорсткого обмеження кількості оброблюваних ключових точок (до 500 для базового експерименту та до 250 для перевірочного) є не випадковим, а математично обґрунтованим рішенням, яке відображає фундаментальний компроміс (trade-off) між швидкодією та репрезентативністю ознак. Алгоритм AKAZE, завдяки низькому порогу чутливості (threshold=1e-4), здатний виділяти тисячі локальних екстремумів на одному зображенні високої роздільної здатності (наприклад, 1280×934 пікселів). Однак переважна більшість цих точок припадає на низькоконтрастний фоновий шум або дрібні відблиски, які не несуть семантичного навантаження для визначення геометрії ножа.

Використання метрики відгуку (response), що визначається як детермінант матриці Гессе в просторі нелінійних масштабів, дозволяє алгоритму select_top алгоритмічно «відсікти» цей шум, залишивши лише найбільш стабільні та виразні кути (наприклад, кінчик леза, переходи руків’я). Обмеження вибірки фіксованим числом (500 точок) виконує ще одну критично важливу функцію: воно гарантує постійний розмір матриць відстаней геммінга (500×500) [15]. У свою чергу, це забезпечує сталість розмірності вихідних інтегральних векторів гістограм, що є обов’язковою математичною умовою для їх подальшого коректного порівняння за допомогою манхеттенської відстані. Без такої фіксації, обчислення манхеттенської відстані між гістограмами різного обсягу потребувало б складної процедури динамічного вирівнювання або нормалізації площ, що суттєво уповільнило б процес класифікації та могло б внести додаткову похибку.

Стандартний дескриптор AKAZE формує масив із 61 байта на кожну ключову точку, що математично відповідає вектору з 488 бітів. З метою оптимізації процесу порівняння та зменшення обчислювального навантаження у роботі розроблено модуль для трансформації дескрипторів на основі випадкових проєкцій [22].

Суть методу полягає у проєктуванні багатовимірного вектора у простір значно меншої розмірності (з 488 до 16) зі збереженням відносних відстаней між векторами [21]. Для цього на етапі запуску програми ініціалізується глобальна матриця проєкцій розмірністю 488×16 за допомогою нормального (гауссового) розподілу. Для забезпечення математичної коректності подальших операцій кожен стовпець цієї матриці нормується (евклідова норма *L2* зводиться до одиниці).

Процес трансформації окремого дескриптора передбачає кілька математичних кроків, програмну реалізацію яких наведено на лістингу 3.1.

Лістинг 3.1 Метод математичної обробки та хешування дескриптора:

```
def transform_descriptor(desc_uint8):
    bits = np.unpackbits(desc_uint8).astype(np.float32)
    centered = bits - 0.5
    norm = np.linalg.norm(centered)
    normalized = centered / norm if norm > 0 else centered
    projected = np.dot(normalized, PROJECTION_MATRIX)
    hash_bits = (projected > 0).astype(np.uint8)
    return hash_bits[::-1]
```

Програмна реалізація процесу трансформації вихідних дескрипторів у компактні хеш-коди інкапсульована у функції transform_descriptor. З метою максимізації обчислювальної швидкодії, алгоритм повністю відмовляється від використання традиційних ітеративних циклів інтерпретатора Python.

Натомість обробка даних здійснюється за допомогою серії оптимізованих векторизованих методів бібліотеки NumPy.

На першому етапі функція np.unpackbits миттєво розгортає 61-байтний масив ознак OpenCV у бітовий вектор, виконуючи цю операцію суцільним блоком безпосередньо в оперативній пам’яті. Наступним кроком відбувається паралельне центрування елементів та їх швидка стандартизація за допомогою методу np.linalg.norm. Ключовим архітектурним рішенням стиснення є застосування матричного множення np.dot для обчислення скалярного добутку між нормалізованим вектором та глобальною матрицею проєкцій. Цей підхід дозволяє делегувати найбільш ресурсомісткі обчислення оптимізованим низькорівневим C-бібліотекам, які використовують апаратне прискорення процесора. У результаті такого програмного конвеєра, початковий масив байтів екстремально швидко та стабільно перетворюється на цільовий 16-бітний LSH-хеш, що є критично важливим для безперебійної обробки великих баз зображень у реальному часі.

Основним критерієм подібності бінарних дескрипторів є відстань геммінга (hamming distance) – кількість позицій, у яких відповідні символи двох слів однакової довжини різні. Традиційно в OpenCV для цього використовується BFMatcher або функція cv2.batchDistance. Проте, у процесі розроблення було виявлено, що стандартні засоби часто працюють нестабільно під час масового порівняння бінарних масивів (через особливості виділення пам’яті на рівні C++).

Для розв’язання цієї проблеми було розроблено авторський метод обчислення матриці відстаней на основі алгебри матриць бібліотеки NumPy [9, 11].

Програмний механізм оптимізованого обчислення матриці відстаней геммінга можна детально проаналізувати, спираючись на код, наведений у лістингу 3.2. У цій функції реалізовано повну відмову від повільного побітового порівняння в ітеративних циклах на користь масових паралельних операцій (векторизації).

На першому кроці алгоритму вбудована функція np.unpackbits розгортає масиви байтів дескрипторів (desc1 та desc2) у двовимірні матриці бітів. Приведення цих матриць до типу np.int32 за допомогою методу .astype() є обов'язковим архітектурним рішенням, яке запобігає апаратному цілочисельному переповненню під час виконання подальших математичних операцій над великими масивами.

Далі система обчислює суми одиничних бітів для кожного дескриптора (sum1 та sum2). Використання специфічних зрізів [:, None] та [None, :] ініціює вбудований механізм «трансляції» (broadcasting) бібліотеки NumPy. Це дозволяє віртуально розширити масиви до розмірності фінальної матриці ($N \times M$), не виконуючи при цьому фізичного дублювання даних в оперативній пам'яті, що суттєво економить ресурси.

Головним рушієм оптимізації виступає матричне множення np.dot(bits1, bits2.T). Замість того, щоб перебирати кожну пару дескрипторів окремо, програма делегує цю операцію низькорівневим математичним C-бібліотекам (таким як BLAS). Вони, у свою чергу, залучають апаратні SIMD-інструкції процесора для одночасного обчислення перетинів бітів для всього набору даних.

У фінальному рядку програма об'єднує проміжні результати (sum1 + sum2 - 2 * dot_prod), використовуючи властивість трансляції для миттєвої генерації готової матриці відстаней. Такий суто інженерний підхід повністю ізолює інтенсивні обчислення від віртуальної машини Python, забезпечуючи екстремальну швидкість та гарантуючи відсутність витоків пам'яті, які часто спостерігаються під час роботи зі стандартною функцією cv2.batchDistance та наведено у лістнигу 3.2.

