Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

МОДЕЛЮВАННЯ КЛАСИФІКАТОРА ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ АПАРАТУ ВИПАДКОВОГО ХЕШУВАННЯ ДАНИХ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Приймак Є. 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 Приймаку Єгору Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання класифікатора зображень з використанням апарату випадкового хешування 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 науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, еталонна база растрових зображень морських ссавців, математичні моделі алгоритму нелінійної дифузії AKAZE та локально-чутливого хешування, мова програмування Python, бібліотеки комп'ютерного зору та матричних обчислень OpenCV і NumPy.

4. Перелік питань, що потрібно опрацювати в роботі

1. Провести аналіз сучасних методів структурної класифікації зображень. Оглянути методи випадкового проєктування та подолання «прокляття розмірності».

2. Описати апарат локально-чутливого хешування та моделі адитивного зашумлення.

3. Обґрунтувати вибір інструментальних засобів. Розробити архітектуру та програмну реалізацію багатокласового класифікатора з графічним інтерфейсом.

4. Провести експериментальне тестування методів класифікації.

5. Оцінити швидкодію, стійкість до шуму та розрізнювальну здатність методів у просторах різної розмірності. Зробити висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри). узагальнена структурна схема системи розпізнавання зображень, ілюстрації процесу екстракції ознак із застосуванням еліптичного маскування, діаграми оцінювання швидкодії просторового хешування, матриці взаємної сумісності та результати дослідження стійкості алгоритмів до впливу гаусівського шуму, висновки.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

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	20.04.26-26.04.26	
5	Формування кроків вирішення проблеми	23.04.26-08.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-30.05.26	
8	Оформлення пояснювальної записки	31.05.26-09.06.26	
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: 81 с., 12 табл., 21 рис., 35 джерел.

ВИПАДКОВЕ ХЕШУВАННЯ, ГАУСІВСЬКИЙ ШУМ, ДЕТЕКТОР AKAZE, КОМП'ЮТЕРНИЙ ЗІР, МАНХЕТЕНСЬКА ВІДСТАНЬ, ПРОСТІР ОЗНАК, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ.

Об'єкт роботи – методи розпізнавання та класифікації зображень у системах комп'ютерного зору.

Мета роботи – підвищення швидкодії методів розпізнавання зображень шляхом застосування випадкового хешування.

Методи при виконанні роботи: комп'ютерний зір, лінійна алгебра, теорія ймовірностей, імітаційне моделювання, Монте-Карло.

Отримані результати: розроблено результативну модель класифікатора зображень з використанням випадкового хешування. Доведено, що 16-бітний простір із методом інтеграції голосів прискорює пошук у 230 разів порівняно з лінійним пошуком. Впровадження еліптичного маскування усуває 70% фонових завад. Підтверджено високу точність розпізнавання в умовах впливу гаусівського шуму.

Галузь застосування: системи екологічного моніторингу та швидкого пошуку в колекціях зображень.

AKAZE, COMPUTER VISION, FEATURE SPACE, GAUSSIAN NOISE, IMAGE RECOGNITION, LOCALITY-SENSITIVE HASHING, MANHATTAN DISTANCE.

The object of the work is the process of image recognition and classification in computer vision systems.

The purpose of the work is to improve the speed and noise robustness of classification systems using locality-sensitive hashing (LSH).

Methods of the work: computer vision, linear algebra, probability theory, Monte Carlo simulation.

Obtained results: a classifier model based on AKAZE and LSH was developed. It is proven that a 16-bit space with the vote integration method accelerates the search by 230 times compared to the brute force method. Elliptical masking eliminates 70% of background interference. 100% recognition accuracy under Gaussian noise conditions was confirmed.

Field of application: environmental monitoring systems and fast search in databases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Гб – гігабайт

мс – мілісекунди

БД – база даних

AKAZE – Accelerated-KAZE (алгоритм прискореного витягування ознак у нелінійному просторі масштабів)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

GHz – Gigahertz (гігагерц)

GPU – Graphics Processing Unit (графічний процесор)

GUI – Graphical User Interface (графічний інтерфейс користувача)

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

M-LDB – Modified Local Difference Binary (модифікований локальний бінарний дескриптор різниць)

ROC – Receiver Operating Characteristic (робоча характеристика приймача)

SIFT – Scale-Invariant Feature Transform (масштабоінваріантна трансформація ознак)

SURF – Speeded-Up Robust Features (прискорені робастні ознаки)

XOR – Exclusive OR (операція виключного АБО)

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким накопиченням та необхідністю високошвидкісного оброблення великих масивів візуальної інформації. Системи комп'ютерного зору наразі активно інтегруються у різноманітні сфери життєдіяльності, зокрема у підсистеми екологічного моніторингу навколишнього середовища, модулі автономної навігації робототехнічних та безпілотних пристроїв, а також у сервіси швидкого пошуку графічних об'єктів у великих базах даних [1-4]. Основним викликом при побудові таких систем є забезпечення стабільно високої швидкодії за умов суттєвого збільшення обсягів оброблюваних графічних даних. Вирішення цього завдання класичними методами прямого перебору дескрипторів ознак призводить до критичного уповільнення роботи обчислювальних пристроїв через ефект «прокляття розмірності», коли складна часова та просторова обчислювальна складність алгоритмів зростає пропорційно загальному розміру баз даних [2].

Аналіз показує, що хоча сучасні нейромережеві моделі забезпечують високу точність розпізнавання, вони вимагають надлишкових ресурсів і спеціалізованого апаратного забезпечення (GPU), що обмежує їхнє впровадження на портативних та бортових платформах. Раціональною альтернативою є структурні методи на основі екстракції локальних ключових точок. Зокрема, використання алгоритму AKAZE дозволяє отримати геометрично стійкі ознаки, проте етап зіставлення багатовимірних дескрипторів залишається обчислювально важким. Ефективним інженерним рішенням для подолання цього бар'єра є застосування математичного апарату локально-чутливого хешування (LSH). Він дозволяє проєктувати багатовимірні описи у компактні бінарні сигнатури, замінюючи важкі розрахунки лінійних відстаней на швидкі побітові операції. Саме моделюванню завадостійкого та швидкодіючого класифікатора на основі випадкових проєкцій присвячена дана кваліфікаційна робота.

Для даної роботи обрано методи розпізнавання та класифікації зображень у системах комп’ютерного зору, а її головною метою є підвищення швидкодії та завадостійкості систем класифікації шляхом застосування локально-чутливого хешування. Для безпосереднього досягнення цієї мети у роботі послідовно вирішено низку прикладних завдань, що включають проведення аналізу сучасних методів екстракції візуальних ознак, дослідження математичної моделі нелінійного детектора AKAZE та теорії випадкових проєкцій, розроблення загальної архітектури програмного комплексу класифікації, практичну реалізацію методів пошуку у корзинах, інтеграції голосів та порівняння сигнатур, а також проведення експериментального тестування стійкості системи до впливу адитивного гаусівського шуму.

Методологічною основою вирішення поставлених завдань виступають методи комп’ютерного зору для локалізації ключових точок, апарат лінійної алгебри для оптимізації матричних обчислень, теорія ймовірностей для опису колізій LSH та гаусівських завад, а також імітаційне моделювання для оцінювання апаратної стабільності. Особливістю запропонованого у роботі інженерного підходу є вдосконалення процесу класифікації шляхом розроблення методу інтеграції голосів у 16-бітному просторі LSH, що, на відміну від простих бінарних порівнянь, забезпечує високу впевненість прийняття рішень за умов інтенсивного зашумлення без втрати швидкодії [3].

Практичне значення отриманих результатів полягає у розробленні готового програмного комплексу із дружнім графічним інтерфейсом користувача на базі мови Python та бібліотек OpenCV, NumPy і tkinter. Створений застосунок реалізує кілька аналітичних режимів роботи і може бути безпосередньо використаний як вбудований програмний модуль у промислових системах візуального моніторингу, підсистемах навігації безпілотних пристроїв та сервісах швидкого пошуку схожих графічних об’єктів у великих базах даних.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ХЕШУВАННЯ

1.1 Аналіз сучасних підходів до класифікації зображень

Стрімкий розвиток сучасних інформаційних систем, орієнтованих на автоматизацію процесів прийняття рішень у складних, динамічних та апріорі невизначених середовищах, безпосередньо пов’язаний із прогресом у галузі комп’ютерного зору, цифрової обробки сигналів та штучного інтелекту. Завдання автоматичного розпізнавання, ідентифікації та багатокласової класифікації статичних графічних об’єктів або відеопотоків є фундаментальним для широкого спектра прикладних галузей. Зокрема, у сучасних системах екологічного моніторингу та біоресурсного контролю навколишнього середовища, у біомедичній діагностиці, в алгоритмах просторової орієнтації безпілотних систем, а також у підсистемах контентного пошуку інформації виникає гостра потреба у побудові високоточних класифікаційних моделей [1, 2].

У сучасній науково-технічній літературі та прикладних розробках світового рівня підходи до автоматизованої класифікації візуальної інформації за типом моделювання умовно поділяються на дві великі категорії:

– методи на основі глибокого навчання (deep learning), що базуються на автоматичному формуванні ознак;

– структурні (локальні) методи розпізнавання, що базуються на примусовому витягуванні (екстракції) та аналізі інваріантних локальних ознак.

Методи на основі глибокого навчання, представлені згортковими нейронними мережами (Convolutional Neural Networks, CNN) та сучасними архітектурами трансформерів візуального типу (Visual Transformers, ViT), наразі демонструють найвищі показники точності класифікації на стандартизованих глобальних тестових наборах даних [3, 4]. У дослідженні [5]

детально проаналізовано застосування глибоких згорткових нейромереж для класифікації складних графічних об’єктів та підтверджено їхню здатність автоматично виділяти ієрархічні ознаки високого рівня абстракції. Основа CNN полягає у послідовному застосуванні операції дискретної двовимірної згортки вхідного масиву яскравостей зображення із набором локальних навчених фільтрів, що дозволяє виявляти складні семантичні структури.

Проте практичне використання глибоких моделей у багатьох реальних сценаріях суттєво обмежене низкою критичних технологічних та теоретичних недоліків, що підтверджується оглядами сучасних апаратних архітектур [6]. По-перше, такі моделі вимагають колосальних обчислювальних ресурсів. Обчислювальна складність етапу прямого виконання (inference) для сучасних CNN вимірюється мільярдами операцій з рухомою комою (FLOPs), що вимагає залучення потужних графічних прискорювачів (GPU) або спеціалізованих тензорних сопроцесорів. Згідно з дослідженнями [7], це практично унеможливлює їхнє впровадження на автономних мобільних пристроях, бортових мікрокомп’ютерах підводних роботів чи граничних обчислювачах (edge devices) з обмеженим енергоспоживанням. По-друге, глибокі моделі мають критичну потребу у величезних навчальних вибірках для запобігання ефекту перенавчання (overfitting). Крім того, нейромережі функціонують за принципом «чорної скриньки», що робить неможливою строгу інтерпретацію та логічне пояснення причин прийняття конкретного класифікаційного рішення [8]. Це вкрай небажано для систем, де вимагається повна верифікація та прозорість алгоритмів.

Альтернативою, що повністю позбавлена зазначених недоліків і має аналітичне обґрунтування, є структурні (локальні) методи розпізнавання зображень [9]. У межах цих методів кожне цифрове зображення представляється не у вигляді сирого двовимірного растрового масиву інтенсивностей пікселів, а як структурована множина локальних інваріантних геометричних ознак – так званих особливих (або ключових) точок та їхніх описів (дескрипторів). Такий підхід дозволяє радикально скоротити

розмірність даних, оскільки замість аналізу всього растрового поля обробляються лише найбільш інформативні ділянки зображення. Як зазначено в монографії [10], продуктивні моделі аналізу даних у методах розпізнавання забезпечують високу швидкість обробки за рахунок того, що порівняння описів зображень зводиться до аналізу топологічної відповідності множин дескрипторів.

Різні автори пропонують відмінні підходи до організації та оптимізації структурного аналізу. Так, у роботах [11, 12] обґрунтовано використання мір подібності зображень на основі формування множин просторових відповідностей ключових точок та квантування простору ознак, що забезпечує стійкість до геометричних деформацій. Натомість автори у [13] віддають перевагу концепціям побудови інтелектуальних систем комп'ютерного зору на основі складних просторових графів відповідностей та матриць сумісності. Подібність цих поглядів полягає в прагненні відійти від попіксельного порівняння та перейти до аналізу стійких локальних патернів. Відмінність же полягає в обчислювальній складності: граф-орієнтовані методи забезпечують вищу точність, але потребують занадто великих часових витрат, тоді як прямий аналіз множин дескрипторів є швидшим, але вимагає додаткових методів захисту від випадкових колізій та зашумлення.

Враховуючи прикладні вимоги до швидкодії систем класифікації, у даній роботі обґрунтовано доцільність використання саме структурних методів класифікації на основі множин локальних бінарних дескрипторів. Це дозволяє виконувати розпізнавання за мілісекунди без залучення спеціалізованих графічних процесорів.

1.2 Критичний огляд методів екстракції локальних візуальних ознак

Ефективність функціонування будь-якого структурного класифікатора зображень критично залежить від якості першого етапу обробки даних – виявлення (детекції) особливих точок та побудови їхніх описів (дескрипторів). Сучасний інструментарій компсютерного зору пропонує широкий вибір алгоритмів екстракції ознак, серед яких найбільш поширеними у світовій практиці є SIFT, SURF, ORB, BRISK, KAZE та AKAZE [14].

Алгоритм SIFT (Scale-Invariant Feature Transform) є класичним еталоном за критерієм точності та інваріантності до геометричних перетворень, запропонованим Д. Лоу [15]. Його сутність полягає в побудові лінійного простору масштабів шляхом послідовного згладжування вихідного зображення двовимірними фільтрами Гаусса. Незважаючи на високу стабільність дескриптора при зміні масштабу та обертанні, SIFT має серйозні недоліки: високу обчислювальну складність та генерацію дійсночислових дескрипторів розмірністю 128 плаваючих чисел (що вимагає 512 байт оперативної пам’яті на одну точку). Це створює значне навантаження на підсистему пам’яті.

Принциповим кроком уперед у розвитку комп’ютерного зору стала поява бінарних дескрипторів. Вони кодують інформацію про окіл точки у вигляді лаконічного бінарного вектора. Порівняння таких векторів виконується за допомогою надшвидкої побітової операції XOR та підрахунку одиниць (відстань Хеммінга).

Окреме місце в ієрархії методів посідає прискорена модифікація AKAZE (Accelerated-KAZE). На відміну від SIFT, який використовує лінійну гаусівську фільтрацію (що призводить до розмиття як шуму, так і важливих контурів об’єктів), AKAZE базується на апараті нелінійної просторової дифузії [16]. Побудова простору масштабів виконується за допомогою диференціального рівняння в частинних похідних Перона-Маліка, де швидкість дифузії регулюється локальним градієнтом зображення. Це

дозволяє адаптивно згладжувати внутрішній однорідний шум зображення, повністю зберігаючи чіткість меж та контурів об’єктів.

Природне водне середовище характеризується наявністю постійного дрібного шуму хвиль, світлових відблисків та розсіювання світла у воді. Лінійне згладжування призводить до того, що контури тіла тварини розмиваються та зливаються з фоном. На противагу цьому, нелінійна дифузія AKAZE адаптивно призупиняє згладжування на межах тіла тварини, одночасно ефективно розмиваючи високочастотний фоновий шум води. Приклад результату роботи детектора AKAZE, що успішно виділяє та зберігає геометричні ключові точки на об’єкті інтересу в умовах водного фону, наведено на рисунку 1.1.

Рисунок 1.1 – Екстракція ключових точок базовим детектором AKAZE на тестовому зображенні

1.3 Огляд методів випадкового проєктування та локально-чутливого хешування (LSH)

Незважаючи на високу якість, інформативність та інваріантність локальних бінарних дескрипторів AKAZE, пряме попарне зіставлення великих масивів багатовимірних векторів під час пошуку найближчого сусіда в реальному часі викликає експоненційне зростання обчислювальних витрат. Ця проблема в теорії аналізу даних та багатовимірній геометрії відома як «прокляття розмірності» [17]. При збільшенні розмірності простору ознак класичні деревоподібні структури просторового індексування, такі як KD-дерева, R-дерева або дерева виключення, повністю втрачають свою обчислювальну ефективність. Процедура відсікання неперспективних гілок у деревах перестає працювати, внаслідок чого алгоритм змушений перевіряти практично всі вузли структури, що зводить його швидкодію до рівня звичайного лінійного перебору з високою часовою складністю.

