Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСОБІВ БІНАРНОГО ОБРОБЛЕННЯ ДАНИХ У СТРУКТУРНИХ МЕТОДАХ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-18-2

Солодченко К.Г.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

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

Допускається до захисту

Зав. кафедри _____ Путятін Є.П.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Солодченко Кирилу Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження ефективності засобів бінарного оброблення даних у структурних методах класифікації об'єктів .

затверджена наказом по університету від "_21__" Жовтня 2019 року № 1506 ст.

2. Термін подання студентом роботи до екзаменаційної комісії _23_ __Квітня_____2019 р.

3. Вихідні дані до роботи *Методи машинного навчання, детектування особливих точок на зображенні, методи класифікації об'єктів за допомогою детекторів особливих точок, класифікація даних, апарат бінарного оброблення векторних даних, бази зображень, центральний дескриптор, вагові коефіцієнти.*

4. Перелік питань, що потрібно опрацювати в роботі
1. Постановка задачі класифікації
2. Апарат бінарної обробки дескрипторів ключових точок
3. Моделі оброблення особливих точок на зображенні
4. Методи класифікації зображень на основі бінарних дескрипторів
5. Програмна реалізація, дослідження результативності методів та аналіз результатів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Дескриптор BRISK - схема вибірки BRISK, Дескриптор BRISK – коротка пара BRISK, Діаграма послідовності, Результати першого досліду, Результати другого досліду, Результати третього досліду, Зведена таблиця результатів дослідження.*

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	21.10.2019	
2	Аналіз завдання, підбір літератури	05.11.19-10.11.19	
3	Аналіз літератури з досліджуваної проблеми	11.11.19-15.11.19	
4	Аналіз технічних засобів	16.11.19-18.11.19	
5	Розробка методу	19.11.19-22.11.19	
6	Програмна реалізація	22.11.19-24.11.19	
7	Оформлення пояснювальної записки	25.11.19-01.12.19	
8	Перевірка на плагіат	02.12.19	
9	Рецензування		
10	Підготовка презентації та доповіді		
11	Занесення роботи в електронний архів		
12	Попередній захист атестаційної роботи	23.04.19	

Дата видачі завдання _21_ ___жовтня___2019 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка атестаційної роботи: 76 с., 48 рис., 4 табл., 1 додаток, 30 джерел.

КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, КЛЮЧОВІ ТОЧКИ, ЦЕНТРАЛЬНИЙ ДЕСКРИПТОР, ЕТАЛОН, ДЕСКРИПТОР BRISK, ВАГОВІ КОЕФІЦІЄНТИ, РЕЛЕВАНТНІСТЬ ОПИСІВ, БАЗА ЕТАЛОНІВ

Метою атестаційної магістерської роботи є дослідження та модернізація методу класифікації зображень на підставі множини ключових точок. За описом зображення у вигляді множини дескрипторів BRISK як бінарних векторів обчислюється вектор центрального дескриптора, аналізуються варіанти його побудови. Проведено аналіз працездатності запропонованих алгоритмів, дослідження на варіативних вибірках та різних первинних обробках вхідних та еталонних зображень.

Об’єктом дослідження є методи класифікації значення центрального дескриптора із врахуванням вагових коефіцієнтів та проведення логічного аналізу важливості окремих бітів.

Розроблено програмне забезпечення для класифікації зображень варіантами алгоритмів, підтверджено їх працездатність та результативність.

IMAGE CLASSIFICATION, KEY POINTS, CENTRAL DESCRIPTOR, STANDARD, BRISK DESCRIPTOR, WEIGHTS, RELEVANCE OF DESCRIPTIONS, BASE

The purpose of the master's thesis is to research and modernize the method of image classification based on a set of key points. By describing the image in the form of a set of BRISK descriptors as binary vectors, the vector of the central descriptor is calculated and variants of its construction are analyzed. The analysis of the efficiency of the proposed algorithms, the study of variant samples and various initial processing of input and reference images.

The object of the study is methods of classifying the value of the central descriptor, taking into account the weights and conducting a logical analysis of the importance of the individual bits.

Software for classification of images by variants of algorithms is developed, their efficiency and effectiveness are confirmed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AVG – Середнє значення

BRIEF – Binary Robust Independent Elementary Features

BRISK – Binary Robust Invariant Scalable Keypoints

EBNF – Extended Backus–Naur form

FAST – Features from Accelerated Segment Test

ID3 – Identify an MP3

OpenCV – Open Source Computer Vision Library

SIMD – Single Instruction, Multiple Data

SSE – Streaming SIMD Extensions

STL – Standard Template Library

SURF – Speeded up Robust Features

SUSAN – Smallest Univalue Segment Assimilation Nucleus

SWIFT – Society for Worldwide Interbank Financial Telecommunications

XOR – eXclusive OR, додавання за модулем два

ВК – метод Вагових коефіцієнтів.

ВВС – Відмова від слабкого

МПЦД – Метод побудови центрального дескриптора

ООП – Об'є́ктно-орієнто́ване програмува́ння

ОТ – особливі точки

ВСТУП

У системах комп’ютерного зору при розпізнаванні візуальних об’єктів за їх структурним описом у вигляді множини дескрипторів особливих точок зображення постає задача побудови функції (правила, процедури) класифікації, що задає відношення еквівалентності на множині дескрипторів бази зображень, де класи представлені окремими описами еталонів. Зважаючи на подібність значень дескрипторів для різноманіття візуальних об’єктів, така еквівалентність на практиці досягається лише наближено [1, 14-17].

У задачі розпізнавання об’ємних баз візуальних об'єктів у системах комп'ютерного зору важливими показниками є результативність і час оброблення. Тому останнього часу набули прикладного застосування бінарні детектори особливих точок, як BRISK, ORB [7,2], що отримують дескриптор ключових точок у вигляді бінарного вектора з розміром, кратним ступені двійки. Бінарне подання даних значно прискорює процес порівняння дескрипторів за рахунок можливості застосування двійкових операцій та відповідно спрощує апаратну реалізацію системи розпізнавання. Крім того, бінарна арифметика дає потенцію застосувати ефективний апарат оброблення бінарних даних та синтезувати нові підходи для визначення подібності дескрипторів ОТ при побудові правил класифікації.

Метою атестаційної магістерської роботи є дослідити та реалізувати модернізації методу класифікації зображень за допомогою бінарного центрального дескриптора. Провести аналіз працездатності алгоритмів, провести досліди на варіативних вибірках та при різних первинних обробках вхідних чи еталонних зображень. Та на основі отриманих даних дати оцінку як алгоритмам, так й проведеним дослідам.

Важлива перевага бінарного дескриптора полягає у швидкості та простоті оброблення. Об’єкти у реальному світі характеризується великою кількістю станів кожного параметра системи. В таких системах

неможливо визначити точні значення параметрів функціонування. Тоді використовують, наприклад, інтервальні оцінки або нечіткі числа. В нашому випадку ми ціле зображення приймаємо як послідовність чітких значень, як дескриптор, що має довжину у 512 (біт) значень, кожне значення може бути 1 або 0. Можливість побудови центру один раз без подальшої його зміни та можливість пришвидшити процес класифікації у стільки разів, скільки ОТ детектується для конкретного зображення, робить даний підхід досить актуальним під час роботи з великим об’ємом даних. Вже не говорячи про можливість порівняння центра з центром, що дозволяє ще додатково пришвидшити процес класифікації в стільки разів, скільки ОТ містить зображення, що порівнюється з множиною еталонних описів.