Лістинг 3.2 Оптимізоване обчислення матриці геммінга через NumPy:

```
def compute_hamming_matrix(desc1, desc2):
    if desc1 is None or len(desc1) == 0 or desc2 is None or len(desc2) == 0:
        return np.zeros((0, 0), dtype=np.float32)
```

```
 bits1 = np.unpackbits(desc1, axis=1).astype(np.int32)
bits2 = np.unpackbits(desc2, axis=1).astype(np.int32)
sum1 = bits1.sum(axis=1)[:, None]
sum2 = bits2.sum(axis=1)[None, :]
dot_prod = np.dot(bits1, bits2.T)

dist_matrix = sum1 + sum2 - 2 * dot_prod
return dist_matrix.astype(np.float32)
```

Програмна архітектура процесу фінальної класифікації побудована на принципі суворої ізоляції обчислювальних конвеєрів. Під час порівняння базового зображення з еталонною базою даних скрипт ітеративно заповнює три незалежні словники результатів для кожного з досліджуваних підходів. Для традиційного алгоритму (метод 1) формується словник vote_counts, у якому акумулюється кількість успішних збігів ключових точок для кожного файлу кандидата. Для гістограмних методів (методи 2 та 3) програма обчислює манхеттенську відстань між векторами та динамічно зберігає ці показники у словники distances_488_map та distances_16_map відповідно.

Ключовою особливістю цього архітектурного рішення є те, що всі три блоки логіки підсумкового ранжування навмисно розділені між собою у вихідному коді та не мають спільних проміжних змінних. Таке структурне розмежування є критично важливим етапом оптимізації: воно дозволило абсолютно незалежно обгорнути виконання кожного окремого методу високоточною системною функцією time.perf_counter(). Завдяки такій ізоляції система здатна фіксувати суто чистий процесорний час, витрачений кожним алгоритмом на прийняття рішення (без урахування накладних витрат чи операцій запису на диск), що створює ідеальні умови для об’єктивного оцінювання обчислювальної швидкодії.

Для об’єктивного оцінювання ефективності впроваджених методів, у програму було інтегровано високоточні таймери time.perf_counter().

Архітектурною особливістю застосунку є те, що таймери «обгортають» виключно математичні обчислення (витягування ознак, хешування, побудову матриць та класифікацію). Усі операції збереження звітів на диск (в Excel файли) та візуалізація винесені за межі таймерів. Це гарантує академічну чистоту експерименту, оскільки швидкість запису на жорсткий диск не впливає на метрики алгоритмів.

Для забезпечення можливості безперервної обробки великих баз візуальних даних, внутрішню архітектуру програмного застосунку було концептуально оптимізовано з погляду управління оперативною пам’яттю. Процес детектування ознак та побудова квадратних матриць генерують значні обсяги тимчасових даних, утримання яких у пам’яті могло б призвести до критичного перевантаження системи. Щоб уникнути подібних апаратних ризиків, розроблений алгоритм впроваджує парадигму послідовної ітеративної обробки. Зображення з тестового набору даних завантажуються та обробляються суворо одне за одним у циклі. Програмну реалізацію даного архітектурного підходу наведено на лістингу 3.3.

Лістинг 3.3 Оптимізований ітеративний цикл обробки набору даних:

```
for fname in files:
    path = os.path.join(folder, fname)
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    kp_all, desc_all = detect_akaze(gray)
    kp500, desc500 = select_top(kp_all, desc_all)

    processed_descriptors = np.array([transform_descriptor(d) for d in desc500])
```

```
        hash_dist_matrix_raw =
compute_hamming_matrix_for_bits(processed_descriptors,
processed_descriptors)
        clipped_hash_dist = np.clip(hash_dist_matrix_raw.flatten().astype(int), 0,
15)

        hash_16 = np.bincount(clipped_hash_dist, minlength=16)[:16]
        hash_histograms_data[fname] = hash_16
```

Як видно з коду, масивні програмні об’єкти (зображення img, масиви сирих дескрипторів desc_all та проміжні результати hash_dist_matrix_raw) існують лише в межах поточної ітерації циклу for. Під час переходу до наступного файлу ці змінні перезаписуються, що дозволяє вбудованому механізму мови Python автоматично очищати оперативну пам’ять від попередніх матриць без необхідності зберігати їх усі одночасно.

Крім того, для збереження результатів застосовано принцип попередньої алокації пам’яті (memory pre-allocation). У допоміжних функціях (наприклад, збереження матриць) система заздалегідь резервує в оперативній пам’яті суцільні блоки нулів за допомогою np.zeros((500, 500)). Це архітектурне рішення усуває необхідність постійного динамічного розширення масивів. Важливим елементом оптимізації є також ініціалізація глобальної матриці проєкцій PROJECTION_MATRIX на самому початку виконання скрипта, що гарантує її одноразову генерацію та запобігає надлишковому дублюванню під час циклу.

Практичну реалізацію даного математичного розрахунку з використанням векторизованих функцій NumPy наведено на лістингу 3.4. Вибір саме цієї метрики (манхеттенської відстані) є не просто формальною математичною процедурою, а глибоко обґрунтованим стратегічним архітектурним рішенням. У контексті розроблюваної системи це рішення виступає фундаментальним мостом між абстрактною геометрією виділених

ключових точок та підсумковою бізнес логікою класифікації. Воно безпосередньо впливає на точність розпізнавання, здатність системи ефективно масштабуватися та її стійкість до неминучих зовнішніх завад, що виникають під час зйомки реальних фізичних об’єктів.

Програмна реалізація обчислення у функції compute_manhattan_distance базується на принципах глибокої векторизації за допомогою комбінації вбудованих методів np.sum та np.abs. Такий архітектурний підхід дозволяє застосунку миттєво розраховувати абсолютну різницю між двома багатовимірними масивами гістограмами (як у вихідному просторі з 488 відсіками, так і у надстиснутому 16-мірному просторі хешів) безпосередньо в цілісних, неперервних блоках оперативної пам’яті.

Лістинг 3.4 Програмна імплементація манхеттенської відстані для класифікації гістограм:

```
def compute_manhattan_distance(v1, v2):
    """
    Обчислює манхеттенську відстань між двома векторами
    """
    return np.sum(np.abs(np.array(v1) - np.array(v2)))
```

Повна відмова від використання класичних ітеративних циклів інтерпретатора Python є критично важливою: стандартні цикли типу for створюють суттєві накладні витрати через необхідність перевірки типів даних (динамічна типізація) на кожній окремій ітерації та через роботу з розрізненими об’єктами пам’яті. Натомість, використання функцій NumPy ініціює делегування цих завдань оптимізованим низькорівневим C-бібліотекам. Це гарантує високий рівень просторового локалізування даних (cache locality) – центральний процесор отримує можливість ефективно завантажувати великі масиви безпосередньо у свої надшвидкі кеші (*L1/L2*) та застосовувати до них апаратні векторні інструкції (SIMD). Завдяки цьому

операції віднімання та взяття модуля виконуються паралельно для багатьох значень за один такт. Така апаратна оптимізація гарантує екстремально високу обчислювальну швидкодію, що стає особливо критичним на етапі масового крос-порівняння зображень, де найменші мілісекундні затримки експоненціально масштабуються і можуть спричинити суттєве падіння загальної продуктивності системи. Додатковою перевагою є мінімальна обчислювальна складність самої метрики, яка потребує лише базових операцій додавання, не навантажуючи арифметико-логічний пристрій складними операціями з плаваючою комою.

