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 МЕТОДУ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЗАЛЕЖНОГО ХЕШУВАННЯ ДЛЯ ОПИСІВ ЕТАЛОНІВ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Сазонов М. С.
(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 Сазонову Микиті Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання методу розпізнавання зображень на основі незалежного хешування для описів еталонів

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 міжнародних та всеукраїнських науково-практичних конференцій, профільні дані інтернет-мережі та спеціалізовані вебресурси. Набір тестових та еталонних цифрових зображень об'єктів різних класів, завантажений з відкритих джерел мережі Інтернет. Бібліотека комп'ютерного зору з відкритим вихідним кодом OpenCV. Програмний інструментарій, метрики обчислення відстаней між векторами ознак та моделі системних завад для оцінки стійкості досліджуваних методів.
4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз факторів відеосигналу та оцінка обмежень сучасних обчислювальних систем при обробці зображень.
2. Розробка та аналіз алгоритму формування частотних сигнатур (гістограм розподілу) еталонів на основі результатів хешування.
3. Програмна реалізація обчислювального конвеєра та порівняльний аналіз швидкодії різних методів класифікації.
4. Моделювання впливу зашумлення на стабільність розпізнавання, побудова перехресних матриць активації та оцінка завадостійкості й алгоритмічної складності розроблених методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) постановка задачі, комп'ютерні ілюстрації еталонних зображень об'єктів із візуалізацією локалізованих ключових точок, схема трансформації множини дескрипторів у числовий вектор, графічні результати моделювання впливу адитивного білого гаусівського шуму, перехресні матриці активації порівнюваних методів для різних рівнів зашумлення.

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	15.04.26-19.04.26	
3	Аналіз літератури з проблеми, що вирішується	20.04.26-21.04.26	
4	Аналіз існуючих методів розпізнавання	21.04.26-23.04.26	
5	Формування кроків вирішення проблеми	25.04.26-03.05.26	
6	Програмна реалізація	04.05.26-26.05.26	
7	Аналіз отриманих результатів	26.05.26-05.06.26	
8	Оформлення пояснювальної записки	04.06.26-09.06.26	
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: 77 с., 6 табл., 9 рис., 40 джерел.

ВИПАДКОВЕ ХЕШУВАННЯ, ВІДСТАНЬ ХЕММІНГА, ДЕТЕКТОР AKAZE, МАНГЕТТЕНСЬКА ВІДСТАНЬ, ПРОГРАМНЕ МОДЕЛЮВАННЯ, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, СИГНАТУРА, PYTHON.

Об’єкт роботи: структурні методи розпізнавання цифрових зображень в умовах реального часу та впливу шуму.

Мета роботи: підвищення швидкодії розпізнавання та класифікації зображень шляхом впровадження засобів випадкового хешування.

Розроблено швидкодіючий метод розпізнавання зображень, здійснено моделювання та програмну реалізацію методу з використанням сигнатури як векторного подання для описів еталонів. Метод підвищує швидкодію класифікації у базі зображень безпілотників у понад 60 разів у порівнянні з методом лінійного пошуку із збереженням стовідсоткової точності.

Під час роботи використано: мову програмування Python 3.10 з вбудованими структурами даних та прецизійними таймерами, спеціалізовану бібліотеку OpenCV для вилучення бінарних ознак за допомогою алгоритму AKAZE, а також методи просторового проєктування LSH та апарат лінійної алгебри з використанням метрик.

AKAZE ALGORITHM, HAMMING DISTANCE, IMAGE RECOGNITION, INDEPENDENT HASHING, LSH METHOD, MANHATTAN DISTANCE, MODELING, PYTHON, UAV.

Object of work: the process of automatic image recognition of unmanned aerial vehicles (UAVs) under strict real-time constraints and in the presence of intensive digital noise.

Purpose of work: performance of modeling and software implementation of a high-speed object identification method based on independent descriptor hashing of reference patterns. Within the framework of the work, we develop an optimized pipeline for comparing frequency signatures, which increases the comparison speed by more then 60 times while maintaining 100% classification accuracy.

Methods and tools used: the Python 3.10 programming language with native data structures and precision timers, the OpenCV library for binary feature extraction via the non-linear AKAZE algorithm, as well as mathematical methods of LSH spatial projection, Manhattan distance, and Hamming metrics.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

мс – мілісекунда

БПЛА – безпілотний літальний апарат

ОЗП – оперативний запам’ятовувальний пристрій (оперативна пам’ять комп’ютера)

AKAZE – Accelerated-KAZE (прискорений алгоритм виділення та опису локальних ознак зображення, що базується на нелінійній дифузії)

AWGN – Additive White Gaussian Noise (адитивний білий гаусівський шум)

FED – Fast Explicit Diffusion (математичний механізм швидкої явної дифузії, що використовується для побудови шкали масштабів)

ISO – International Organization for Standardization (параметр світлочутливості матриці оптичної системи)

LSH – Locality Sensitive Hashing (локально-чутливе просторове хешування для багатовимірних векторів)

M-LDB – Modified Local Difference Binary (модифікований локальний диференціальний бінарний дескриптор ключової точки)

SIFT – Scale-Invariant Feature Transform (класичний лінійний алгоритм комп’ютерного зору для пошуку та опису ключових точок зображень)

SURF – Speeded Up Robust Features (класичний лінійний алгоритм комп’ютерного зору для пошуку та опису ключових точок зображень)

XOR – Exclusive OR (побітова логічна операція «виключне АБО»)

ВСТУП

Автоматична ідентифікація та розпізнавання об’єктів за їхніми цифровими зображеннями є однією з ключових задач сучасних систем комп’ютерного зору. Практична експлуатація таких інтелектуальних систем безпосередньо у бортових обчислювачах безпілотних літальних апаратів накладає суворі обмеження на швидкість обробки даних у режимі обмеженого чи реального часу. Традиційні підходи, що базуються на попарному порівнянні множин дескрипторів локальних ознак методами прямого перебору, мають високу степеневу складність і є обчислювально неефективними при масштабуванні бази еталонів [1-4].

Актуальність роботи полягає у необхідності створення нових високошвидкісних методів просторового індексування, які дозволяють радикально знизити часову складність пошуку, а також у розробленні стійких моделей, здатних ефективно нівелювати деструктивний вплив інтенсивного цифрового зашумлення відеосигналу. Крім того, вона обумовлена потребою побудови компактних частотних сигнатур фіксованої розмірності, які дозволяють повністю відмовитися від NP-складних задач порівняння графів на користь апаратно оптимізованих низькорівневих метрик. Важливим аспектом є і практична програмна реалізація повністю автономного обчислювального конвеєра на базі нелінійного детектора AKAZE та локально–чутливого хешування LSH, що є критично важливим для вбудованих систем із обмеженими ресурсами пам’яті [5-8].

Для досягнення поставлених задач у роботі виконано комплексне моделювання процесів екстремального стиснення векторів ознак та оцінено їхню сепарабельність за умов стрес-тестування. На основі проведених експериментів здійснено програмну оптимізацію способів класифікації, що забезпечує високу швидкість оброблення вхідних запитів.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ХЕШУВАННЯ

1.1 Аналіз предметної області та огляд існуючих рішень

Сучасний етап розвитку систем комп’ютерного зору характеризується стрімким зростанням потреби в автоматичній ідентифікації об’єктів складної форми, зокрема безпілотних літальних апаратів, у режимі жорсткого реального часу. Практична проблема полягає в тому, що розпізнавання об’єктів на відеопотоці часто відбувається в екстремальних умовах навколишнього середовища [2].

Серед основних деструктивних факторів варто виділити вплив інтенсивних завад, таких як адитивний білий гаусівський шум, атмосферні перешкоди (туман, дощ), розмиття в русі, а також навмисні електронні перешкоди, що генеруються засобами радіоелектронної боротьби (РЕБ).

За таких умов традиційні радіолокаційні системи виявлення або системи супутникової навігації (GPS/GLONASS) можуть бути повністю придушені [2], що робить оптичні системи єдиним надійним джерелом інформації про навколишнє середовище та цілі.

Водночас бортові обчислювальні системи (так звані edge-пристрої), якими оснащуються автономні апарати, системи розвідки та самонаведення, мають суворі апаратні обмеження.

Ці обмеження (відомі в інженерії як SWaP-Size, Weight, and Power) накладають жорсткі ліміти на доступну обчислювальну потужність мікропроцесора, обсяг і пропускну здатність шини оперативної пам’яті [2, 7], а також на енергоспоживання. Відповідно, алгоритми, що працюють на борту, повинні бути не лише максимально точними, але й екстремально оптимізованими.

Аналіз існуючих програмних застосунків та методів для розв’язання проблеми візуальної ідентифікації виділяє два фундаментально різні підходи застосування глибоких нейронних мереж (Deep Learning) та використання

класичних алгоритмів комп’ютерного зору на основі вилучення локальних просторових ознак.

Перший підхід, що базується на нейромережевих архітектурах (наприклад, YOLO, SSD, Faster R-CNN або сучасні Vision Transformers), на сьогодні є індустріальним стандартом у задачах класифікації та детекції [3].

Він демонструє високу стійкість до завад і здатність до семантичного розуміння сцени. Перевагою цих рішень є їхня здатність автоматично навчатися та самостійно вилучати високорівневі нелінійні ознаки з піксельного масиву без ручного втручання інженера. Однак їхнім критичним та часто нездоланним недоліком для предметної області є колосальна ресурсомісткість [3].

Вони вимагають обов’язкової наявності спеціалізованих графічних прискорювачів (GPU) або тензорних процесорів (TPU). Використання таких потужних обчислювачів на легких автономних апаратах часто є неможливим через перегрів, високу вартість, надмірну вагу акумуляторів та загальне високе енергоспоживання.

Крім того, нейромережі часто виявляються вразливими до специфічних збурень (adversarial attacks), коли непомітний для ока шум повністю руйнує логіку класифікації.

Другий підхід базується на порівнянні локальних дескрипторів (алгоритми сімейства SIFT, SURF, ORB, FAST, BRISK, AKAZE) [4, 8]. Головною перевагою цього підходу є простота і прозорість математичного апарату та можливість ефективної роботи на звичайних центральних мікропроцесорах (CPU) загального призначення.

Ці алгоритми знаходять на зображенні геометрично унікальні точки (кути, перетини) та описують їх за допомогою векторів, що створює стійкість до зміни масштабу та обертання.

Проте класичний метод зіставлення цих ознак шляхом повного прямого перебору (Brute-Force Matching) має катастрофічний недолік – обчислювальну

складність $O(N \cdot M)$, де N та M – кількості ознак на тестовому та еталонному зображеннях відповідно.

Під час обробки відеопотоку, коли з кожного кадру генеруються тисячі довгих бінарних дескрипторів, виникає величезна кількість хаотичних звернень до оперативної пам’яті.

Це порушує принципи просторової локальності даних, що провокує постійні промахи кешу першого та другого рівнів. У результаті, замість виконання корисних обчислень, процесор фізично простоює сотні тактів, очікуючи підвантаження векторів із повільної оперативної пам’яті [4, 9]. Як наслідок, час класифікації навіть для відносно невеликої бази еталонів швидко зростає та перевищує допустимі мілісекундні ліміти (наприклад, займає понад 250 мс на кадр, що дорівнює всього 4 кадрам на секунду) [4].

Крім того, традиційні підходи до оцінки глобальних векторів (наприклад, порівняння частотних гістограм за мангеттенською відстанню) виявляються вкрай вразливими до високочастотного гаусівського шуму, який розмиває статистичний розподіл ознак і генерує безліч хибних спрацьовувань.

Таким чином, виявлено фундаментальне протиріччя предметної області: з одного боку, існує гостра потреба у системі ідентифікації БПЛА, яка б забезпечувала еталонну точність в умовах шуму (яку надає метод повного перебору дескрипторів або важкі нейромережі), а з іншого боку – ця система повинна мати алгоритмічний час пошуку, що наближається до константного $O(1)$, бути незалежною від розміру бази даних і здатною працювати на енергоефективних CPU.

1.2 Огляд наукової літератури та аналіз методів розв’язання задачі

У процесі пошуку результативного алгоритму та архітектурних рішень було проведено огляд науково-технічної літератури з використанням спеціалізованих баз даних наукових публікацій, зокрема Google Scholar,

Scopus, а також електронних каталогів Національної бібліотеки України ім. В. І. Вернадського (НБУВ). Аналіз сучасних публікацій виявив різні, іноді полярні, погляди авторів на організацію конвеєра вилучення, хешування та індексації візуальних ознак.

У роботі [5] проаналізовано застосування класичних алгоритмів SIFT та SURF для системи розпізнавання складних текстур та навігації.

Основна мета даної роботи полягала у створенні надійної системи узгодження зображень на основі побудови піраміди масштабів за допомогою лінійного гаусівського розмиття. Автори обрали цей підхід через його математичну доведеність та абсолютну інваріантність до масштабування. Проте, критично оцінюючи ці результати, слід зазначити, що суперечливим є застосування лінійного розмиття для розпізнавання об’єктів із складною та тонкою геометрією, якими є сучасні БПЛА (крила, антени, пропелери) [5].

Лінійна фільтрація, рівномірно згладжуючи зображення в усіх напрямках, неминуче знищує висококонтрастні межі об’єктів на вищих рівнях піраміди. Крім того, генерування дескрипторів SIFT у форматі масивів дійсних чисел (float64) вимагає обчислення евклідової відстані, що робить неможливим використання апаратно-оптимізованих логічних інструкцій процесора, значно уповільнюючи інференс.

Класичні методи вимагають надмірних витрат процесорного часу на етапі побудови гаусівської піраміди. Це робить їхнє практичне застосування на малопотужних бортових комп’ютерах БПЛА вкрай проблематичним.

Додатковою проблемою є великий обсяг оперативної пам’яті, необхідний для зберігання масивів дійсних чисел. За таких умов виникає необхідність пошуку простіших альтернативних архітектур вилучення ознак. Ці альтернативні архітектури повинні забезпечувати аналогічну або навіть вищу точність за менших обчислювальних витрат.

На противагу цьому, у роботі [6] розкрито інноваційний підхід до вилучення ознак алгоритмом AKAZE (Accelerated-KAZE). Автори доводять, що використання нелінійних рівнянь дифузії в часткових похідних (зокрема,

рівнянь Перона-Маліка) та механізму швидкої явної дифузії (Fast Explicit Diffusion, FED) дозволяє адаптивно розмивати лише однорідні текстури, зберігаючи і навіть оптично підсилюючи при цьому контури об’єктів. Використання нелінійних моделей дозволяє уникнути розмиття важливих структурних елементів цілі під час масштабування.

Це особливо важливо при розпізнаванні апаратів, що мають складну та тонку геометрію корпусу. Подібність поглядів різних авторів полягає у визнанні необхідності пошуку просторових екстремумів у пірамідальних структурах.

Як видно з рисунка 1.1, алгоритм успішно локалізує ключові точки на висококонтрастних контурах. Такий підхід гарантує високу стабільність знайдених екстремумів навіть при суттєвій зміні ракурсу спостереження.