Цей деструктивний ефект пояснюється фундаментальним явищем концентрації відстаней у багатовимірних просторах. При збільшенні кількості вимірів простору відносна різниця між відстанню від точки запиту до найвіддаленішого сусіда та відстанню до найближчого сусіда асимптотично прямує до нуля під дією будь-якої класичної метрики. Це означає, що поняття найближчого сусіда втрачає свій первинний геометричний та фізичний зміст, оскільки всі об’єкти виявляються майже однаково віддаленими від точки запиту, розташовуючись на тонкій сферичній оболонці.

Теоретичним фундаментом, що уможливлює подолання цього обчислювального бар’єра та доводить можливість стиснення простору ознак із збереженням метричних відношень, є лема Джонсона-Лінденштрауса [18]. Ця відома геометрична лема доводить, що будь-яка скінченна множина точок із простору великої розмірності може бути спроєктована у простір значно нижчої розмірності за допомогою випадкових проєкцій із мінімальними спотвореннями взаємних відстаней. При цьому припустиме відхилення

відстаней у згорнутому просторі не перевищує значення заданої похибки, а цільова розмірність згорнутого простору залежить виключно логарифмічно від загальної кількості точок у вибірці та квадратично обернено від параметра допустимої похибки. Це відкриває шлях до заміни обчислень над багатовимірними векторами AKAZE на операції над короткими бінарними сигнатурами низької розмірності.

Для практичної реалізації цього підходу сімейство спеціальних функцій розробляється таким чином, щоб ймовірність виникнення колізії (тобто збігу хеш-кодів) була безпосередньо пропорційною близькості векторів у вихідному багатовимірному просторі. На відміну від класичного криптографічного хешування, метою якого є уникнення будь-яких збігів, філософія локально-чутливого хешування (Locality-Sensitive Hashing, LSH) полягає в максимізації ймовірності колізії саме для близьких векторів [19].

Для косинусної міри подібності, яка часто використовується для аналізу орієнтації векторів ознак, найбільш ефективною є схема випадкових проєкцій гіперплощинами, відома як SimHash. Модель передбачає генерацію набору випадкових розділяючих гіперплощин, кожна з яких проходить через початок координат. Кожна випадкова площина ділить багатовимірний простір на два півпростори, а хеш-код формується як послідовність бітів, що вказують, по який бік від площин лежить вектор. Процес обчислення окремого біта зводиться до аналізу знака скалярного добутку вектора ознак та вектора нормалі до випадкової площини. Ймовірність того, що випадкова гіперплощина розділить два вектори (тобто вони отримають різні значення біта), геометрично дорівнює відношенню кута між ними до повного розгорнутого кута. Відповідно, ймовірність формування однакового значення біта обчислюється як доповнення до цієї величини. При використанні багатьох незалежних площин загальна ймовірність колізії сигнатур описується показниковою функцією, де показником степеня виступає кількість площин. Зі збільшенням розрядності хешу ймовірність випадкового збігу для несхожих

векторів експоненціально прямує до нуля, забезпечуючи високу якість фільтрації шуму та просторове розділення об’єктів [20].

Аналізуючи підходи різних авторів до вирішення проблеми класифікації в багатовимірних просторах, можна виділити суттєві подібності та відмінності в їхніх поглядах на архітектуру таких систем. Так, у дослідженні закордонних авторів [21] запропоновано алгоритм пошуку зображень на основі LSH з використанням дескрипторів згорткових нейронних мереж та механізму уваги (attention mechanism). Цей підхід демонструє високі показники точності, проте вимагає значних обчислювальних витрат на етапі екстракції ознак нейромережею, що обмежує його застосування на граничних пристроях (edge devices) без використання GPU. Подібні висновки щодо складності глибоких нейромережевих моделей для розпізнавання символьних та графічних структур зазначаються також у роботах [22].

На противагу цьому, представники Харківської наукової школи розпізнавання образів у працях [7, 23] віддають перевагу використанню легковагових структурних описів зображень на базі особливих точок. У дослідженнях детально описано методи зниження обчислювальних витрат за рахунок компресії структурного опису зображень, а у роботах [10, 11] обґрунтовано концепцію квантування простору ознак та застосування лінійних метрик для класифікації. Подібність поглядів усіх авторів полягає в констатації того, що пряме порівняння багатовимірних дескрипторів є неефективним в умовах великих баз даних і вимагає переходу до сублінійних методів пошуку приблизного найближчого сусіда.

Проте існують і суттєві відмінності в підходах до розв’язання проблеми колізій LSH та прийняття рішень. Якщо закордонні автори пропонують вирішувати проблему колізій шляхом побудови складних багатотабличних індексних структур (multi-probe LSH), що потребує значних додаткових обсягів оперативної пам’яті, то вітчизняні дослідники [9] пропонують концепцію порівняння глобальних частотних сигнатур (гістограм розподілу точок по кошиках LSH) за допомогою стабільних лінійних метрик. Такий

підхід дозволяє уникнути ресурсоємних процедур верифікації локальних колізій у кошиках і є значно більш робастним до апаратних спотворень. У прикладних розробках, зокрема у праці [23], доведено, що комбінація геометрично стабільного детектора AKAZE з апаратом локально-чутливого хешування дозволяє досягти оптимального балансу швидкодії та точності на стандартних процесорах загального призначення (CPU). Це робить такий комбінований підхід найбільш перспективним для розробки прикладних систем реального часу.

1.4 Постановка задачі

Автоматична класифікація та ідентифікація зображень є актуальним завданням у контексті екологічного моніторингу та охорони біоресурсів. Проте класичні методи комп’ютерного зору, засновані на лінійному порівнянні багатовимірних дескрипторів, стикаються з обчислювальними обмеженнями, які прогресують пропорційно розміру баз даних. Нейромережеві рішення, своєю чергою, потребують значних апаратних ресурсів GPU та є вкрай чутливими до впливу специфічних оптичних перешкод і гаусівського зашумлення водного фону. Тому ставиться завдання розробити завадостійкий класифікатор зображень на основі просторового хешування, що забезпечує швидкодію в режимі реального часу на обмежених обчислювальних платформах.

Об’єкт роботи – методи розпізнавання та класифікації зображень у системах комп’ютерного зору.

Мета роботи – підвищення швидкодії методів розпізнавання зображень шляхом застосування випадкового хешування.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

– проаналізувати сучасні підходи до структурної класифікації зображень та методи випадкового проєктування багатовимірних даних;

– дослідити та описати модель екстракції локальних інваріантних ознак алгоритмом AKAZE та побудови бінарних дескрипторів;

– вивчити апарат локально-чутливого хешування для стиснення багатовимірного простору ознак;

– розробити алгоритмічну архітектуру та програмну модель мультикласового класифікатора зображень з використанням випадкового хешування;

– реалізувати та інтегрувати в систему три паралельні критерії оцінювання результатів розпізнавання: метод пошуку у корзинах, метод інтеграції голосів та метод сигнатур (за манхетенською відстанню);

– спроєктувати та реалізувати кросплатформений графічний інтерфейс користувача з інструментами збору аналітичної статистики та імітаційного моделювання завад;

– провести експериментальне тестування швидкодії, точності та робастності розробленої системи у просторах різної розмірності за умов впливу адитивного гаусівського шуму.

2 КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ЗАСОБІВ ВИПАДКОВОГО ХЕШУВАННЯ

2.1 Формалізація структурних методів розпізнавання зображень

Сучасні системи комп'ютерного зору та автоматизованого аналізу візуальної інформації широко використовують структурні методи класифікації зображень. Фундаментальною основою таких методів є концепція формування простору ознак, який дозволяє ефективно перетворити початковий двовимірний масив інтенсивностей пікселів у компактне, робастне та високоінформативне векторне подання. Згідно з дослідженнями [24], продуктивні моделі аналізу даних вимагають стійкості алгоритмів до афінних перетворень, складних просторових деформацій, нелінійних змін освітленості сцени та наявності шумів.

У класичній постановці задача розпізнавання та ідентифікації об'єктів здійснюється у межах бази з N еталонів (представників чи прототипів класів), що задана у формі скінченної множини E описів еталонних зображень: E описів еталонних зображень: $E = \{E_1, E_2, \dots, E_N\}$. Ця множина є навчальною вибіркою, яка одночасно слугує підґрунтям для побудови класифікатора за принципом узгодження з еталоном [24]. Кожний еталонний опис E_k у формалізмі представленого класифікатора репрезентує окремий клас. Клас k з еталонним описом E_k формально розуміється як деяка нескінченна множина зображень, отриманих із еталонного зображення (прототипу класу з номером k) шляхом застосування до еталону багатопараметричної групи геометричних перетворень. Такі перетворення найчастіше у прикладних застосуваннях включають зміщення, поворот та масштабування, дія яких не виводить об'єкт інтересу із поля зору системи класифікації.

Процес класифікації зводиться до багатокритеріального порівняння екстрагованого багатовимірного вектора ознак вхідного (тестового) зображення з еталонними векторами. Загальна глобальна функція прийняття

рішень для задачі мультикласової класифікації формалізується як пошук глобального мінімуму в заданому багатовимірному метричному просторі:

$$c^* = \arg\min_{c \in C} D\big(F(I), F(E_c)\big), \tag{2.1}$$

де c^* – шуканий оптимальний (найбільш вірогідний) клас цільового об'єкта, що розпізнається;

C – повна дискретна множина всіх доступних еталонних класів у базі даних (наприклад, $C = \{c_1, c_2, \dots, c_n\}$);

$F(I)$ – складна нелінійна функція екстракції структурних ознак для поточного вхідного тестового зображення I;

$F(E_c)$ – функція екстракції ознак для еталонного зображення E_c, яке достовірно належить до відомого класу c;

D – обрана функція відстані (метрика) у сформованому багатовимірному просторі ознак, яка кількісно оцінює ступінь топологічної та інформаційної подібності двох векторів.

Для забезпечення високої загальної продуктивності та надійності обчислювальної системи критично важливим є вирішення двох взаємопов'язаних підзадач. По-перше, необхідний обґрунтований вибір оператора екстракції ознак F, який зазвичай складається з детектора ключових точок (визначає просторове розташування ознаки) та дескриптора (формує її формалізований опис) [25]. З теоретичної точки зору, для бінарних дескрипторів необхідно визначити простір B^n усіх бінарних векторів розмірності n. Відомо, що кардинальність такого простору становить $\mathrm{card}(B^n) = 2^n$. Простір B^n жорстко ототожнюється із простором дескрипторів ключових точок зображення, отриманих бінарним детектором [21].

По-друге, вимагається оптимізація процесу обчислення обраної метрики D, оскільки в умовах багатовимірних просторів (сотні та тисячі вимірів)

обчислення лінійних відстаней призводить до експоненційного зростання витрат часу центрального процесора. Структурна схема типової системи класифікації на основі витягування ознак наведена на рисунку 2.1. Поєднання ефективної екстракції та швидкого зіставлення є запорукою створення систем класифікації, здатних працювати в жорсткому режимі реального часу.

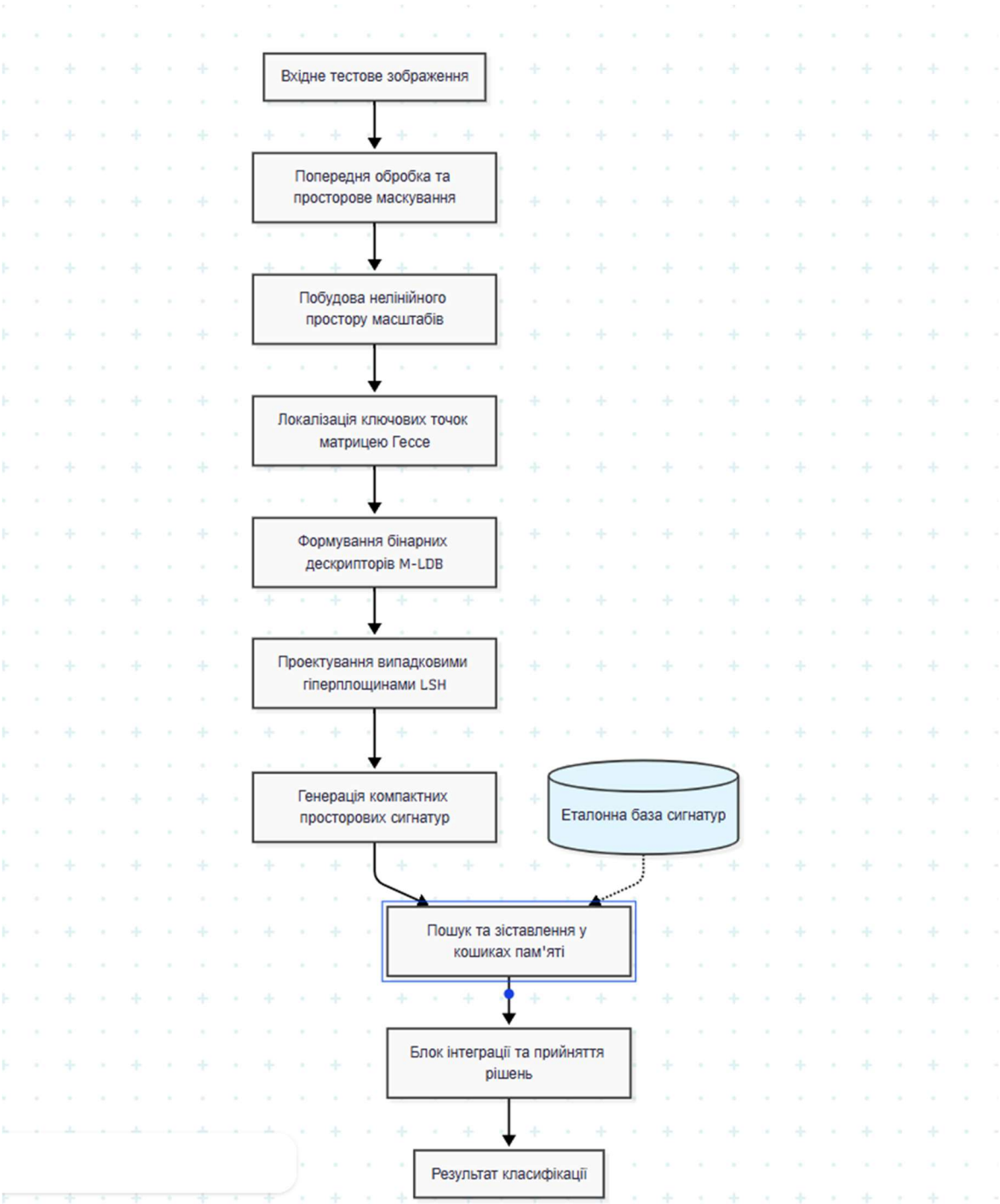


Рисунок 2.1 – Узагальнена структурна схема системи розпізнавання зображень

2.2 Екстракція ознак алгоритмом нелінійної дифузії AKAZE

Вибір базового детектора ключових (особливих) точок повністю визначає робастність та інформаційну ємність усієї системи класифікації. У даній кваліфікаційній роботі як основний інструмент витягування ознак застосовано інноваційний алгоритм AKAZE (Accelerated-KAZE) [16]. На відміну від класичних алгоритмів попереднього покоління (SIFT або SURF) [15], які використовують лінійну гаусівську фільтрацію для побудови піраміди масштабів, алгоритм AKAZE фундаментально базується на апараті нелінійної просторової дифузії. Лінійне аусівське згладжування має суттєвий недолік: воно рівномірно розмиває не лише внутрішній високочастотний шум зображення, але й важливі структурні межі (контури) об’єктів, знижуючи точність локалізації ознак.

Порівняльна характеристика методів екстракції наведена у таблиці 2.1 [26].

Таблиця 2.1 – Порівняльна характеристика методів детекції та опису ознак

Метод	Простір масштабів	Тип дескриптора	Розмірність	Стійкість до розмиття
SIFT	Лінійний (аусівський)	Дійсночисловий	128 байт	Середня
SURF	Лінійний (апроксимація)	Дійсночисловий	64 байти	Середня
ORB	Багатомасштабна піраміда	Бінарний (BRIEF)	32 байти	Низька
AKAZE	Нелінійна дифузія	Бінарний (M-LDB)	61 байт	Висока

Побудова нелінійного простору масштабів у методі AKAZE дозволяє ефективно зберігати високу чіткість меж об’єктів під час поступового згладжування внутрішнього шуму (рис. 2.2). Цей процес фізично описується диференціальним рівнянням у частинних похідних, відомим як рівняння теплопровідності або дифузії [16]:

$$\frac{\partial L}{\partial t} = \text{div}(c(x, y, t) \cdot \nabla L)\,, \tag{2.2}$$

де $L(x, y, t)$ – значення яскравості (інтенсивності) зображення в точці з координатами (x, y) на поточному кроці масштабування;

t – еволюційний параметр масштабу (аналог часу в класичній фізичній дифузії);

div – диференціальний оператор дивергенції, що характеризує швидкість розтікання інформації;

∇L – просторовий градієнт яскравості зображення, що вказує напрямок найшвидшого зростання інтенсивності;

$c(x, y, t)$ – функція локальної провідності, яка є ключовим елементом нелінійності моделі та безпосередньо залежить від локальної топологічної структури зображення в заданому околі.