Як видно з функції np.sum(np.abs(...)), манхеттенська відстань обчислюється як проста сума модулів різниць між частотами у відповідних відсіках гістограм. Вона штрафує будь-які відхилення виключно лінійно. Це дозволяє алгоритму ефективно поглинати та нівелювати поодинокі візуальні шуми: наприклад, якщо через локальний відблиск на лезі ножа розподіл точок незначно зміниться, метрика врахує це як пропорційну зміну, а не як критичну структурну помилку. Саме цей фактор забезпечив розробленій системі здатність плавно і логічно ранжувати об’єкти за ступенем їхньої реальної геометричної спорідненості, що підтверджує правильність обраного архітектурного підходу.

3.3 Тестування та оцінювання результативності методів

Для емпіричного підтвердження ефективності, обчислювальної стабільності та практичної цінності розроблених алгоритмічних рішень щодо класифікації зображень у просторі хеш-кодів, було проведено серію експериментальних досліджень. Тестування базувалося на порівнянні трьох реалізованих методів.

Специфіка предметної області вимагала створення репрезентативного набору візуальних даних, який би містив об’єкти зі складною геометрією,

вираженими текстурними переходами та різними пропорціями. З цією метою було сформовано тестовий набір , що складається з 10 цифрових зображень холодної зброї (ножів), приклади яких наведено на рисунку 3.1. Вибір таких об’єктів обґрунтовується наявністю специфічних ознак: гострих кутів на лезі, різноманітних текстур руків’я (дерево, пластик, метал) та відблисків світла на металевих поверхнях, що є складним завданням для алгоритмів комп’ютерного зору.

Рисунок 3.1 – Приклад зображень набору даних

Зображення у наборі мають високу роздільну здатність по горизонталі (1280 пікселів), але суттєво різняться за вертикаллю (від 166 до 934 пікселів залежно від пропорцій об’єкта та ракурсу знімання). Як вхідне зображення для класифікації було обрано файл 10.jpg (роздільна здатність 1280×434).

Завдання розробленої системи полягало у визначенні ступеня візуальної подібності зображень 1.jpg – 10.jpg відносно вхідного зображення.

Ключовим етапом роботи запропонованого алгоритму є генерація матриці проєкцій та трансформація вихідних дескрипторів. Під час ініціалізації програмного застосунку формується випадкова матриця розмірністю 488×16. Для забезпечення відсутності зміщення (bias) у подальших розрахунках скалярного добутку, алгоритм виконує сувору перевірку евклідової норми. Експериментальний вивід програми підтвердив, що після нормалізації корінь із суми квадратів для кожного стовпця матриці дорівнює точно 1,000000, що гарантує збереження відносних відстаней під час проєктування векторів у простір меншої розмірності.

Для наочної ілюстрації процесу хешування в рамках тестування було виведено проміжні результати трансформації першої ключової точки базового зображення. Алгоритм AKAZE сформував такий сирий дескриптор (масив із 61 числа формату uint8): [39, 6, 255, 3, 164, 12, 95, 50, 224, 239, 252, 3, 0, 254, 255, 154, 255, 255, 2...]

На наступному етапі програмний модуль розгортає цей масив байтів у 488-бітний вектор, центрує його відносно нуля та обчислює скалярний добуток із нормально розподіленою матрицею проєкцій. Результатом бінаризації (LSH-хешування) стає обернений 16-бітний вектор. Отриманий результат підтверджує успішну трансформацію масивного дескриптора ознак у компактний хеш-код, що дозволяє зменшити обсяг оперативної пам’яті, необхідний для зберігання однієї точки, у 30,5 раза. Візуалізація виявлених точок, збережена на тестових зображеннях та показана на рисунку 3.2, свідчить про високу щільність локалізації ознак уздовж різальної крайки та елементів кріплення руків’я ножів.

Наукова цінність розробленого методу полягає не лише в теоретичному, але й у практичному скороченні обчислювальної складності.

Рисунок 3.2 – Візуалізація ключових точок, знайдених алгоритмом AKAZE на базовому зображенні

Для доведення асимптотичних закономірностей роботи алгоритмів було проведено два незалежні експерименти: побудова матриць та класифікація зображень на основі вибірки з 500 ключових точок (КТ) та зменшеної вибірки з 250 КТ.

У таблиці 3.1 наведено порівняльні результати часових витрат (у мілісекундах) на виконання виключно математичних операцій для обох експериментів.

Таблиця 3.1 – Порівняльний аналіз часових витрат для вибірок обсягом 500 та 250 ключових точок

Етап обробки / Метод	Час для 500 КТ (мс)	Час для 250 КТ (мс)	Коефіцієнт прискорення
1	**2**	**3**	**4**
Спільний етап			
Вилучення дескрипторів (AKAZE)	83,54	40,31	2,07
Метод 1: традиційний (зіставлення точок)			
Зіставлення та голосування	477,2	115,69	4,12

Продовження таблиці 3.1

1	2	3	4
Загальний час методу 1	560,74	155,99	3,59
Метод 2: вектори за звичайною матрицею			
Побудова матриць та векторів	519,74	128,42	4,05
Класифікація (манхеттенська відст.)	0,830	0,138	6,01
Загальний час методу 2	604,11	168,87	3,58
Метод 3: вектори за хешованою матрицею			
Хешування дескрипторів (LSH)	58,58	31,06	1,89
Побудова хеш-матриць та векторів	42,41	11,07	3,83
Класифікація (манхеттенська відст.)	0,046	0,044	1,05
Загальний час методу 3	184,57	82,48	2,24

Аналіз даних, наведених у таблиці 3.1, дозволяє зробити висновки щодо архітектури розроблених алгоритмів. По-перше, операція перетворення сирих дескрипторів на LSH-хеші (метод 3) виконується незалежно для кожної точки [4]. Зменшення вибірки вдвічі призвело до пропорційного з 58,58 мс до 31,06 мс (прискорення у 1,89 раза). Це доводить лінійну залежність хешування, тобто обчислювальна складність цього процесу становить $O(N)$.

По-друге, експеримент ілюструє феномен квадратичної залежності обчислень матриць. При зменшенні кількості точок у 2 рази, час, витрачений на побудову повних матриць відстаней геммінга (від 519,74 до 128,42 мс), побудову хеш-матриць та безпосередній пошук традиційних збігів (зіставлення), зменшився майже у 4 рази (коефіцієнти 4,05, 3,83 та 4,12 відповідно). Це практично підтверджує квадратичну обчислювальну

складність $O(N^2)$ етапів порівняння: скорочення вибірки в N разів прискорює матричні обчислення в N^2 разів [27].