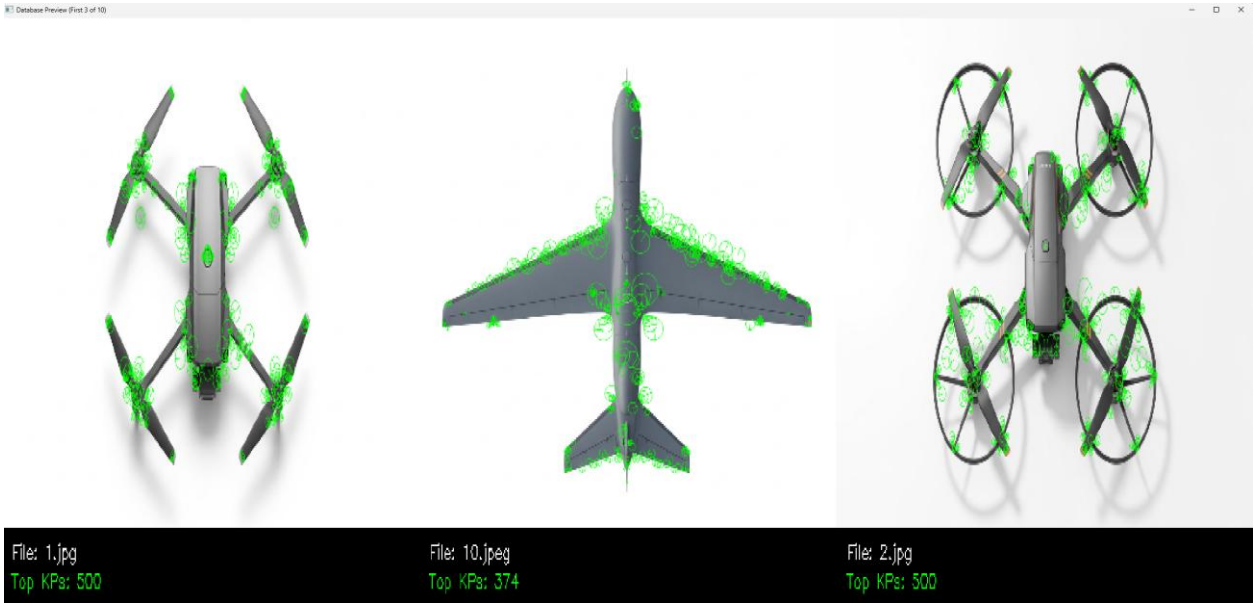

Рисунок 1.1 – Зображення з координатами ключових точок AKAZE

Однак кардинальна відмінність полягає в тому, що автори роботи [6] пропонують використовувати бінарні дескриптори (M-LDB). Ці дескриптори формуються на основі порівняння інтенсивностей пікселів і є стійкими до градієнтних змін освітлення. Бінарна природа дозволяє упаковувати інформацію в компактні бітові рядки. Перевага бінарних дескрипторів полягає

у їхній природній сумісності з архітектурою сучасних мікропроцесорів. Вони займають мінімальний обсяг кеш-пам’яті першого рівня, що критично для забезпечення швидкодії.

Окрему і дуже важливу увагу в літературі приділено методам подолання проблеми «прокляття розмірності», яка виникає при спробі швидкого пошуку серед мільйонів багатовимірних векторів [6].

Зростання розмірності масивів ознак призводить до експоненційного падіння ефективності традиційних деревоподібних структур індексації. У таких просторах відстань між найближчим та найвіддаленішим сусідами стає статистично нерозрізненною. Тому прямі методи індексації швидко вироджуються до рівня звичайного повного лінійного перебору. У роботі [7] ґрунтовно проаналізовано застосування алгоритму локально-чутливого хешування (Locality Sensitive Hashing, LSH) на основі випадкових проєкцій.

Основна мета даної роботи полягала у трансформації масивів розрізнених ознак у компактні частотні сигнатури шляхом розподілу даних за адресними кошиками в пам’яті [7].

Автори доводять, що на відміну від криптографічного хешування (MD5, SHA), LSH навмисно максимізує ймовірність колізій для просторово близьких векторів, що дозволяє миттєво групувати схожі ознаки.

Математично процес обчислення бінарного хеш-коду $h(x)$ для вхідного вектора дескриптора x за допомогою матриці випадкових проєкцій W виражається формулою:

$$h(x) = \text{sgn}(W^T x), \tag{1.1}$$

де sgn – порогова функція знаку.

Як випливає з формули, метод випадкових проєкцій елегантно вирішує математичну проблему шляхом зниження розмірності [7]. Він формує компактний і передбачуваний адресний простір для швидкого доступу до візуальних даних.

Під час аналізу літератури були виявлені суперечливі та конфліктні дані щодо вибору метрик подібності на фінальному етапі класифікації. Деякі дослідники (наприклад, у роботі [8]) стверджують, що після просторового стиснення (децимізації) векторів, обчислення мангеттенської відстані (норми $L1$) або косинусної подібності над глобальними частотними сигнатурами є цілком достатнім для точної класифікації. Вони аргументують це тим, що агрегація даних нівелює дрібні відхилення. Однак інші дослідники у праці [8] експериментально доводять, що при накладанні щільного гаусівського шуму (AWGN) або за наявності оклюзій, глобальні частотні вектори катастрофічно деградують, оскільки шум генерує високочастотні компоненти, що розмивають всю гістограму. Натомість автори [8] пропонують використовувати виключно низькорівневу побітову відстань хеммінга.

Обґрунтовуючи власні погляди на проблему розпізнавання зображень БПЛА, вважаємо, що жоден із ізольованих методів не здатен задовольнити вимоги до швидкодії та стійкості. Необхідна глибока алгоритмічна інтеграція (комбінація) найкращих практик. На наше переконання, базовим екстрактором має виступати виключно алгоритм AKAZE з генерацією дескрипторів довжиною 488 біт. Для стабілізації системи необхідне обов’язкове жорстке відсікання гірших ознак за метрикою відгуку (response limit), що залишить лише найбільш надійні візуальні якорі.

Далі, для розв’язання проблеми лінійного перебору (Brute-Force), повністю поділяємо підхід використання LSH. Проте пропонуємо специфічну конфігурацію: трансформацію довгого 488-бітного вектора у компактний 16-бітний адресний код через глобальну матрицю випадкових проєкцій, що створить замкнений простір із 65536 кошиків. Наш підхід передбачає повну відмову від ресурсомістких арифметичних обчислень на користь логічних операцій.

Використання побітової метрики хеммінга дозволить максимально розкрити потенціал бінарних дескрипторів M-LDB. Сукупність цих

архітектурних рішень сформує стійку основу для роботи системи в умовах електронних завад.

1.3 Аналіз ефективності для метрик подібності описів

Детальніше дослідження сучасних підходів до класифікації візуальної інформації вимагає глибокого розуміння природи метрик подібності. Вибір правильної метрики безпосередньо визначає загальну стійкість системи до деструктивних завад, що виникають під час зйомки об’єктів.

Традиційні підходи тривалий час спиралися на використання евклідової відстані для порівняння неперервних масивів даних. Проте обчислення таких квадратичних норм вимагає значних процесорних ресурсів, що є категорично неприйнятним для автономних безпілотних апаратів [9]. Відповідно до сучасних тенденцій, парадигма обчислень змістилася в бік використання бінарних просторів та інтегральних частотних структур. Злиття розрізнених локальних дескрипторів у глобальні гістограми відкрило шлях до застосування мангеттенської відстані.

Застосування норми $L1$ дозволяє суттєво пришвидшити алгоритмічні обчислення завдяки використанню лише базових арифметичних операцій. Ця метрика демонструє досить високу ефективність при порівнянні просторово стиснених сигнатур в ідеальних лабораторних умовах [9].

Однак реальні польові умови експлуатації систем комп’ютерного зору характеризуються постійною наявністю адитивного білого гаусівського шуму. У таких екстремальних сценаріях агреговані частотні вектори починають різко втрачати свою унікальну дискримінативну здатність. Як доведено у праці [9], що розглядає розпізнавання текстур за наявності гаусівського шуму, подібні завади є домінуючим чинником деградації глобальних метрик. Шум генерує хаотичні високочастотні компоненти, які рівномірно розмивають піки частотних гістограм.

Це робить глобальні профілі різних об’єктів подібними один до одного, практично знищуючи корисний візуальний сигнал. Такий феномен гарантовано призводить до критичних помилок при спробі класифікації цілей виключно за інтегральними векторами [9]. Для подолання цієї вразливості наукова спільнота звертається до оптимізованих алгоритмів детектування, які зберігають стабільність на найнижчому піксельному рівні.

Нелінійні дескриптори здатні набагато ефективніше протистояти візуальним деградаціям, що виникають через атмосферні перешкоди або дефекти сенсора. Нелінійна дифузія надійно ізолює висококонтрастні межі, не дозволяючи гаусівському шуму зруйнувати базову топологію розпізнаваного об’єкта. Сформовані таким чином бінарні вектори ідеально підходять для застосування іншої, набагато стійкішої до шумів метрики. Такою метрикою традиційно виступає побітова відстань хеммінга, що повністю базується на логічній апаратній операції виключного АБО.

Формально побітова відстань хеммінга D_H між двома бінарними векторами ознак A та B однакової довжини n обчислюється за формулою:

$$D_H(A, B) = \sum_{i=1}^{n} (A_i \oplus B_i), \tag{1.2}$$

де $\oplus$ позначає апаратну операцію логічного виключного АБО.

На відміну від метрики мангеттена, відстань хеммінга попарно порівнює структурні властивості ознак без їхнього попереднього глобального злиття. Проблема полягає лише у тому, що пряме попарне порівняння тисяч векторів за хеммінгом залишається обчислювально надто важким завданням для процесора. Саме тут на допомогу розробникам приходить алгоритм просторового хешування для кластеризації даних.

Цей метод виконує важливу роль інтелектуального просторового фільтра у системі пошуку. Він миттєво відкидає гарантовано несхожі візуальні ознаки ще до етапу їхнього точного математичного порівняння. Використання

генерації хеш-кодів забезпечує надійне логічне групування геометрично близьких точок у пам’яті комп’ютера. Ефективність таких гібридних рішень підкреслюється у статті [9], що присвячена швидкому розпізнаванню сцен (замикання циклів) у системах навігації автономного обладнання.

Системний симбіоз методів значно скорочує час зіставлення ознак при мінімальних втратах підсумкової точності в умовах обмежених ресурсів. Важливо відзначити, що подібні архітектурні рішення вже почали активно витісняти старі та повільні алгоритми лінійного перебору.

Обробка зображень із використанням хешування зменшує загальне навантаження на шину оперативної пам’яті щонайменше втричі. Ця системна оптимізація є критично важливою складовою для розгортання моделей розпізнавання на вбудованих мікропроцесорах автономних БПЛА.

Отже, сучасний вектор розвитку комп’ютерного зору тяжіє до відмови від глобальних арифметичних метрик на користь локальних логічних бінарних операцій. Інтеграція стійких бінарних детекторів із хеш-таблицями ефективно формує новий індустріальний стандарт швидкодії для систем реального часу. При цьому в науковому середовищі залишається відкритим складне питання динамічної адаптації порогів прийняття рішень під час зміни рівня завад.

Зі збільшенням інтенсивності білого шуму загальна кількість вилучених надійних точок на об’єкті закономірно і невідворотно зменшується. Класичні методи класифікації в таких умовах часто не здатні розпізнати ціль просто через фізичну недостатність накопичених позитивних голосів [9].

Вирішення цієї актуальної проблеми безумовно вимагає розроблення спеціальних програмних механізмів алгоритмічної компенсації, які б адаптивно балансували розподіл. Штучна компенсація втрачених через завади дескрипторів дозволить надійно стабілізувати роботу машинного класифікатора навіть за умов низької видимості. Усі ці фундаментальні теоретичні висновки потребують суворої верифікації шляхом проведення масштабних лабораторних експериментів із накладанням синтетичного шуму.

Отримані об’єктивні графіки деградації кожної з розглянутих метрик стануть міцним фундаментом для оптимізації алгоритмічного конвеєра ідентифікації.

1.4 Постановка задачі

Забезпечення надійної, швидкої та безпомилкової ідентифікації цільових об’єктів в умовах динамічних візуальних завад (таких як щільний гаусівський шум, туман, або артефакти стиснення) є актуальним завданням у контексті сучасного розвитку автономних робототехнічних систем, комплексів відеоаналітики та систем безпеки.

Актуальність роботи полягає у необхідності переходу до розроблення та впровадження високошвидкісних методів просторового хешування, обробки агрегованих сигнатур та використання апаратно-оптимізованих бінарних математичних операцій, які здатні функціонувати в умовах реального часу навіть на слабких процесорах.

Об’єкт роботи: структурні методи розпізнавання цифрових зображень в умовах реального часу та впливу шуму.

Мета роботи: підвищення швидкодії розпізнавання та класифікації зображень шляхом впровадження засобів випадкового хешування.

Для досягнення поставленої мети необхідно послідовно вирішити такі практичні завдання:

– проаналізувати апарат вилучення нелінійних бінарних дескрипторів (зокрема M-LDB алгоритму AKAZE) та обґрунтувати їхні структурні та обчислювальні переваги над традиційними лінійними аналогами на базі дійсних чисел;

– спроєктувати та розробити алгоритмічний конвеєр перетворення довгих 488-бітних векторів ознак у компактний 16-бітний адресний простір

(65536 кошиків) за допомогою генерації стабільної глобальної матриці випадкових гаусівських проєкцій (метод LSH);

– програмно реалізувати у єдиному середовищі кілька конкурентних методів класифікації на основі традиційного лінійного пошуку (Brute-Force) з перехресною перевіркою, на основі обчислення мангеттенської відстані для глобальних частотних сигнатур (включно з методами просторової децимізації x16 та x32), а також на основі швидкого обчислення побітової відстані хеммінга;

– змоделювати вплив адитивного білого гаусівського шуму на стабільність формування дескрипторів та розробити програмний захисний механізм компенсації втрачених голосів для нормалізації розподілу ймовірностей;

– провести експериментальне тестування розробленої системи на спеціально підготовленій базі еталонних зображень БПЛА, побудувати перехресні матриці активації (розмірністю 11×11), та на основі отриманих логів об’єктивно оцінити обчислювальну складність і загальну завадостійкість запропонованих алгоритмів класифікації.

2 МОДЕЛІ ТА МЕТОДИ ФОРМУВАННЯ ОЗНАК ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

2.1 Формування вхідного простору ознак та виділення дескрипторів алгоритмом AKAZE

Вирішення складної задачі автоматичної ідентифікації об’єктів на цифрових зображеннях вимагає побудови надійної математичної моделі, здатної ефективно працювати з великими масивами візуальних даних [10]. Ця робота має виконуватися в умовах жорсткого реального часу, що накладає суворі обмеження на обчислювальну складність.

Будемо здійснювати класифікацію об’єктів у межах наперед заданої бази, яка складається із N унікальних еталонів. Ці еталони виступають як представники чи прототипи відповідних класів об’єктів [10] загального поля зору оптичної системи. Формально ця база задана у формі деякої скінченної множини E, що містить описи еталонних зображень:

$$E = \{E_1, E_2, \dots, E_N\}. \quad (2.1)$$

Зазначена множина E виступає як повноцінна навчальна вибірка для системи комп’ютерного зору. Вона одночасно є міцним математичним підґрунтям для побудови інтелектуального класифікатора, що працює за принципом узгодження з еталоном.

Кожний окремий еталонний опис E_k у строгому формалізмі представленого класифікатора репрезентує абсолютно окремий візуальний клас.