1 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ НА ПІДСТАВІ МНОЖИНИ КЛЮЧОВИХ ТОЧОК

1.1 Основи задачі класифікації

Класифікація - один з розділів машинного навчання, присвячений вирішенню проблеми відношення об’єкту до групи конкретних об’єктів ,що мають спільні відзнаки від групи інших об’єктів. Є множиною об'єктів (ситуацій), розділених деяким чином на класи. Визначено кінцеву множину об'єктів, для яких відомо, до яких класів вони належать. Ця множина називається навчальною вибіркою. Класова приналежність інших об'єктів не відома. Потрібно побудувати алгоритм, здатний класифікувати довільний об'єкт з початкової множини [26].

Класифікувати об'єкт - значить, вказати номер (або найменування класу), до якого належить даний об'єкт.

Класифікація об'єкта - номер або найменування класу, що видається алгоритмом класифікації в результаті його застосування до даного конкретного об'єкту.

Математичне формулювання завдання наступне: Нехай X — множина описів об'єктів, Y —множина номерів (чи назв) класів. Існує невідома цільова залежність - відображення $y^*: X \to Y$, значення якої відомі лише на елементах скінченної навчальної вибірки $X^m = \{(x_1, y_1), \cdots, (x_m, y_m)$. Потрібно побудувати алгоритм $a: X \to Y$, здатний класифікувати довільний об'єкт $x \in X$ [9,10].

Загальнішим є імовірнісне формулювання завдання. Припускається, що множина пар «об'єкт, клас» $X \times Y$ є ймовірнісним простором з невідомою ймовірнісною мірою P. Є скінченна навчальна вибірка спостережень $X^m = \{(x_1, y_1), \cdots, (x_m, y_m)$, з генерована згідно з ймовірнісною мірою P.

Необхідно побудувати алгоритм $a: X \to Y$, здатний класифікувати довільний об'єкт $x \in X$ [9,10].

У математичній статистиці завдання класифікації називаються також завданнями дискримінантного аналізу.

У машинному навчанні завдання класифікації відноситься до розділу навчання з учителем. Існує також навчання без вчителя, коли поділ об'єктів навчальної вибірки на класи не задається, і потрібно класифікувати об'єкти тільки на основі їх подібності між собою. У цьому випадку прийнято говорити про завдання кластеризації або таксономії, і класи називати, відповідно, кластерами або таксонами [26].

Різновидності вхідних даних у задачі класифікації [26]:

- ознаковий опис - найбільш поширений випадок. Кожен об'єкт описується набором своїх характеристик, які називаються ознаками. Ознаки можуть бути числовими або нечисловими;
- матриця відстаней між об'єктами. Кожен об'єкт описується відстанями до всіх інших об'єктів навчальної вибірки. З цим типом вхідних даних працюють деякі методи, зокрема, метод найближчих сусідів, метод парзеновского вікна, метод потенційних функцій;
- часовий ряд або сигнал являє собою послідовність вимірювань в часі. Кожен вимір може представлятися числом, вектором, а в загальному випадку - признаковим описом досліджуваного об'єкта в даний момент часу;
- зображення або відеоряд;
- зустрічаються і більш складні випадки, коли вхідні дані подаються у вигляді графів, текстів, результатів запитів до бази даних, й т.д. Як правило, вони наводяться до першого або другого випадку шляхом попередньої обробки даних і вилучення ознак.

Типи класів [26]:

- двокласова класифікація. Найбільш простий в технічному відношенні випадок, який служить основою для вирішення більш складних завдань;
- багатокласова класифікація. Коли число класів досягає багатьох тисяч (наприклад, при розпізнаванні ієрогліфів або злитого мовлення), завдання класифікації стає істотно більш важким;
- класи що не перетинаються;
- класи що перетинаються. Об'єкт може належати одночасно до кількох класів;
- нечіткі класи. Потрібно визначати ступінь належності об'єкта кожному з класів, зазвичай це дійсне число від 0 до 1.

Правила класифікації [26]:

- ділити потрібно по одному конкретному основи. Якщо порушити це правило, відбудеться перетин понять і моторошна плутанина. Автомобілі бувають декількох видів: хетчбеки, седани, червоні, чорні, білі, що працюють на бензині і газі. Тут ми змішали аж три ознаки: тип кузова, колір, вид споживаного палива;
- класифікація повинна бути повною. Якщо ми складемо всі підвиди, то повинен вийти вид. Це означає, що при розбитті важливо не забути вказати всі частини. Лікарі діляться на стоматологів, терапевтів, педіатрів та кардіологів. Так? Не так. Ми забули хірургів, косметологів і ще багато кого;
- класифікація не повинна містити зайвих елементів. Типова помилка, коли при поділі одного поняття ми перестаралися і додали туди предмет, який не має відношення до цілого. Наприклад, цитрусові бувають такі: апельсини, лимони, банани. Цілком очевидно, що банан - НЕ цитрус. Цей елемент тут зайвий;
- елементи повинні взаємно виключати один одного. Ось ви створили в пам'яті комп'ютера дві папки - «Пісні» і «Музика». Тепер сидите і роздумуєте, куди зберігати альбом улюбленої музичної групи;

- ділячи поняття, слід діяти по порядку, безперервно переходити від одного підкласу до іншого найближчого. Перестрибувати не можна.

1.2 Методи формування ключових точок зображення

Особливі точки – це те ж саме, що і точки інтересу на зображенні [5]. Це просторові розташування або точки в зображенні, які визначають, що цікаво чи що виділяється на зображенні. Причина, по якій ці точки є особливими, полягає в тому, що незалежно від того, як змінюється зображення, стискається або розширюється, (все це будуть афінні трансформації) або схильна до спотворення (тобто проективне перетворення або homography), ви все одно зможете знайти ті ж ключові точки в цьому зміненому зображенні в порівнянні з вхідним зображенням.

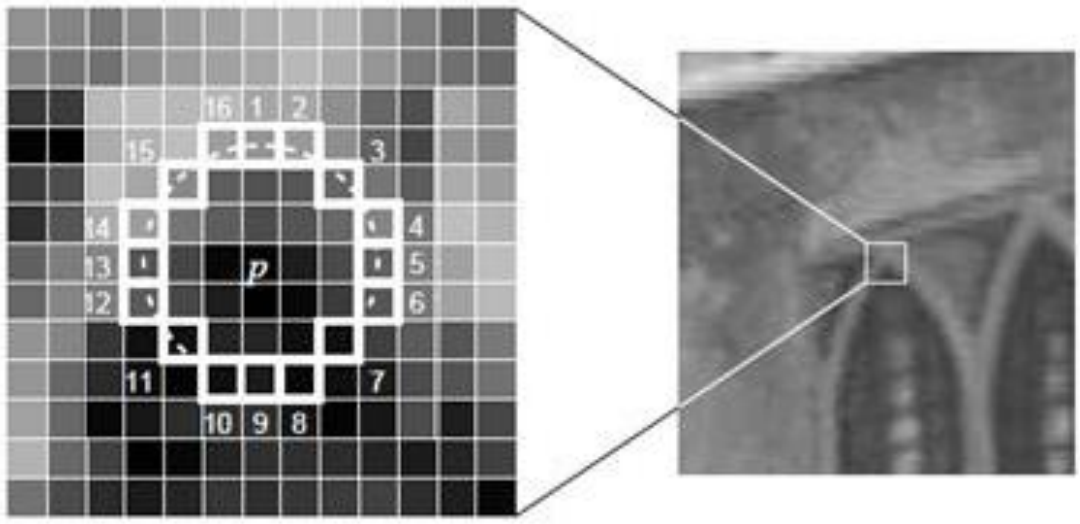


Рисунок 1.1 – Точка-кандидат й окружність з 16 пікселів навколо неї

FAST - це алгоритм, запропонований Ростеном та Драммодом [18] для виявлення ОТ у зображенні. ОТ зображення – це піксель, який має чітко визначену позицію і може бути надійно виявлений. ОТ мають високий інформаційний вміст у своєму форматі та ідеально повторюються між

різними зображеннями [15]. Визначення ОТ має програмну реалізацію для знаходження збігу зображень, розпізнавання об'єктів, відстеження, тощо.