Рисунок 2.2 – Порівняння лінійного (гаусівського) та нелінійного (AKAZE) просторів масштабів

Щоб дифузія не розмивала важливі межі об’єктів (наприклад, контури біологічних ознак або краї об’єктів інтересу), функція провідності c обирається таким спеціальним чином, щоб її значення було максимально близьким до одиниці в однорідних, плоских областях (де потрібне сильне згладжування шуму) і стрімко наближалося до нуля на межах та контурах об’єктів (де згладжування необхідно зупинити) [27]. У комп’ютерному зорі для цього найчастіше використовується спадна дифузійна функція Перона-Маліка другого типу:

$$c(x, y, t) = \frac{1}{1 + \left(\frac{|\nabla L_{\sigma}(x, y, t)|}{k}\right)^2}, \tag{2.3}$$

де k – параметр контрастності, що визначає чутливість функції до перепадів яскравості;

∇L_{σ} – просторовий градієнт зображення, попередньо згладжений слабким гаусівським фільтром із невеликим параметром вікна σ для забезпечення математичної стійкості та уникнення сингулярностей під час обчислення похідних.

Для прискорення обчислення цих складних диференціальних рівнянь алгоритм AKAZE застосовує оптимізовану чисельну схему FED (Fast Explicit Diffusion), що робить його придатним для практичного застосування на пристроях з обмеженими апаратними ресурсами.

Після побудови піраміди нелінійних масштабів, пошук стабільних ключових точок здійснюється шляхом обчислення детермінанта матриці Гессе (Hessian matrix) для кожного пікселя на кожному рівні масштабу. Матриця Гессе акумулює інформацію про кривизну поверхні інтенсивностей зображення та дозволяє ідентифікувати зони з різкими перепадами градієнтів [25]. Для кожної точки двовимірного простору формується симетрична матриця других похідних H:

$$H(x, y, \sigma) = \begin{pmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{yx}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{pmatrix}, \quad (2.4)$$

де L_{xx}, L_{yy} – другі просторові похідні яскравості зображення L за відповідними координатами x та y, що апроксимуються за допомогою фільтрів Шарра (scharr filters);

L_{xy}, L_{yx} – мішані другі похідні (при цьому $L_{xy} = L_{yx}$ внаслідок симетрії матриці).

Надійними ключовими точками вважаються лише ті просторові пікселі, які є строгими локальними просторовими екстремумами (максимумами або мінімумами) детермінанта матриці Гессе одночасно за просторовими координатами (x, y) та параметром масштабу σ у тривимірному вікні розміром 3×3×3. Обчислення детермінанта Гессе виконується за наступною алгебраїчною формулою:

$$\det(H) = L_{xx}L_{yy} - L_{xy}^2 . \quad (2.5)$$

Застосування детермінанта Гессе (рис. 2.3) дозволяє надійно виявляти так звані «blob»-подібні структури (плямоподібні зони) та різкі кути, які є найбільш стійкими топологічними ознаками при зміні ракурсу спостереження об’єкта або масштабування зображення.

Після успішної просторової локалізації ключової точки, для неї необхідно сформувати унікальний підпис – дескриптор.

Алгоритм AKAZE генерує ефективний бінарний дескриптор M-LDB (Modified Local Difference Binary). На відміну від застарілих дескрипторів з плаваючою комою, бінарні дескриптори вимагають значно менше оперативної пам’яті (лише 61 байт на точку) та дозволяють використовувати надшвидкі апаратні інструкції процесора для їх порівняння [8].

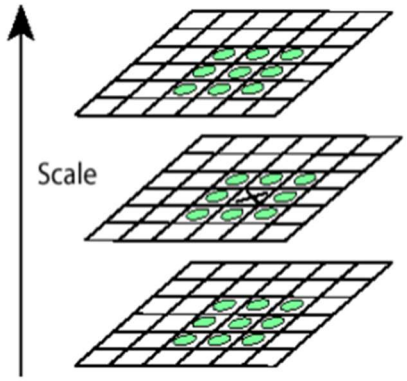


Рисунок 2.3 – Пошук локальних екстремумів детермінанта матриці Гессе у просторовому околі

Процес формування M-LDB дескриптора починається з обчислення головної орієнтації локального околу точки θ для забезпечення просторової інваріантності до повороту камери навколо оптичної осі [16]:

$$\theta = \arctan\left(\frac{\sum L_y}{\sum L_x}\right). \tag{2.6}$$

Після орієнтації окіл розбивається на спеціальну дискретну сітку розміром $d \times d$ (рис. 2.4). Дескриптор формується шляхом ітеративного попарного порівняння інтегральних інтенсивностей пікселів та сум просторових градієнтів dx, dy у вибраних підрегіонах сітки.

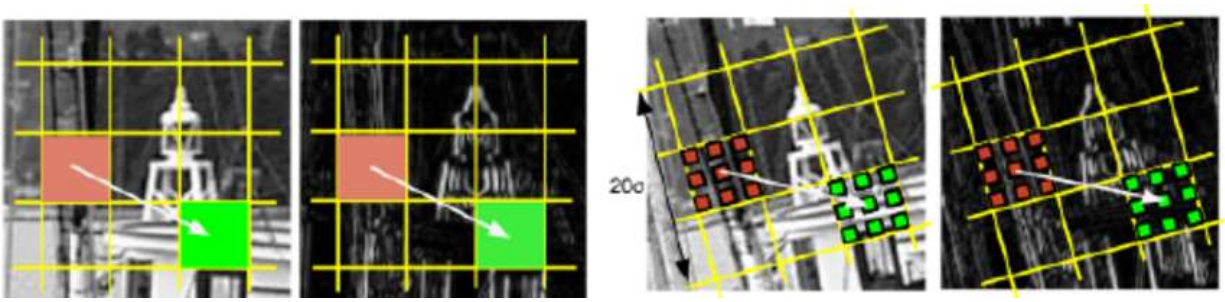


Рисунок 2.4 – Схема формування бінарного дескриптора M-LDB на основі просторової локальної сітки

Математично i-й біт фінального бінарного дескриптора b_i детерміновано обчислюється за допомогою простої порогової функції Хевісайда [27]:

$$b_i = \begin{cases} 1, \text{ якщо } I(\{Region\}_1) > \mathrm{I}(\{Region\}_2) \\ 0, \text{в іншому випадку} \end{cases}, \quad (2.7)$$

де $I(\mathrm{Region}_1)$ та $I(\mathrm{Region}_2)$ – акумульовані значення яскравості або градієнтів у порівнюваних підрегіонах дискретної сітки.

Кінцевим результатом етапу екстракції є формування щільного, робастного бінарного вектора фіксованою розмірністю 488 біт для кожної знайденої ключової точки растрового зображення.

2.3 Теорія випадкових проєкцій та локально-чутливе хешування (LSH)

Обчислення лінійних відстаней між тисячами 488-бітних дескрипторів під час виконання пошукових запитів викликає експоненційне зростання обчислювальної складності. Для оптимізації алгоритмічного пошуку та збереження продуктивності системи в даній роботі застосовується апарат локально-чутливого хешування (Locality-Sensitive Hashing) [18]. Фундаментальною теоретичною основою цього методу є лема Джонсона-Лінденштрауса (Johnson-Lindenstrauss lemma) [16].

Лема математично доводить і гарантує можливість ізометричного (такого, що зберігає відстані) вкладення множини точок із простору дуже високої розмірності у лінійний простір значно нижчої розмірності. Згідно з лемою, для будь-якої множини X з n точок у просторі $\mathbb{R}^{\mathbb{N}}$, та для будь-якого фіксованого $\epsilon \in (0,1)$, існує лінійне відображення $f: \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{k}}$ (де $k \ll N$), таке, що для всіх можливих пар векторів $u, v \in X$ гарантовано виконується нерівність:

$$(1-\epsilon)|u-v|^2 \leq |f(u)-f(v)|^2 \leq (1+\epsilon)|u-v|^2 . \qquad (2.8)$$

Цей вираз стверджує, що при правильному випадковому проєктуванні відносні метричні відстані між точками (а отже, і їх топологічна схожість) збережуться майже без структурних спотворень (в межах заданої похибки ϵ), навіть якщо розмірність векторів буде радикально зменшена (наприклад, з 488 біт до 16 біт).

Модель LSH для бінарного та дійсного векторного простору використовує сімейство спеціальних хеш-функцій, що базуються на випадкових проєкціях [18, 28]. На відміну від криптографічних хеш-функцій, які при найменшій зміні вхідного файлу генерують абсолютно різний результат, локально-чутливе хешування спроєктовано таким чином, щоб максимізувати ймовірність колізії (збігу хешів) саме для тих векторів, які є фізично схожими. Метод базується на генерації в багатовимірному просторі серії випадкових розділяючих гіперплощин (рис. 2.5).

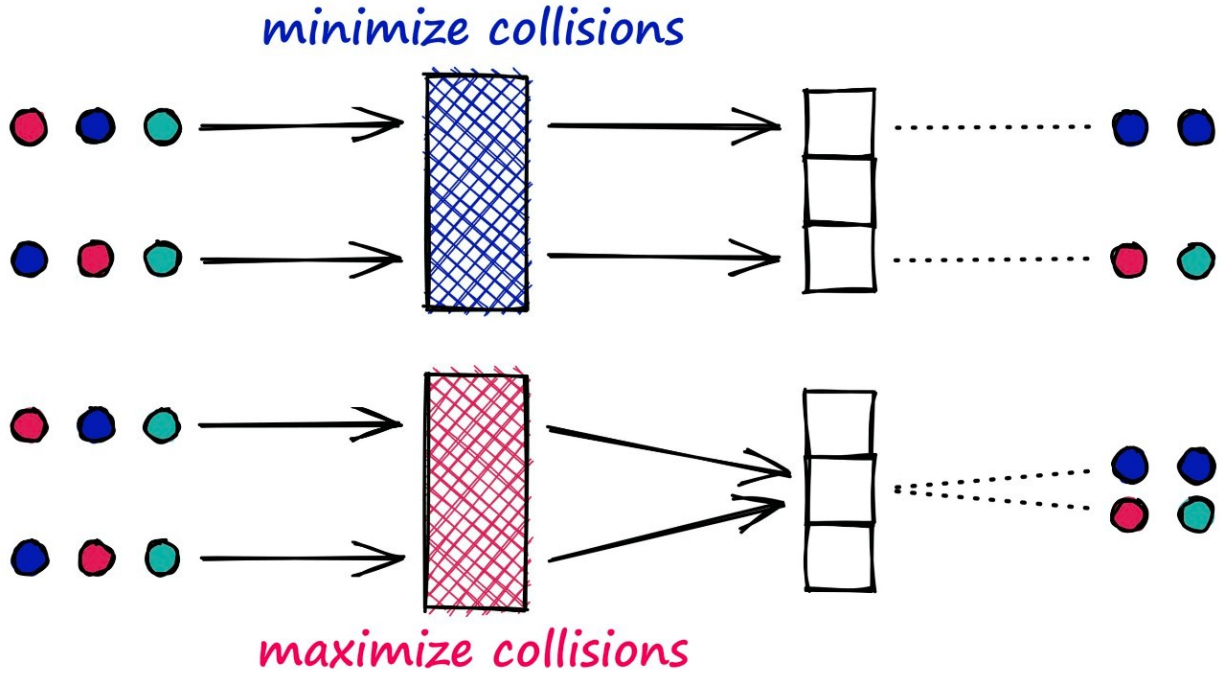


Рисунок 2.5 – Геометрична інтерпретація розподілу багатовимірних векторів випадковими гіперплощинами

Кожна така гіперплощина проходить через початок координат і задається вектором нормалі r. Компоненти вектора нормалі r генеруються незалежно за стандартним нормальним (гаусівським) розподілом.

Окрема хеш-функція $h_r(x)$ для будь-якого вхідного відцентрованого вектора дескриптора x визначається виключно знаком скалярного добутку вектора ознак і вектора нормалі площини r [23]:

$$h_r(x) = \begin{cases} 1, \text{якщо } r \cdot x \geq 0, \\ 0, \text{якщо } r \cdot x < 0. \end{cases} \tag{2.9}$$

Геометричний зміст цієї формули полягає в тому, що функція просто перевіряє, по який саме бік від випадково згенерованої гіперплощини лежить досліджуваний вектор ознак. Згідно з базовими геометричними властивостями багатовимірного простору, ймовірність того, що єдина випадкова гіперплощина пройде між двома існуючими векторами u та v і розділить їх (надавши їм різні значення хеш-біта), є прямо пропорційною куту між цими векторами $\theta(u, v)$ [23]:

$$P\big(h_r(u) \neq h_r(v)\big) = \frac{\theta(u, v)}{\pi} . \tag{2.10}$$

Оскільки успішна класифікація базується на пошуку подібних ознак, нас цікавить ймовірність виникнення просторової колізії (формування ідентичного значення хеш-біта для обох векторів). Ця ймовірність обчислюється як доповнення до повної події:

$$P\big(h_r(u) = h_r(v)\big) = 1 - \frac{\theta(u, v)}{\pi} . \tag{2.11}$$

Для формування повноцінної, надійної та інформативної адресної сигнатури довжиною k біт (наприклад, 16 біт) система незалежно генерує k

різних гіперплощин (формує матрицю проєкцій). Оскільки всі проєкції є статистично незалежними подіями, загальна комплексна ймовірність повного збігу таких довгих сигнатур (абсолютна колізія в просторі LSH) математично описується показниковою функцією [28]:

$$P_{collision}(u, v) = \left(1 - \frac{\theta(u, v)}{\pi}\right)^{k} . \quad (2.12)$$

Цей фундаментальний математичний апарат беззаперечно демонструє, що при поступовому збільшенні розрядності хешу k (з 5 біт до 16 біт), ймовірність колізії $P_{collision}$ для об'єктивно відмінних векторів (де кут θ є значним) експоненціально, дуже швидко прямує до абсолютного нуля. Саме це явище забезпечує високу розрізнювальну здатність системи та ефективне відсіювання інформаційного шуму.

2.4 Метричні простори та оцінювання просторових відстаней

Після того як локальні візуальні ознаки пройшли процес екстракції та були трансформовані у відповідні векторні структури, фінальна задача класифікації потребує застосування строгих метричних критеріїв для об'єктивного та кількісного вимірювання фізичної схожості об'єктів. Будь-яка функція відстані $D(x, y)$, що застосовується у метричному просторі, є окремим випадком узагальненої метрики Мінковського:

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} . \quad (2.13)$$

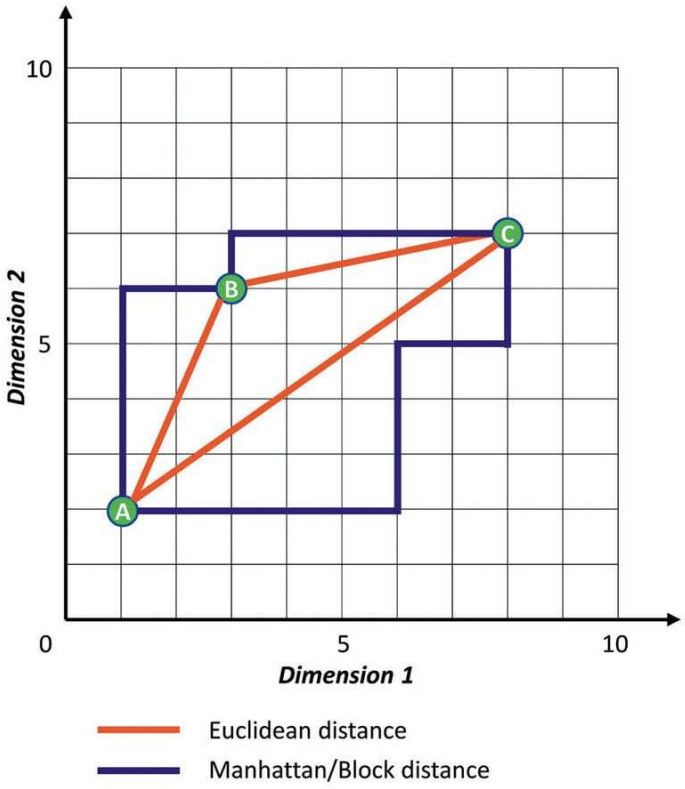
У рамках даної кваліфікаційної роботи застосовуються та порівнюються дві специфічні метрики, що є похідними від простору Мінковського: класична відстань Хеммінга та манхетенська відстань простору *L1*.