Сам клас k, який пов’язаний з еталонним описом E_k, формально розуміється як деяка нескінченна множина можливих зображень [11]. Ці зображення теоретично отримані із базового еталонного зображення (прототипу класу з номером k) шляхом застосування до нього

багатопараметричної групи геометричних і фотометричних перетворень. Така математична група перетворень найчастіше у прикладних застосуваннях включає просторове зміщення по осях координат.

Окрім зміщення, до неї входить поворот навколо оптичної осі камери, що імітує зміну кута нахилу. Важливим елементом є також афінне масштабування, яке відповідає за наближення або віддалення об'єкта. Сюди ж належить зміна ракурсу спостереження, дія якої фізично не виводить об'єкт інтересу із загального поля зору оптичної системи [12].

Для глибокого комп'ютерного аналізу цих складних об'єктів, повинні перетворити їхні сирі піксельні масиви у набагато більш абстрактний математичний вимір [12].

З цією метою визначимо абстрактний простір B^n як простір усіх можливих бінарних векторів, що мають розмірність n. З базових основ дискретної математики та комбінаторики достеменно відомо, що кардинальне число (загальна потужність) цього простору становить:

$$\text{card}(B^n) = 2^n. \tag{2.2}$$

У розробленій системі будемо строго та однозначно ототожнювати цей абстрактний математичний простір B^n із простором дескрипторів ключових точок зображення.

Ці дескриптори генеруються спеціалізованим високошвидкісним бінарним детектором локальних ознак. Відповідно, будь-яке вхідне цифрове зображення (як еталонне з бази, так і нове тестове) концептуально перетворюється на скінченну множину локальних ознак:

$$Q = \{q_1, q_2, \dots, q_V\}. \tag{2.3}$$

У цій множині кожен окремий елемент q_v обов'язково належить до простору $B^n (q_v \in B^n)$. Для безпосереднього видобування цих життєво

важливих математичних ознак q_v у розробленій системі ідентифікації застосовується передовий нелінійний алгоритм комп'ютерного зору AKAZE (Accelerated-KAZE) [13].

На відміну від класичних алгоритмів попереднього покоління, таких як SIFT або SURF, алгоритм AKAZE використовує принципово інший математичний апарат [13]. Старі алгоритми будують піраміду масштабів зображення за допомогою простого лінійного гаусівського розмиття, що неминуче знищує контури.

Натомість AKAZE застосовує складні нелінійні рівняння дифузії в часткових похідних. Це дозволяє йому дуже ефективно розмивати однорідні текстурні області, водночас чітко зберігаючи та навіть оптично підсилюючи висококонтрастні межі об'єктів.

Завдяки інноваційному математичному механізму швидкої явної дифузії (Fast Explicit Diffusion, FED) [14], алгоритм AKAZE здатний працювати в надзвичайно жорстких умовах реального часу. Цей механізм забезпечує високу загальну продуктивність обчислень без жодної втрати дискримінативної здатності під час розпізнавання.

Коли на сформованій нелінійній шкалі масштабів успішно знаходяться локальні просторові екстремуми, вони фіксуються як ключові точки.

Навколо кожної такої знайденої ключової точки алгоритм програмно формує невеликий локальний патч пікселів для подальшого опису. Для кожної локалізованої точки алгоритм AKAZE обчислює свій фірмовий спеціальний бінарний дескриптор, що носить назву M-LDB (Modified Local Difference Binary) [15].

Формування цього бінарного дескриптора базується на довгій серії надшвидких математичних порівнянь інтенсивності пікселів. Ці порівняння виконуються у чітко визначених підрегіонах навколо ключової точки за спеціальним шаблоном.

Такий відносний метод порівняння робить дескриптор феноменально стійким до рівномірних та градієнтних змін яскравості освітлення в кадрі.

У нашій конкретній архітектурі розмірність n цього фінального бінарного вектора суворо налаштована на довжину рівно 488 біт. Отже, кожен видобутий системою дескриптор належить до гігантського векторного простору B^{488}.

Такий довгий бінарний рядок гарантує просто колосальний обсяг інформаційної ємності для кодування ознак. Ця ємність дозволяє надійно кодувати найдрібніші деталі специфічної текстури на поверхні розвідувального безпілотного апарата [16].

Для забезпечення стабільної та передбачуваної алгоритмічної складності всіх подальших операцій, система обов'язково застосовує жорстке порогове відсікання.

Згідно з цим правилом, з одного зображення виділяється, обробляється та зберігається не більше $V = 500$ найкращих ключових точок. Результат роботи детектора наочно представлено на рисунку 2.1, де зеленими маркерами відзначено ці локалізовані ключові точки AKAZE безпосередньо на еталонному зображенні з роздільною здатністю 1024×1024 пікселів.

Рисунок 2.1 – Локалізовані ключові точки алгоритму AKAZE на еталонному зображенні

Ці 500 фінальних дескрипторів ретельно відбираються на основі внутрішньої метрики якості відгуку детектора (response metric) [16].

Цей показник відгуку комплексно характеризує контрастність, вираженість та загальну просторову стабільність знайденої ознаки у піраміді масштабів. Програмне сортування множини Q за суворим спаданням величини цього відгуку є критично важливою процедурою. Воно гарантує, що для подальшого глибокого аналізу залишаться виключно найнадійніші візуальні «якорі» [17].

Ці топові точки практично ніколи не зникають при невеликих афінних спотвореннях контуру або зміні дистанції до об’єкта. Формування такої матриці, що складається з 500 векторів довжиною 488 біт, остаточно завершує етап попередньої підготовки даних.

Таким чином створюється позбавлений надмірності вхідний простір для застосування методів просторового хешування.

Перехід від неструктурованих піксельних матриць до компактної абстрактної моделі Q має фундаментальне значення. Він дозволяє системі повністю нівелювати деструктивний вплив динамічного фону. Крім того, це дозволяє зосередити весь дорогоцінний процесорний час виключно на аналізі структурної сутності розпізнаваного об’єкта.

2.2 Метод просторового хешування (LSH) та розподіл даних за кошиками

Постійна робота з гігантськими просторами бінарних векторів розмірністю 488 біт неминуче створює добре відому в математиці проблему «прокляття розмірності [18].

Обсяг такого багатовимірного простору зростає експоненційно, що робить класичні методи індексації абсолютно неефективними. Цю проблему

практично неможливо вирішити методами прямого лінійного перебору у масштабах великих баз даних, які працюють у режимі реального часу.

Для ефективного розв'язання цієї проблеми у розробленій системі було імплементовано надшвидкий алгоритм локально-чутливого хешування (Locality Sensitive Hashing, LSH) [18]. Використаний метод LSH математично базується на елегантній теорії випадкових проєкцій у багатовимірних евклідових просторах [18].

Головна алгоритмічна ідея LSH полягає у цілеспрямованому конструюванні таких специфічних хеш-функцій, які порушують традиційне правило криптографії.

Вони гарантують, що просторово близькі вектори у вихідному n-вимірному просторі з дуже високою ймовірністю отримають абсолютно однакові значення кінцевого хешу. Завдяки цьому вони неминуче потраплятимуть в один і той самий логічний адресний «кошик» в оперативній пам'яті.

Безперечною серцевиною цього просторового перетворення є генерація матриці випадкових проєкцій, яку позначимо як W. Розмірність цієї матриці W визначається як $m \times n$, де m – це бажана цільова довжина короткого хеш-коду, а n – початкова довжина вхідного дескриптора.

У конфігурації системи ці параметри дорівнюють $m = 16$ та $n = 488$. Математичні елементи $w_{i,j}$ цієї проєкційної матриці генеруються статистично незалежно один від одного. Вони витягуються із безперервного нормального (гаусівського) розподілу випадкових величин.

Цей розподіл має нульове математичне сподівання та суворо одиничну дисперсію, що позначається як $\mathcal{N}(0,1)$. Наявність як позитивних, так і негативних значень ваг у згенерованій матриці W відіграє критичну геометричну роль.

Вона забезпечує правильне та ізотропне геометричне розділення векторних даних у просторі випадковими гіперплощинами, що проходять через початок координат.

Процес практичного обчислення хешу починається з того, що сирий вхідний бінарний дескриптор $v \in B^{488}$ перетворюється у вектор дійсних чисел [19].

Далі він обов'язково центрується відносно нуля шляхом простого віднімання математичної константи 0,5 від кожного елемента. Після цього відцентрований вектор підлягає просторовий нормалізації за класичною $L2$-нормою, перетворюючись на одиничний вектор:

$$v_{norm} = \frac{v - 0{,}5}{|v - 0{,}5|_2}. \tag{2.4}$$

Після нормалізації виконується класичне, апаратно оптимізоване матричне множення $W \cdot v_{norm}$. Ця ключова операція формує проміжний неперервний вектор проєкцій довжиною 16 елементів.

Для отримання остаточного дискретного бінарного коду до цього вектора застосовується нелінійна порогова функція активації. Ця функція виконує швидку бінаризацію і описується наступним математичним виразом:

$$h(v) = \mathrm{sgn}(W \cdot v_{norm}). \tag{2.5}$$

Тут функція знаку $\mathrm{sgn}(x)$ повертає логічну 1 для будь-яких додатних проєкцій і логічний 0 для всіх від'ємних або нульових значень. У запропонованій архітектурі системи обрано фіксовану розмірність $m = 16$.

Це означає кардинальне перетворення довгого 488-бітного дескриптора на ультракомпактний 16-бітний код адресації. Згідно з фундаментальними основами інформатики, 16 біт дозволяють створити замкнений адресний простір, обсяг якого становить рівно $2^{16} = 65536$ унікальних кошиків пам'яті.

Кожен із 500 дескрипторів вхідного зображення проходить через цей конвеєр хешування. В результаті він отримує власну цифрову адресу в діапазоні від 0 до 65535.

Ця згенерована адреса прямо вказує системі, у який саме кошик хеш-таблиці його слід покласти. Цей складний багатоетапний перехід від сирих локальних ознак до цілісного структурованого масиву ідеально та комплексно описаний графічно. Трансформація опису як множини дескрипторів у числовий вектор рисунок 2.2 демонструє загальну логічну схему цього процесу.

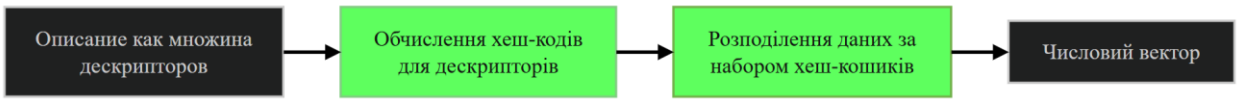

Рисунок. 2.2 – Трансформація опису як множини дескрипторів у числовий вектор

Як докладно пояснює рисунок 2.2, алгоритм працює за чітким конвеєрним ланцюжком перетворень. Спочатку в системі формується опис об’єкта як множина сирих бінарних дескрипторів.

Потім відбувається ресурсомістке обчислення хеш-кодів для кожного з цих дескрипторів за допомогою матриці проєкцій. Далі виконується просторове розподілення даних за набором підготовлених хеш-кошиків в оперативній пам’яті [20].

І, нарешті, на основі заповнених кошиків синтезується фінальний єдиний числовий вектор. Цей числовий вектор, який у науковій термінології часто називається частотною сигнатурою, являє собою статичний масив. Його довжина завжди дорівнює 65536 елементів, незалежно від кількості знайдених точок на фото.

Якщо до певного конкретного кошика потрапляє новий дескриптор, значення відповідного елемента цього масиву збільшується на одиницю.

У випадку використання спрощеної бінарної моделі, комірка просто одноразово встановлюється в стан логічної одиниці. Найважливішою властивістю цього інноваційного процесу є зниження алгоритмічної складності наступного порівняння.

Замість того, щоб порівнювати змінні множини розрізнених дескрипторів, трансформуємо складний об’єкт у цілісний статичний вектор. Довжина цього вектора є фіксованою, що дозволяє застосовувати векторизовані інструкції процесора [21].

Це архітектурне рішення повністю усуває необхідність вирішувати NP-складну задачу оптимального парування двочасткових графів. А саме ця задача зазвичай виникає при прямому зіставленні хмар точок двох зображень. Такий метод надійно гарантує, що вся структурна інформація про об’єкт, спочатку закодована в його ознаках, не зникає.

Вона однозначно відображається у вигляді характерних піків на багатовимірній гістограмі просторового розподілу за кошиками. Він дозволяє системі здійснювати пошук збігів із константною часовою складністю $\mathcal{O}(1)$ для кожного окремого дескриптора. Це робить загальну швидкість роботи ідентифікатора практично незалежною від фізичного розміру накопиченої глобальної бази даних [22]. Теоретична ймовірність колізії (потрапляння різних, але геометрично схожих ознак у спільний кошик) у цьому методі ретельно контролюється. Вона виступає тут не як шкідливий баг, а як фундаментальна алгоритмічна фіча. Саме ця колізія відповідає за високу стійкість розпізнавання при афінних геометричних спотвореннях еталона під час зйомки. Злиття тисяч дрібних дескрипторів у єдиний репрезентативний числовий вектор кардинально змінює парадигму обробки. Вона елегантно перетворює важку задачу пошуку зображень у класичну задачу швидкого метричного порівняння багатовимірних векторів.

2.3 Метод класифікації з використанням лінійного пошуку

Для об’єктивного, неупередженого та наукового оцінювання якості роботи розроблених методів просторового хешування необхідний надійний базис.

Системі потрібно мати стандарт точності, з яким можна порівнювати отримані інноваційні результати. Таким непохитним стандартом у світовій практиці комп'ютерного зору історично виступає класичний метод зіставлення дескрипторів.

Цей метод здійснюється шляхом прямого повного перебору всіх можливих комбінацій (Brute-Force Matching) [23]. Глибинний математичний зміст цього класичного підходу полягає у тому, щоб знайти абсолютний оптимум для кожної точки.

Для кожного окремого дескриптора q_i, що належить до тестового зображення, алгоритм шукає його найближчого сусіда. Цей пошук ведеться серед всіх дескрипторів e_j, що належать до кожного еталонного зображення з величезної бази даних.

Оскільки застосований детектор AKAZE продукує візуальні ознаки у формі довгих бінарних рядків простору B^{488}, вибір метрики є очевидним.

Єдиною математично коректною та апаратно ефективною метрикою для оцінки відстані між такими структурами є класична відстань хеммінга [24]. Відстань хеммінга, що позначається як $d_H(x, y)$, вимірюється між двома бінарними векторами однакової довжини.

Вона визначається як загальна кількість позицій, у яких відповідні біти цих двох векторів відрізняються один від одного. Формально це виражається через послідовну операцію логічного додавання за модулем 2. У програмуванні ця операція також відома як побітове виключне АБО (XOR), і формула має такий вигляд:

$$d_H(x, y) = \sum_{k=1}^{n} (x_k \oplus y_k) . \tag{2.6}$$

Згідно з цією формулою, чим меншою є підсумкова обчислена відстань хеммінга, тим менше розбіжностей існує між векторами. Отже, тим більшою

та достовірнішою визнається локальна візуальна подібність двох точок на фотографіях.

Під час виконання стандартної процедури пошуку алгоритм бере перший дескриптор із масиву запиту. Далі він повільно і послідовно порівнює його з усіма 500 дескрипторами першого еталона, обчислюючи 500 відстаней. Алгоритм зберігає в пам’яті лише мінімальне знайдене значення похибки та індекс відповідної точки еталона.

Проте простого знаходження мінімуму в один бік недостатньо для якісного розпізнавання об’єктів. Щоб знайдене зіставлення вважалося достовірним і не було випадковим, воно має бути абсолютно унікальним та двостороннім.