Ідея виявлення ОТ або виявлення кутів (обидва використовуються в літературі), не нова. Існують і інші алгоритми, такі як: Moravec алгоритм виявлення кутів, алгоритм виявлення кутів Harris & Stephens, детектор куту SUSAN. Причиною створення алгоритму FAST було створення детектора ОТ для використання в програмах реального часу, таких як SLAM на мобільному роботі, що мають обмежені обчислювальні ресурси [19].

Алгоритм FAST полягає в наступних кроках.

1. Виберіть піксель p у зображенні. Припустимо, що інтенсивність цього пікселя має бути I_p. Це є піксель, який слід визначити як ОТ, або визначити протилижне. (Рисунок 1.3)

2. Встановіть величину інтенсивності порогу T (скажімо, 20% випробуваного пікселя).

3. Розглянемо кола з 16 пікселів навколо пікселя p. (Це кола Брес-Енахм [4] з радіусом 3).

4. Помітні пікселі N з 16, повинні бути вище або нижче I_p за значенням T, якщо піксель потрібно визначити як ОТ. (Автори використовували $N = 12$ у першій версії алгоритму)

5. Щоб зробити алгоритм швидким, спочатку порівнюйте інтенсивність пікселів 1, 5, 9 і 13 кола з I_p. Як можна побачити з наведених вище цифр, принаймні три з цих чотирьох пікселів повинні задовольняти пороговому критерію, щоб існувала ОТ.

6. Якщо принаймні три з чотирьох значень пікселів - $I1$, $I5$, $I9$ чи $I13$ не вище або нижче $I_p + T$, то p не є ОТ (кутом). У цьому випадку відхилюємо піксель p як можливу ОТ. Іншими словами, якщо принаймні три пікселі перебувають вище або нижче $I_p + T$, то перевірте або всі 16 пікселів й перевірте, чи 12 сусідніх пікселів потрапляють у критерій.

7. Повторення процедури для всіх пікселів у зображенні.

Існує декілька обмежень алгоритму. По-перше, для $N<12$ алгоритм працює не дуже добре, оскільки при $N<12$ кількість виявлених ОТ дуже висока. По-друге, порядок, в якому запитується 16 пікселів, визначає швидкість алгоритму.

Підхід до машинного навчання був доданий до алгоритму для вирішення цих проблем [3,5].

1. Виберіть набір зображень для тренувань.

2. У кожному зображенні запускається алгоритм FAST для виявлення ОТ, приймаючих один піксель одночасно та оцінюючи всі 16 пікселів у колі.

3. Для кожного пікселя p зберігайте 16 пікселів навколо нього, як вектор. (Рис. 1.4)

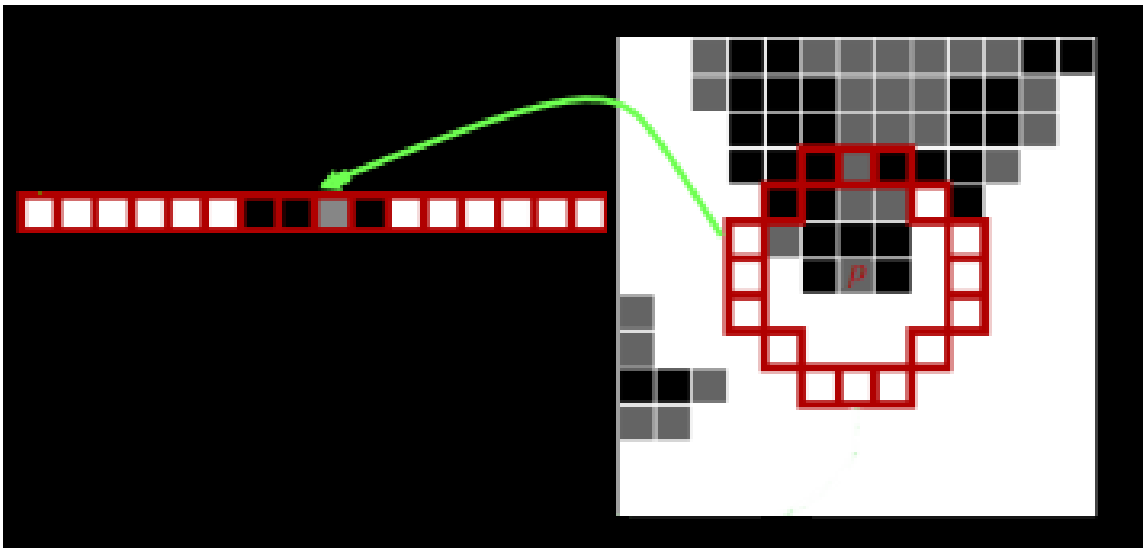


Рисунок 1.2 – 16 значень, що оточують піксель p, зберігаються у векторній формі (зображення скопійовано з [5])

4. Повторіть це для всіх пікселів у всіх зображеннях. Це вектор P, який містить всі дані для розпізнавання. (Зверніть увагу на різницю між p - пікселем та P - вектор)

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

Математично,

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

$S_{p->x}$ – це стан, $I_{p->x}$ – інтенсивність пікселя x, t - поріг.

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

7. Визначте змінну K_p, яка є істиною, якщо p є ОТ або помилковою, якщо p не є ОТ.

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

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

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

$$\textit{where} \quad c = \left|\left\{p \middle| K_p \text{ is true}\right\}\right| \quad (number\ of\ corners)$$

$$\textit{and} \quad \bar{c} = \left|\left\{p \middle| K_p \text{ is false}\right\}\right| \quad (number\ of\ non\,corners)$$

10. Рекурсивно застосовуйте цю мінімізацію ентропії до всіх трьох підмножин.

11. Припиніть процес, коли ентропія підмножини дорівнює нулю.

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

Мета створення BRIEF-дескриптора (Binary Robust Independent Elementary Features) [17] полягала в тому, щоб забезпечити розпізнавання однакових ділянок зображення, які були зняті з різних точок обзору. При цьому ставилося завдання максимально зменшити кількість виконуваних обчислень. Алгоритм розпізнавання зводиться до побудови випадкового лісу (randomize classification trees) або наївного байєсівського класифікатора на деякій тренувальній множині зображень з подальшою класифікацією ділянок тестових зображень. У спрощеному варіанті може використовуватися метод найближчого сусіда для пошуку найбільш схожого патча в тренувальній вибірці. Невелика кількість операцій забезпечується за рахунок подання вектора ознак у вигляді бінарного рядка, а як наслідок, використання в якості міри схожості відстані Хеммінга.