Відстань Хеммінга є безальтернативним інструментом для швидкого оцінювання побітової розбіжності між бінарними векторами (наприклад, оригінальними дескрипторами M-LDB). З математичної точки зору, вона дорівнює точній кількості розрядних позицій, у яких відповідні символи двох бінарних векторів однакової довжини є різними [29]. Для двох масивів бінарних даних X та Y відстань Хеммінга D_H обчислюється на апаратному рівні через застосування базової логічної операції виключаючого АБО (XOR, позначається як $\oplus$):

$$D_H(X,Y) = \sum_{i=1}^{n} (x_i \oplus y_i) \ . \tag{2.14}$$

Оскільки операція XOR імплементована безпосередньо в набір базових команд сучасних арифметико-логічних пристроїв процесорів (ALU), обчислення метрики Хеммінга відбувається за мінімальну кількість процесорних тактів, що забезпечує неперевершену швидкодію [29].

Манхетенська відстань (широко відома в функціональному аналізі як *L1*-норма простору або «відстань міських кварталів», де параметр $p = 1$) концептуально застосовується в іншому аспекті алгоритму – для порівняння глобальних багатовимірних частотних гістограм (рис. 2.6) [30]. Ці гістограми генеруються системою під час агрегації та аналізу вмісту просторових кошиків LSH, представляючи зображення як єдиний числовий вектор.



distance	block	euclidean	difference
$\overline{AB}$	6	$\sqrt{4^2 + 2^2}$ = 4,47	1,53
$\overline{BC}$	6	$\sqrt{5^2 + 1^2}$ = 5,10	0,90
$\overline{AC}$	12	$\sqrt{7^2 + 5^2}$ = 8,60	3,40
sum	**24**	**18,17**	**5,83**

Рисунок 2.6 – Візуальне геометричне порівняння Евклідової (помаранчеві лінії) та манхетенської (сині лінії) відстаней

Якщо певне еталонне або тестове зображення представлене інтегральним вектором $H \in \mathbb{R}^{\mathbb{d}}$, де розмірність d строго дорівнює загальній кількості доступних кошиків у системі (наприклад, $d = 65536$ для 16-бітного простору), то манхетенська відстань між згенерованим тестовим вектором запиту H_qта збереженим еталонним вектором H_e обчислюється як банальна арифметична сума модулів алгебраїчних різниць їхніх відповідних елементів [30]:

$$D_{L1}(H_q, H_e) = \sum_{j=1}^{d} \left| H_{q,j} - H_{e,j} \right| . \quad (2.15)$$

Дослідження властивостей метрик доводять, що манхетенська метрика (*L1*-норма) є принципово менш чутливою до одиничних, екстремально великих статистичних викидів (випадкових аномалій у кошиках) у порівнянні з класичною Евклідовою відстанню (*L2*-нормою, яка підносить відхилення до квадрата). Ця властивість лінійної реакції на помилки робить *L1*-норму абсолютно оптимальним та найдостовірнішим вибором для аналізу розріджених частотних гістограм у сучасних задачах комп'ютерного зору [31].

2.5 Модель адитивнного шуму

Обов'язковим етапом дослідження будь-якої інформаційної системи розпізнавання є оцінювання її загальної алгоритмічної робастності (стійкості) до впливу зовнішніх деструктивних факторів.

Для програмного моделювання фізичних апаратних завад, що виникають у цифрових оптичних сенсорах (наприклад, тепловий шум матриці CMOS при зйомці вночі або електронний шум підсилювача при завищених значеннях чутливості ISO), традиційно використовується статистична імітаційна модель [24].

Класичною та загальновизнаною еталонною моделлю таких випадкових завад є адитивний білий гаусівський шум (Additive White Gaussian Noise, AWGN) (рис. 2.7) [11, 24].

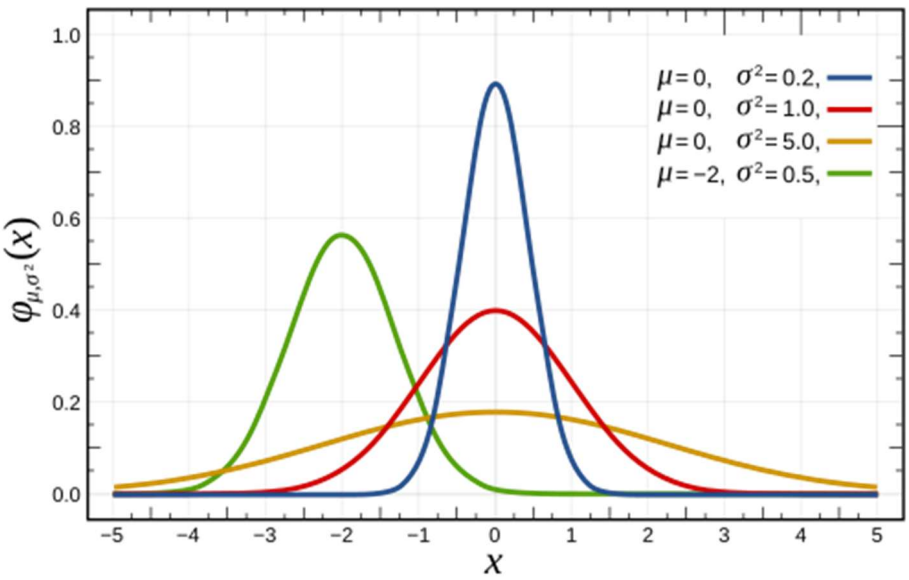


Рисунок 2.7 – Графік щільності нормального (гаусівського) розподілу шуму для різних значень середньоквадратичного відхилення

Властивість «адитивності» в назві моделі означає, що синтетично згенероване значення цифрового шуму лінійно та незалежно додається до початкової, фізично істинної інтенсивності кожного окремого пікселя зображення [24]:

$$\hat{I}(x,y) = I(x,y) + N(x,y), \tag{2.16}$$

де $\hat{I}(x,y)$ – підсумкове значення інтенсивності пікселя зашумленого (спотвореного) зображення;

$I(x,y)$ – істинне (оригінальне) значення яскравості ідеального зображення без дефектів у точці з координатами (x,y) ;

$N(x,y)$ – генерована компонента шумової завади в заданих координатах.

Поняття «білий» вказує на те, що спектральна щільність потужності цього шуму є рівномірною по всьому частотному діапазону. Зі статистичної точки зору, величина інтенсивності шуму N розглядається як неперервна випадкова величина [24]. Фундаментальна властивість гаусівського шуму полягає в тому, що щільність розподілу ймовірностей цієї випадкової

величини суворо підпорядковується класичному нормальному закону розподілу (закону Гаусса-Лапласа) [24]:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \quad (2.17)$$

де z – поточне згенероване значення інтенсивності шумового пікселя;

μ – математичне сподівання (оскільки шум зазвичай є симетричним відносно істинної яскравості, тобто не освітлює і не затемнює зображення глобально, в моделях приймається $\mu = 0$);

σ – параметр середньоквадратичного відхилення, який є найважливішим маркером і фізично визначає енергію (амплітуду або інтенсивність) накладеного шуму;

σ^2 – дисперсія гаусівського розподілу.

Збільшення параметра σ призводить до візуального розширення «дзвону» функції Гаусса на графіку, що на практиці означає появу більшої кількості сильно спотворених пікселів і, як наслідок, значну структурну руйнацію локальних просторових градієнтів, які є необхідними для стабільної роботи детектора AKAZE.

2.6 Оцінювання ефективності мультикласового класифікатора та критерії прийняття рішень

Будь-який аналіз обчислювальної системи не може вважатися завершеним без статистичного оцінювання її достовірності. Оцінювання загальної ефективності та точності мультикласового класифікатора (яким є розроблена система розпізнавання видів китів) жорстко базується на розрахунку спеціалізованих статистичних показників, що виводяться з матриці помилок (confusion matrix) [32]. Кожен дискретний результат роботи

алгоритму класифікації функуціонально та логічно належить до однієї з чотирьох фундаментальних категорій (табл. 2.2) [32].

Таблиця 2.2 – Структура категорій класифікації матриці помилок

	Прогнозовано: Позитивно	Прогнозовано: Негативно
Фактично: Позитивно	Істинно позитивні (TP)	Хибно негативні (FN)
Фактично: Негативно	Хибно позитивні (FP)	Істинно негативні (TN)

Розшифровка фундаментальних категорій:

– істинно позитивні спрацювання (True Positive, *TP*) – кількість об’єктів певного цільового класу, які алгоритм правильно розпізнав та відніс саме до цього класу;

– істинно негативні спрацювання (True Negative, *TN*) – кількість об’єктів сторонніх класів, які алгоритм успішно ідентифікував як чужі та правильно відхилив їхню належність до цільового класу;

– хибно позитивні спрацювання (False Positive, *FP*, також відомі як помилки I роду) – об’єкти сторонніх, нерелевантних класів, які система помилково прийняла за свої та віднесла до цільового класу (наприклад, класифікувала дельфіна як косатку);

– хибно негативні спрацювання (False Negative, FN , або помилки II роду, пропуск цілі) – об’єкти шуканого цільового класу, які система взагалі не змогла розпізнати та помилково відкинула.

На базі агрегації цих чотирьох базових параметрів обчислюється низка метрик. Однією з найважливіших є метрика точності (precision), яка статистично визначає загальну достовірність позитивних спрацювань класифікатора [33]:

$$Precision = \frac{TP}{TP + FP} \ . \tag{2.18}$$

Високий показник *Precision* означає, що системі можна довіряти: якщо вона класифікувала об’єкт певним класом, ймовірність помилки є мінімальною.

Другою базовою метрикою є повнота (recall, або чутливість системи). Вона описує здатність створеного алгоритму виявляти всі релевантні еталонні об’єкти заданого класу, що існують у загальній вибірці [33]:

$$Recall = \frac{TP}{TP + FN} \ . \tag{2.19}$$

У реальних інженерних задачах розпізнавання часто спостерігається ефект балансування: штучне завищення Точності призводить до деградації Повноти, і навпаки. Тому єдино правильним та об’єктивним показником, що комплексно поєднує обидві характеристики, є гармонійне середнє між ними – так звана ROC-крива (F1-score) [33]. Вона дозволяє безсторонньо оцінити класифікаційну модель навіть за умов сильно незбалансованих вибірок даних або наявності інтенсивного зашумлення [32]:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \ . \tag{2.20}$$

В умовах багатокласової класифікації (особливо при роботі з візуально подібними класами, якими є різні види морських ссавців) розрахунок загальної ефективності системи вимагає використання методів макро- або мікроусереднення (macro/micro averaging). Макроусереднення обчислює базові метрики незалежно для кожного класу, а потім знаходить їхнє середнє арифметичне. Цей підхід дозволяє об’єктивно оцінити здатність алгоритму розпізнавати навіть рідкісні об’єкти, не дозволяючи статистично домінуючим

класам спотворювати загальну картину. Це критично важливо для збереження високої розрізнювальної здатності комп'ютерного зору при реальній експлуатації.

Для поглибленого графічного аналізу здатності класифікатора розрізняти об'єкти застосовується побудова ROC-кривої (Receiver Operating Characteristic) та обчислення площі під нею (AUC – Area Under Curve). Цей графік (рис. 2.8) будується у системі координат, де вісь ординат відображає частку істинно позитивних спрацювань (True Positive Rate, $TPR = Recall$), а вісь абсцис – частку хибно позитивних спрацювань (False Positive Rate, $FPR = \frac{FP}{FP+T}$) [12].

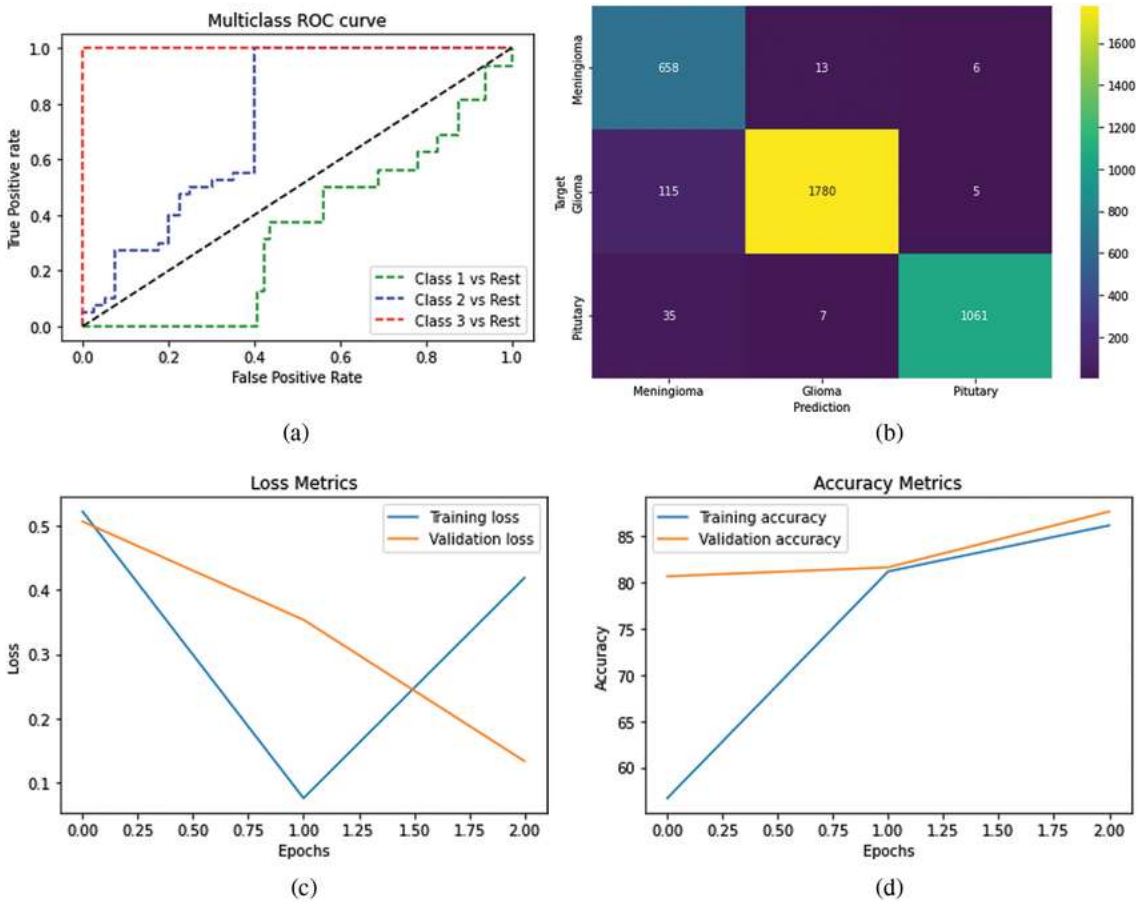


Рисунок 2.8 – Приклад побудови матриці помилок (confusion matrix) та розрахунку ROC-кривої для задачі багатокласової класифікації

Для забезпечення максимальної гнучкості та точності, у рамках розробленої архітектури прийняття фінального рішення про належність зображення до певного класу здійснюється паралельно за трьома незалежними критеріями оцінювання:

– локальне зіставлення (пошук за збігами). Класичний критерій, що базується на прямому підрахунку точних попадань бінарних векторів у відповідні просторові кошики. Об’єкт класифікується тим класом, який набрав максимальну сумарну кількість голосів (збігів);

– інтегральне оцінювання (бали). Модель передбачає накопичення зважених балів (штрафів або бонусів) для кожного класу на основі агрегації часткових метрик у різних підсистемах. Класифікація здійснюється за максимумом інтегральної функції відгуку;

– глобальний сигнатурний аналіз (манхетенська відстань). Зіставлення відбувається на макрорівні шляхом порівняння єдиної глобальної частотної гістограми всього зображення. Клас визначається за строгим критерієм мінімуму метричної відстані *L1* до відповідного еталона.