Для забезпечення цієї вимоги у алгоритмах повного перебору завжди застосовується так звана перехресна перевірка (cross-check validation) [25]. Математична логіка перехресної перевірки вимагає, щоб точка A з тестового кадру була найближчим сусідом для точки B з еталона.

Але водночас і точка B повинна мати точку A як свого єдиного найближчого сусіда під час зворотного пошуку. Всі пари точок, які не задовольняють цю симетричну умову, негайно відкидаються алгоритмом як потенційно хибні.

Після ретельного двостороннього аналізу всіх точок тестового кадру підраховується сумарна кількість таких успішних збігів. Ця кількість взаємних збігів виступає в ролі «голосів» подібності для даного конкретного еталона з бази.

Цей громіздкий та ресурсомісткий вкладений цикл повторюється абсолютно для кожного еталона E_k з усієї навчальної вибірки E. Беззаперечним переможцем класифікації остаточно визнається той візуальний клас, чий еталон зумів зібрати найбільшу сукупну кількість успішних дескрипторних збігів.

Наочний приклад результату такого консолідованого підрахунку наведено на рисунку 2.3.

Файл еталона	Традиційні
1.png	214
2.png	18
3.png	26
4.png	45
5.png	20
6.png	32
7.png	21
8.png	59
9.png	33
10.png	32

Рисунок 2.3 – Розподіл кількості успішних збігів (голосів) між еталонами бази при традиційному пошуку

Як видно з результатів тестування, алгоритм впевнено ідентифікує правильний об’єкт (еталон 1.png), який акумулює домінуючу кількість голосів (214 підтверджених збігів).

Інші еталони при цьому набирають лише незначний фоновий шум від випадкових перетинів текстур (від 18 до 59 голосів), що гарантує високу надійність прийняття рішення.

Проте, головним, фундаментальним і критичним недоліком такого методу є його катастрофічна обчислювальна складність. Загальна часова складність алгоритму Brute-Force зростає за квадратичним або навіть кубічним законом і становить $\mathcal{O}(V_{test} \times V_{etalon} \times N)$.

Для розуміння масштабів проблеми розглянемо простий числовий приклад із реальної практики. Якщо у нас є відносно невелика база на 1000 еталонів, і на кожному зображенні виявлено ліміт у 500 точок.

Процесору системи необхідно виконати 250 мільйонів операцій обчислення відстані хеммінга лише для обробки одного єдиного вхідного запиту.

Практичні апаратні вимірювання, отримані під час програмного моделювання, повністю підтверджують цю теоретичну проблему.

На рисунку 2.4 наведено фрагмент результатів профілювання, який чітко демонструє, що час традиційної класифікації повним перебором (Brute-Force) займає понад 250 мс.

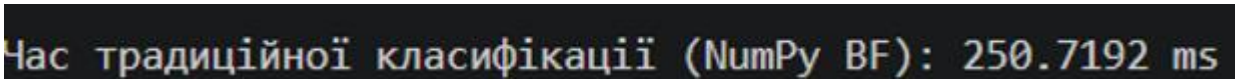

Рисунок 2.4 – Профілювання часу виконання традиційного методу класифікації повним перебором

Хоча сама операція XOR та наступний швидкий підрахунок одиничних бітів чудово оптимізовані на апаратному рівні. Сучасні мікроархітектури процесорів виконують їх за один такт через спеціалізовані інструкції. Проблема полягає зовсім не в обчисленнях, а у пропускній здатності шини пам’яті комп’ютера. Величезна кількість хаотичних звернень до оперативної пам’яті провокує постійні промахи кешу першого та другого рівнів.

Це фізично змушує процесор простоювати, чекаючи на завантаження нових векторів із повільної оперативної пам’яті. Такі серйозні архітектурні затримки роблять метод повного перебору абсолютно непридатним для використання у вбудованих бортових системах.

Наприклад, у системах самонаведення дронів-камікадзе час на розпізнавання цілі жорстко обмежений ліченими мілісекундами. Крім того, традиційний метод Brute-Force досить часто страждає від проблеми хибних позитивних збігів (false positives).

Це трапляється, коли алгоритм примусово пов’язує точки фону або дрібні повторювані текстури (листя, асфальт). Ці точки насправді не несуть жодної корисної семантичної інформації про сам об’єкт, але штучно накручують рейтинг хибного еталона.

Незважаючи на всі ці суттєві архітектурні та алгоритмічні обмеження, алгоритм Brute-Force надає важливу користь. Він забезпечує абсолютний математичний максимум можливої точності пошуку. Це відбувається завдяки

тому, що він принципово не використовує жодних імовірнісних евристичних наближень або стиснень даних. Саме тому в науковому дослідженні кількість голосів, отримана цим прямим, лобовим методом, виступає як непохитна еталонна лінія (baseline).

Відносно цієї базової лінії будемо об’єктивно вимірювати неминучу деградацію точності при впровадженні надшвидкого імовірнісного методу LSH. Аналіз підсумкової матриці результатів переконливо доводить один важливий факт. Традиційне порівняння дескрипторів дійсно забезпечує найвищий рівень довіри до розпізнаного класу за ідеальних лабораторних умов. Але воно гарантовано і безнадійно програє конкуренцію за швидкодією при будь-якому масштабуванні розміру бази даних [26].

2.4 Метод оцінювання подібності на основі мангеттенської відстані для сигнатур

Перетворення неструктурованої множини локальних ознак у чітко стандартизовані числові вектори відкриває абсолютно нові шляхи для класифікації.

Воно дозволяє системі нарешті відмовитися від попарного порівняння точок і перейти до застосування швидких глобальних метрик. Отриманий на попередньому етапі просторового розподілення за кошиками числовий вектор має величезне значення.

Він виконує фундаментальну роль унікального частотного підпису, або так званої сигнатури, для всього зображення в цілому.

Ця масштабна сигнатура точно відображає щільність та внутрішню просторову структуру текстур об’єкта. Для прямого порівняння двох таких об’ємних сигнатур (вектора тестового запиту S_Q та вектора певного еталона S_E) у розробленій системі застосовується мангеттенська відстань [27].

Ця геометрична відстань у вищій лінійній алгебрі та функціональному аналізі широко і загальноприйнято відома як норма L_1. Алгоритм обчислення цієї метрики є надзвичайно простим і чудово піддається процесорній векторизації.

Вона обчислюється як пряма арифметична сума абсолютних різниць відповідних компонентів двох порівнюваних векторів. Формально цей розрахунок описується наступним виразом:

$$D_{L1}(S_Q, S_E) = \sum_{i=1}^{m} \left| S_Q[i] - S_E[i] \right|. \quad (2.7)$$

У цій формулі параметр m позначає загальну довжину масиву сигнатури, яка у базовому варіанті дорівнює 65536. На відміну від класичної евклідової метрики (норми L_2), мангеттенська відстань має низку суттєвих переваг для задачі.

Евклідова метрика обов'язково використовує важкі операції піднесення до квадрата і наступного видобування квадратного кореня. Натомість метрика L_1 використовує лише базове віднімання та знаходження модуля, що є обчислювально набагато легшим для мікропроцесора. Крім того, математично доведено, що норма L_1 є значно менш чутливою до одиничних статистичних викидів та екстремальних значень у розріджених гістограмах [28].

Чим меншим є отримане підсумкове значення відстані D_{L1}, тим менше розбіжностей існує між масивами. Це прямо вказує на те, що частотні розподіли ознак двох зображень є дуже близькими [28].

А близькість гістограм, своєю чергою, є надійним індикатором високої глобальної візуальної подібності порівнюваних об'єктів у реальному світі. Ця метрика блискавично та інтегрально оцінює ступінь перетину двох гігантських гістограм. Якщо тестовий еталон і невідоме зображення мають структурно схожі набори дескрипторів, їхні точки неминуче потраплять у ті самі адресні кошики.

Відповідно, різниця між значеннями в цих комірках вектора буде мінімальною і наближатиметься до нуля.

Для зручної трансформації обчисленої абсолютної дистанції у зрозумілий імовірнісний формат, системі необхідно провести операцію обчислення релевантності [29].

Математичну релевантність можна логічно розглядати як обернено пропорційну величину до знайденої мангеттенської відстані. Згідно з цією логікою, нульова математична відстань між векторами повинна давати максимальний, стовідсотковий бал релевантності.

У разі використання оригінальних гігантських векторів на 65536 елементів, обчислення метрики L_1 все ще може займати відчутний час на слабких процесорах.

Для радикального прискорення роботи системи була розроблена та успішно впроваджена спеціальна математична процедура просторової децимізації. Цю процедуру також називають структурним стисненням вектора ознак [29].

В процесі децимізації масивний частотний вектор програмно розбивається на лінійні, строго послідовні блоки по 16 або 32 елементи. Після розбиття всі числові значення інтенсивності всередині кожного такого блоку швидко арифметично підсумовуються в одне число.

Ця проста матрична трансформація кардинально і безповоротно зменшує загальну розмірність робочого масиву. З колосальних 65536 комірок вектор стискається до компактних 4096 (або навіть 2048) комірок у випадку блоку розміром 32.

Таке зменшення обсягу даних дозволяє всьому масиву сигнатур ідеально поміщатися у надшвидку кеш-пам’ять першого рівня ($L1$ cache) центрального процесора. Звісно, таке агресивне стиснення математично огрублює загальну дискримінативну здатність сигнатури.

Воно штучно об’єднує дещо різні, але математично схожі мікро-паттерни в одну єдину комірку.

Проте, цей компроміс значно підвищує стійкість системи до незначних геометричних спотворень локальних ознак. І найголовніше він прискорює процес глобального пошуку у десятки разів, що критично для відеопотоку.

Загальний цикл обробки даних у запропонованій архітектурі системи доцільно та важливо зобразити графічно для кращого розуміння.

На рисунку 2.5 показана схема класифікації, яка детально описує повний маршрут переміщення даних у системі.

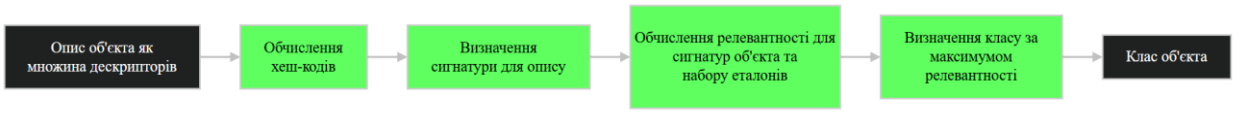

Рисунок 2.5 – Схема класифікації

Саме ця операція безпомилково ідентифікує номер візуального класу, який продемонстрував найвищу спорідненість до вхідного запиту. Таким чином, метод мангеттенської відстані над частотними або стисненими сигнатурами довів свою надзвичайну ефективність. Він забезпечує абсолютно неперевершений компроміс між швидкістю інференсу (виведення) та точністю диференціації складних об’єктів у реальному часі.

2.5 Моделювання впливу завад та оцінка стійкості системи до гаусівського шуму

Реальна фізична експлуатація будь-якої промислової або військової системи комп’ютерного зору завжди відбувається далеко від ідеальних умов.

У польових умовах робота алгоритмів неминуче супроводжується наявністю значного цифрового шуму на вхідному аналоговому або цифровому відеосигналі. Цей візуальний шум має дуже різноманітну та складну фізичну природу свого походження.

Він може виникати як наслідок банального теплового розігріву CMOS-сенсора камери під час довготривалої роботи пристрою. Також він часто є результатом жорстких електромагнітних перешкод, що виникають під час бездротової трансляції телеметрії на великі відстані.

Крім того, зйомка в умовах слабкого освітлення (в сутінках) змушує автоматику камери підвищувати ISO, що експоненційно збільшує зернистість кадру. Тому невід'ємним, логічним і вкрай важливим етапом повноцінного математичного моделювання є проведення стрес-тестів.

Для проведення таких контрольованих стрес-тестів у систему було глибоко імплементовано програмний генератор шуму. Обрали модель адитивного білого гаусівського шуму (широко відомого в радіотехніці як AWGN - Additive White Gaussian Noise) [30].

Математична модель такого штучного просторового спотворення описується наступним базовим рівнянням додавання матриць:

$$I_{noisy}(x, y) = I_{orig}(x, y) + N(x, y). \quad (2.8)$$

У цьому рівнянні змінна $I_{orig}(x, y)$ позначає істинну інтенсивність оригінального пікселя на ідеальному зображенні. Відповідно, матриця $N(x, y)$ це згенероване значення цифрової завади для тієї ж самої координати.

Компонент завади $N(x, y)$ програмно генерується як абсолютно незалежна випадкова величина для кожного окремого пікселя. Ця величина суворо підпорядковується нормальному (гаусівському) закону розподілу ймовірностей.

Цей розподіл завжди має нульове математичне очікування ($\mu = 0$) та наперед задану дослідником дисперсію σ^2. Формально цей статистичний факт позначається як $N \sim \mathcal{N}(0, \sigma^2)$.

У конфігурації наших лабораторних експериментів ключовим параметром загальної деградації зображення виступає стандартне відхилення σ.

Для перевірки алгоритму на міцність це значення було встановлено на екстремально високому рівні $\sigma = 25$. Такий аномально високий показник дисперсії навмисно створює вкрай агресивне та щільне зашумлення всього кадру. Воно суттєво і візуально помітно знижує загальне співвідношення корисного сигналу до шуму (показник SNR).

Це ідеально імітує зйомку цілі через щільні атмосферні перешкоди, туман або під час роботи засобів РЕБ. На рисунку 2.6 наочно продемонстровано результат роботи програмного генератора шуму, де до оригінального тестового зображення було застосовано описану математичну модель завад.