Схема побудови векторів ознак досить проста. Зображення розбивається на патчі (окремі перекриті ділянки). Припустимо патч P має розміри $S*S$ пікселів. Із патча деяким чином вибирається множина пар пікселів $\{(X,Y), \forall X, Y \, в \, околиці\}$ для яких будується набір бінарних тестів:

$$\tau(P,X,Y) = \begin{cases} 1, & I(X < I(Y)) \\ 0, & інакше \end{cases},$$

де $I(X)$ - інтенсивність пікселя X. Для кожного патча вибирається множина, що містить n_d пар точок, які однозначно визначають набір бінарних тестів. Далі на підставі цих тестів будується бінарна рядок:

$$f_{nd}(P) = \sum_{1 \le i \le n_d} 2^{i-1} \tau(P, X_i, Y_i) .$$

Автори [17] наводять результати експериментів (якості розпізнавання) при виборі пар точок відповідно до закону рівномірного розподілу в патчі, а також нормального розподілу з різними значеннями математичного

очікування і середньоквадратичного відхилення. Відзначимо, що при однакових умовах проведення експериментів на деяких тестових зображеннях точність детектування за допомогою BRIEF майже в 1.5 рази вище, ніж з використанням SURF-дескрипторів.

Почнемо з наступного зображення, що показує приклад використання BRISK для порівняння зображення без змін й над яким провели деякі перетворення. Зелені лінії - дійсні співвідношення між точками, червоні кола виявляються ключовими точками.

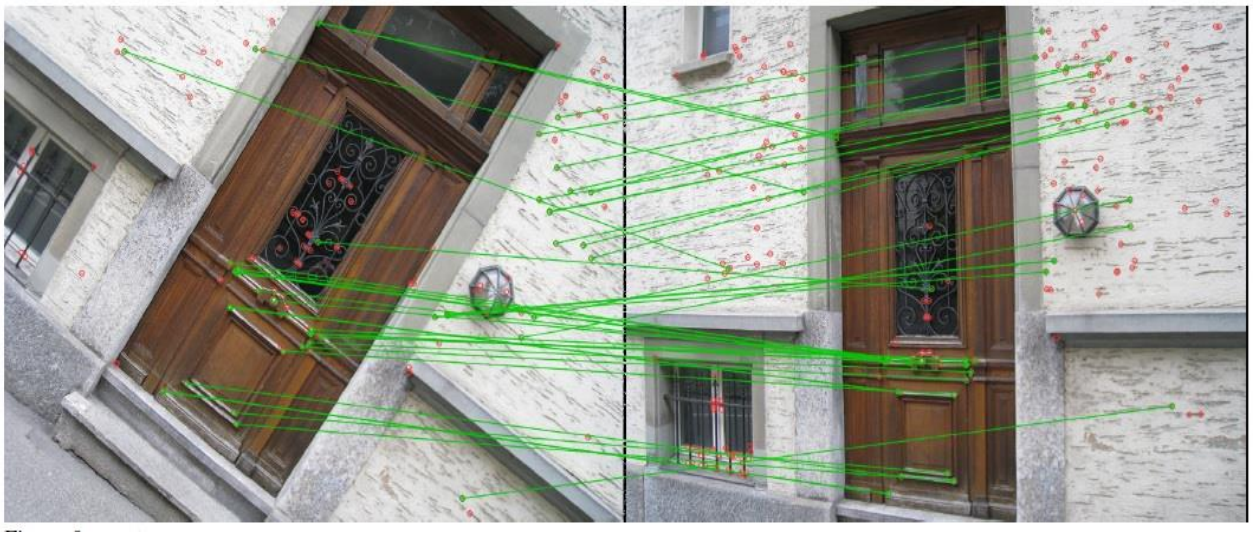


Рисунок 1.3 – Дескриптор BRISK - приклад точок збігу за допомогою BRISK

Двійковий дескриптор BRISK складається з трьох частин.

1. Зразок вибірки: де взяти зразок точок у регіоні навколо дескриптора.

2. Компенсація орієнтації: якийсь механізм для вимірювання орієнтації ОТ та його обертання, щоб компенсувати зміни обертання.

3. Пари вибірки: які пари порівнювати при побудові остаточного дескриптора.

Щоб побудувати двійковий рядок, що представляє регіон навколо ключової точки, нам потрібно перейти за всіма парами і для кожної пари ($p1$,

$p2$) - якщо інтенсивність у точці $p1$ перевищує інтенсивність у точці $p2$, ми пишемо 1 в двійковий рядок і 0 в іншому випадку.

Дескриптор BRISK э модифікованим алгоритмом FAST (розділ 1.2), та має ручний шаблон вибірки. Зразки вибірки BRISK складаються з концентричних кілець:

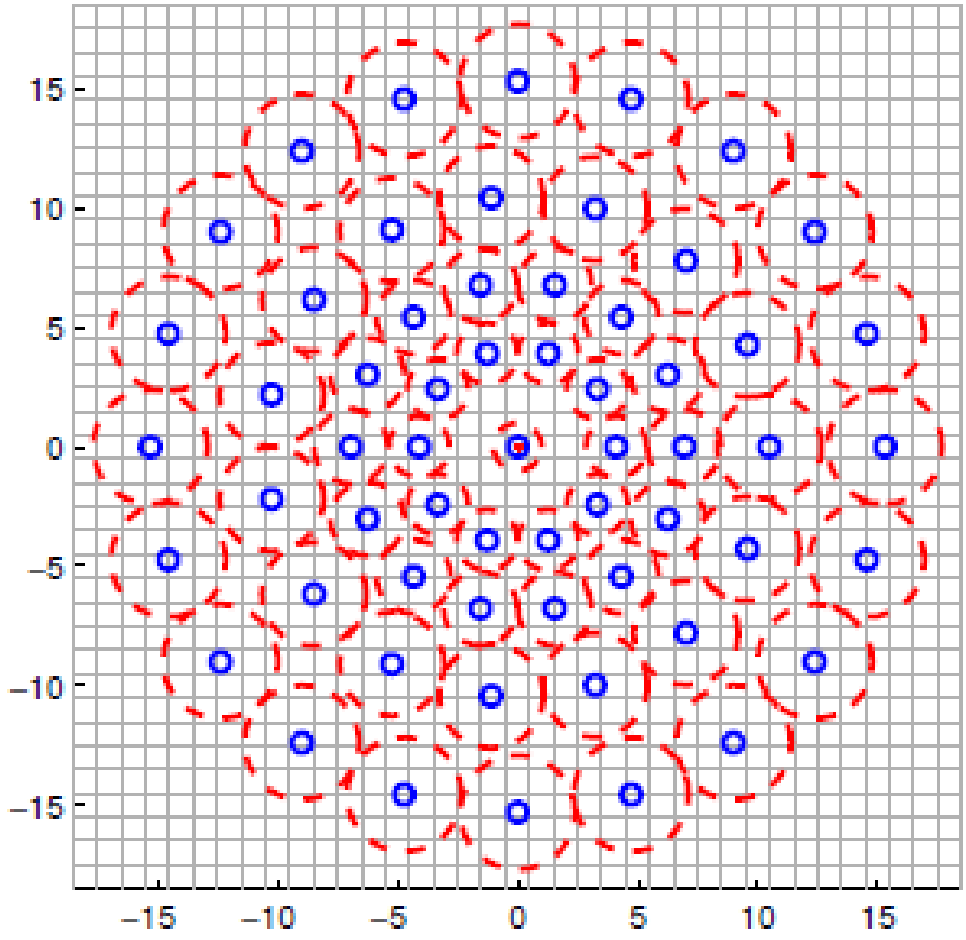


Рисунок 1.4 – Дескриптор BRISK - схема вибірки BRISK

Розглядаючи кожну точку відбору, ми беремо навколо неї невеликий патч і застосовуємо гаусове згладжування. Червоне коло на рисунку 1.6 - розмір стандартного відхилення гаусового фільтра, застосованого до кожної точки. Використовуючи цей шаблон вибірки, ми розрізняємо короткі пари та довгі пари. Коротка пара - це пара точок відбору проб, що їхня відстань нижча за певну порогову величину $d\_\max$, а довгими парами є пари точок

відбору проб, що їх відстань перевищує певний інший поріг $d\_\min$, де $d\_\min > d\_\max$, так що немає коротких пар, які також є довгими пари

Довгі пари використовуються в BRISK для визначення орієнтації, а короткі пари використовуються для порівняння інтенсивності, що складають дескриптор, як у BRIEF (пункт 1.3) на основі якого й було збудовано алгоритм порівняння ОТ у BRISK.

Щоб проілюструвати це і зрозуміти, рисунок 1.5 – фігура коротких пар BRISK - кожна червона лінія представляє одну пару. Кожна фігура показує 100 пар.

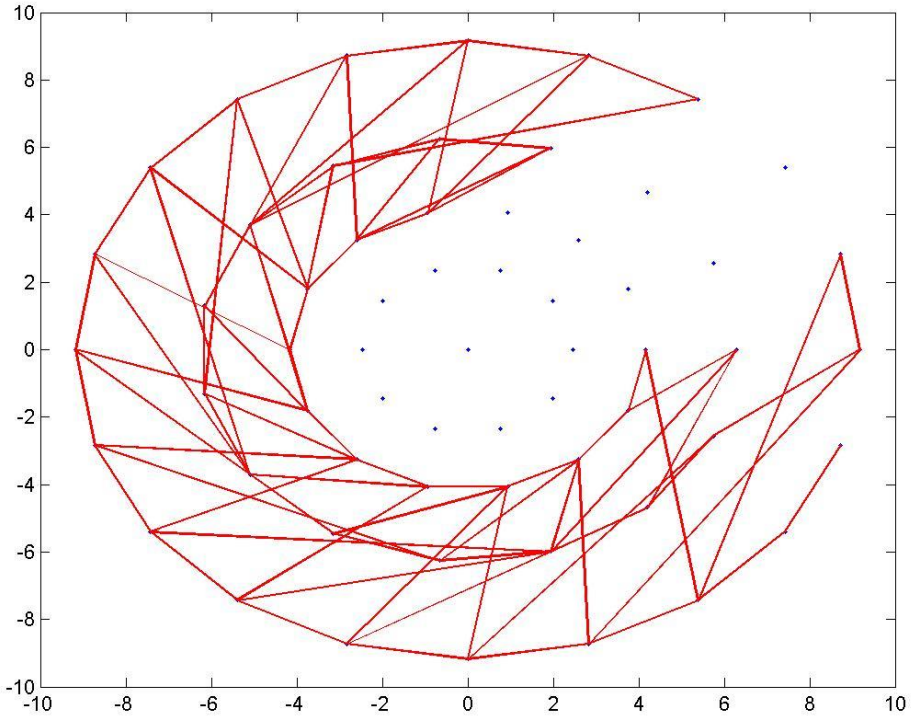


Рисунок 1.5 – Дескриптор BRISK, коротка пара BRISK

BRISK оснащений механізмом компенсації орієнтації; намагаючись оцінити орієнтацію ключової точки та обертання шаблону вибірки за цією орієнтацією, BRISK стає дещо інваріантним до обертання.

Для обчислення орієнтації ключової точки BRISK використовує локальні градієнти між парами вибірки, які визначаються за допомогою

$$g(p_i, p_j) = \left(p_j - p_i\right) \times \frac{I\left(p_j, \sigma_j\right) - I\left(p_i, \sigma_i\right)}{\left\| p_j - p_i \right\|^2},$$

де $g(p_i, p_j)$ – локальний градієнт між парою вибірки (p_i, p_j), I - згладжена інтенсивність у відповідній точці відбору з урахуванням відповідного стандартного відхилення.

Щоб обчислити орієнтацію, підсумовуємо локальні градієнти між усіма довгими парами і вирахуємо $\arctan(gy / gx)$ - де арктангенс y - компоненти градієнта, поділений на x компоненту градієнта. За допомогою арктангенсу вираховуємо кут даного арктангенсу. Далі потрібно повернути короткі пари за цим кутом, щоб дескриптор став інваріантним до обертання. BRISK використовує лише довгі пари для обчислення орієнтації, виходячи з припущення, що локальні градієнти скасовують один одного, таким чином, вони не є необхідними для визначення глобального градієнта.

Як і у випадку з усіма двійковими дескрипторами, побудова дескриптора здійснюється шляхом порівняння інтенсивності. BRISK бере набір коротких пар, обертає пари орієнтацією, обчисленою раніше, і робить порівняння форми:

$$b = \begin{cases} 1, & I(p_j^{\alpha}, \sigma_j) > I(p_i^{\alpha}, \sigma_i) \\ 0, & otherwise \end{cases}.$$

Це означає, що для кожної короткої пари він згладжує інтенсивність точок відбору проб і перевіряє, чи згладжена інтенсивність першої точки в парі більша, ніж у другої точки. Якщо це так, то він записує 1 у відповідний біт дескриптора та іншим чином 0. Пам'ятайте, що BRISK використовує лише короткі пари для побудови дескриптора.

Як завжди, відстань між двома дескрипторами визначається як кількість різних бітів двох дескрипторів, і їх можна легко обчислити як суму оператора XOR між ними.

1.3 Постановка задачі дослідження

Класифікація зображень за допомогою бінарних центрів є актуальною задачею завдяки своїй простоті та забезпеченню високої швидкодії. Ставиться завдання розробки та дослідження алгоритмів побудови бінарного центрального дескриптора з його подальшою обробкою для отримання оцінки швидкодії та якості класифікації зображень. Для отримання ОТ з бінарними дескрипторами будемо використовувати алгоритм BRISK. Він має довжину дескриптора у 512 біт. Планується модернізувати даний алгоритм ваговими коефіцієнтами. Далі планується порівняти нові алгоритми з методом побудови центрального дескриптора та отримати порівнювальні оцінки працездатності та конкурентоздатності алгоритмів.

Потрібно програмно реалізувати алгоритмічну частину та розробити програму, що буде використовувати алгоритми для розпізнавання зображень у базі еталонів, провести досліди, на основі яких можна буде провести оцінювання результативності.