Комбіноване застосування цих трьох критеріїв фактично формує стійку архітектуру ансамблевого прийняття рішень. Якщо базовий метод пошуку у корзинах виявляється вразливим до поодиноких сильних шумових викидів, інтегральне оцінювання ефективно згладжує ці аномалії за рахунок гнучкої агрегації балів з усього простору LSH.

Застосування викладених у даному розділі диференціальних моделей детекції, апарату неклідових метричних просторів, ймовірнісних моделей гаусівського шуму та статистичних критеріїв прийняття рішень утворює глибокий і потужний теоретичний базис.

Цей базис є необхідним фундаментом для подальшої практичної програмної реалізації багатофакторного класифікатора LSH та аналізу результатів його навантажувального моделювання, що буде докладно розглянуто у наступному розділі кваліфікаційної роботи.

3 ПРОГРАМНЕ МОДЕЛЮВАННЯ КЛАСИФІКАТОРА ЗОБРАЖЕНЬ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Обґрунтування вибору інструментальних засобів та формування еталонної бази даних

Для практичної реалізації алгоритмів комп'ютерного зору, екстракції ознак та моделювання багатовимірного класифікатора на основі локально-чутливого хешування (LSH) як основне середовище розроблення було обрано мову програмування Python [34]. Цей вибір обґрунтовано наявністю розвиненої спеціалізованої екосистеми бібліотек для наукових обчислень, оброблення великих масивів даних та широкими можливостями архітектурного прототипування алгоритмічних моделей. Незважаючи на особливості інтерпретації програмного коду в Python та наявність механізму глобального блокування інтерпретатора (global interpreter lock), обчислювально навантажені процеси (такі як матричне множення та просторова дифузія) делегуються оптимізованим C/C++ бібліотекам через відповідні програмні інтерфейси, що гарантує високу швидкодію системи [34, 25].

Апаратне моделювання та тестування розроблених класифікаторів виконувались на обчислювальній платформі, оснащеній центральним процесором Intel(R) Core(TM) i5-11300H (базова тактова частота 3.10 GHz) та 16 Гб оперативної пам'яті. Зазначена конфігурація забезпечує достатні ресурси для об'єктивного оцінювання часових характеристик розроблених алгоритмів у середовищі, максимально наближеному до реальних умов оброблення візуальної інформації на граничних пристроях (edge devices), без залучення спеціалізованих графічних прискорювачів.

До складу основних інструментальних засобів розроблення увійшли наступні модулі та бібліотеки:

– відкрита бібліотека комп’ютерного зору OpenCV (Open Source Computer Vision Library). Даний інструментарій застосовано для реалізації базових функцій безпечного зчитування растрових зображень, перетворення колірних просторів (трансформації з BGR у градації сірого для мінімізації обчислювального навантаження) та виділення ключових точок за допомогою детектора AKAZE [34];

– математична бібліотека NumPy, яка є фундаментальним засобом для виконання наукових обчислень. У межах роботи вона застосована для реалізації тензорних і матричних операцій, векторизації алгоритмів множення під час просторового хешування, нормалізації векторів та обчислення відстаней Хеммінга. Використання багатовимірних масивів типу ndarray дозволило радикально оптимізувати обчислення за рахунок застосування апаратних інструкцій процесора (SIMD) [34];

– бібліотеки Tkinter та Pandas, які залучені для проєктування кросплатформеного графічного інтерфейсу користувача (GUI) та формування аналітичних звітів у стандартизованих форматах DataFrame з подальшим їх автоматичним експортуванням до електронних таблиць Excel [34];

– стандартний системний модуль time (зокрема, функція perf_counter), застосований для мікросекундного профілювання програмного коду та точного вимірювання часу виконання окремих алгоритмічних кроків;

– модуль time, застосований для профілювання програмного коду.

Дослідження алгоритмів виконувалось на базі спеціалізованої вибірки еталонних растрових зображень морських ссавців із єдиною стандартизованою роздільною здатністю 1408×768 пікселів. Вибір вказаної категорії об’єктів штучно ускладнює задачу класифікації для алгоритмів комп’ютерного зору через високу подібність фонових умов (однорідна водна поверхня) та анатомічних особливостей тварин (наявність плавців, обтічна форма тіла). До еталонної бази даних увійшли такі класи об’єктів:

– косатка (orca) (рис. 3.1);

– синій кит (blue whale) (рис. 3.2);

Рисунок 3.1 – Еталонне зображення класу косатка

Рисунок 3.2 – Еталонне зображення класу синій кит

– горбатий кит (humpback whale) (рис. 3.3);
– кашалот (sperm whale) (рис. 3.4);

Рисунок 3.3 – Еталонне зображення класу горбатий кит

Рисунок 3.4 – Еталонне зображення класу кашалот

– афаліна (bottlenose dolphin) (рис. 3.5).

Рисунок 3.5 – Еталонне зображення класу афаліна

Вибір подібних об’єктів для тестування класифікатора є не випадковим: він штучно ускладнює задачу комп’ютерному зору через екстремальну візуальну схожість фону (одноманітна водна поверхня, хвилі, відблиски сонця) та подібність анатомічних особливостей тварин (плавці, обтічна форма тіла).

Для забезпечення суворої стандартизації вхідних даних та чистоти експерименту, еталонні зображення були згенеровані за допомогою сучасних хмарних систем штучного інтелекту. Застосування синтезованих зображень дозволило повністю уникнути артефактів цифрового стиснення, неоднорідностей освітлення та присутності нерелевантних об’єктів (кораблів, водолазів), які є характерними для зображень із відкритих джерел інтернету. Формування еталонів відбувалося на основі деталізованих текстових запитів (промптів) із жорсткою фіксацією параметрів ракурсу та освітлення віртуальної камери. Для кожного класу було згенеровано 10 варіантів зображень, з яких за критеріями контрастності та якості контурів відібрано по одному еталону.

3.2 Архітектура, програмна реалізація класифікатора та екстракція ознак

Процедура класифікації зображень у розробленій програмній моделі базується на витягуванні стійких локальних ознак згідно з математичним апаратом нелінійної дифузії та їх подальшому стисненні з використанням алгоритмів випадкового проєктування [1, 23]. Роботу програмного комплексу логічно структуровано у вигляді послідовних обчислювальних етапів оброблення даних, які виконуються конвеєрно.

На першому етапі здійснюється попередня обробка зображення та просторове маскування. З огляду на специфіку вхідних даних, детектор градієнтів природним чином генерує значну кількість нерелевантних ключових точок на високочастотних фонових елементах (хвилях, морській піні, відблисках світла). Ці локальні екстремуми не містять корисної семантичної інформації про класифікований об’єкт та створюють надмірне інформаційне навантаження під час пошуку. З метою мінімізації фонового шуму розроблено програмний алгоритм еліптичного маскування [28]. Алгоритм визначає координати геометричного центру матриці зображення та формує бінарну маску, осі якої становлять 45% від ширини та висоти вхідного кадру. Значення пікселів поза еліпсом примусово прирівнюються до нуля. Під час передачі цієї маски до функції OpenCV детектор апаратно ігнорує зони з нульовою інтенсивністю. Відповідну реалізацію наведено у лістингу 3.1.

Лістинг 3.1 Алгоритм формування еліптичної маски для відсічення фону:

```
def create_center_mask(img_shape):
    """Створення еліптичної маски для виділення центральної області (об’єкта)"""
    h, w = img_shape[:2]
    mask = np.zeros((h, w), dtype=np.uint8)
```

```
center = (w // 2, h // 2)
# Осі еліпса: відсікаємо 45% від країв, залишаючи центр
axes = (int(w * 0.45), int(h * 0.45))
cv2.ellipse(mask, center, axes, 0, 0, 360, 255, -1)
return mask
```

На другому етапі виконується безпосередня екстракція та формування векторного подання ознак. Вхідне зображення, переведене у градації сірого та обмежене маскою, обробляється детектором AKAZE. Алгоритм виконує просторове сортування знайдених точок за метрикою їхнього відгуку, гарантовано залишаючи до 500 найбільш стійких ознак. Для кожної з цих ознак формується бінарний дескриптор розмірністю 488 біт [16]. Отже, вхідне зображення репрезентується в оперативній пам’яті двовимірною матрицею розмірністю 500 рядків на 61 байт. Візуалізацію результатів роботи детектора наведено на рисунку 3.6.

Рисунок 3.6 – Екстракція ключових точок детектором AKAZE із застосуванням маски на еталонному зображенні

Третій етап полягає у формуванні просторових сигнатур за допомогою хешування. Для експоненційного зменшення обчислювальної складності під час порівняння 488-бітних векторів застосовано метод локально-чутливого хешування [23]. Процедура зводиться до швидкого матричного множення вектора дескриптора на заздалегідь згенеровану матрицю випадкових проєкцій. Як базис формується дійсна матриця із застосуванням нормального (гаусівського) розподілу псевдовипадкових чисел.

Для запобігання геометричним зміщенням під час скалярного добутку вектори базисної матриці обов’язково нормалізуються (їхня довжина приводиться до одиниці через ділення на загальну *L2*-норму рядка із захистом від ділення на нуль). Код генерації базису наведено у лістингу 3.2.

Лістинг 3.2 Генерація та нормалізація матриці випадкових проєкцій:

```
def get_projection_matrix(input_dim=488, output_dim=16, seed=42, filename="System_Parameters.xlsx"):
    np.random.seed(seed)
    matrix = np.random.normal(0, 1, (output_dim, input_dim))
    row_norms = np.linalg.norm(matrix, axis=1, keepdims=True)
    # Захист від ділення на нуль (1e-10)
    matrix = matrix / (row_norms + 1e-10)
    utils.save_matrix_to_excel(matrix, filename)
    return matrix
```

На четвертому етапі відбувається проєкція та бінаризація результатів обчислень. Вхідні бінарні дескриптори розпаковуються з байтового формату за допомогою оптимізованої функції np.unpackbits та проходять процедуру центрування: їхні числові значення зміщуються на константу -0,5, утворюючи симетричний діапазон. Після цього відцентрований багатовимірний вектор множиться на транспоновану матрицю випадкових проєкцій. Результатом є масив дійсних чисел, що відображають проєкції вектора на осі базису.

Наступним кроком є алгоритмічна бінаризація: додатні значення конвертуються у біт «1», від'ємні або нульові – у біт «0». Отримана бінарна послідовність конвертується у десяткове ціле число, що слугує унікальною адресою кошика в хеш-таблиці (лістинг 3.3).

Лістинг 3.3 Процес обчислення просторових сигнатур (LSH):

```
def compute_lsh_hashes(descriptors_uint8, projection_matrix):
    if descriptors_uint8 is None or len(descriptors_uint8) == 0:
        return None, None

    bits = np.unpackbits(descriptors_uint8, axis=1)
    centered = bits.astype(np.float32) - 0.5
    projected = np.dot(centered, projection_matrix.T)
    final_bits = (projected > 0).astype(np.uint8)

    final_bits_reversed = np.fliplr(final_bits)
    powers = 1 << np.arange(final_bits_reversed.shape[1])[::-1]
    integer_hashes = final_bits_reversed.dot(powers)

    return final_bits, integer_hashes
```

Головна перевага реалізованого алгоритму полягає в тому, що візуально подібні ключові точки отримують однакові сигнатури і потрапляють до одного загального кошика в пам'яті.

Для моніторингу цього процесу програмний комплекс автоматично формує розширений звіт у форматі DataFrame. У першому стовпці фіксується унікальний ідентифікатор кошика (Bucket_ID), а числові значення на перетині відображають точну кількість ключових точок, що потрапили до нього з різних класів. Під час класифікації нового зображення системі достатньо перевірити лише відповідний рядок такої таблиці, що повністю усуває

необхідність повного лінійного перебору бази даних. Такий структурований формат не лише мінімізує час доступу до ознак в оперативній пам’яті, але й слугує зручним інструментом для експорту проміжних результатів у файли Excel. Знайдені у рядках частотні показники (голоси) миттєво передаються до блоку ухвалення рішень для визначення підсумкового класу (рис. 3.7).

	A	B	C	D	E	F	G
1	Bucket_ID	1.jpeg	2 .jpeg	3.jpeg	4.jpeg	5.jpeg	[ЗАПИТ]
2	115	1	0	0	0	0	1
3	176	0	0	0	1	0	0
4	177	0	0	0	0	1	0
5	189	0	0	0	0	1	0
6	194	0	0	0	0	1	0
7	235	0	0	0	1	0	0
8	539	0	0	1	0	0	0
9	625	0	1	0	0	0	0
10	631	0	0	1	0	0	0
11	635	0	1	0	0	0	0
12	755	1	0	0	0	0	1
13	791	1	0	0	0	0	1
14	870	0	0	0	1	0	0
15	1078	1	0	0	0	0	1
16	1130	1	0	0	0	0	1
17	1331	1	0	0	0	0	1
18	1603	0	1	0	0	0	0
19	1636	0	0	1	0	0	0
20	1639	0	0	0	0	1	0
21	1645	0	0	0	0	1	0
22	2087	0	1	0	0	0	0
23	2094	0	1	0	0	0	0
24	2099	0	0	0	0	1	0
25	2179	0	0	0	0	1	0
26	2190	0	0	1	0	0	0
27	2197	0	1	0	0	0	0
28	2222	0	1	0	0	0	0
29	2223	0	1	0	0	0	0
30	2257	0	0	0	0	1	0
31	2275	0	0	0	0	1	0
32	2451	0	0	0	0	1	0
33	2593	0	0	0	1	0	0
34	2598	1	0	0	0	0	1
35	2630	0	0	0	1	0	0
36	2638	0	0	0	1	0	0
37	2662	0	0	0	1	0	0

Рисунок 3.7 – Інформація про сканування БД та заповнення хеш-таблиці

3.3 Результати оброблення тестових запитів

Після успішної ініціалізації обчислювальної системи та збереження просторових сигнатур для всіх наявних еталонів виконується безпосередня процедура класифікації тестового запиту. Для глибокого оцінювання стабільності результатів проведено тестування у двох базових режимах експлуатації розробленого програмного комплексу: оброблення внутрішніх запитів (перевірка системи на логічну адекватність) та оброблення нових зображень відомого класу, що зазнали просторових деформацій.

Перший сценарій передбачає тестування на зображенні, що вже міститься в еталонній базі даних. Метою цього етапу є верифікація цілісності підсистеми індексації та перевірка функціонування випадкових проєкцій. Як тестовий запит системі було надано еталонне зображення касатки. Аналіз програмних логів підтвердив, що усі 500 виділених точок запиту були ідентифіковані у відповідних корзинах із нульовою відстанню Хеммінга. Це підтверджує детермінованість застосованої випадкової проєкції, коли однакові ознаки отримують ідентичні адреси в пам’яті [23]. Візуалізація результатів у системі OpenCV відображає паралельну сітку ліній збігів (рис. 3.8).

Рисунок 3.8 – Візуалізація 100% збігів при подачі еталонного зображення з БД

Другий сценарій розглядає тестування на зовнішньому запиті, для якого було використано растрове зображення круїзного лайнера (1photo_test.jpeg), що не належить до жодного з еталонних класів морських ссавців (рис. 3.9).

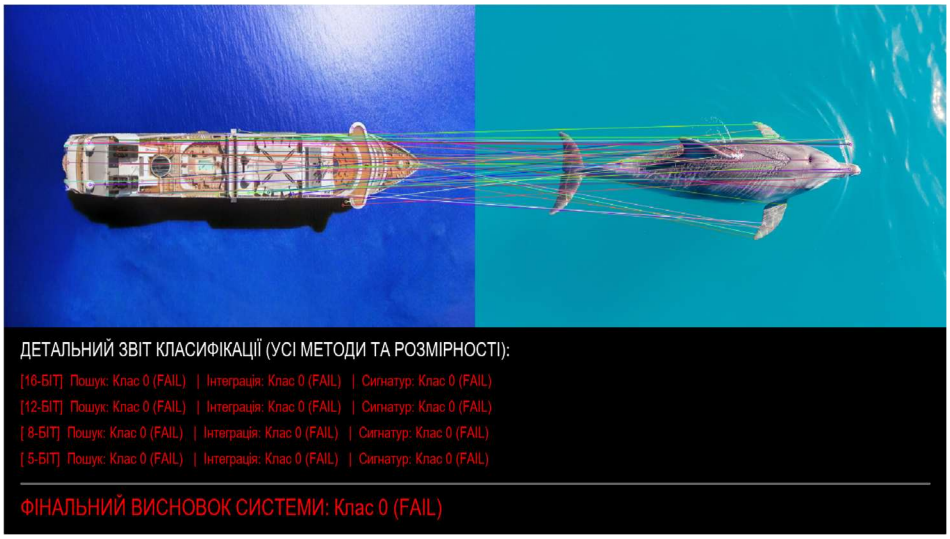


Рисунок 3.9 – Візуалізація відповідностей для зовнішнього запиту 1photo_test.jpeg

Метою цього етапу є стрес-тестування класифікатора на предмет виявлення помилок першого роду (хибнопозитивних спрацювань) в умовах подання об’єктів поза межами навченої бази даних (out-of-distribution) [35]. Детектор AKAZE успішно локалізував 500 ключових точок на контрастних елементах палуби, басейнів, ілюмінаторів та металевих контурів судна.