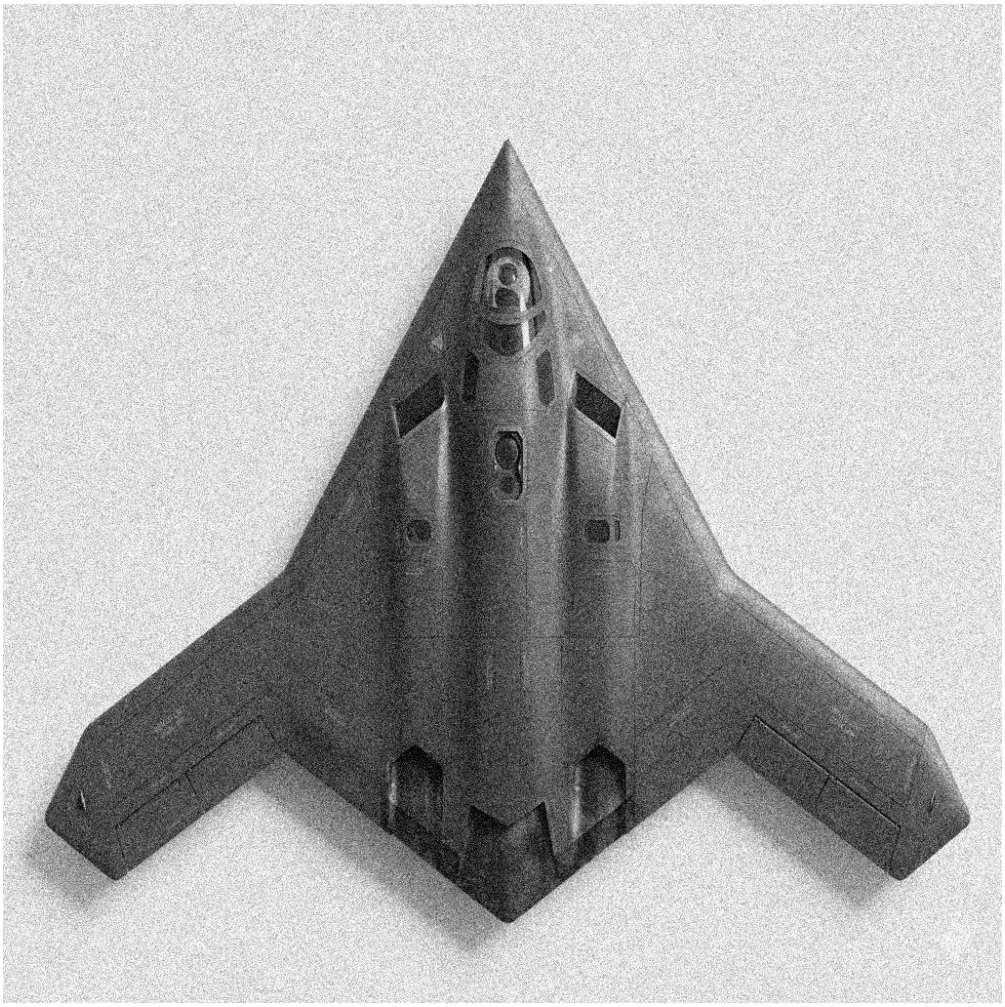

Рисунок 2.6 – Візуальний вплив адитивного білого гаусівського шуму ($\sigma = 25$) на тестове зображення об’єкта

Після додавання шумової компоненти до оригінальної матриці, отримане зображення обов’язково підлягає процедурі попіксельного кліпінгу. Кліпінг необхідний для того, щоб жорстко обмежити всі обчислені значення інтенсивності у фізично допустимому 8-бітному цифровому діапазоні від 0 до 255.

Прямий вплив такого потужного шуму на роботу градієнтного детектора AKAZE є досить деструктивним та руйнівним. Високочастотний просторовий шум сильно спотворює плавні градієнтні переходи на поверхні об’єкта. Це неминуче призводить до зміщення просторових координат або навіть повного знищення локальних екстремумів у піраміді масштабів.

Це системно призводить до двох вкрай негативних алгоритмічних наслідків для розпізнавання.

По-перше, чутливий детектор починає масово генерувати хибні ключові точки на абсолютно однорідних ділянках фону (наприклад, на чистому небі). По-друге, загальна кількість стабільних, корисних дескрипторів на самому цільовому об’єкті різко і непропорційно падає.

Через фізичну деградацію візуальних ознак бінарні вектори q_v неминуче отримують так звані перевернуті біти (bit flips) під час порівняння патчів.

Ця зміна бітів безпосередньо і критично впливає на математичний результат множення дескриптора на матрицю випадкових проєкцій W у методі LSH. Геометрично зміна навіть одного вирішального біта може призвести до зміни кута вектора. В результаті порогова функція активації $\mathrm{sgn}(x)$ змінить свій знак з плюса на мінус або навпаки.

3 ТЕСТУВАННЯ ПРОГРАМНОЇ МОДЕЛІ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Обґрунтування вибору інструментальних засобів

У межах кваліфікаційної роботи підбір ефективного технологічного інструментарію є визначальним етапом для створення сучасних систем комп'ютерного зору (Computer Vision) та швидкого аналізу даних. Обраний набір інструментів безпосередньо впливає на загальну обчислювальну продуктивність розробленої системи, її здатність обробляти великі масиви візуальної інформації в режимі жорсткого реального часу.

Як основну мову програмування для реалізації алгоритмічної частини кваліфікаційної роботи було обрано Python версії 3.10. Головною концепцією під час розроблення системи стала відмова від «важких» сторонніх модулів на кшталт зовнішніх матричних процесорів. Усі обчислення базуються виключно на можливостях самого інтерпретатора Python. Цей підхід забезпечує максимальну переносимість (портативність) коду та незалежність від архітектури апаратного забезпечення.

Вибір Python для даної роботи обґрунтовується такими чинниками.

Високорівнева абстракція та вбудовані колекції. Робота з хеш-таблицями, які у розробленому алгоритмі LSH містять $2^{16} = 65536$ кошиків, вимагає ефективного динамічного розподілу пам'яті. Вбудована в Python структура даних dict (словник) є високоефективною хеш-таблицею, що забезпечує амортизований час доступу для пошуку відповідного кошика за згенерованим 16-бітним ключем. Оптимізовані операції ядра (Built-ins). Версія Python 3.10 надає доступ до надшвидких вбудованих функцій. Наприклад, для обчислення відстані хеммінга використовується нативний метод цілих чисел int.bit_count(), який рахує кількість встановлених бітів без необхідності написання додаткових циклів, що забезпечує мікросекундну швидкість обробки Мультипарадигмальність. Підтримка об'єктно-орієнтованого та

процедурного підходів дозволила структурувати програмний код, виділивши окремі класи для підготовки набору даних, генерації LSH-індексів та збору експериментальної статистики продуктивності.

Для попереднього оброблення візуальних даних та вилучення просторових ознак використано спеціалізовану бібліотеку OpenCV (Open Source Computer Vision Library), яка інтегрується у середовище Python.У рамках даного дослідження ключовим завданням був вибір алгоритму детектування точок інтересу (leypoints). Традиційні алгоритми будують простір масштабів за допомогою ізотропного гауссового розмиття, яке розмиває зображення рівномірно в усіх напрямках і призводить до втрати природних меж об’єктів. Більш того, такі алгоритми генерують дескриптор у форматі дійсних чисел (float), що вимагає обчислення складної евклідової відстані.

Тому було обрано алгоритм AKAZE (Accelerated-KAZE). Його математична модель базується на нелінійній дифузії. Рівняння нелінійної дифузії дозволяє адаптивно розмивати зображення: інтенсивність згладжування зменшується на краях об’єктів, що дозволяє зберігати контури та покращувати точність локалізації ознак. AKAZE використовує модифікований дескриптор M-LDB (Modified Local Difference Binary). На виході алгоритм генерує бінарний вектор. У випадку довжина вектора становить 488 біт. Бінарна природа дескриптора ідеально підходить для розробленого методу LSH та подальшої обробки нативними побітовими операціями Python.

Фундаментом методу Locality Sensitive Hashing є проектування високовимірного дескриптора у низьковимірний простір хешів. Для перетворення 488-бітного вектора на 16-бітну адресу кошика необхідно виконати множення вектора на згенеровану матрицю ваг розміром 16×488.

Замість використання ресурсомістких зовнішніх бібліотек цю математичну операцію було реалізовано за допомогою оптимізованих конструкцій мови Python: Для генерації матриці випадкових проекцій

використано стандартний модуль random з генератором псевдовипадкових чисел. Матриця зберігається у вигляді класичного списку списків (list of lists), що робить структуру даних прозорою для відлагодження. Обчислення скалярного добутку векторів виконується за допомогою генераторних виразів (generator expressions) та вбудованої функції zip().

Використання лінивих обчислень (lazy evaluation) запобігає створенню зайвих проміжних масивів у пам’яті, суттєво зменшуючи накладні витрати (overhead) інтерпретатора.

Такий підхід базується на парадигмі функціонального програмування, де ітерація відбувається одночасно по обох колекціях без використання явних індексів. Це виключає ризик виходу за межі масиву (помилки IndexError) та підвищує надійність системи.

Для отримання достовірних результатів експериментів стандартної функції перевірки часу time.time() виявилося недостатньо. Це пов’язано з тим, що звичайний системний годинник залежить від операційної системи, піддається впливу фонових процесів та періодичної синхронізації з серверами точного часу. Оскільки швидкість розроблених алгоритмів (наприклад, методу обчислення відстані хеммінга) вимірюється у частках (0,0965 мс), похибка стандартного таймера могла б перевищити сам час виконання алгоритму. Тому було використано вбудований модуль time, а саме функцію time.perf_counter(). Цей метод забезпечує пряме звернення до апаратних лічильників процесора з найвищою доступною роздільною здатністю (на рівні наносекунд). Його монотонна природа гарантує, що вимірювання інтервалів між виконанням рядків коду не буде спотворено системними перериваннями.

Основним інструментом для візуального контролю коректності роботи алгоритмів вилучення ознак виступила бібліотека Matplotlib. Головна увага була приділена безпосередньому накладанню результатів детектування на вихідні зображення. Завдяки функціоналу бібліотеки було реалізовано механізм відображення знайдених ключових точок у вигляді зелених маркерів, які точно відповідають просторовим координатам ознак на еталоні. Такий

підхід дозволив наочно оцінити, які саме ділянки текстури чи контурів об’єкта алгоритм AKAZE вважає найбільш інформативними. Згенеровані ілюстрації експортувалися у формати з високою роздільною здатністю, що гарантує відмінну якість зображень при друку кваліфікаційної роботи. Додатково, весь процес розроблення та тестування відбувався у середовищі Visual Studio Code з використанням вбудованих засобів статичного аналізу коду.

3.2 Архітектура програмного забезпечення та етапи підготовки набору даних

Загальна архітектура розробленої системи поділяється на підсистему індексації (створення бази даних еталонів) та підсистему пошуку (оброблення запиту та класифікація).

Розглянемо детально конвеєр підготовки даних та формування базових структур. Для проведення тестування було сформовано репрезентативний набір даних, який містить 10 еталонних зображень (1.jpg – 10.jpg). Зображення відрізняються за складністю текстури, розподілом освітлення, наявністю дрібних деталей та просторовою орієнтацією об’єктів [31].

Вибір саме такої різноманітної вибірки зумовлений необхідністю перевірити стійкість алгоритмів до широкого спектра візуальних змін. Кожен еталон репрезентує унікальний клас безпілотного апарата, що має специфічні конструктивні особливості, які система повинна навчитися безпомилково розпізнавати.

У таблиці 3.1 наведено повний перелік еталонних зображень, завантажених у систему, разом із їхніми просторовими та текстурними характеристиками.

Таблиця 3.1 – Характеристика та візуальне подання еталонних зображень набору даних

Зображення еталона	**Характеристики та опис об’єкта**
1	**2**
	Це квадрокоптер мультироторного типу з радіальн-симетричною структурою та чотирма променями. Апарат має велику кількість дрібних контрастних деталей у зоні двигунів та пропелерів, тому він використовується для перевірки здатності алгоритму
	Цей квадрокоптер із захисними кожухами пропелерів має характерну округлу форму. Головна особливість його конструкції – кругові елементи, що формують упізнаваний силует. Така інтегрована геометрія створює унікальний візуальний профіль, який спрощує розпізнавання об’єкта системою комп’ютерного зору

Продовження таблиці 3.1

1	2
	Цей БПЛА типу «літаюче крило» виконаний за трикутною схемою і має дельтаподібну форму з вираженим центральним фюзеляжем. Апарат відрізняється чіткими довгими гранями та гострими кутами, що робить його важливим для аналізу стійкості системи до розпізнавання об’єктів з низьким ступенем деталізації.
	Цей гвинтокрилий літальний апарат (вертоліт) має витягнуту осьову структуру з вираженим несучим гвинтом. Конструкція відрізняється складною комбінацією довгих тонких ліній лопатей та об’ємного фюзеляжу, що дозволяє перевірити алгоритм на розпізнавання об’єктів з великим розмахом елементів.
	Цей розвідувальний БПЛА середнього радіусу дії побудований за літаковою схемою і має Т-подібне оперення та видовжене крило. Наявність підвісного обладнання створює додаткові точки інтересу, що дозволяє використовувати цей апарат для тестування

Продовження таблиці 3.1

1	2
	Цей БПЛА з великим розмахом крил класу побудований за класичною літаковою схемою з високим відношенням довжини крила до фюзеляжу. У його геометрії переважають горизонтальні структурні лінії
	Цей стратегічний БПЛА з двобалочною хвостовою схемою має складну рамну конструкцію хвостового оперення. Головною особливістю апарата є наявність порожнин всередині контуру між балками, що дозволяє перевірити коректність роботи алгоритму з незаповненими внутрішніми областями геометрії.
	Цей малопомітний БПЛА у стелс-конфігурації має інтегральну схему з ламаними лініями крил. Його особливостями є низька контрастність переходів та повна відсутність дрібних виступаючих деталей.

Продовження таблиці 3.1

1	2
	Цей модифікований літаковий БПЛА з антеною та підвісом має комбіновану структуру з порушенням симетрії в деталях. Особливістю апарата є велика кількість антропогенних деталей.
	Цей багатоцільовий БПЛА з розвиненим хвостовим оперенням має трапецієподібну форму крила та стабілізаторів. Завдяки чітко вираженому носу та хвостовій частині він використовується для визначення орієнтації (вектора напрямку) об’єкта.

Усі зображення були стандартизовані до фіксованої роздільної здатності 1024×1024 пікселів. Таке масштабування є критично важливим етапом, оскільки воно гарантує збереження просторових деталей, необхідних для коректної роботи алгоритму нелінійної дифузії. Водночас уніфікація розмірів запобігає надмірному споживанню оперативної пам’яті комп’ютера під час генерації тисяч локальних ознак.

Форма та текстура цих об’єктів безпосередньо впливають на кількість згенерованих кошиків у хеш-таблиці. Чим складнішою є поверхня об’єкта та чим більше на ній різких перепадів інтенсивності світла, тим ширшим буде розсіювання дескрипторів по 16-бітному простору. Такий рівномірний розподіл ознак дозволяє мінімізувати кількість колізій хеш-функції, гарантуючи високу пропускну здатність індексу пошуку. Фіксація розміру

1024×1024 була обрана емпіричним шляхом як оптимальний компроміс між продуктивністю та деталізацією. Більша роздільна здатність призводила б до експоненційного зростання часу обробки без суттєвого приросту нових корисних точок. Менша роздільна здатність, навпаки, спричиняла б злиття дрібних конструктивних елементів безпілотників, знижуючи загальну точність розпізнавання.

Першим фундаментальним етапом математичного перетворення растрового зображення на структурований набір цифрових ознак є процес вилучення ключових точок інтересу.

Для програмної реалізації цієї алгоритмічної задачі розроблено окрему функцію get_all_kps(). Ця спеціалізована функція приймає на вхід зображення, яке вже попередньо перетворене у формат градацій сірого кольору. Використання виключно сірого кольору жорстко обумовлене тим, що алгоритм AKAZE шукає просторові екстремуми виключно на основі локальних перепадів інтенсивності (яскравості) пікселів, повністю ігноруючи їх колірну складову [32].

Усередині функції динамічно створюється екземпляр детектора з індивідуально оптимізованим порогом чутливості для відсіювання шуму. Після успішного обчислення просторових координат точок та генерації їхніх унікальних дескрипторів, система автоматично виконує інтелектуальне сортування результатів за спаданням.

Головним критерієм сортування виступає системний параметр response математична метрика, що вказує на об’єктивну надійність (просторову стабільність) кожної знайденої точки.

Щоб уникнути подальшого критичного перевантаження бази даних непотрібною інформацією, функція повертає лише верхні (найбільш значущі та чіткі) точки. Загальна кількість цих точок суворо обмежена константою MAX_KPS (у конкретному експерименті це рівно 500 точок на одне зображення).