Для цього необхідно вирішити такі завдання:

- провести аналіз недоліків та переваг алгоритму побудови центрального дескриптора;
- розробити нові алгоритми чи модернізацію побудованого алгоритму класифікації з бінарними даними;
- реалізувати комп'ютерну модель детектування ОТ та обчислення множини дескрипторів;.
- модернізувати систему тестування та аналізу даних для оцінювання роботи алгоритму;

- провести комп’ютерне тестування алгоритму на підготовленій реальній базі зображень;
- після побудови графіків встановити оцінки працездатності та конкурентоздатності для алгоритмів з розглядом їх подальшої модернізації;
- провести аналіз результатів дослідження.

2 МОДЕЛІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ НАВЧАННЯ

2.1 Задача класифікації для множини бітових рядків

Нехай $Z=\{Z^j\}_{j=1}^{J}$ – це множина дескрипторів бази зображень, яку складають описи еталонів, J – число класів, Z^j – еталон, $Z \subseteq B^n$, B^n – множина бінарних векторів (ланцюжків, рядків) розмірності n, $s(j) = \text{card } Z^j$ – кількість елементів еталону Z^j. Кількість розпізнаваних класів визначається числом J еталонів.

Розглянемо правило класифікації $L: B^n \to [1,...,J]$ у вигляді функції, що $\forall z \in Z$ виконується $L(z) \in [1,...,J]$, так що кожний елемент буде зараховано до одного із еталонів Z^j. При цьому конструктивно та параметрично функція L опирається на апріорні дані конкретної бази, через те що приналежність кожного з елементів $z \in Z$ до відповідного еталону всередині бази задана до початку класифікації.

Для побудови методу класифікації застосуємо дискретну модель простору B^n ознак [14], де для кожного з еталонів Z^j попередньо сформуємо так званий «центр класу» або «центр маси класу», $m_j \in B^n$, $j = 1,..., J$. У процесі здійснення класифікації довільний вектор $b \in B^n$ будемо конкурентно відносити до класу υ у відповідності до найменшої відстані

$$\upsilon = \arg \min_{j=1,...,J} \rho(b, m_j), \tag{2.1}$$

де $\rho(b, m_j)$ – метрика у просторі B^n.

Якщо розпізнавання відбувається на універсумі B^n, то в даному випадку правило (2.1) треба параметрично доповнити логічним аналізом із застосуванням порогу [2]: віднесення дескриптора ОТ до класу υ

здійснюється тільки у тому випадку, коли для вектора b буде також виконано умову $\rho(\mathrm{b},\mathrm{m}_j) \le \delta$, де δ – заданий поріг еквівалентності елементів у просторі. Якщо таке порогове обмеження до класу відсутнє, то буде віднесено зовсім не схожий елемент, але мінімум в (2.1) буде завжди досягнуто.

Відповідно з результатами у запропонованої процедури класифікації, множина дескрипторів $O = \{o_1,...,o_v\}$, $o_i \in B^n$, що описує довільний візуальний об'єкт, буде подана у вигляді цілочисельного вектора $h[\mathrm{O}] = \{\mathrm{h}_1,...,\mathrm{h}_J\}$, де $\mathrm{h}_j = card\ \{\mathrm{o}_i \in \mathrm{O} \mid \mathrm{o}_i \to Z^j\}$ – кількість елементів еталону, віднесених до класу з номером . Одним із методів визначення класу об'єкта є парадигма

$$\mathrm{d} = \arg \max_{j=1,...,J} \mathrm{h}_j \ . \qquad (2.2)$$

Правило (2.2) безпосередньо можна застосувати тільки у ситуації, де для подання $h[Z^j]$ кожного з еталонів, максимум відповідає номеру j еталона, що класифікується. Коли ця умова не виконується й для даних бази еталонів, то необхідно застосовувати інші, більш універсальні підходи [1]. У той же час правила класифікації побудовані із застосуванням (2.2) дають ефективну можливість суттєво спростити прийняття рішення, через те що кожний із структурних елементів може безпосередньо бути класифікований без додаткового аналізу їх сукупних властивостей. Застосування бінарного оброблення та подання дає можливість синтезувати такий спосіб.

Взагалі підхід з визначенням центрів класів, забезпечує значне збільшення швидкодії класифікації за рахунок переходу від обчислення міри релевантності множин до порівняння векторів, або окремих їх елементів [6]. Бінарне подання даних суттєво спрощує процес обробки на програмному рівні, даючи можливість, наприклад в (2.1), застосовувати метрику Хемінга замість більш суттєвих за обсягом обчислень метрик.

Застосування методів кластеризації є традиційним підходом до обчислення центрів, спираючись на ознаки, на основі яких різняться значення дескрипторів ОТ зображення. Беручи до уваги те, що раніш практикувалося використання дескрипторів SIFT та SURF, які взагалі нормовані. Вагомість норми для дескрипторів BRISK не є достатньо значимою характеристикою окремого класу, так для кластеризації застосовувалися безпосередньо значення елементів множини Z [14].

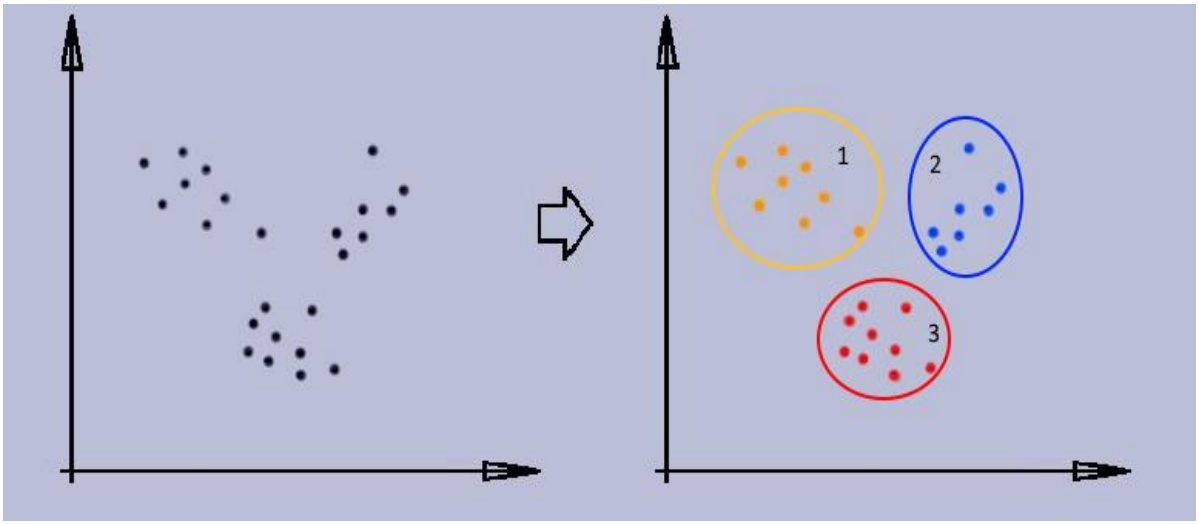


Рисунок 2.1 – Приклад розбиття точок по кластерам

У межах апарату бінарних даних викликає інтерес застосування націленого оброблення множини дескрипторів із визначенням одного з елементів, що буде представляти множину як клас. Двійковий вид дескрипторів дає підвалини застосувати апарат логічного оброблення. Зважаючи на бінарний вид BRISK-дескрипторів, для кожного з еталонів Z^j визначимо бінарний вектор m_j центра класу на підставі такого логічного правила

$$m_j(a) = \begin{cases} 1, & f(Z^j, a) \geq s(j)/2, \\ 0, & \text{інакше}, \end{cases} \qquad (2.3)$$

де $m_j(a)$ – значення біту на позиції з номером a для центру m_j, $f(Z^j,a)$ – функція, що підраховує кількість одиничних бітів на позиції з номером a у множині дескрипторів еталону Z^j, $a=1,...,n$.