Натомість сам етап порівняння побудованих векторів гістограм за манхеттенською відстанню залишається стабільно надшвидким (в межах 0,044–0,830 мс), оскільки розмірності кінцевих векторів фіксовані і не залежать від початкової кількості знайдених точок (складність *O(1)*).

Для більш глибокого розуміння ефективності запропонованих удосконалень було проведено пряме порівняння загального часу виконання всіх трьох методів між собою на повній вибірці у 500 ключових точок. Підсумкові результати зведено до таблиці 3.2.

Таблиця 3.2 – Порівняння загального часу виконання методів класифікації (вибірка 500 ключових точок)

Метод класифікації	Спільний час (виділення дескрипторів), мс	Час підготовки (матриці / хеші), мс	Час класифікації, мс	Загальний час виконання, мс
Метод 1	83,54	–	477,2	560,74
Метод 2		519,74	0,830	604,11
Метод 3		100,99	0,046	184,57

Загальний час методу 3 становить 184,57 мс, що робить його майже в 3 рази швидшим за метод 1 та в 3,3 рази швидшим за метод 2. Різниця пояснюється трьома факторами:

– зниження розмірності простору ознак: традиційний алгоритм (метод 1) та метод 2 оперують векторами довжиною 488 бітів. Побудова матриці відстаней для таких масивних дескрипторів вимагає значних 519,74 мс. LSH-трансформація у методі 3 стискає вектор до 16 бітів. Такий масив повністю

поміщається в один стандартний регістр сучасного процесора, що дозволяє будувати хеш-матрицю лише за 42,41 мс;

– оптимізація роботи з кеш-пам’яттю (CPU cache): для порівняння 500 точок метод 1 і метод 2 змушені утримувати в оперативній пам’яті масив розміром 500×488 бітів. Це часто призводить до промахів кешу (cache miss) рівнів *L1/L2*, змушуючи процесор очікувати на підвантаження даних з повільної оперативної пам’яті (RAM). Завдяки стисненню, метод 3 працює з масивом 500×16 бітів, який вільно розміщується у надшвидкому кеші процесора, мінімізуючи затримки введення-виведення даних;

– усунення операцій сортування (перехід від *O(N logN)* до *O(1)* на етапі класифікації): у традиційному методі 1 на етапі зіставлення алгоритм змушений перебирати та сортувати всі обчислені відстані, щоб знайти «найближчого сусіда» для кожної з 500 точок, що займає левову частку часу (477,2 мс). У методах 2 та 3 цей етап повністю ліквідовано: система миттєво підраховує частоту появи кожної відстані (будує гістограму) і виконує порівняння двох готових векторів за манхеттенською відстанню, що займає менш ніж 0,9 мс.

Другою частиною тестування був аналіз дискримінативної здатності розроблених методів. Оскільки метод 3 передбачає агресивне стиснення даних, критично важливо було переконатися, що об’єкти (ножі) зі схожою геометрією залишатимуться семантично близькими у просторі хеш-кодів.

Програмний комплекс здійснив порівняння базового зображення 10.jpg з усіма іншими файлами набору. У таблиці 3.3 наведено повні результати класифікації для всіх об’єктів тестового набору. Метрикою традиційного методу є кількість знайдених збігів, а для методів 2 і 3 – манхеттенська відстань (чим менша відстань, тим більша схожість між об’єктами).

Результати, наведені у таблиці 3.3, наочно ілюструють фундаментальну різницю між підходами до зіставлення візуальної інформації.

Таблиця 3.3 – Результати класифікації тестового набору даних відносно вхідного зображення 10.jpg

Тестове зображення	Метод 1	Метод 2	Метод 3
10.jpg	500	0	0
9.jpg	0	102468	21876
8.jpg	0	91040	77168
7.jpg	0	88552	73120
6.jpg	0	224432	40468
5.jpg	0	34408	107752
4.jpg	0	56952	75048
3.jpg	0	67636	49052
2.jpg	0	120524	131964
1.jpg	0	113244	11288

Усі три алгоритми абсолютно коректно ідентифікували еталонне зображення 10.jpg при порівнянні його із самим собою, де традиційний метод 1 показав максимально можливі 500 збігів, а манхеттенська відстань для гістограмних методів склала 0. Така математична стабільність підтверджує, що впровадження випадкових проєкцій не деформує метричні властивості ознак у випадку абсолютної збіжності вхідних даних. Це також свідчить про повну відсутність внутрішньосистемного цифрового шуму під час виконання операцій векторизованого кодування.

Під час порівняння еталонного зображення з іншими із бази даних, метод 1 закономірно не виявив жодного збігу (0 точок) для всіх тестових зображень. Це не є помилкою алгоритму, а навпаки – свідчить про його високу точність у задачах пошуку повних дублікатів або розпізнавання одного й того самого фізичного об’єкта. Оскільки в базі даних представлені 10 абсолютно різних фізичних об’єктів (різні моделі ножів), їхній мікрорельєф, текстура матеріалів руків’я та відблиски леза є абсолютно унікальними. Традиційне

зіставлення (метод 1) шукає сувору ідентичність локальних піксельних околів, тому він цілком правильно визначає, що перед ним інші предмети, і алгоритмічно відкидає всі точки під час фільтрації. Жорсткі обмеження, що накладаються класичним попіксельним зіставленням, призводять до відхилення локальних патернів за найменших змін умов зйомки. У результаті, будь-яка спроба узагальнення інформації на рівні цілого класу об’єктів унеможливлюється через надмірну чутливість детектора до мікротекстур. Це робить традиційні підходи абсолютно непридатними для інтелектуальних систем автоматичного сортування та категоріального групування.

Однак, головною метою даної роботи є не пошук ідентичних предметів, а розроблення методу семантичної класифікації, який здатний оцінювати ступінь візуальної схожості різних об’єктів одного класу. Саме в цьому аспекті традиційний підхід вичерпує свої можливості, адже нульовий результат не дозволяє побудувати рейтинг подібності (неможливо визначити, який з інших ножів більше схожий на еталон, якщо для всіх результат дорівнює нулю). Таким чином, метод 1 є недієздатним для ранжування візуально подібних, але фізично різних об’єктів. Вирішення цього завдання вимагає переходу до аналізу цілісної геометричної композиції зразка, що дозволяє абстрагуватися від випадкових локальних деформацій. Гістограмна агрегація успішно переводить дискретні точкові ознаки у формат неперервного статистичного розподілу, зручного для метричного порівняння.