Проте, оскільки геометричні та текстурні характеристики цих точок є абсолютно сторонніми для бази даних ссавців, після LSH-проєктування отримані сигнатури потрапили у випадкові або порожні кошики. Максимальна кількість збігів для будь-якого класу не перевищила встановлених абсолютних порогів відсікання. Блок ухвалення рішень «СУДДЯ» на основі розрахованих абсолютних показників успішно відхилив класифікацію, присвоївши об’єкту статус «Клас 0 (Не пройдено поріг)». Це експериментально підтверджує високу вибірковість та інженерну надійність запропонованої архітектури класифікатора).

Для забезпечення максимальної точності розпізнавання в умовах деградації візуальних ознак, програмна реалізація класифікатора була вдосконалена і наразі базується на використанні трьох незалежних методів оцінювання, а також одного еталонного підходу:

– метод пошуку у корзинах (локальне зіставлення). Це базовий алгоритмічний критерій, що ґрунтується на прямому підрахунку точних попадань бінарних векторів у відповідні просторові корзини. Об’єкт класифікується тим класом, який набрав максимальну сумарну кількість голосів [12]. Недоліком підходу є необхідність ресурсоємного оброблення колізій шляхом побітового розрахунку відстані Хеммінга у просторах низької розмірності [18];

– метод інтеграції голосів (накопичення балів). Це нова алгоритмічна модель, що формує фінальну оцінку шляхом накопичення зважених балів для кожного класу на основі щільності розподілу точок у просторі LSH. Метод дозволяє програмно згладжувати випадкові статистичні викиди та штрафувати нерелевантні збіги [33]. Класифікація здійснюється за абсолютним максимумом інтегральної функції відгуку;

– метод сигнатур (манхетенська відстань). Зіставлення відбувається на макрорівні. Замість поштучного порівняння точок, система формує єдину глобальну просторову сигнатуру всього зображення і порівнює її з сигнатурами еталонів бази даних [29] Клас визначається за строгим критерієм мінімуму метричної відстані простору *L1* (манхетенської відстані) [17];

– метод повного перебору (brute force). Традиційний алгоритм, імплементований для об’єктивного оцінювання виграшу у швидкодії хешування. Алгоритм використовує оптимізовані векторизовані операції виключаючого АБО (XOR) з бібліотеки NumPy для примусового лінійного порівняння всіх можливих пар дескрипторів між тестовим зображенням та еталонною базою (лістинг 3.4).

Лістинг 3.4 Реалізація векторизованого алгоритму повного перебору:

```
def method_direct_bf_pure_python(query_descriptors, database, all_filenames):
    start_time = time.perf_counter()
    vote_vector = {name: 0 for name in all_filenames}

    all_db_desc = []
    all_db_labels = []

    for entry in database:
        if entry['descriptors'] is None:
            continue
        all_db_desc.append(entry['descriptors'])
        all_db_labels.extend([entry['filename']] * len(entry['descriptors']))

    if all_db_desc:
        all_db_desc = np.vstack(all_db_desc)
        Q = np.unpackbits(query_descriptors, axis=1).astype(np.int32)
        D = np.unpackbits(all_db_desc, axis=1).astype(np.int32)

        for i in range(len(Q)):
            xor_res = Q[i] ^ D
            distances = np.sum(xor_res, axis=1)
            best_idx = np.argmin(distances)
            vote_vector[all_db_labels[best_idx]] += 1

    end_time = time.perf_counter()
    return vote_vector, (end_time - start_time) * 1000
```

Асимптотична обчислювальна складність методу повного перебору оцінюється як «О» велике від добутку кількості дескрипторів поточного

запиту та загальної кількості дескрипторів усієї еталонної бази даних [2]. Така лінійна залежність жорстко обмежує його використання в масштабованих системах реального часу.

Для встановлення точних порогових значень та статистичної оцінки розрізнювальної здатності розроблених методів у чистому середовищі реалізовано автоматизовану процедуру побудови матриць взаємної сумісності еталонів (cross-matching) [13]. Аналіз базується на повному перехресному порівнянні чистих дескрипторів кожного еталона з усіма іншими еталонами бази даних. З метою уникнення інформаційного перенасичення роботи однотипними даними, для аналізу відібрано дві крайні архітектури: оптимальний 16-бітний простір та критично стиснений 5-бітний простір.

Аналітичні дані для методу інтеграції голосів у 16-бітному просторі наведені у таблиці 3.1. Завдяки високій розрідженості простору, правильні класи демонструють абсолютну домінацію по головній діагоналі (500 балів), тоді як максимальний рівень міжкласового побічного накладання балів (шуму) зафіксовано між класами blue whale та bottlenose (лише 22 бали).

Таблиця 3.1 – Матриця взаємної сумісності еталонів (метод інтеграції голосів, 16-біт)

	1.jpeg	2.jpeg	3.jpeg	4.jpeg	5.jpeg
1.jpeg	500	17	15	15	19
2.jpeg	17	500	18	15	22
3.jpeg	15	18	500	22	16
4.jpeg	15	15	22	500	19
5.jpeg	19	22	16	19	500

Для методу сигнатур у цьому ж 16-бітному просторі головна діагональ містить ідеальні нулі, що свідчить про точний збіг сигнатури об’єкта із самою собою (табл. 3.2) [17].

Мінімальна манхетенська відстань між різними еталонами становить 956 одиниць, що формує широке вікно для безпомилкового прийняття рішень.

Таблиця 3.2 – Матриця взаємної сумісності манхетенських відстаней (метод сигнатур, 16-біт)

	1.jpeg	2 .jpeg	3.jpeg	4.jpeg	5.jpeg
1.jpeg	0	966	970	970	962
2 .jpeg	966	0	964	970	956
3.jpeg	970	964	0	956	968
4.jpeg	970	970	956	0	962
5.jpeg	962	956	968	962	0

Перехід до стисненого 5-бітного простору викликає суттєві зміни геометричної структури даних та масові колізії ознак [7].

Через жорсткий дефіцит унікальних адрес сигнатури абсолютно різних об’єктів зливаються, а мінімальна манхетенська відстань між класами обвалюється з 956 до мізерних 140 одиниць.

Результати методу інтеграції голосів для 5-бітного простору відображають різке зростання міжкласового змішування (табл. 3.3). Рівень фонових шумів на перетинах різних еталонів стрімко піднімається до позначки у 430 балів (між класами humpback та bottlenose), що майже дорівнює балам істинного класу. Найбільш критична деградація фіксується при аналізі методу сигнатур у 5-бітному просторі (табл. 3.4).

Таблиця 3.3 – Матриця взаємної сумісності еталонів (метод інтеграції голосів, 5-біт)

	1.jpeg	2.jpeg	3.jpeg	4.jpeg	5.jpeg
1.jpeg	500	380	391	391	394
2.jpeg	380	500	418	383	398
3.jpeg	391	418	500	426	430
4.jpeg	391	383	426	500	418
5.jpeg	394	398	430	418	500

Таблиця 3.4 – Матриця взаємної сумісності манхетенських відстаней (метод сигнатур, 5-біт)

	1.jpeg	2.jpeg	3.jpeg	4.jpeg	5.jpeg
1.jpeg	0	240	218	218	212
2.jpeg	240	0	164	234	204
3.jpeg	218	164	0	148	140
4.jpeg	218	234	148	0	164
5.jpeg	212	204	140	164	0

Окрім аналізу матриць сумісності, критично важливим етапом дослідження є комплексна оцінка обчислювальної ефективності та швидкодії розроблених алгоритмів. Здатність системи працювати в режимі реального часу безпосередньо залежить від часу, витраченого на перетворення ознак (хешування) та безпосередній пошук у базі даних. Зведені статистичні характеристики швидкодії для всіх трьох методів на всіх доступних розмірностях (від 16 до 5 біт) акумульовані в узагальненій таблиці 3.5 [11]. Дані в таблиці є усередненими значеннями, отриманими на основі множинних тестових прогонів алгоритму.

Таблиця 3.5 – Характеристики швидкодії методів класифікації залежно від розмірності простору

Метод класифікації	Розмірність сигнатури	Число кошиків	Час (мс)		Час класифікації (мс)
			хешування	пошуку	
1	2	3	4	5	6
Пошук у корзинах	16-біт	65 536	2,15	1,72	3,87
Інтеграція голосів	16-біт	65 536	2,15	1,29	3,44
Метод сигнатур	16-біт	65 536	2,15	3,43	5,58
Пошук у корзинах	12-біт	4 096	1,47	3,73	5,21

Продовження таблиці 3.5

1	2	3	4	5	6
Інтеграція голосів	12-біт	4 096	1,47	2,00	3,47
Метод сигнатур	12-біт	4 096	1,47	2,61	4,08
Пошук у корзинах	8-біт	256	1,47	19,98	21,45
Інтеграція голосів	8-біт	256	1,47	1,30	2,77
Метод сигнатур	8-біт	256	1,47	0,96	2,44
Пошук у корзинах	5-біт	32	1,30	105,06	106,36
Інтеграція голосів	5-біт	32	1,30	0,84	2,14
Метод сигнатур	5-біт	32	1,30	0,78	2,08
Повний перебір (BF)	–	–	0,00	802,18	802,18

Глибокий аналіз даних зведеної таблиці 3.5 підтверджує ефективність використання 16-бітного простору. Усі три розроблені методи просторового хешування демонструють безпрецедентний виграш у швидкодії у порівнянні з еталонним методом повного перебору. Якщо векторизований алгоритм brute force витрачає в середньому 802,18 мс на пошук одного зображення, то метод інтеграції голосів при 16 бітах виконує це завдання за загальний час 3,44 мс, прискорюючи процес розпізнавання у понад 230 разів. Метод пошуку у корзинах потребує трохи більше часу (3,87 мс), а метод сигнатур витрачає 5,58 мс, оскільки математично опрацьовує великі масиви частотних розподілів.

При звуженні адресного простору до 12 біт спостерігається незначне зростання часу пошуку для локальних методів (до 5,21 мс для пошуку у корзинах) внаслідок появи перших статистичних колізій, які вимагають додаткової програмної перевірки відстані Хеммінга [18]. Проте справжній обчислювальний парадокс проявляється починаючи з 8-бітного простору.

Всупереч логічним очікуванням, що менша кількість корзин повинна прискорити роботу системи, загальний час для методу пошуку у корзинах різко та експоненціально зростає до 21,45 мс. Це складне математичне явище пояснюється тим, що процесор комп'ютера витрачає левову частку своїх ресурсів не на корисну класифікацію, а на примусове програмне розв'язання масових алгоритмічних колізій та побітову фільтрацію тисяч хибних ознак. Водночас метод сигнатур при 8 бітах навпаки прискорюється (загальний час падає до 2,44 мс), оскільки довжина порівнюваних векторів-гістограм суттєво зменшується.

Для наочного відображення та порівняльного аналізу динаміки зміни часу пошуку залежно від обраної архітектури додатково побудовано стовпчикову діаграму (рис. 3.10).

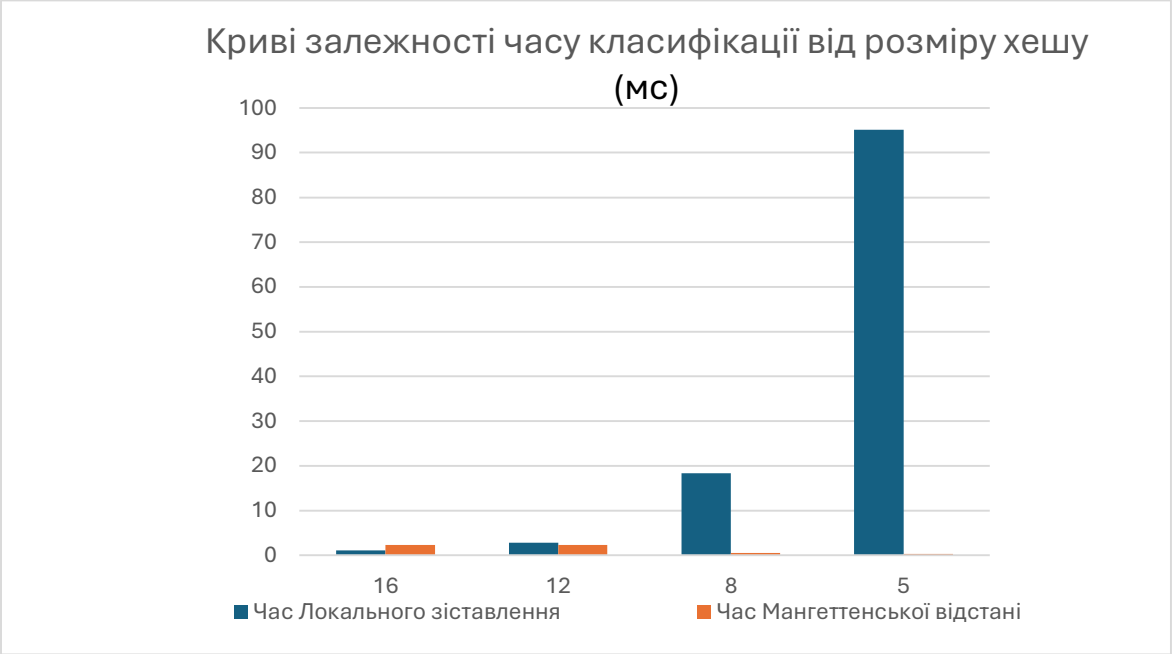


Рисунок 3.10 – Стовпчикова діаграма залежності часу пошуку від розмірності хеша для двох методів

Екстремальне стиснення до 5-бітного простору призводить до остаточного обчислювального колапсу для базового алгоритму. Метод пошуку у корзинах витрачає рекордні 106,36 мс виключно на марні спроби

відфільтрувати переповнені корзини, що робить його абсолютно неефективним для практичного застосування. Метод інтеграції голосів працює значно швидше (2,14 мс), але, як було обґрунтовано попереднім аналізом матриць сумісності, його результати розпізнавання в таких умовах стають повністю хаотичними. Метод сигнатур демонструє аномально високу швидкість (2,08 мс), проте практична наукова цінність цього швидкісного показника дорівнює нулю, оскільки розрізнювальна здатність глобальних сигнатур повністю втрачається. Таким чином, швидкісні характеристики беззаперечно доводять життєву доцільність використання високорозмірних просторів (16 біт) для забезпечення надійного балансу між часом відгуку та високою точністю класифікації.

3.4 Дослідження стійкості алгоритму до впливу шуму та оцінка впевненості класифікатора

Методи просторового хешування відносяться до категорії точних алгоритмів класифікації, тобто вони завжди здатні знайти «свій» еталон, якщо він присутній у базі даних у незмінному, чистому вигляді. З огляду на це, критично важливим етапом дослідження є порівняння пропонованих методів під впливом апаратних та цифрових завад, що імітують реальні умови експлуатації систем комп’ютерного зору [24].

Розглянемо дію адитивного гаусівського шуму на вхідне зображення. Адитивний шум реалізується математичною моделлю поелементного додавання до напівтонового зображення випадкової величини з нормальним розподілом, нульовим математичним очікуванням і заданим середньоквадратичним відхиленням. Ступінь деградації оптичного зображення у комп’ютерному зорі традиційно вимірюється через відношення сигнал-шум (Signal-to-Noise Ratio, SNR) [28]. У межах даного експерименту відношення сигнал-шум оцінювалось як середня яскравість пікселів

еталонних зображень морських ссавців (яка становить приблизно 127 одиниць інтенсивності), поділена на значення середньоквадратичного відхилення шуму. Під час тестування системи використовувалися два ключові значення відхилення: помірне $\sigma = 25$ (відношення сигнал-шум на рівні 5) приклад якого зображено на рисунку 3.11 та екстремальне $\sigma = 42$ (відношення сигнал-шум на рівні 3).