Функція $f(Z^j,a)$ може бути обчислена безпосередньо додаванням бітів

$$f(Z^j,a)=\sum_{d=1}^{s(j)} x_d(a), \qquad x_d \in Z^j, \tag{2.4}$$

де $x_d(a)$ – біт з номером a для дескриптора з номером d в описі еталона.

За виразом (2.3) значення кожного з бітів центру m_j для j-го класу визначається переважною більшістю значень відповідних розрядів усіх дескрипторів ОТ, які належать еталону Z^j. Бітове подання можна ефективно використати при роботі з множинами даних. Підрахунок числа одиничних бітів у (2.4) можна здійснити також застосуванням відповідних бінарних функцій [5].

Аналізуємо елементи $x \in Z$ загального вмісту структурних описів бази еталонів (навчальна вибірка) шляхом віднесення їх до відповідного класу з використанням конкуренції (1). У якості ρ застосуємо метрику Хемінга

$$\rho(x,m_j)=\sum_{a=1}^{n} |x(a)-m_j(a)|, \tag{2.5}$$

що визначає кількість розбіжних бітів для двійкових послідовностей однакової довжини.

Відстань Хемінга - число позицій, в яких відповідні символи двох слів однакової довжини різні. У більш загальному випадку відстань Хемінга застосовується для рядків однакової довжини будь-яких q-ічних алфавітів і служить метрикою відмінності (функцією, визначальною відстань в метричному просторі) об'єктів однакової розмірності.

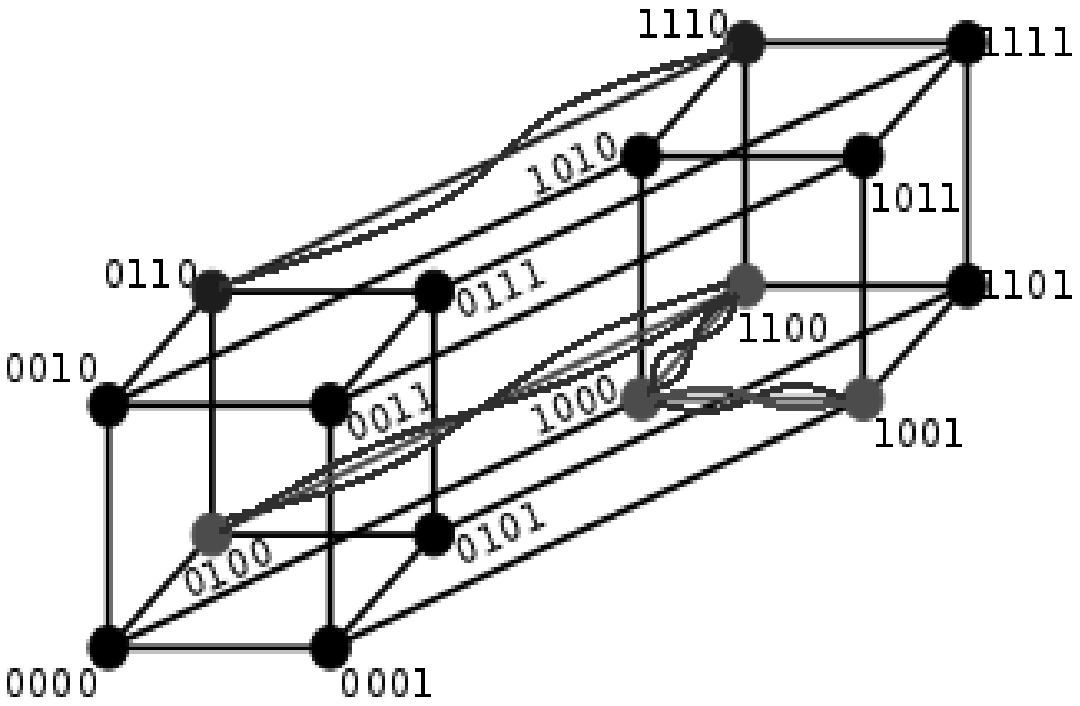


Рисунок 2.2 – Приклади відстаней в двійковому Тессеракті: відстань 1(подвійна риска) 0110 → 1110, відстань 3(потрійна риска) 0100 → 1001

У результаті оброблення з використанням попередньо обчислених центрів одержуємо подання для кожного еталона: $h[Z^i]=(h_1^i,...,h_J^i)$, яке відповідає розподілу елементів множини Z^i за класами еталонів, де h_a^i – цілі числа, причому $\sum_{a=1}^{s(i)} h_a^i = s(i)$, так як кожний з елементів опису буде віднесено до одного з фіксованих класів.

Важливою характеристикою множини отриманих центрів є симетрична матриця відстаней між ними $\Upsilon=\{\rho(m_i,m_j)\}_{i,j=1}^{J}$, яка відображає їх сумісну наближеність у просторі B^n та фактично визначає фінальну якість класифікації. Чим більше віддалені центри між собою, тим вище досяжна ступінь розрізненості дескрипторів із різних класів.

Загальним же критерієм якості квантування простору у відповідності до апріорних даних – опису бази зображень у вигляді множин дескрипторів ОТ може бути нормоване значення розкиду даних навколо сформованих центрів

$$E = \frac{1}{sN} \sum_{j=1}^{J} \sum_{\upsilon=1}^{s(\mathrm{j})} \rho(\mathrm{x}_{\upsilon}, m_j), \tag{2.6}$$

де N – розмір дескриптора ($N = 512$ для BRISK), $s = \sum_{j=1}^{J} s(\mathrm{j})$ – загальне число елементів навчальної вибірки, а для спрощення та приведення до інтервалу 0…1 замість традиційного квадрата метрики вибрано її значення [20].

Другим критерієм при умові правильної класифікації є відношення загального числа дескрипторів, які за результатом оброблення віднесені до правильного класу, до гуртового числа дескрипторів бази

$$\beta = \sum_{i=1}^{J} h_i^i / \mathrm{s}. \tag{2.7}$$

Критерій (2.7) – це відношення кількості правильно класифікованих дескрипторів бази до їх загального числа. Ідеальним варіантом є значення $\beta = 1$.

Зважаючи на бітове подання даних, виникає можливість також застосувати по-байтовий аналіз, використавши замість виразів (2.5), (2.6) міри, обчислені на підставі подібності окремих байтів.

З метою порівняння показників класифікації також розглянемо у якості центру множини її узагальнену медіану [17]. Медіана в математичній статистиці - число, що характеризує вибірку (наприклад, набір чисел). Якщо всі елементи вибірки різні, то медіана - це таке число вибірки, що рівно половина з елементів вибірки більше нього, а інша половина менше нього. У більш загальному випадку медіану можна знайти, упорядкувавши елементи вибірки за зростанням або спаданням і взявши середній елемент. Наприклад, вибірка {11, 9, 3, 5, 5} після упорядкування перетворюється в {3, 5, 5, 9, 11} і її медіаною є число 5. Якщо у вибірці парне число елементів, медіана може

бути не визначена однозначно: для числових даних найчастіше використовують напів-суму двох сусідніх значень (тобто медіану набору {1, 3, 5, 7} приймають рівною 2.4).