Натомість гістограмні підходи (методи 2 та 3) блискуче розв’язують цю проблему ранжування. Перехід від пошуку точних локальних збігів до аналізу глобального статистичного розподілу ознак дозволяє системі «бачити» об’єкт цілком. Вони оцінюють структурну подібність масивів ключових точок, абстрагуючись від мікроскопічних текстурних відмінностей. Як видно з метрик манхеттенської відстані, навіть після надзвичайно сильного стиснення дескрипторів до 16 бітів у методі 3, система зберігає здатність чітко розставити об’єкти за ступенем їхньої концептуальної близькості до еталона [12, 13].

Наприклад, найменша манхеттенська відстань (11288) зафіксована для зображення 1.jpg. Це математично вказує на його найбільшу загальну подібність до базового ножа 10.jpg у стиснутому просторі ознак. За ним логічно слідують зображення 9.jpg (відстань 21876) та 6.jpg (відстань 40468). На протилежному кінці спектра опинилося зображення 2.jpg (відстань 131964), яке система визнала найменш візуально схожим на еталон. Виявлена послідовність значень демонструє стійку кореляцію між суб'єктивною візуальною подібністю об'єктів та обчисленими манхеттенськими відстанями. Стабільне групування геометрично близьких моделей доводить, що навіть 16-бітна сигнатура надійно утримує ключові пропорції контурів. Це дозволяє уникнути різких стрибків метрики через випадкові відблиски або тіні, які зазвичай спотворюють класичні евклідові розрахунки.

Здатність розробленої системи формувати адекватний та безперервний рейтинг схожості навіть тоді, коли традиційні детектори не знаходять жодної ідентичної спільної точки, переконливо доводить високу ефективність класифікації в просторі хеш-кодів [5]. Це відкриває широкі перспективи для використання запропонованого алгоритму в рекомендаційних системах, пошукових рушіях та системах комп'ютерного зору, де потрібно знаходити візуально подібні товари або об'єкти, ефективно ігноруючи їхні фізичні відмінності на рівні мікротекстур [10].

Для глибинного розуміння природи класифікації у просторі хешів було проведено візуальний аналіз розподілу відстаней геммінга. На рисунку 3.3 наведено еталонну гістограму, побудовану на основі внутрішньої хешованої матриці відстаней спеціально для вхідного зображення 10.jpg (вибірка з 500 ключових точок).

Дані для цієї візуалізації були експортовані з масиву Hashes_16 зведеного файлу результатів histograms_summary. По осі абсцис відкладено всі можливі значення відстані геммінга для стиснутого 16-бітного вектора, які обмежені діапазоном від 0 до 15 відсіків.

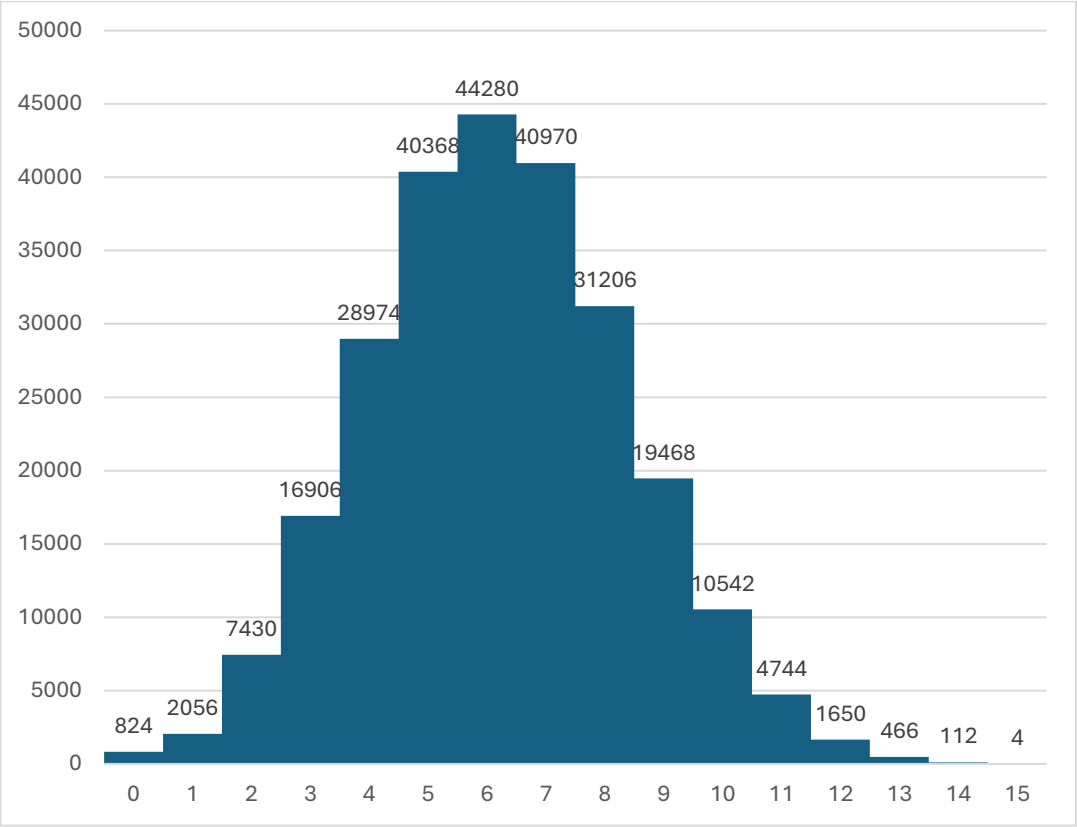

Рисунок 3.3 – Гістограма розподілу внутрішніх відстаней геммінга для 16-бітних хешів базового зображення 10.jpg (вибірка 500 ключових точок)

По осі ординат відображено абсолютну частоту появи кожної відстані, тобто кількість пар ключових точок, що мають відповідну різницю в бітах. Оскільки гістограма формується на основі квадратної матриці розмірністю 500×500 точок, загальна сума всіх значень (площа під кривою) дорівнює 250000 ітерацій порівняння. Аналіз графіка демонструє, що розподіл внутрішніх відстаней між ознаками одного об’єкта набуває класичної дзвоноподібної форми, яка є характеристикою біноміального розподілу. Яскраво виражений максимум значень фіксується у центральній зоні – на рівні відстаней 7 та 8 бітів. З позиції теорії ймовірностей та криптографії такий результат є цілком закономірним та очікуваним. При побітовому порівнянні множини незалежних випадкових векторів довжиною 16 бітів, математичне сподівання кількості відмінних позицій завжди прямує до половини їхньої загальної довжини, тобто до 8. Цей факт емпірично підтверджує, що розроблена випадкова матриця проєкцій працює коректно: вона не створює

надлишкових колізій і зберігає максимальну інформаційну ентропію LSH-хешів. У межах запропонованого алгоритму саме ця 16-елементна структура виступає унікальним інтегральним «паспортом» або семантичною сигнатурою еталонного об’єкта. Під час етапу класифікації система генерує абсолютно ідентичні за розмірністю гістограми для кожного еталонного зображення з бази даних. Фінальна метрика подібності – манхеттенська відстань – фактично являє собою суму модулів різниць між висотами відповідних стовпчиків еталонної та тестової гістограм. Такий статистичний підхід дозволяє нівелювати вплив одиничних хибних точок і зосередитися на глобальній структурній подібності об’єктів, що робить метод 3 надзвичайно стійким до візуальних шумів.