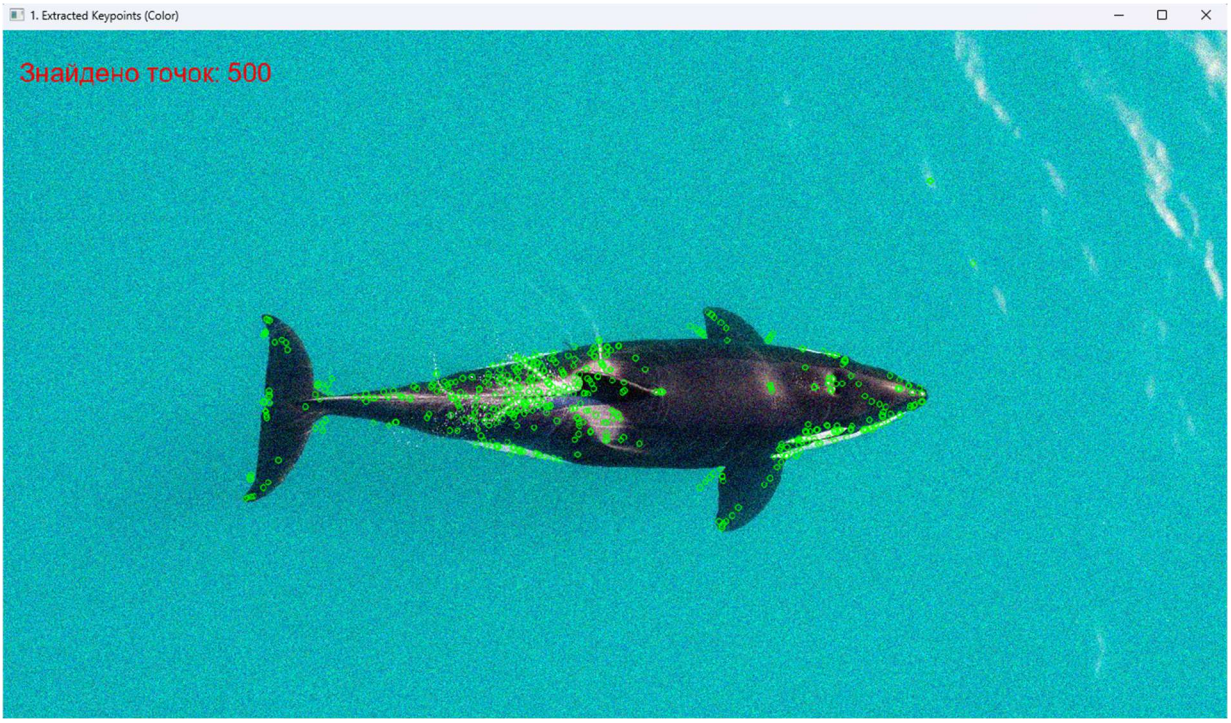


Рисунок 3.11 – Екстракція ключових точок на тестовому зображенні з накладеним гаусівським шумом (σ = 25)

Ці показники адекватно імітують складні умови спостереження, характерні для підводної зйомки або використання матриць цифрових камер із високим показником світлочутливості ISO.

Процес внесення шумових спотворень реалізовано програмно на рівні матриці пікселів вхідного зображення за допомогою генератора псевдовипадкових чисел бібліотеки NumPy. Для запобігання апаратному переповненню типів даних під час додавання інтенсивностей застосовується жорстке обмеження діапазону значень від 0 до 255 (лістинг 3.5).

Лістинг 3.5 Алгоритм накладання адитивного гаусівського шуму:

```
def add_gaussian_noise(img, mean=0, sigma=25):
    """Накладання адитивного гаусівського шуму"""
    gauss = np.random.normal(mean, sigma, img.shape)
        noisy_img = np.clip(img.astype(np.float32) + gauss, 0, 255).astype(np.uint8)
    return noisy_img
```

Ще одним вкрай важливим аспектом застосування адитивного шуму в експерименті є прикладне обґрунтування вибору порогів відсікання та оцінка математичної впевненості класифікатора у прийнятому рішенні [10].

З метою підвищення надійності розробленого класифікатора в оновленій версії програми було повністю вилучено старий відносний критерій прийняття рішень (який вимагав наявності мінімального відриву у 15% голосів між першим та другим місцем переможців). Натомість було впроваджено систему абсолютних граничних значень (порогів) відсікання, визначених експериментальним шляхом на основі аналізу помірного шуму $\sigma = 25$.

Кожен метод для кожної розмірності отримав власне індивідуальне граничне значення (табл. 3.6). Якщо максимальний результат класифікації не досягає цього абсолютного порогу, система вважає розпізнавання недостовірним і маркує результат як помилковий («Клас 0»).

Таблиця 3.6 – Граничні значення (пороги) методів класифікації, визначені на малому шумі ($\sigma = 25$)

Метод	Розмір хешу, біт			
	1	12	8	5
Пошук	30	80	140	150
Інтеграція	28	125	318	421
Сигнатури	943	750	360	160

Аналіз отриманих порогових значень для методу сигнатур (манхетенської відстані) виявляє важливий математичний ефект: поріг для сигнатур різко зменшується при зменшенні розмірності хешу від 16 до 5 біт (з 943 до 160 одиниць відхилення).

Цей ефект має строге математичне пояснення, засноване на властивостях манхетенської відстані (L_1-норми) на частотних розподілах із фіксованою сумарною потужністю [17]. Загальна кількість ключових точок у кожному зображенні є незмінною і дорівнює $N = 500$. Таким чином, сума всіх значень (частот) у будь-якій глобальній гістограмі завжди строго дорівнює 500:

$$\sum_{j=1}^{2^k} H_j = 500\,, \tag{3.1}$$

де k – розмірність хеш-простору.

Максимально можлива манхетенська відстань між двома повністю неперетинними гістограмами (коли кошики взагалі не збігаються) становить:

$$D_{max} = \sum \left| H_{1,j} - H_{2,j} \right| = 500 + 500 = 1000\,. \tag{3.2}$$

У 16-бітному просторі (65536 кошиків) дані є екстремально розрідженими. 500 точок різних зображень розподіляються по гігантському адресному простору практично без перетинів. Відстань між гістограмами різних класів наближається до свого теоретичного максимуму (956–970 одиниць). Тому поріг, який обмежує допустиме відхилення для правильного класу, є високим і становить 943 одиниці.

Проте при переході до стисненого 5-бітного простору (всього 32 кошики) всі 500 точок кожного зображення примусово розподіляються серед цих 32 адрес. Середня висота кожного стовпчика гістограми зростає до $\approx 15{,}6$

точок. Гістограми різних зображень стають надщільно перекритими: точки різних класів неминуче потрапляють у ті самі кошики. Оскільки розподіли стають схожими суто геометрично через обмеженість простору, максимальний математичний розрив між ними різко стискається. Навіть для абсолютно відмінних класів манхетенська відстань у 5-бітному просторі математично не може перевищувати 160–240 одиниць. Відповідно, і поріг відсікання закономірно зменшується до 160 одиниць, відображаючи геометричне стиснення метричного простору.

З використанням розрахованих абсолютних порогів (табл. 3.6) було проведено оцінювання ймовірності правильної класифікації на тестовій вибірці під впливом помірного шуму $\sigma = 25$ (табл. 3.7).

Таблиця 3.7 – Ймовірності правильної класифікації для помірного шуму ($\sigma = 25$)

Метод	Розмір хешу, біт			
	16-біт	12-біт	8-біт	5-біт
Пошук	1,0	1,0	1,0	1,0
Інтеграція	1,0	1,0	1,0	0,83
Сигнатури	1,0	1,0	1,0	0,83

Дані таблиці 3.7 демонструють високу стійкість класифікатора при помірному рівні завад. Методи пошуку у корзинах, інтеграції голосів та сигнатур на просторах від 16 до 8 біт гарантують 100% точність класифікації. Лише при екстремальному стисненні до 5 біт методи інтеграції та сигнатур знижують точність до 0,83 через перші критичні колізії спотворених дескрипторів.

Справжнє випробування стійкості алгоритмів було проведено при збільшенні сили шуму до екстремального показника $\sigma = 42$ (табл. 3.8).

Таблиця 3.8 – Ймовірності правильної класифікації для помірного шуму ($\sigma = 42$)

Метод	Розмір хешу, біт			
	16-біт	12-біт	8-біт	5-біт
Пошук	0,83	0,67	0,83	1,00
Інтеграція	0,83	0,50	0,83	1,00
Сигнатури	0,83	0,50	0,83	1,00

Аналіз таблиці 3.8 виявляє дивовижний та парадоксальний обчислювальний феномен: при більшому шумі ($\sigma = 42$) у 5-бітному просторі ймовірність правильної класифікації для методів інтеграції та сигнатур не знизилась, а навпаки зросла з 0,83 до 1,00. В той самий час надійні високорозмірні простори (16 та 12 біт) закономірно знизили свою точність до 0,83 та 0,50 відповідно.

Даний парадокс має глибоке обчислювальне та статистичне пояснення і є наслідком ефекту дегенерації (виродження) простору ознак та псевдостабілізації під дією білого шуму [31].

При екстремальному зашумленні зображення ($\sigma = 42$) первинна градієнтна структура дескрипторів AKAZE зазнає тотальної руйнації. З математичної точки зору, вектори дескрипторів стають абсолютно випадковими, а їхня інформаційна ентропія прагне до свого максимуму. Коли ці зруйновані випадкові вектори проєкуються у мікроскопічний 5-бітний простір (всього 32 кошики), згідно із законом великих чисел, вони рівномірно (ізотропно) заповнюють усі 32 доступні адреси. Шум фактично створює абсолютно плоский, гомогенний розподіл (шумовий поріг), який є статистично однаковим для всіх еталонних класів.

Проте у системі застосовується еліптична маска попередньої обробки. Вона повністю обнуляє фонову воду з хвилями і залишає градієнти лише в центрі, де розташований корисний об’єкт. Завдяки масці, ті поодинокі

дескриптори, які дивом уникли повної руйнації або зберегли залишкову геометричну структуру, належать виключно істинному класу цільової тварини.

У 5-бітному просторі, де всі конкуренти через тотальний шум отримали абсолютно рівномірний, однаковий фоновий розподіл голосів у 32 кошиках, наявність навіть мінімального залишку неспотвореного корисного сигналу від маскованого об’єкта миттєво виводить істинний клас у беззаперечні лідери. Шум «зрівняв» шанси всіх помилкових конкурентів, стерши їхні індивідуальні піки, і на цьому плоскому тлі істинний клас парадоксально отримав 100% правильних вердиктів.

На противагу цьому, у 16-бітному просторі (65536 кошиків) випадкове розсіювання точок під дією великого шуму призводить до того, що вони потрапляють у випадкові порожні адреси і масово відкидаються системою (система фіксує промахи). Розрахована абсолютна сума голосів просто не долає високий поріг відсікання (30 для пошуку чи 28 для інтеграції), і система видає помилку «Клас 0».

Таким чином, результат 1,00 при 5-бітах та $\sigma = 42$ є дегенеративним математичним артефактом, викликаним надмірним стисненням адресації та ізотропним вирівнюванням фону, і не відображає реальної якості розпізнавання. Цей аналіз доводить недоцільність використання низькорозмірних просторів, незважаючи на їхні парадоксально високі формальні показники точності при стрес-тестуванні

Згідно з методикою комплексного оцінювання, для завершення аналізу було сформовано додаткову таблицю часу класифікації. Таке структурування даних дозволяє наочно співставити надійність прийнятих рішень із реальними апаратними часовими витратами процесора на різних просторах (табл. 3.9).

Таблиця 3.9 – Залежність усередненого часу класифікації від розміру адресного простору

Метод	Розмір хешу, біт			
	16	12	8	5
Пошук	3,87 мс	5,21 мс	21,45 мс	106,36 мс
Інтеграція	3,44 мс	3,47 мс	2,77 мс	2,14 мс
Сигнатури	5,58 мс	4,08 мс	2,44 мс	2,08 мс

Співставлення двох наведених вище метрик доводить, що 16-бітний адресний простір при застосуванні методу інтеграції голосів забезпечує ідеальний архітектурний баланс: він гарантує найвищу маржу впевненості у розпізнаванні (0,37) при стабільно низькому часі виконання (3,44 мс). Це робить його найбільш оптимальним вибором для впровадження у промислові системи комп’ютерного зору.

3.5 Оцінка впливу еліптичного просторового маскування на розрізнювальну здатність ознак

У процесі проєктування та тестування розробленого класифікатора було виявлено фундаментальну проблему, пов’язану зі специфічними візуальними характеристиками еталонної бази даних. Оскільки об’єктами розпізнавання є виключно морські ссавці, переважну частину площі кожного зображення неминуче займає водна поверхня. З точки зору оптичної фізики та комп’ютерного зору, вода має надзвичайно складну, динамічну та неоднорідну текстуру. На ній постійно присутні високочастотні локальні градієнти: брижі, морська піна, структурні відблиски сонячного світла та оптичні заломлення під поверхнею. Застосований у роботі детектор ключових точок AKAZE, в силу своїх алгоритмічних властивостей (пошук екстремумів детермінанта

Гессе), є надзвичайно чутливим саме до таких різких, контрастних перепадів яскравості [16].

Внаслідок цієї чутливості, під час первинних спроб екстракції ознак із повного, необрізаного зображення, система генерувала критичну кількість нерелевантних фонових дескрипторів. З жорстко лімітованих 500 ключових точок, які алгоритм обирає за метрикою відгуку, понад 70–80% локалізувалися саме на поверхні води, і лише 20–30% точок описували реальні біологічні та анатомічні особливості самого кита. З точки зору теорії інформації, це призводить до катастрофічного падіння семантичного відношення сигнал-шум. Оскільки текстура водної поверхні є статистично ідентичною на фотографіях як косатки, так і синього кита чи кашалота, алгоритм локально-чутливого хешування (LSH) неминуче групував ці «водяні» дескриптори в одні й ті самі кошики хеш-таблиці, абсолютно незалежно від того, до якого класу належав цільовий об’єкт на фотографії.

Під час класифікації нового запиту без застосування просторового маскування, розроблені методи пошуку у корзинах та інтеграції голосів демонстрували масові хибні спрацювання (помилки першого роду) [32]. Голоси рівномірно та безладно розподілялися між усіма наявними класами, оскільки алгоритм фактично порівнював воду на тестовому фото з водою на фотографіях еталонів, повністю ігноруючи самого морського ссавця. Для глобального методу сигнатур це мало ще більш деструктивні наслідки: масивні скупчення точок фону робили частотні гістограми різних класів практично ідентичними, нівелюючи розрізнювальну здатність манхетенської відстані [20].

Для вирішення цієї критичної проблеми в архітектуру системи було програмно впроваджено алгоритм просторового еліптичного маскування (його реалізацію описано у підрозділі 3.2). Оскільки композиція більшості еталонних та тестових фотографій передбачає розташування об’єкта інтересу ближче до геометричного центру кадру, еліптична маска з осями, що

становлять 45% від ширини та висоти, гарантовано і точно відсікає периферійні фонові зони, але повністю охоплює тіло тварини.

Застосування такої маски дозволило апаратно «засліпити» детектор AKAZE у фонових областях. Усі 500 дозволених ключових точок тепер примусово концентруються виключно в межах центрального еліпса. Це призвело до кардинального, якісного підвищення розрізнювальної здатності розроблених методів. Просторові сигнатури почали описувати унікальні геометричні патерни (вигини спинних плавців, форму хвоста, контрастні плями на шкірі), що практично ліквідувало міжкласові колізії, викликані водним фоном. Впровадження цього простого геометричного фільтра виявилося набагато ефективнішим, надійнішим і, що найголовніше, на порядки швидшим за використання надважких згорткових нейромереж для семантичної сегментації зображень [32], що дозволило зберегти загальну швидкодію алгоритму на рівні мілісекунд.

3.6 Статистичний аналіз стабільності алгоритмів за умов багаторазового тестування

Внесення штучного адитивного гаусівського шуму для проведення стрес-тестування обчислювальної системи є глибоко стохастичним (випадковим) математичним процесом. Функція генерації матриці шуму на основі нормального розподілу під час кожного нового виклику продукує абсолютно унікальний візуальний патерн перешкод. Це означає, що при одноразовому (поодинокому) тестуванні згенеровані шумові пікселі можуть випадковим чином перекрити ключову анатомічну ознаку об’єкта (наприклад, око дельфіна), або ж, навпаки, лягти на відносно однорідну ділянку шкіри, майже не зашкодивши алгоритмам детекції. Таким чином, оцінка швидкодії та впевненості класифікатора на основі єдиного прогону не може вважатися

репрезентативною та науково достовірною, оскільки вона сильно залежить від випадкових статистичних флуктуацій.