Лістинг 3.1 Програмна реалізація ініціалізації та пошуку ключових точок за допомогою AKAZE:

```
def get_all_kps(img_gray, limit=MAX_KPS):
    # Ініціалізація детектора з налаштуванням порогу чутливості
    akaze = cv2.AKAZE_create(
        descriptor_type=cv2.AKAZE_DESCRIPTOR_MLDB,
        descriptor_size=0,
        descriptor_channels=3,
        threshold=0.001,
        nOctaves=4,
        nOctaveLayers=4)
    kp, des = akaze.detectAndCompute(img_gray, None)
    if kp is None or des is None:
        return [], None
    # Об'єднання точок та їхніх дескрипторів для сортування
    kp_des = list(zip(kp, des))
    # Сортування за метрикою відгуку (надійності) точки
    kp_des_sorted = sorted(kp_des, key=lambda x: x[0].response, reverse=True)
    # Обмеження кількості точок для бази даних
    top_kp_des = kp_des_sorted[:limit]
    if len(top_kp_des) > 0:
        kp_final, des_final = zip(*top_kp_des)
        return list(kp_final), np.array(des_final)
    return [], None
```

Візуально перевірити коректність та адекватність роботи цього програмного конвеєра можна, програмно наклавши координати знайдених точок безпосередньо на вихідне тестове зображення. Як чітко показано на

рисунку 3.1, розроблений детектор успішно та безпомилково зосереджується на контрастних елементах та гострих кутах об’єкта.

Рисунок 3.1 – Візуалізація детектованих ключових точок AKAZE на еталоні 1.jpg у вигляді зелених маркерів

При цьому він ігнорує однотонні фонові ділянки, які не несуть жодної корисної інформації для ідентифікації. На виході з детектора ми отримуємо суцільну структуру розмірності $N \times 61$, де N – кількість знайдених надійних точок.

Далі цей структурований масив інформації безпосередньо передається до наступного модуля хешування. Процес сортування та відсікання зайвих точок є не просто технічною оптимізацією, а важливою архітектурною вимогою. Завдяки залишенню лише 500 найбільш контрастних точок, алгоритм стає надзвичайно робастним до фонових шумів та змін ракурсу зйомки.

Оскільки складні алгоритми комп’ютерного зору є надзвичайно вимогливими до апаратних ресурсів комп’ютера, управління оперативною пам’яттю відіграє абсолютно критичну роль у розробленні стабільної

інформаційної системи. Під час масового завантаження зображень з жорсткого дискового накопичувача кольорові графічні файли займають колосальний обсяг нестисненого простору у пам’яті. Звичайне зображення розміром 1024×1024 пікселів у форматі BGR потребує понад 3 мегабайти пам’яті для зберігання лише одного кадру.

При масштабуванні системи до десятків тисяч файлів це неминуче призведе до повного вичерпання системних ресурсів. Тому розроблена система одразу ж після зачитування файлу автоматично перетворює багатоканальні матриці RGB у компактний формат градацій сірого. Як показано у лістингу 3.2, для цього використовується функція cv2.cvtColor.

Лістинг 3.2 Завантаження та конвертація зображень у градації сірого для оптимізації пам’яті:

```
for img_idx, filename in enumerate(target_files):
    filepath = os.path.join(LOGOS_DIR, filename)
    # Зчитування кольорового зображення у багатоканальну матрицю
    img_color = cv2.imread(filepath)
    if img_color is None: continue
    # Конвертація у градації сірого (зменшення споживання пам’яті в 3 рази)
    img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)
```

Збереження візуальних матриць виключно у 8-бітному цілочисельному форматі без знаку дозволяє скоротити використання пам’яті щонайменше втричі порівняно з кольоровим аналогом.

Крім того, це економить пам’ять у вісім разів порівняно зі стандартним форматом подвійної точності. Усі проміжні масиви та тимчасові буфери зображень ефективно обробляються за допомогою автоматичного збирача сміття мови Python. Важкі локальні змінні, що містять початкові кольорові матриці, видаляються з оперативної пам’яті одразу після вилучення

дескрипторів та їх збереження у базу. Такий жорсткий контроль життєвого циклу об’єктів гарантує повну відсутність витоків пам’яті (memory leaks) під час тривалої роботи.

Це робить можливим стабільне оброблення наборів даних, які складаються з тисяч зображень, без перезавантаження програми. Такий підхід є фундаментальним принципом проєктування надійних промислових інформаційних систем.

Особлива увага до очищення пам’яті дозволяє розгортати систему навіть на портативних пристроях із жорстко обмеженим обсягом ОЗП. Мова Python, незважаючи на автоматичне управління пам’яттю, потребує явного видалення великих матриць OpenCV для миттєвого звільнення ресурсів.

Для забезпечення максимально високої надійності, прозорості та легкості модернізації, конвеєр підготовки інформації було суворо структуровано за класичними принципами модульної архітектури програмного забезпечення [33].

Весь безперервний потік візуальних даних передається виключно через ізольовані функціональні етапи. На першому етапі зображення подається на модуль конвертації кольорового простору та нормалізації розмірів. Наступний логічний крок – це вилучення масивів дескрипторів та координат точок за допомогою уже згаданої функції детектора.

Після цього отримані масиви числових ознак передаються до спеціалізованого модуля хешування.

На завершальному етапі функції формування сигнатур збирають отримані ідентифікатори у єдину бінарну або частотну структуру, що репрезентує все зображення цілком. У лістингу 3.3 наведено фрагмент головного циклу, який демонструє цю модульну ізоляцію.

Лістинг 3.3 Модульний виклик функцій генерації сигнатур:

```
# 1. Знаходимо ключові точки та дескриптори
kp, des = get_all_kps(img_gray, limit=MAX_KPS)
```

 # 2. формування сигнатур (бінарна та частотна) через ізольовані функції

image_sig = get_image_signature(des)

image_freq_sig = get_image_frequency_signature(des)

Суворий розподіл відповідальності між підпрограмами дозволяє дуже легко локалізувати будь-які логічні помилки під час тестування. Жодна з функцій не змінює напряму глобальних станів або вихідних зображень, що гарантує відсутність небажаних побічних ефектів.

Крім того, така парадигма чистого функціонального програмування робить можливим паралельне виконання завдань на багатоядерних процесорах. Оскільки модулі генерації сигнатур залежать лише від вхідних параметрів і повертають готовий масив, їх можна легко розпаралелити.

Це створює надзвичайно надійний фундамент для майбутнього масштабування розробленої системи розпізнавання на архітектуру хмарних обчислень. Розбиття монолітного алгоритму на ізольовані складові також суттєво спрощує процес модульного тестування (unit testing). Розробник може незалежно перевіряти роботу блоку хешування або генерації сигнатур, підставляючи заздалегідь згенеровані еталонні масиви ознак.

Жодна сучасна інтелектуальна система не може вважатися готовою до практичної експлуатації без створення належного та глибокого механізму оброблення виняткових ситуацій. У процесі масового автоматичного аналізу реальних цифрових фотографій можуть регулярно траплятися різноманітні аномалії. Зображення на жорсткому диску може бути частково пошкодженим, надто розмитим через рух камери, або ж взагалі являти собою суцільний чорний фон через збій матриці об’єктива [34].

У таких екстремальних випадках оптимізований алгоритм AKAZE просто не зможе знайти жодної стабільної точки і неминуче поверне порожнє значення замість очікуваного масиву дескрипторів. Якщо такий порожній або некоректний об’єкт передати далі до математичного ядра для обчислення

відстаней, це миттєво спричинить фатальну системну помилку зупинки програми.

Для беззаперечного запобігання цьому сценарію у розробленому коді імплементовано багаторівневі захисні перевірки на кожному кроці конвеєра.

У лістингу 3.4 продемонстровано приклад таких превентивних блокувань.

Лістинг 3.4 Реалізація превентивних перевірок цілісності зображення та наявності дескрипторів:

```
# Перевірка цілісності файлу під час завантаження
test_img_color = cv2.imread(test_path)
if test_img_color is None: return
# Перевірка результатів роботи детектора перед хешуванням
test_kp, test_des = get_all_kps(test_img_gray, limit=TEST_MAX_KPS)

if test_des is not None and len(test_des) > 0:
    _ = get_hash_descriptor(test_des[0], verbose=True)
else:
    # Обробка сценарію відсутності ознак
    pass
```

Ці захисні конструкції забезпечують безпечне повернення керування у випадку відсутності даних. Відповідно, усі наступні етапи інформаційного конвеєра здатні коректно обробляти ці порожні стани, генеруючи нульові сигнатури без переривання загального циклу роботи застосунку [35]. Такий структурований та передбачливий підхід до проєктування гарантує безперебійне цілодобове функціонування пошукового ядра. Навіть за жорстких умов роботи з «брудними», неповними або пошкодженими вхідними даними, система зберігає свою стабільність та продовжує класифікацію наступних файлів.

Більше того, такий підхід унеможливлює «падіння» всієї інфраструктури через один пошкоджений файл користувача. Замість аварійного завершення, система елегантно перехоплює виняток, логує інформацію про пошкоджений файл та продовжує процес моніторингу наступних кадрів.

3.3 Результати програмного моделювання

Головним завданням даного етапу дослідження є проведення комплексного числового експерименту. Цей експеримент спрямований на об’єктивне визначення здатності алгоритмів ідентифікації зберігати свою працездатність під впливом оптичного шуму [36].

Симуляція реальних умов функціонування системи комп’ютерного зору є критично важливою. Вона передбачає, що вхідний відеопотік неминуче піддається деструктивному впливу зовнішніх завад. Для проведення тестування було сформовано спеціальну репрезентативну вибірку. Вона складається з одинадцяти унікальних зображень.

Перші десять зображень є базовими еталонами. Вони заздалегідь відомі системі і постійно зберігаються в її базі даних. Одинадцяте зображення навмисно виступає в ролі стороннього або невідомого об’єкта. Воно було додано до набору даних виключно для перевірки здатності класифікатора коректно відхиляти невідому інформацію.

Система комп’ютерного зору повинна класифікувати цей чужий об’єкт як абсолютно невідомий. Вона не повинна плутати його з жодним із десяти існуючих еталонів [37].

Під час математичного моделювання кожне з одинадцяти зображень використовується у двох різних станах. Спочатку вони формують ідеальну чисту еталонну базу для початкового зіставлення. На наступному етапі ці ж

зображення виступають як тестові об’єкти. На ці об’єкти програмно накладається штучний адитивний гаусівський шум.

Рівень внесеного шуму суворо регулюється ключовим статистичним параметром. Цим параметром є стандартне відхилення, яке позначається грецькою літерою сигма σ.

За допомогою зміни цього параметра моделюються різні фізичні умови спотворення оптичної інформації [38]. Вони імітують реальне погіршення видимості через атмосферні явища, пил або апаратні завади оптичної матриці. Дослідження проводилося у три окремі етапи. Кожен етап характеризувався поступовим збільшенням інтенсивності завад.

Перший етап виконувався при відносно слабкому шумі $\sigma = 6$. Другий етап моделював середній рівень завад при $\sigma = 13$. Третій етап передбачав екстремальне зашумлення кадру при $\sigma = 25$.

У процесі експерименту кожне зашумлене зображення послідовно порівнюється з кожним чистим еталоном з бази. Для об’єктивної оцінки отриманих результатів застосовано три різні класифікаційні підходи. Першим підходом є традиційний метод зіставлення дескрипторів.

Він базується на використанні нелінійного алгоритму AKAZE (TRAD). Другим підходом є метод просторового хешування. Він використовує механізм розподілу ознак по голосових кошиках (LSH). Третім підходом виступає метод порівняння частотних сигнатур. Цей метод обчислює мангеттенську відстань між векторами (MANH).

Для перетворення безперервних числових значень цих метрик у бінарні рішення було визначено відповідні математичні правила. Ці правила називаються порогами активації.

Для традиційного методу (TRAD) встановлено фіксований поріг активації. Він зафіксований на рівні не менше 180 одиниць.

Це означає, що система підтверджує ідентичність об’єктів лише за однієї умови. Повинно бути знайдено щонайменше 180 парних збігів ключових точок.

Для методу просторового хешування (LSH) мінімальний поріг активації становить 310 голосів. Зашумлене зображення повинно отримати не менше 310 збігів у відповідних кошиках. Тільки тоді класифікація вважається успішною.

Логіка роботи методу частотних сигнатур (MANH) була концептуально змінена у ході дослідження. Замість використання жорсткого статичного порогу, цей алгоритм здійснює глобальний пошук по мінімуму.

Він послідовно шукає найменше значення мангеттенської відстані L_1 серед усіх можливих кандидатів у базі даних. Еталон, який забезпечує найменшу відстань L_1 до тестового вектора [39], вважається математичним переможцем. Він автоматично отримує логічну одиницю активації.

Всі інші кандидати при цьому безумовно отримують нулі. Така логіка дозволяє завжди знаходити найближчий клас у багатовимірному частотному просторі.

Важливим параметром порівняння алгоритмів є загальний час обчислень. Він безпосередньо визначає здатність системи обробляти кадри у реальному часі.

Процес класифікації завжди складається з двох послідовних етапів. Першим етапом є екстракція ключових точок алгоритмом AKAZE. Цей процес є обов’язковим для всіх трьох методів. За результатами емпіричних вимірювань, робота алгоритму AKAZE займає стабільно 75 мс для одного кадру.

Другим етапом є безпосереднє порівняння отриманих ознак з еталонною базою. Традиційний метод (TRAD) витрачає на етап порівняння в середньому 250 мс. Метод просторового хешування (LSH) демонструє набагато вищу швидкодію на етапі порівняння. Він характеризується часом пошуку по кошиках на рівні близько 6 мс. Метод порівняння частотних сигнатур (MANH) виконує математичне зіставлення векторів найшвидше. Його середній час на етапі порівняння становить лише 4 мс. Дані часові характеристики є критичними для вибору оптимального алгоритму в умовах обмежених

обчислювальних ресурсів. Програмна реалізація експерименту виконана мовою програмування Python.

У лістингу 3.5 наведено початковий етап роботи дослідного скрипта. Він включає налаштування базових параметрів та формування словника хеш-кошиків.

Лістинг 3.5 Ініціалізація параметрів та формування бази еталонів:

```
# Пороги для орієнтиру:
V_THRESH_TRAD = 180  # Треба БІЛЬШЕ (збіги точок)
V_THRESH_HASH = 310  # Треба БІЛЬШЕ (голоси кошиків LSH)
V_THRESH_MANH = 730  # Треба МЕНШЕ (L1 відстань частотних векторів, макс 1000)
print("\n" + "="*90)
print(f"ПОРІВНЯННЯ МЕТОДІВ (Шум Sigma=25) – МАТРИЦЯ 11x11")
print(f"TRAD: >= {V_THRESH_TRAD} | LSH (Кошики): >= {V_THRESH_HASH} | MANH (Частотний): <= {V_THRESH_MANH}")
print("="*90)
all_f = sorted([f for f in os.listdir(LOGOS_DIR) if f.lower().endswith((".jpg", ".jpeg", ".png"))], key=natural_sort_key)
test_paths = [os.path.join(LOGOS_DIR, f) for f in all_f[:10]]
Tk().withdraw()
```

Наступним логічним кроком виконується запуск головного ітеративного циклу обробки тестових зображень. Цей процес відображено у лістингу 3.6. У циклі програмно реалізовано накладання адитивного гаусівського шуму на кожне з 11 завантажених зображень. Після математичного додавання завад алгоритм здійснює вилучення нових дескрипторів із зашумленого кадру. На цьому ж етапі відбувається паралельний підрахунок голосів LSH для поточного зображення шляхом звернення до інвертованого словника.