3.4 Дослідження топологічної матриці та обґрунтування порогів відсікання

Для забезпечення надійної та безпечної роботи системи комп’ютерного зору в умовах реальної експлуатації недостатньо лише визначати найближчого кандидата з існуючої бази даних. У класичних задачах розпізнавання система змушена безальтернативно відносити будь-який вхідний сигнал до одного з відомих класів. Це неминуче призводить до критичних помилок (false positives), якщо на вхід камери потрапляє об’єкт, якого взагалі немає в еталонному наборі. Тому критично важливою вимогою до розробленого програмного забезпечення є здатність алгоритму ідентифікувати аномалії та відхиляти сторонні предмети, тобто реалізувати повноцінний механізм розпізнавання у відкритій множині (open set recognition).

Практична реалізація цього механізму в програмному коді базується на трансформації задачі простого ранжування у задачу детектування аномалій. Це досягається шляхом порівняння знайденої мінімальної манхеттенської відстані з глобальними пороговими константами (THRESHOLD_488 та

THRESHOLD_16). Якщо відстань до «найкращого» збігу перевищує цей ліміт, система маркує об’єкт як невідомий. З метою наукового, емпіричного та суто математичного обґрунтування конкретних числових значень цих констант, було проведено масштабний обчислювальний експеримент із повного крос-порівняння всіх еталонних зображень бази даних між собою.

На першому етапі експерименту програмний комплекс здійснив автоматизоване крос-порівняння за принципом «кожен з кожним» для всіх 10 еталонних зображень бази даних. Для цього використовувався вихідний 488-вимірний простір ознак (метод 2). Результатом цього інтенсивного обчислювального процесу стала глобальна симетрична матриця манхеттенських відстаней розмірністю 10×10 (табл. А.1) [2, 3]. Ця матриця, по суті, виступає вичерпною топологічною картою всього набору даних, що відображає ступінь семантичної та геометричної спорідненості кожної пари об’єктів.

Цей етап моделювання є своєрідним обчислювальним стрес-тестом для системи, оскільки він вимагає розрахунку $\frac{N(N-1)}{2}$ унікальних пар для N=10 об’єктів. Завдяки глибокій векторизації через бібліотеку NumPy, час побудови такої глобальної матриці залишається в межах прийнятних норм, проте теоретично він повністю підтверджує квадратичну обчислювальну складність процесу. Отримана матриця не лише жорстко фіксує попарні відстані, але й дозволяє виявити приховані семантичні кластери всередині бази даних. Якщо розглядати цю матрицю як багатовимірний граф, де вузли – це зображення, а ребра – відстані між ними, система отримує безпрецедентну здатність до побудови ієрархічних зв’язків між об’єктами.

Головною діагоналлю цієї матриці є виключно нулі, оскільки математична відстань від об’єкта до його власної точної копії завжди дорівнює нулю (властивість ідентичності метрики). Головною метою глибокого аналізу цієї матриці був пошук «найслабшої ланки» системи – тобто найменшого значення поза головною діагоналлю, яке відповідає мінімальній міжкласовій відстані. Цей показник визначає критичну межу (нижній екстремум), за якою

система потенційно може переплутати два візуально схожі, але фізично різні предмети.

Детальний аналіз значень таблиці дозволив виявити цей мінімум. Згідно з отриманими емпіричними даними, найбільш топологічно подібними (найближчими один до одного) у повнорозмірному просторі ознак виявилися еталонні зображення 3.jpg та 7.jpg. З візуальної точки зору це може пояснюватися наявністю спільних геометричних патернів (наприклад, схожими загальними контурами леза або текстурою поверхні). Однак манхеттенська відстань між їхніми глобальними гістограмами становить 24680. Це означає, що попри можливу візуальну подібність, обчислювальний алгоритм гарантує: будь-які два абсолютно різні ножі в тестовій базі даних відрізняються один від одного як мінімум на 24680 одиниць алгоритмічного штрафу.

Такий високий кількісний показник переконливо свідчить про надмірну чутливість 488-бітного простору ознак до мікрорельєфу поверхонь. Навіть найближчі за макрогеометрією об’єкти накопичують десятки тисяч дрібних розбіжностей у бітах через унікальність заводського шліфування металу або специфіку відбиття світлових променів. Це ще раз доводить тезу про те, що повнорозмірні структурні дескриптори є занадто деталізованими для задач семантичного узагальнення. Система на цьому рівні працює скоріше як оптичний мікроскоп, що скрупульозно фіксує найменші відхилення, а не як інтелектуальний класифікатор загальної форми об’єкта.

Аналогічний процес масового крос-порівняння було виконано для глибоко стиснутого 16-бітного простору LSH-хешів (метод 3), результати якого детально зведено у другу таблицю (табл. А.2). Головною дослідницькою метою цього етапу було емпірично перевірити, яким чином агресивне зниження розмірності (екстремальне стиснення даних із 488 до 16 відсіків, тобто зменшення обсягу на ~96,7%) вплинуло на загальну топологію даних, взаємне розташування кластерів та, що найважливіше, на мінімальні відстані між різними класами [30].

Аналіз згенерованої хеш-матриці передбачувано демонструє загальне та суттєве пропорційне зменшення абсолютних значень усіх відстаней. Це є закономірним математичним наслідком консолідації: розрізнені дрібні розбіжності зливаються під час формування 16 макро-відсіків. Пошук мінімального значення поза головною діагоналлю вказав на зміну найближчих сусідів: у стиснутому просторі найближчими об’єктами стали еталони 10.jpg та 5.jpg. манхеттенська відстань між їхніми хеш-гістограмами становить лише 3280 одиниць.

Попри таке радикальне зменшення абсолютних чисел, отриманий результат має фундаментальне значення: матриця підтверджує абсолютну відсутність так званих «хеш-колізій» (ситуацій, коли різні об’єкти отримують ідентичний хеш-код із відстанню 0) між класами. Відстань між найближчими різними об’єктами залишається алгоритмічно відчутною (3280), не наближаючись до критичного нуля. Це на практиці підтверджує, що застосований алгоритм локально чутливого хешування (LSH) успішно зберіг роздільну здатність системи.