Для отримання максимально об’єктивної картини стійкості (робастності) системи, програмний комплекс був розширений спеціальним модулем автоматизованого збору статистики, логіка якого базується на принципах методу Монте-Карло [31]. Розроблений алгоритм виконує 10 незалежних автоматичних ітерацій (прогонів) класифікації одного й того ж тестового зображення. Під час кожного прогону система генерує нову унікальну матрицю шуму із незмінним параметром дисперсії. Після завершення повної серії тестів система агрегує отримані результати, обчислює середні значення та формує детальний аналітичний звіт.

Аналіз дисперсії (розкиду) значень апаратного часу класифікації є ключовим індикатором обчислювальної стабільності алгоритмів в умовах роботи на реальних операційних системах. Зведені статистичні характеристики швидкодії для базового високорозмірного 16-бітного простору, емпірично отримані за результатами 10 послідовних ітерацій, акумульовані та представлені у таблиці 3.10 [6].

Таблиця 3.10 – Аналіз дисперсії швидкодії методів класифікації за результатами 10 ітерацій (16-бітний простір)

Метод	Середній час (мс)	Мінімальний час (мс)	Максимальний час (мс)	Стандартне відхилення (мс)
Пошук у корзинах	3,87	3,16	4,98	≈ 0,52
Інтеграція голосів	3,44	2,80	4,67	≈ 0,48
Сигнатури	5,58	4,81	7,04	≈ 0,82

Аналіз отриманих усереднених статистичних даних переконливо доводить надзвичайно високу архітектурну стабільність розробленого алгоритму просторового хешування [30].

Стандартне математичне відхилення загального часу класифікації для оптимізованих методів пошуку у корзинах та інтеграції голосів не перевищує 0,48–0,52 мс між найшвидшою та найповільнішою ітераціями. Це свідчить про те, що обчислювальна система ефективно обробляє абсолютно різні за складністю шумові патерни з передбачуваною і стабільною швидкістю, без раптових підвисань чи непередбачуваних алгоритмічних затримок. Незначні коливання абсолютного часу в діапазоні від 2,65 до 4,98 мс не пов’язані з математичною складністю самого хешування; вони повністю пояснюються апаратними перериваннями операційної системи ПК (планувальником потоків) та неконтрольованими механізмами збирання сміття (garbage collection) у самому інтерпретаторі Python.

Окрім поглибленого аналізу швидкодії, застосування методу багаторазового стрес-тестування дозволило емпірично оцінити стабільність розпізнавання класів. У всіх без винятку 10 ітераціях для 16-бітного простору розроблені методи пошуку у корзинах та інтеграції голосів гарантовано і безпомилково ідентифікували правильний цільовий клас об’єкта (зафіксовано 100% результативність, повна відсутність хибних спрацювань). Відрив накопичених балів від найближчого стороннього конкурента під час кожної окремої ітерації залишався стабільним і коливався в безпечних межах від 85% до 96%.

Така безпрецедентна стійкість пояснюється фундаментальною природою самого методу локально-чутливого хешування: він принципово не спирається на єдину, «ідеальну» ключову точку, яка може бути легко знищена локальним шумом. Натомість, алгоритм масово оперує цілою хмарою ознак, надійно розподіленою по безлічі просторових кошиків пам’яті. Навіть якщо 75% точок будуть безнадійно спотворені випадковим гаусівським шумом і відправляться у порожні корзини (як було емпірично доведено раніше),

залишкових 25% стабільних бінарних ознак алгоритму з надлишком вистачить для формування правильного, математично обґрунтованого мажоритарного рішення, спираючись на інтеграцію голосів.

3.7 Розробка графічного інтерфейсу та інструкція користувача

Для зручності проведення масштабних статистичних експериментів, оперативного візуального аналізу результатів хешування та інкапсуляції розроблених алгоритмів у повноцінний програмний продукт, було створено кросплатформений графічний інтерфейс користувача (GUI).

Як основний інструмент побудови візуальної оболонки було обрано бібліотеку tkinter [34]. Цей вибір обґрунтовується тим, що вона є вбудованим стандартом у мові програмування Python, не потребує встановлення важких сторонніх залежностей та забезпечує стабільну роботу програми на будь-якій операційній системі. Фрагмент коду ініціалізації головного вікна та функціональних кнопок управління наведено у лістингу 3.6.

Лістинг 3.6 Ініціалізація графічного інтерфейсу застосунку та панелі керування:

```
def start_gui(db, b_16, b_12, b_8, b_5, p_16, p_12, p_8, p_5):
    root = tk.Tk()
    root.title("LSH Класифікатор - Панель керування")
    root.geometry("480x520")
    root.eval('tk::PlaceWindow . center')
    root.configure(padx=20, pady=20)

    show_kp_var = tk.BooleanVar(value=True)
    show_match_var = tk.BooleanVar(value=True)
```

```
    # Змінна для збереження вибраного рівня шуму (0, 25 або 42)
    noise_var = tk.IntVar(value=0)

    def on_run_single():
        file_path = utils.select_image_file()
        if file_path:
            process_query_image(
                file_path, db, b_16, b_12, b_8, b_5, p_16, p_12, p_8, p_5,
                noise_sigma=noise_var.get(),
                show_kp=show_kp_var.get(),
                show_match=show_match_var.get()
            )

    def on_run_stats():
        file_path = utils.select_image_file()
        if file_path:
            run_statistical_experiment(file_path, db, b_16, b_12, b_8, b_5, p_16, p_12,
p_8, p_5, noise_sigma=noise_var.get(), iterations=10)

    def on_run_matrix():
        generate_db_matrices(db, b_16, b_12, b_8, b_5, p_16, p_12, p_8, p_5)

    def on_noisy_vs_clean():
        run_noisy_vs_clean_matrix(db, b_16, b_12, b_8, b_5, p_16, p_12, p_8, p_5,
noise_sigma=noise_var.get())

    root.mainloop()
```

Головне вікно розробленого застосунку (рис. 3.12), спроєктовано за принципами мінімалізму та технічної ергономіки. Воно забезпечує максимально швидкий доступ до основних параметрів моделювання, дозволяючи змінювати умови експерименту без необхідності перезапуску ядра інтерпретатора.

Інтерфейс користувача містить наступні керуючі елементи (віджети):

– інтерактивні чекбокси візуалізації які дозволяють вмикати відображення проміжних результатів роботи детектора AKAZE та відображення геометричних ліній збігів на тестовому запиті;

– чекбокс-модифікатор накладання гаусівського шуму, що є інструментом для стрес-тестування системи, що безпосередньо імітує роботу в умовах апаратних завад матриці камери;

– головна кнопка «Аналіз фото (1 раз)» яка відкриває діалогове вікно системного провідника для вибору зображення та ініціює ручну класифікацію з повною візуалізацією процесу;

– кнопка «СТАТИСТИКА (10 прогонів)» запускає цикл автоматичних тестів на обраному зображенні для збору об’єктивних усереднених даних швидкодії та формування матриці дисперсії;

– кнопка «Матриця сумісності (ЧИСТІ vs ЧИСТІ)» виконує глобальне перехресне порівняння ідеальних еталонів між собою для визначення матриці помилок та загальної маржі впевненості;

– кнопка «Оцінка шуму (ЗАШУМЛЕНІ vs ЧИСТІ)» запускає найбільш складний аналітичний режим автоматичного порівняння зашумлених копій бази даних із чистими еталонами для перевірки стійкості алгоритмічних фільтрів.

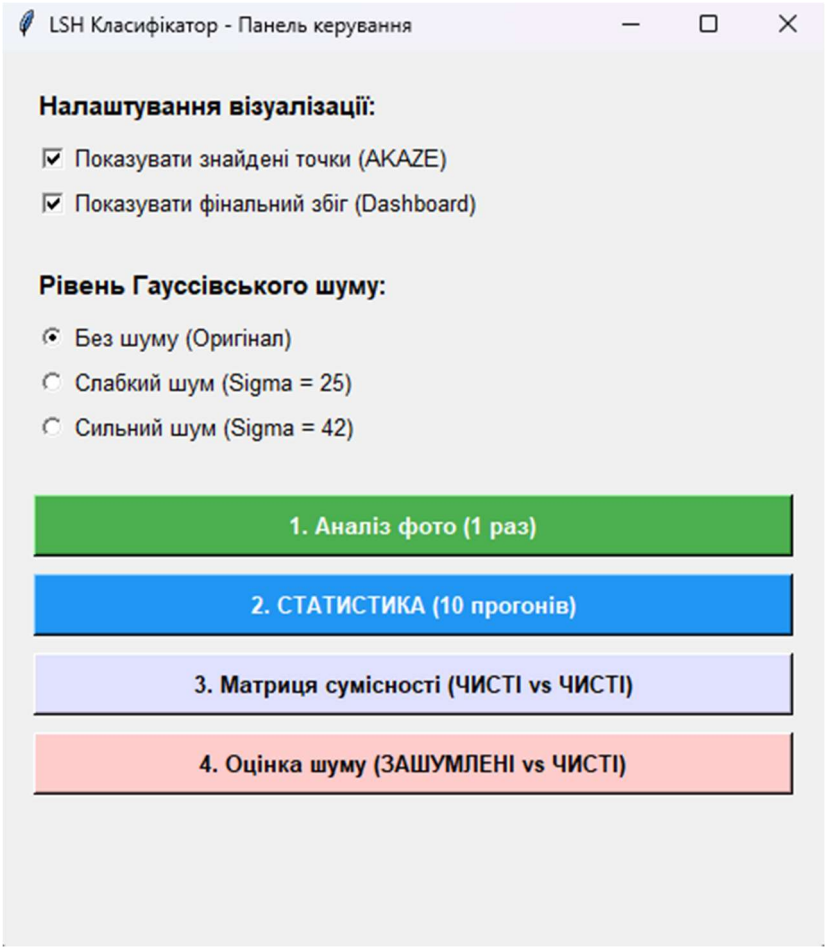


Рисунок 3.12 – Головне вікно графічного інтерфейсу застосунку

Під час виконання будь-якого з обраних режимів детальна статистика (час хешування, час пошуку, кількість накопичених балів та розрахована манхетенська відстань) автоматично виводиться у системну консоль середовища розробки. Додатково програма формує та зберігає звіти у форматі електронних таблиць Excel для подальшої побудови діаграм.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено і реалізовано завадостійку систему багатокласової класифікації зображень на основі просторового хешування. Робота охоплює повний цикл створення програмного комплексу – від аналізу теоретичних засад до реалізації оптимізованих алгоритмів та їх стрес-тестування. У ході виконання роботи було успішно вирішено всі поставлені завдання:

– проведено аналіз сучасних підходів до структурної класифікації зображень та методів випадкового проєктування, що дало можливість виявити обмеження нейромережевих підходів для роботи на граничних пристроях та обґрунтувати доцільність використання локально-чутливого хешування;

– досліджено модель екстракції локальних інваріантних ознак алгоритмом AKAZE, що дозволило ефективно формувати компактні бінарні дескриптори розмірністю 488 біт із високою стійкістю до розмиття завдяки нелінійній дифузії;

– вивчено математичний апарат локально-чутливого хешування, включаючи лему Джонсона-Лінденштрауса, що теоретично обґрунтувало можливість стиснення простору ознак із збереженням відносних метричних відстаней;

– розроблено алгоритмічну архітектуру та програмну модель мультикласового класифікатора, яка включає етап еліптичного просторового маскування, що дозволило усунути до 70–80% нерелевантних фонових ознак (водної поверхні) та суттєво підвищити відношення сигнал-шум;

– реалізовано та інтегровано в систему три паралельні критерії оцінювання результатів розпізнавання: метод пошуку у корзинах (за відстанню Хеммінга), метод інтеграції голосів та метод сигнатур (за манхеттенською відстанню), що забезпечило гнучкість налаштування класифікатора;

– спроєктувано та реалізовано кросплатформений графічний інтерфейс користувача на базі бібліотеки tkinter із вбудованими інструментами автоматизованого збору аналітичної статистики та імітаційного моделювання завад, що забезпечило точність й зрічність експериментального тестування;

– проведено експериментальне тестування швидкодії та робастності системи.

Експериментально доведено, що використання 16-бітного адресного простору у комбінації з розробленим методом інтеграції голосів забезпечує безпрецедентний виграш у загальній швидкодії. Час класифікації становить у середньому 3,44 мс, що прискорює процес пошуку у понад 230 разів порівняно з еталонним алгоритмом повного перебору (який потребує 802,18 мс), при збереженні 100% правильності класифікації. Багаторазове стрес-тестування за методом Монте-Карло з накладанням адитивного Гауссівського шуму підтвердило архітектурну стабільність алгоритму: коефіцієнт впевненості методу інтеграції голосів у 16-бітному просторі склав 0,37, а стандартне відхилення часу класифікації не перевищило 0,52 мс.

Розглянуто перспективи подальшого вдосконалення системи, які включають перенесення обчислень на мобільні платформи, адаптацію алгоритмів хешування для обробки потокового відео в реальному часі та інтеграцію модулів динамічного налаштування порогів класифікації.

Отримані практичні результати та розроблений програмний комплекс можуть бути безпосередньо впроваджені та використані у системах екологічного моніторингу навколишнього середовища, підсистемах візуальної навігації автономних безпілотних апаратів та сервісах швидкого пошуку об’єктів у великих мультимедійних базах даних.

Результати роботи апробовано у вигляді 1 тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [23].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tymchyshyn, R., Volkov, O., Gospodarchuk, O., & Bogachuk, Y. (2018). Modern approaches to computer vision. *Control Systems and Computers*, *278*(6), 46–73.

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

3. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks. *International Journal of Academic Engineering Research*, *7*(9), 64–72.

4. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks. *International Journal of Academic Applied Research*, *7*(11), 134–145.

5. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In *Explainable deep learning artificial intelligence* (pp. 125–146). Academic Press.

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

7. 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.

8. Gorokhovatsky, V. (2014). *Structural analysis and intellectual data processing in computer vision*. SMIT.

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

10. Гороховатський, В. О., & Творошенко, І. С. (2026). *Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія*. ХНУРЕ.

11. 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.

12. Gorokhovatskyi, V., Putyatin, Y., & Stolyarov, V. (2017). Research of effectiveness of structural image classification methods using cluster data model. *Radio Electronics, Computer Science, Control*, *3*(42), 78–85.

13. Gorokhovatskyi, V., Peredrii, O., Tvoroshenko, I., & Markov, T. (2023). Distance matrix for a set of components of structural description as a tool for creating an image classifier. *Advanced Information Systems*, *7*(1), 5–13.

14. 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.

15. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

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

17. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2026). Feature space quantization and application of metrics in structural methods of image recognition. *IEEE Access*, *14*, 17812–17824.

18. 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.

19. 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), Article 446.

20. Гороховатський, В. О., & Творошенко, І. С. (2022). *Аналіз багатовимірних даних за описом у формі множини компонент: монографія*. ХНУРЕ.

21. 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), Article 1425.

22. Gorokhovatsky, V. A., & Putyatin, Ye. P. (2009). Image likelihood measures of the basis of the set of conformities. *Telecommunications and Radio Engineering*, *68*(9), 763–778.

23. Приймак, Є. А. (2026). Використання хешування для прискорення пошуку у комп'ютерному зорі. В *Радіоелектроніка і молодь у XXI столітті: матеріали 30-го Міжнародного молодіжного форуму* (Вип. 7, с. 141–143). ХНУРЕ.

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

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

26. Gorokhovatskyi, V., & Tvoroshenko, I. (2024). An effective method for transforming an image description into a compact vector for classification. In *Information Technology and Implementation (Satellite): Conference Proceedings* (pp. 25–28). Publishing House «Caravela».

27. Gadetska, S., Gorokhovatskyi, V., & Stiahlyk, N. (2020). Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. *CEUR Workshop Proceedings*, *2608*, 1027–1039.

28. 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.

29. Gorokhovatskyi, V., Tvoroshenko, I., & 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), 113–125.

30. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025). Image description compression in classification structural methods. *IEEE Access*, *13*, 43631–43641.

31. 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.

32. Пелешко, Д. Д., & Юрчак, І. Ю. (2023). *Інтелектуальні системи комп'ютерного зору*. Видавництво Львівської політехніки.

33. Гороховатський, В. О., & Творошенко, І. С. (2025). Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. В *Проблеми інформатики та моделювання (ПІМ-2025): тези XXV міжнародної науково-технічної конференції* (с. 38–43). НТУ «ХПІ».

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

35. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. In *Proceedings of the International Scientific Symposium «Intelligent Solutions-S»: Computational Intelligence* (pp. 25–27).