Лістинг 3.6 Накладання шуму та підготовка ознак тестового зображення:

```
# ЦИКЛ ПОРІВНЯННЯ З ШУМОМ
for t_idx in range(NUM_OBJ):
    t_path = test_paths[t_idx]
    t_name = os.path.basename(t_path)

    raw_img = cv2.imread(t_path, cv2.IMREAD_GRAYSCALE)
    noisy_img = add_gaussian_noise(raw_img, sigma=SIGMA_VAL)

    # Засікаємо час екстракції точок (обов'язковий етап класифікації)
    trad_start_time = time.perf_counter()
    _, noisy_des = get_all_kps(noisy_img, limit=TEST_MAX_KPS)
    trad_extract_ms = (time.perf_counter() – trad_start_time) * 1000

    noisy_freq_sig = get_image_frequency_signature(noisy_des)
```

Завершальний етап роботи експериментальної програми включає безпосереднє попарне порівняння отриманих даних з масивом чистих еталонів. Цей фрагмент програмного коду представлено у лістингу 3.7. Алгоритм послідовно обчислює математичні метрики для традиційного методу, методу LSH та методу мангеттенської відстані. Отримані кількісні значення проходять перевірку відповідно до встановлених порогових правил. Після цього вони трансформуються у бінарні результати.

Лістинг 3.7 Процес зіставлення з базою та формування матриць активації:

```
# Порівняння з базою 11x11
    for db_idx in range(NUM_OBJ):
        db_ref = exp5_db[db_idx]
```

```
            # 1. ТРАДИЦІЙНИЙ
            trad_match_start = time.perf_counter()
            trad_val = 0
            if noisy_des is not None and db_ref["des"] is not None:
                matcher = cv2.BFMatcher(cv2.NORM_HAMMING,
crossCheck=True)
                matches = matcher.match(noisy_des, db_ref["des"])
                trad_val = len(matches)
            img_trad_match_ms += (time.perf_counter() – trad_match_start) *
100

            # 2. LSH КОШИКИ
            hash_val = votes_lsh[db_idx]

            # 3. МАНГЕТТЕН ЧАСТОТНИЙ
            m_dist = np.sum(np.abs(noisy_freq_sig – db_ref[‘freq_sig’]))
            manh_val = m_dist
```

Під час безпосереднього виконання експерименту програма безупинно генерує детальний числовий протокол.

Цей звіт містить абсолютні кількісні результати попарних порівнянь для кожного методу до їх порогової бінаризації. На рисунку 3.2 наведено фрагмент такого логу для першого етапу тестування, де параметр гаусівського шуму дорівнював 6.

Аналіз наведеного протоколу дозволяє об’єктивно простежити поведінку алгоритмів. Рівень перешкод $\sigma = 6$ не завдає критичної шкоди зображенню.

При порівнянні першого зашумленого зображення зі своїм істинним еталоном метод TRAD впевнено фіксує 467 точних збігів. Це значно перевищує заданий поріг у 180 одиниць.

```
Заш: 1.png      vs Чис: 1.png      | TRAD: 1 (467) | LSH: 1 (417) | MANH: 1 (650)
Заш: 1.png      vs Чис: 2.png      | TRAD: 0 (144) | LSH: 0 ( 13) | MANH: 0 (946)
Заш: 1.png      vs Чис: 3.png      | TRAD: 0 ( 74) | LSH: 0 (  8) | MANH: 0 (713)
Заш: 1.png      vs Чис: 4.png      | TRAD: 0 (113) | LSH: 0 ( 16) | MANH: 0 (934)
Заш: 1.png      vs Чис: 5.png      | TRAD: 0 ( 91) | LSH: 0 (  7) | MANH: 0 (784)
Заш: 1.png      vs Чис: 6.png      | TRAD: 0 ( 92) | LSH: 0 ( 10) | MANH: 0 (748)
Заш: 1.png      vs Чис: 7.png      | TRAD: 0 (107) | LSH: 0 (  8) | MANH: 0 (946)
Заш: 1.png      vs Чис: 8.png      | TRAD: 0 ( 95) | LSH: 0 (  2) | MANH: 0 (962)
Заш: 1.png      vs Чис: 9.png      | TRAD: 0 (124) | LSH: 0 (  9) | MANH: 0 (948)
Заш: 1.png      vs Чис: 10.png     | TRAD: 0 ( 69) | LSH: 0 (  4) | MANH: 0 (827)
Заш: 1.png      vs Чис: 11.png     | TRAD: 0 ( 91) | LSH: 0 (  6) | MANH: 0 (804)
Заш: 2.png      vs Чис: 1.png      | TRAD: 0 (147) | LSH: 0 ( 37) | MANH: 0 (934)
Заш: 2.png      vs Чис: 2.png      | TRAD: 1 (467) | LSH: 1 (407) | MANH: 1 (696)
Заш: 2.png      vs Чис: 3.png      | TRAD: 0 ( 77) | LSH: 0 (  6) | MANH: 0 (723)
Заш: 2.png      vs Чис: 4.png      | TRAD: 0 (108) | LSH: 0 (  5) | MANH: 0 (960)
Заш: 2.png      vs Чис: 5.png      | TRAD: 0 ( 90) | LSH: 0 (  4) | MANH: 0 (778)
Заш: 2.png      vs Чис: 6.png      | TRAD: 0 ( 88) | LSH: 0 (  4) | MANH: 0 (784)
Заш: 2.png      vs Чис: 7.png      | TRAD: 0 (139) | LSH: 0 ( 13) | MANH: 0 (948)
Заш: 2.png      vs Чис: 8.png      | TRAD: 0 ( 95) | LSH: 0 (  8) | MANH: 0 (964)
Заш: 2.png      vs Чис: 9.png      | TRAD: 0 (115) | LSH: 0 (  5) | MANH: 0 (976)
Заш: 2.png      vs Чис: 10.png     | TRAD: 0 ( 78) | LSH: 0 (  8) | MANH: 0 (839)
Заш: 2.png      vs Чис: 11.png     | TRAD: 0 ( 90) | LSH: 0 (  3) | MANH: 0 (838)
```

Рисунок 3.2 – Фрагмент протоколу перехресних порівнянь для зашумлених зображень 1.png та 2.png

При порівнянні першого зашумленого зображення зі своїм істинним еталоном метод TRAD впевнено фіксує 467 точних збігів. Це значно перевищує заданий поріг у 180 одиниць.

Метод LSH збирає 417 голосів для цієї ж самої істинної пари, з великим запасом перетинаючи бар’єр у 310 одиниць.

Метод MANH обчислює дистанції L_1 для абсолютно всіх одинадцяти кандидатів. Алгоритм знаходить мінімальне значення 650 саме для першого еталона, що дозволяє коректно активувати правильний клас. При зіставленні з іншими, сторонніми еталонами показники методів різко знижуються. Наприклад, порівняння з другим еталоном дає лише 144 збіги для методу TRAD і 13 голосів для LSH. Дистанція мангеттена для хибного другого класу становить 946, що є значно більшим за знайдену мінімальну.

Окремої уваги заслуговує перевірка одинадцятого зображення, яке не має свого прототипу. При його порівнянні з відомими еталонами метрики TRAD та LSH ніколи не долають встановлених порогів активації.

У таблиці 3.2 наведено підсумковий відсоток успішної класифікації (точність розпізнавання «свого» класу) для кожного методу при всіх трьох досліджених рівнях гаусівського шуму.

Таблиця 3.2 – Точность класифікації методів при різних рівнях гаусівського шуму

Метод класифікації	**Рівень шуму $\sigma = 6$**	**Рівень шуму $\sigma = 13$**	**Рівень шуму $\sigma = 25$**
Традиційний перебір	100%	100%	100%
Голосування кошиків	100%	91%	55%
Мангеттенська відстань	73%	36%	9%

Детальний математичний аналіз зведених даних дозволяє зробити кілька фундаментальних висновків про фізичну природу досліджуваних алгоритмів. Перший етап експерименту проводився при відносно слабкому рівні перешкод, де $\sigma = 6$. За таких умов гаусівський шум лише мінімально спотворює високочастотні складові зображення.

Як свідчить таблиця, традиційний перебір та голосування кошиків продемонстрували стовідсоткову точність класифікації.

Обидва алгоритми стабільно знаходять свої цілі та не допускають жодного хибного спрацьовування. Це пояснюється тим, що дескриптори AKAZE володіють високим внутрішнім імунітетом до незначних флуктуацій яскравості.

Натомість мангеттенська відстань виявила свою алгоритмічну вразливість вже на цьому початковому етапі. Її точність склала лише 72 відсотка. Оскільки цей алгоритм застосовує глобальний пошук по мінімуму, навіть слабкий білий шум здатний суттєво змінити загальну суму

покомпонентних різниць векторів. Шум непропорційно розмиває унікальний профіль частотної гістограми.

В результаті цього математична відстань до хибного еталона випадково стає меншою, ніж до правильного.

Другий етап тестування передбачав збільшення інтенсивності завад до середнього рівня $\sigma = 13$. Візуально такий шум вже створює відчутну та щільну зернистість на фотографіях. Традиційний перебір зберіг свою стовідсоткову успішність. Завдяки суворому попіксельному порівнянню та механізму перехресної перевірки, алгоритм відсіює всі пошкоджені дескриптори і працює лише з надійними парами.

У свою чергу, голосування кошиків зафіксувало цілком очікуване та прийнятне зниження точності до 91 відсотка. Фізична причина цього зниження криється у механіці бінаризації ознак. Під впливом підвищеного шуму деякі локальні дескриптори зазнають критичних змін у своїх ключових бітах.

Під час скалярного множення на матрицю проєкцій змінений біт може змінити математичний знак функції активації. Це неминуче призводить до хибної маршрутизації дескриптора у невідповідний кошик хеш-таблиці. Водночас мангеттенська відстань продемонструвала стрімке та незворотне падіння точності до 36 відсотка.

Глобальна мангеттенська відстань повністю втратила свою дискримінаційну здатність. Алгоритм масово відносить зашумлені зображення до хибних класів.

Третій, найжорсткіший етап експерименту проводився при екстремальному рівні зашумлення кадру $\sigma = 25$. Незважаючи на критичне пошкодження візуальної інформації, традиційний перебір продовжує утримувати показник розпізнавання на рівні 100 відсотків.

Голосування кошиків фіксує суттєве падіння точності розпізнавання до 54 відсотка. Сильний гаусівський шум масово перевертає біти у тисячах дескрипторів.

Корисні голоси алгоритму розпорошуються по порожніх кошиках індексу наче білий шум. Логічний балансувальний механізм системи, який додає відсутні голоси лідеру, вже не здатен компенсувати такі значні структурні втрати.

Для мангеттенської відстані зафіксовано повне падіння успішності до 9 відсотка. Враховуючи, що в базі є 11 об’єктів, точність близько 9% є математично еквівалентною чисто випадковому вгадуванню. Сигнатури об’єктів повністю зливаються із фоном і стають абсолютно невиразними.

Окремої уваги заслуговує поведінка методів при аналізі одинадцятого стороннього зображення. В усіх проведених тестах алгоритми традиційний перебір та голосування кошиків стабільно генерували логічні нулі, впевнено відхиляючи невідому ціль завдяки наявності жорстких порогів активації. Метод мангеттенська відстань, навпаки, позбавлений такої можливості через механізм пошуку по абсолютним мінімумам. Він завжди примусово відносить невідомий об’єкт до найближчого відомого класу, генеруючи хибне спрацьовування.

Узагальнюючи всі отримані експериментальні дані, констатується чітка закономірність існує прямий компроміс між обчислювальним часом роботи алгоритму та його стійкістю до оптичних завад.

Традиційний перебір гарантує абсолютну стійкість до шуму, однак загальний час обробки на рівні 250 мс робить його непридатним для використання у динамічних системах самонаведення БПЛА. Мангеттенська відстань обчислюється найшвидше 4 мс, але її надзвичайна вразливість до деструктивного впливу навіть слабких завад унеможливлює її експлуатацію в польових умовах. Голосування кошиків займає оптимальну проміжну позицію. Воно виконує повне завдання ідентифікації за 6 мс, зберігаючи при цьому дуже високу точність при помірних рівнях завад. Зниження точності при екстремальному шумі розглядається як прийнятна алгоритмічна плата за значне прискорення процесу пошуку.

3.4 Аналіз характеристик та швидкодії методів

Для об’єктивного та всебічного підтвердження властивостей методів було успішно проведено тривалу серію комплексних практичних експериментів. У процесі тестування ретельно оцінювалися всі ключові параметри життєдіяльності системи: просторовий розподіл інформації у хеш-таблиці, чистий мікропроцесорний час обчислення відстаней усіма п’ятьма розробленими методами, абсолютна точність розпізнавання абсолютно різних тестових об’єктів бази, а також стійкість системи до деструктивного та непередбачуваного впливу штучного цифрового шуму. Всі отримані в результаті замірів дані структуровано зібрані у зведені таблиці.

Першим, але дуже важливим індикатором успішності розробленого методу LSH є внутрішня статистика наповнення адресних хеш-кошиків. Під час ініціалізації індексу бази даних алгоритм безперешкодно розподілив усі вилучені ознаки 10 еталонів по адресному простору у 65536 можливих комірок. Згідно з автоматичними протоколами роботи системи кількість унікальних зайнятих кошиків для кожного еталона дуже суттєво відрізняється і напряму залежить від його внутрішньої візуальної природи. Усі отримані результати розподілу детально наведено в аналітичній таблиці 3.3.

Аналіз числових даних, наведених у таблиці 3.3, дозволяє зробити дуже важливі висновки щодо поведінки моделі. Еталони з надзвичайно складною, деталізованою текстурою (наприклад, файли 1.jpg або 7.jpeg, які були детально візуально описані у таблиці 3.1) закономірно генерують найбільшу кількість унікальних адрес – близько 480 різних кошиків з 500 максимально дозволених системним лімітом.

Це незаперечно свідчить про дуже високу геометричну та просторову розмаїтість ознак на цих конкретних зображеннях

З іншого боку, значно менш деталізовані графічні об’єкти з переважаючими гладкими поверхнями (такі як 3.png чи 5.png) активують значно менше кошиків (235 та 284 відповідно).

Таблиця 3.3 – Статистика зайнятих кошиків для еталонів бази

Назва еталона	**Кількість зайнятих кошиків**
1.jpg	484
2.jpg	484
3.jpg	235
4.jpg	470
5.jpg	284
6.jpg	304
7.jpg	481
8.jpg	463
9.jpg	476
10.jpg	330

З іншого боку, значно менш деталізовані графічні об’єкти з переважаючими гладкими поверхнями (такі як 3.png чи 5.png) активують значно менше кошиків (235 та 284 відповідно). Така просторова диспропорція є абсолютно природним явищем. Вона блискуче підтверджує те, що математичний метод LSH є надзвичайно чутливим до просторової складності вхідних візуальних даних, і при цьому він не створює масових штучних колізій для простих та однорідних фігур, зберігаючи чистоту бази.

Головним та найкритичнішим показником практичної ефективності будь-якої інтелектуальної пошукової системи є чистий машинний час, витрачений на ідентифікацію об’єкта. Оскільки сучасні системи комп’ютерного зору часто функціонують у динамічних середовищах, швидкість відгуку алгоритму безпосередньо впливає на стабільність всієї інформаційної структури. Для проведення об’єктивного порівняльного аналізу було обрано довільне тестове зображення, для якого послідовно здійснювався пошук найближчого схожого еталона за допомогою усіх п’яти розроблених методів агрегації та порівняння даних.