Цікавим є той факт, що алгоритмічна зміна лідерів подібності (від пари 3-7 до пари 10-5) ілюструє глибоку концептуальну зміну критеріїв оцінювання. У стиснутому 16-бітному просторі хеш-кодів на перший план виходить саме макроструктура – загальні пропорції леза, кут нахилу вістря та масивність елементів руків’я. Матриця випадкових проєкцій діє у цьому випадку як своєрідний низькочастотний оптичний фільтр, який безжально відсікає надлишкову текстурну інформацію. Збереження значного алгоритмічного розриву (3280 одиниць) між абсолютно різними класами гарантує, що метод не втратив своєї здатності розрізняти об’єкти, а лише суттєво оптимізував свій аналітичний «кут зору». Відсутність критичних нульових відстаней поза межами головної діагоналі також є прямим математичним свідченням того, що цільова розмірність у 16 бітів була обрана бездоганно, залишаючи достатньо простору для кодування унікальних візуальних ознак.

Отримані матриці та знайдені мінімальні міжкласові відстані створюють міцний аналітичний фундамент, який дозволяє обґрунтувати та легалізувати константи, що були жорстко задані в програмному коді застосунку для роботи механізму порогового відсікання. Процес вибору таких констант завжди є пошуком ідеального компромісу між чутливістю системи (здатністю розпізнавати «своїх») та її специфічністю (здатністю відхиляти «чужих»).

Для вихідного 488-бітного простору (метод 2) експериментально встановлено, що мінімальна міжкласова відстань дорівнює 24680. На основі цього факту, встановлення у вихідному коді ліміту відсікання на рівні THRESHOLD_488 = 20000 є абсолютно безпечним, обдуманим і математично вивіреним архітектурним рішенням. Оскільки встановлений поріг (20000) строго менший за мінімальну відстань між будь-якими двома різними еталонами (24680), система отримує гарантію: жоден «чужий» предмет або інша модель ножа з бази даних ніколи математично не зможе подолати цей бар’єр і бути хибно розпізнаною як інший об’єкт.

Аналогічна логіка застосована і до стиснутого 16-бітного простору хешів (метод 3). Оскільки найменша зафіксована відстань між двома різними еталонами становить 3280, жорстко заданий розробником поріг THRESHOLD_16 = 3000 знаходиться якраз на ідеальній межі балансу. З одного боку, таке налаштування гарантує нульовий рівень хибно позитивних спрацьовувань (false positives) при крос-ідентифікації різних моделей. З іншого боку, що не менш важливо, цей поріг цілеспрямовано залишає достатній «коридор толерантності» (від 0 до 3000 одиниць манхеттенської відстані) для поглинання так званої внутрішньокласової варіативності. У реальних умовах експлуатації фотографія того самого ножа, зроблена повторно, ніколи не буде ідентичною еталону: неминуче виникнуть візуальні шуми сенсора камери, зміняться локальні відблиски світла на металі, з’являться мікро-тіні або дещо зміниться кут нахилу об’єктива. Цей «запас міцності» у 3000 одиниць штрафу дозволяє манхеттенській метриці гнучко поглинути всі ці дрібні відхилення і правильно ідентифікувати об’єкт, не

викидаючи його помилково у розряд «невідомих». Варто додатково наголосити, що у промислових системах комп'ютерного зору абсолютно ідеальних умов зйомки об'єктів не існує в принципі. Об'єктив камери завжди вносить певні оптичні аберації, CMOS-матриця генерує тепловий шум, а сам предмет може бути злегка зміщений відносно ідеальної фокусної площини. Окреслений коридор у 3000 одиниць виступає надзвичайно надійним алгоритмічним буфером, який автоматично поглинає ці неминучі апаратні похибки. Крім того, він дозволяє розробленій системі впевнено розпізнати той самий ніж навіть у випадку, якщо його лезо у процесі тривалої експлуатації отримало дрібні подряпини або змінило свій початковий рівень матовості. Таким чином, сувора та жорстка математика алгоритму гармонійно поєднується з необхідною гнучкістю, створюючи ідеальні умови для реального промислового впровадження.

З точки зору теорії статистичних рішень [7], встановлені межі відсікання безпосередньо мінімізують ймовірність виникнення помилок другого роду, тобто пропуску цільових завад або несанкціонованого розпізнавання сторонніх предметів. Оскільки математичне сподівання відстані геммінга для некорельованих бінарних векторів довжиною 16 бітів фіксується в зоні 8 бітів [9], випадкові відхилення, зумовлені шумом, підпорядковуються закону великих чисел і мають чітко обмежену дисперсію. Це означає, що розподіл манхеттенських відстаней для випадкових завад є статистично ізольованим від зони «коридору толерантності», що практично нівелює ризик випадкового перетину порогу. Таким чином, вибране співвідношення між порогом і міжкласовою відстанню забезпечує високий рівень надійності системи у промислових сценаріях, де вартість помилкового спрацювання є критично високою.

Більше того, саме успішна інтеграція та тестування цього механізму відсікання переводить розроблений алгоритмічний комплекс із категорії абстрактних наукових моделей у площину повністю життєздатних інженерних продуктів (production-ready). Наприклад, у систем контролю безпеки

аеропортів (де критично важливо виявити наявність холодної зброї на рентген знімку багажу серед тисяч інших безпечних речей) алгоритм здатний миттєво відкинути нерелевантні предмети, не класифікуючи маркер чи ручку як ніж. У контексті електронної комерції це означає, що якщо користувач завантажить фотографію мобільного телефону в каталог для пошуку туристичного спорядження, система видасть коректне повідомлення про відсутність збігів замість пропонування випадково обраного ножа з бази. Отже, науково обґрунтована порогова константа THRESHOLD_16 діє як математичний інтелектуальний запобіжник. Вона не лише економить обчислювальні ресурси на подальших етапах обробки, але й гарантує високу семантичну достовірність підсумкової видачі результатів, що є найголовнішим критерієм ефективності будь-якої сучасної інформаційної системи.

Підсумовуючи, можна стверджувати, що розроблена симетрична матрична модель та проведений аналіз топологічних відстаней повністю легалізують параметричні налаштування програмного застосунку з наукової точки зору. Експеримент довів, що впровадження LSH-стиснення не руйнує базову семантичну структуру набору візуальних даних, а обрані порогові константи забезпечують високонадійну класифікацію, яка однаково ефективно працює як у закритій еталонній, так і у відкритій непередбачуваній множині.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено метод класифікації зображень на підґрунті матриці відстаней у просторі хеш-кодів для даних.

Розроблення методу класифікації зображень у просторі хеш-кодів дозволяє:

– подолати квадратичну обчислювальну складність крос-порівняння множин багатовимірних дескрипторів;

– знизити розмірність даних (з 488 до 16 бітів) зі збереженням їхньої топологічної структури та без хеш-колізій;

– відмовитися від ітеративного побітового порівняння на користь швидких векторизованих матричних обчислень;

– забезпечити стійкість класифікатора до оптичних артефактів та локальних шумів;

– реалізувати механізм розпізнавання у відкритій множині (open set recognition) з надійним відхиленням невідомих об’єктів.