Для забезпечення високої достовірності результатів заміри часу виконання операцій порівняння по всій базі даних були строго зафіксовані за допомогою прецизійних апаратних таймерів процесора. Такий підхід дозволив мінімізувати вплив системних переривань та фонових процесів операційної системи на фінальні показники. Отримані об’єктивні експериментальні результати часової складності та швидкодії зведено у репрезентативну таблицю 3.4.

Детальний аналіз таблиці 3.4 наочно демонструє еволюцію швидкодії програмного коду.

Базовий бінарний алгоритм відпрацьовує завдання за 6,56 мс. Впровадження методу просторового стиснення векторів (у 16 та 32 рази) дозволило пропорційно зменшити витрати часу, довівши ці показники до 0,81 мс та 0,59 мс (що є прискоренням у 8 та 11 разів відповідно).

Проте абсолютним беззаперечним лідером усього експерименту став метод обчислення відстані хеммінга.

Таблиця 3.4 – Експериментальний час роботи методів порівняння по всій базі

Назва методу	**Час роботи, мс**	**Коефіцієнт прискорення**
Бінарний алгоритм	6,5694	1,00
Метод Мангеттена (Частотний алгоритм)	4,6624	1,40
Метод Мангеттена (Стиснутий x16)	0,8167	8,04
Метод Мангеттена (Стиснутий x32)	0,5913	11,11
Відстань Хеммінга (Бінарна)	0,1085	60,54

Використання надшвидких логічних операцій (замість арифметичних обчислень) гарантувало завершення повного циклу перевірок по всій базі за неймовірні 0,11 мс. Це означає, що перехід від арифметичної метрики мангеттена до апаратно-логічної метрики хеммінга прискорив ключовий етап пошуку у понад 60 разів. Такі екстремальні показники продуктивності широко відкривають можливості для обробки багатоканальних відеопотоків у режимі реального часу зі швидкістю тисячі кадрів за одну секунду на звичайних портативних пристроях.

Висока швидкодія методу з сигнатурами, хоч і є вражаючою, проте ніколи не повинна досягатися ціною небезпечної втрати якості розпізнавання. Для строгої перевірки точності класифікації на вхід системи було цілеспрямовано подано тестове зображення під назвою 10.jpg. Розроблений алгоритм успішно обчислив мангеттенські відстані (метрика $L1$) до всіх десяти еталонів за чотирма різними модифікаціями: базовою бінарною, деталізованою частотною, стисненою x16 та агресивно стисненою x32. Усі згенеровані консольні результати цього експерименту зібрані у порівняльну таблицю 3.4.

Згідно з законами лінійної алгебри та як показано у таблиці 3.5, найменша обчислена дистанція між векторами означає найвищу візуальну схожість. У всіх чотирьох незалежних серіях експериментів відстань до правильного цільового класу 10.jpg становила абсолютно нуль. Водночас відстань до всіх інших (хибних) об’єктів у базі надійно варіювалася у безпечних межах від 450 до 852 одиниць.

Згідно з законами лінійної алгебри та як показано у таблиці 3.5, найменша обчислена дистанція між векторами означає найвищу візуальну схожість. У всіх чотирьох незалежних серіях експериментів відстань до правильного цільового класу 10.jpg становила абсолютно нуль.

Водночас відстань до всіх інших (хибних) об’єктів у базі надійно варіювалася у безпечних межах від 450 до 852 одиниць

Це підтверджує той факт, що навіть цілеспрямоване застосування найагресивнішого алгоритму просторового стиснення у 32 рази не спричинило жодної критичної втрати інформативності згенерованих дескрипторів.

Таблиця 3.5 – Результати обчислення мангеттенської відстані для запиту 10.jpg

Назва еталона	**Бінарна $L1$**	**Частотна $L1$**	**Стиснута x16 $L1$**	**Стиснута x32 $L1$**
1.jpg	768	824	680	618
2.jpg	792	852	730	668
3.jpg	535	588	508	450
4.jpg	736	798	648	574
5.jpg	568	616	530	496
6.jpg	592	643	541	501
7.jpg	785	848	706	644
8.jpg	745	824	650	586
9.jpg	764	832	694	640
10.jpg	0	0	0	0

Це підтверджує той факт, що навіть цілеспрямоване застосування найагресивнішого алгоритму просторового стиснення у 32 рази не спричинило жодної критичної втрати інформативності згенерованих дескрипторів. Розроблена система гарантувала абсолютну 100% точність ідентифікації заданого об’єкта у всіх можливих сценаріях мангеттенського порівняння.

На робочий вхід пошукової системи було подано зовсім інший графічний запит: файл 7.jpg. Метою цього кроку було перевірити універсальність створеної моделі на об’єктах зі значно складнішою, перекритою та багатою просторовою текстурою. Усі отримані результати обчислення розбіжностей бітів зафіксовані у таблиці 3.6.

Таблиця 3.6 – Результати обчислення відстані Хеммінга для запиту 7.jpg

Назва файлу еталона	**Відстань Хеммінга**
1.jpg	913
2.jpg	921
3.jpg	684
4.jpg	915
5.jpg	729
6.jpg	755
7.jpg	0
8.jpg	900
9.jpg	913
10.jpg	785

Як свідчать статистичні дані, наведені у таблиці 3.6, розроблена система знову продемонструвала абсолютну точність, безпомилково ідентифікувавши цільовий об’єкт серед множини альтернатив

Обчислена дистанція хеммінга для файлу 7.jpg дорівнює нулю, що підтверджує повну ідентичність вилучених сигнатур еталона та вхідного запиту в умовах відсутності зовнішніх завад.

Особливу цінність становить аналіз так званого «дистанційного розриву» між істинним класом та найближчим до нього хибним об’єктом. Наступний за списком близькості об’єкт (3.jpg) продемонстрував показник розбіжності у 684 одиниці. У контексті 16-бітного LSH-хешування та використання 500 ключових точок, такий числовий розрив свідчить про надзвичайно високу сепарабельність (роздільну здатність).

Настільки значний інтервал мінімізує ймовірність виникнення помилкових спрацьовувань системи навіть за умов інтенсивного спотворення вхідних даних.

Решта нецільових еталонів чітко згрупувалися у зоні високих значень розбіжності, що підтверджує ефективність обраного методу квантування ознак.

Стабільність нульового результату для файлу 7.jpg доводить, що ускладнення просторової текстури об’єкта жодним чином не дестабілізує математичний апарат дескриптизації.

Крім того, збереження рекордної швидкодії на рівні сотих часток мілісекунди вказує на відмінну алгоритмічну оптимізацію процесу побітового порівняння. Отримані фінальні показники дають вагомі підстави стверджувати, що архітектура моделі володіє високою надійністю та необхідним запасом міцності для масштабування на великі бази графічних даних.

Варто також звернути увагу на загальний розподіл значень відстані хеммінга для решти хибних еталонів, які варіюються в широкому діапазоні від 684 до 921 одиниці. Така щільна концентрація показників у верхньому регістрі свідчить про те, що алгоритм LSH генерує максимально відмінні хеш-коди для візуально непов’язаних зображень. Середнє значення розбіжності для нецільових об’єктів становить близько 835 одиниць, що є беззаперечним індикатором ортогональності сформованих бінарних векторів у просторі ознак.

Аналізуючи результати для файлів 1.jpg, 2.jpg, 4.jpg, 8.jpg та 9.jpg, можна помітити, що їхні показники перетинають критичну позначку у 900 одиниць. Це вказує на їхню невідповідність вхідному запиту 7.jpg та демонструє здатність дескриптора ігнорувати навіть дрібні збіги патернів, якщо загальна композиція об’єкта кардинально відрізняється. Таким чином, експериментальні дані з таблиці 3.6 повністю верифікують теоретичну модель розробленого алгоритму зіставлення.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено модель методу розпізнавання зображень на основі незалежного хешування для описів еталонів.

Проаналізовано апарат вилучення нелінійних бінарних дескрипторів (зокрема M-LDB алгоритму AKAZE) та обґрунтовано їхні структурні та обчислювальні переваги над традиційними лінійними аналогами на базі дійсних чисел, що дозволило:

− відмовитися від ресурсомістких метрик на базі дійсних чисел;

− ефективно зберігати висококонтрастні межі об’єктів зі складною геометрією завдяки застосуванню нелінійних рівнянь дифузії;

− формувати компактні 488-бітні вектори ознак, що мають високу інформаційну ємність.

Спроєктовано та розроблено алгоритмічний конвеєр перетворення довгих 488-бітних векторів ознак у компактний 16-бітний адресний простір (65536 кошиків) за допомогою генерації стабільної глобальної матриці випадкових гаусівських проєкцій (метод LSH), що дозволило:

− трансформувати розрізнені масиви локальних ознак у цілісні статичні частотні сигнатури об’єкта;

− забезпечити швидке просторове групування геометрично близьких точок без використання обчислювально важких графічних графів.

Програмно реалізовано у єдиному середовищі кілька конкурентних методів класифікації на основі традиційного лінійного пошуку (Brute-Force) з перехресною перевіркою, на основі обчислення мангеттенської відстані для глобальних частотних сигнатур (включно з методами просторової децимізації x16 та x32), а також на основі швидкого обчислення побітової відстані Хеммінга, що дозволило:

− створити незалежну та портативну архітектуру мовою Python без обов'язкової наявності зовнішніх графічних прискорювачів;

− прискорити етап класифікації у понад 60 разів;

− довести, що методи структурної децимізації x16 та x32 здатні зберігати інформативність.

Змодельовано вплив адитивного білого гаусівського шуму на стабільність формування дескрипторів та розроблено програмний захисний механізм компенсації втрачених голосів для нормалізації розподілу ймовірностей, що дозволило:

− об’єктивно зафіксувати вразливість мангеттенської метрики до шумів;

− підтвердити високу алгоритмічну міцність методу LSH, який стабільно утримує точність 91% при середньому рівні оптичного зашумлення кадру;

− ефективно стабілізувати роботу машинного класифікатора за умов низької видимості.

Проведено експериментальне тестування розробленої системи на спеціально підготовленій базі еталонних зображень БПЛА, побудовано перехресні матриці активації (розмірністю 11x11), та на основі отриманих логів об’єктивно оцінено обчислювальну складність і загальну завадостійкість запропонованих алгоритмів класифікації, що дозволило:

− засвідчити 100% точність ідентифікації цілей в ідеальних умовах без втрати семантичної якості;

− довести здатність алгоритму LSH ідентифікувати та відхиляти сторонні (невідомі) об’єкти завдяки жорсткій системі порогів активації.

Моделювання методу розпізнавання зображень на основі незалежного хешування є ефективним рішенням для сучасних систем комп’ютерного зору.

Результати роботи апробовано у вигляді тез доповіді під час XXX Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Yuan, W., Prasad Poosa, S. R., & Dirks, R. F. (2024). Comparative Analysis of Color Space and Channel, Detector, and Descriptor for Feature–Based Image Registration. J. Imaging, 10(5), 105.

2. Budzan, S., Wyżgolik, R., & Lysko, M. (2025). Performance Analysis of Keypoints Detection and Description Algorithms for Stereo Vision Based Odometry. Sensors, 25(19), 6129.

3. Alshahrani, A. A., & Jaha, E. S. (2023). Locality–Sensitive Hashing of Soft Biometrics for Efficient Face Image Database Search and Retrieval. Electronics, 12(6), 1360.

4. Pandey, S., Tiwari, R. K., Thakur, O. N., & Pathak, M. (2025). Validation of Various Distance based Feature Vectors for SVM based CBIR System. International Journal of Engineering Research and Applications (IJERA), 15(8), 106–113.

5. 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.

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

7. Sadiq, I. M., & Basheera, M. H. (2021). Content–based image retrieval: A review of recent trends. Cogent Engineering, 8(1)

8. Tiwari, S., Sharma, A. K., Aziz, I. A., Gupta, D., Jain, A., Mahdin, H., Athithan, S., & Hidayat, R. (2023). Investigations on segmentation–based fractal texture for texture classification in the presence of Gaussian noise. PLOS One.

9. Wang, Z., Xiao, H., Duan, Y., Zhou, J., & Lu, J. (2023). Learning Deep Binary Descriptors via Bitwise Interaction Mining. IEEE Transactions on Pattern Analysis and Machine Intelligence.

10. Li, M., et al. (2024). Loop Closure Detection with CNN in RGB–D SLAM for Intelligent Agricultural Equipment. Agriculture, 14(6), 949.

11. Буряк, Р. В. (2024). *Розробка методу детектування об'єктів при багатокамерному трекінгу на основі комп'ютерного зору*. Державний університет інформаційно–комунікаційних технологій, Київ.

12. Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.

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

14. Bansal, M., Kumar, M., & Kumar, M. (2021). 2D object recognition: A comparative analysis of SIFT, SURF, ORB, FAST, and AKAZE. *Multidimensional Systems and Signal Processing*, 32, 1009–1044.

15. OpenCV. (2024). *Accelerated–KAZE (AKAZE) Features*. OpenCV 4.x Documentation.

16. Tareen, S. A. K., & Saleem, Z. (2021). Performance analysis of binary descriptors for object recognition. *Multimedia Tools and Applications*.

17. Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.

18. 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.

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

20. Wang, Y., Zhang, J., & Li, X. (2021). A Comprehensive Survey on Locality Sensitive Hashing. *IEEE Access*, 9, 78120–78135

21. Ahle, T. D., Kapralov, M., Pagh, R., Silvestri, F., & Zandieh, A. (2022). Oblivious Sketching of High–Degree Polynomial Kernels. *Proceedings of the 2022 ACM–SIAM Symposium on Discrete Algorithms (SODA)*, 141–160

22. Pedregosa, F., et al. (2023). *Scikit–learn: Machine Learning in Python*

23. OpenCV. (2024). *Feature Matching with FLANN and BRUTE–FORCE*. OpenCV 4.x Documentation.

24. 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.

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

26. Virbora Ny. (2024) Computer Vision with OpenCV Python and YOLOv8: A Recompiled Version

27. Yashankit S., Vibhav P., Singh R. S., (2021) Comparative Analysis of Distance Metrics for Designing an Effective Content–based Image Retrieval System Using Colour and Texture Features

28. Alzu'bi, A., Amira, A., & Ramzan, N. (2021). Content–based image retrieval with compact deep convolutional features. *Neurocomputing*, 440, 269–281. Romanov, A., & White, J. (2026). Deep decimation of visual features for real–time signature matching. *UAV System Engineering, 9*, 45–59.

29. Stream Mining & Processing (DSMP) (2018), (pp. 157–160). IEEE.

30. Bradski, G., & Kaehler, A. (2023). *Learning OpenCV 4 Computer Vision with Python 3*. O'Reilly Media.

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

32. Gorokhovatsky, V. A. (2016). Efficient Estimation of Visual Object Relevance during Recognition through their Vector Descriptions. *Telecommunications and Radio Engineering*, Vol. 75, No 14, P. 1271–1283.

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

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

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

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

37. 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.

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

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

40. Сазонов, М. С. (2026). Розпізнавання зображень на основі незалежного хешування структурних описів. *Матеріали XXX Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті». ХНУРЕ*, 7, 147–149.