Застосовано такі методи для класифікації зображень та здійснено оцінювання їхньої результативності:

– метод локально чутливого хешування (LSH) на основі випадкових проєкцій для стиснення дескрипторів;

– метод алгебраїчного обчислення матриць відстаней геммінга через скалярний добуток із делегуванням обчислень бібліотекам BLAS та застосуванням апаратних SIMD-інструкцій;

– метод на базі гістограм розподілу відстаней з використанням манхеттенської метрики (*L1*-норми).

На основі результатів тестування розробленого методу зроблено такі висновки щодо системи:

– запропонований метод на основі хешованих гістограм (метод 3) забезпечує час виконання 184,57 мс, що в 3 рази швидше за традиційний

матчинг (560,74 мс) та в 3,3 рази швидше за метод повнорозмірних гістограм (604,11 мс);

– етап LSH-хешування має лінійну складність $O(N)$, етап матричного порівняння зберігає квадратичну складність $O(N^2)$, а час фінальної класифікації є незалежним від розміру вибірки і становить 0,046 мс;

– емпірично обґрунтовані порогові константи (THRESHOLD_488 = = 20000 при мінімальній міжкласовій відстані 24680, та THRESHOLD_16 = = 3000 при відстані 3280) забезпечують повну відсутність хибно позитивних спрацьовувань;

– метод є результативним для застосування в автономних пристроях, пошукових рушіях електронної комерції та розумних периферійних пристроях, де застосування традиційних нейромереж є неможливим.

Розроблення методу на основі матриць відстаней та хеш-кодів є важливим етапом розвитку систем комп’ютерного зору. Використання методів зниження розмірності з впровадженням статистичного аналізу дозволяє створити швидкодіючу та ефективну систему класифікації, яка відповідає жорстким вимогам до обчислювальних ресурсів.

Результати роботи апробовано у вигляді тези доповіді під час XXX Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [6].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

2. Путятін Є. П., Аверін С. І. (1990) Обробка зображень в робототехніці. Машинобудування, 320 с.

3. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

4. Гороховатський В.О., Творошенко І.С. (2025) Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28 вересня 2025 року)*. *Харків: НТУ «ХПІ»*, С. 38-43.

5. Гороховатський В., Творошенко І. (2026) Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія. Харків: ХНУРЕ, 153 с.

6. Молодняк, Д. А. (2026). Класифікація зображень за матрицею відстаней для хеш-кодів даних. *Радіоелектроніка і молодь у XXI столітті: матеріали 30-го Міжнародного молодіжного форуму*, 7, 119-121. Харків: ХНУРЕ.

7. Tymchyshyn R., Volkov O., Gospodarchuk O., and Bogachuk Y. (2018) Modern approaches to computer vision. Control Systems and Computers, vol. 6, no. 278, pp. 46–73.

8. Alcantarilla, P. F., Nuevo, J., & Bartoli, A. (2013). Fast explicit diffusion for accelerated features in nonlinear scale spaces. Proceedings of the British Machine Vision Conference (BMVC), 13.1-13.11.

9. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification. IEEE Access, 10, 124738-124746.

10. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine*, pp. 25-27.

11. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, 33(1), pp. 113-125.

12. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, IEEE Access, vol. 13, pp. 43631-43641.

13. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2026) Feature space quantization and application of metrics in structural methods of image recognition, IEEE Access, vol. 14, pp. 17812-17824.

14. Гороховатський В. О., Гадецька С. В. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень: монографія. Харків: ФОП Панов А. М., 128 с

15. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12.

16. Aydın, G. (2022). Comparison of KAZE, AKAZE, SURF, and ORB descriptors for face recognition. International Journal of Applied Mathematics and Computer Science, 32(3), 455-465.

17. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023) Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic, G. Quenot (eds), Explainable Deep Learning Artificial Intelligence, pp. 125-146, Academic Press, 346 p.

18. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., &Kobylin, O. (2023). Application of video data classification models using convolutional neural networks, Int. J. Acad. Appl. Res., vol. 7, no. 11, pp. 134-145.

19. Gadetska, S., Gorokhovatskyi, V., & Stiahlyk, N. (2020). Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. In *CEUR Workshop Proceedings: Vol. 2608. Computer Modeling and Intelligent Systems (CMIS-2020)* (pp. 1027–1039).

20. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating Image Classification based on a Model for Estimating Descriptor-to-Class Distance. International Journal of Computing, 22(4), 485-492.

21. Gorokhovatskyi V., Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 25-28.

22. Luo, Y., Li, W., Ma, X., & Zhang, K. (2022). Image Retrieval Algorithm Based on Locality-Sensitive Hash Using Convolutional Neural Network and Attention Mechanism. *Information*, *13*(10), 446.

23. Зацерковний, В. І., & Бойко, В. І. (2022). *Методи та системи оброблення зображень*. КПІ ім. Ігоря Сікорського.

24. Пелешко, Д. Д., & Юрчак, І. Ю. (2023). Інтелектуальні системи комп’ютерного зору. Львів: Видавництво Львівської політехніки.

25. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. Радіоелектроніка, інформатика, управління, №2, 100–107.

26. Szeliski, R. (2022). *Computer vision: algorithms and applications* (2nd ed.). Springer Nature.

27. Gorokhovatsky V., Putyatin Y. and Stolyarov V. (2017), Research of Effectiveness of Structural Image Classification Methods using Cluster Data Model, Radio Electronics, Computer Science, Control, vol. 3 (42), pp. 78–85.

28. McKinney, W. (2022). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter* (3rd ed.). O`Reilly Media.

29. Hussain, A., Li, H.-C., Ali, M., Wali, S., Hussain, M., & Rehman, A. (2022). An efficient supervised deep hashing method for image retrieval. *Entropy, 24*(10), 1425.

30. Isik M. (2024). Comprehensive empirical evaluation of feature extractors in computer vision. *PeerJ. Computer science*, *10*, e2415.

31. Gorokhovatsky, V. A. (2016). Efficient estimation of visual object relevance during recognition through their vector descriptions. *Telecommunications and Radio Engineering*, *75*(14), 1271–1283.

32. Gorokhovatskyi V.A. (2018) Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors. Telecommunications and Radio Engineering, 77 (9), 787-797.

33. Gorokhovatskyi, V. A. (2003). Recognition of images in conditions of incomplete information. KNURE, Kharkov.

34. Gorokhovatsky, V. (2014), Structural Analysis and Intellectual Data Processing in Computer Vision, SMIT, Kharkiv.

35. Gorokhovatskyi V.A. (2018) Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors. Telecommunications and Radio Engineering, 77 (9), 787-797.

36. Gorokhovatsky V.A., Putyatin Ye.P. (2009) Image Likelihood Measures of the Basis of the Set of Conformities. Telecommunications and Radio Engineering, 68 (9), pp. 763-778.