Узагальнена медіана m скінченної множини X={x(i)} визначається шляхом мінімізації функціоналу D:

$$m = \arg \min_{\upsilon \in X} D(\upsilon),\ D(\upsilon)=\sum_{x(i)\in X} \rho(x(i),\upsilon), \qquad (2.8)$$

де $\rho(x(i),\upsilon)$ – відстань між елементами. Обчислити медіану можна шляхом агрегації значень рядків матриці взаємних відстаней елементів множини [3]. У результаті обчислення (2.8) отримуємо медіану m_j як один із елементів еталону $m_j \in Z^j$.

Рисунок 2.3 – Порівняння медіани з середнім значенням

Вектор $h[Z^i]$ також може виступити критерієм якості класифікації на навчальній вибірці. Ідеальним класифікатором є той, для якого

$\arg \max_j \mathrm{h}_j^i = i$, тобто при реалізації (2.2) максимальне значення досягається для номера еталону, що класифікується.

2.2 Метод класифікації за значенням центрального дескриптора

Даний алгоритм був розроблений шляхом експериментальної модернізації над моделлю навчання Кохонена під час бакалаврської атестаційної роботи, експериментальним шляхом було доказано перевагу побудови найбільш приближеного до ідеального центру кластера (центрального дескриптора), як за швидкістю так й за результатами класифікації. Також варто зазначити, що в даному випадку послідовність надходження даних не впливає на побудову центру, що робить алгоритм більш стійким.

Нехай задана навчальна множина даних $W = \{\mathrm{x} \mid \mathrm{x} \in \mathrm{R}^n\}$, $\mathrm{W} \subseteq R^n$, що складає набір дескрипторів ОТ бази еталонів. Тепер побудуємо кортеж k опорних (модельних) векторів $\mathrm{M}=\{\mathrm{m}_i\}_{i=1}^k$, $m_i \in R^n$, $i = 1,2,...,k$. Для сформованого набору M апроксимація довільного вектора $x \in W$ конкурентним способом означає визначення номера υ найближчого до нього (у метриці $\rho(\mathrm{x},\mathrm{m}_i)$) вектора $m_\upsilon \in M$ у просторі модельних векторів:

Модель (2.1) називають конкурентним навчанням Кохонена [14]. Для традиційного варіанту онлайн-навчання, якщо дескриптори ОТ $\mathrm{x}[\mathrm{t}] \in \mathrm{W}$ поступають по черзі, центр m_υ кластера, який став переможцем в (1), на кроці $t = 1,2,...,s$ навчання коригується у відповідності до виразу

$$\mathrm{m}_\upsilon[\mathrm{t}+1] = \mathrm{m}_\upsilon[\mathrm{t}] + \alpha[\mathrm{t}](\mathrm{x}[\mathrm{t}] - \mathrm{m}_\upsilon[\mathrm{t}]), \qquad (2.9)$$

де $s = card\ W$ – повний обсяг навчальної вибірки (загальне число ОТ бази зображень), а параметр $\alpha[\mathrm{t}]$ задається дослідником та визначає швидкість

навчання, причому $\alpha[\mathrm{t}] \to 0$ при $\mathrm{t} \to s$. Існує величезне різноманіття стратегій та технологій навчання (2.9), включаючи навіть моделювання динаміки топології мережі [1,25].

У випадку, коли число k кластерів дорівнює числу еталонів бази зображень, кластерування виконує класифікацію на k класів у просторі W. Множина дескрипторів ОТ, що описує довільний візуальний об'єкт, може бути класифікована як один із еталонів.

Зауважимо, що якість класифікації зображень, як правило, оцінюється іншим критерієм, наприклад, значенням ймовірності правильного розпізнавання об'єктів.

Тепер перейдемо до нашого більш простого евристичного підходу бінарного аналізу (Метод класифікації за допомогою центрального дескриптора), суть якого полягає в наступному:

1. Зважаючи на бінарний вид дескрипторів BRISK, для кожного з еталонів Z^i визначимо вектор m_i центра класу на підставі логічного правила як

$$\mathrm{m}_i(\mathrm{b}) = \begin{cases} 1, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) \geq \mathrm{s(i)}/2, \\ 0, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) < \mathrm{s(i)}/2, \end{cases} \quad x_d \in \mathrm{Z}^i, \qquad b = 1,...,512, \tag{2.10}$$

де $x_d(\mathrm{b})$ – біт з номером b для дескриптора з номером d в описі еталона.

Таким чином значення кожного з бітів центру m_i для i-го класу визначається переважною більшістю значень відповідних бітів усіх дескрипторів ОТ, які належать еталону Z^i.

2. Аналізуємо елементи $x \in \mathrm{Z}$ загального вмісту структурних описів бази еталонів (навчальна вибірка) шляхом віднесення їх до відповідного класу з використанням конкурентного способу (2.1). Тут у якості $\rho(\mathrm{x},\mathrm{m}_i)$ застосуємо метрику Хемінга

$$\rho(x,m_i)=\sum_{b=1}^{512}|x(b)-m_i(b)|, \qquad (2.11)$$

що визначає кількість розбіжних бітів для двох двійкових послідовностей однакової довжини (детекторів ОТ).

У результаті оброблення для кожного еталона отримуємо його кластерне подання $H[Z^i]=(h_1^i,...,h_J^i)$, де h_a^i – цілі числа, яке відповідає розподілу елементів множини Z^i за класами еталонів.

Етап попередньої обробки на базі бінарного аналізу можна вважати різновидом хешування на множині дескрипторів ОТ еталону. Аналогічну обробку можна використати і для аналізу множини ORB-дескрипторів.

Кажучи простими словами ми порівнюємо кожен дескриптор зображення, що надійшло до алгоритму з усіма кластерами, що знаходяться в нашій базі. Нехай у нас буде чотири кластери, а кожне зображення має по 40 дескрипторів. Коли перше дескриптор надійшов до алгоритму, то за відстанню Хемінга ми порівнюємо його з усіма кластерами й обираємо той відстань до якого буде найменшою. Нехай перший дескриптор надійшов до 3-ого кластеру, на відміну від Кохонена, цей алгоритм не змінює центр кластеру, а продовжує свою роботу. Так само надходить один за одним інші дескриптори доки вони не закінчаться. Після того, як усі 40 дескрипторів з нашого прикладу будуть розкидані по кластерам, ми починаємо підраховувати скільки дескрипторів надійшло до якого кластеру. Нехай до першого надійшло 3 дескриптори, до другого 2, до третього 30, а до 4-го 5-ть. Таким чином буде вирішено, що зображення буде віднесено до 3-ого кластеру. В нашому випадку 75% дескрипторів буди віднесені до 3-го кластеру, чим більший відсоток тим більша точність розрахунків й ймовірність того, що робота алгоритму була вірна. При випадку коли декілька кластерів отримали однакову кількість кластерів, або різниця у відсотковому співвідношенні між ними менше 20%, це говорить про невдало під обранні параметри алгоритму BRISK, або невдало обрані еталонні

зображення. А також про те, що спосібність гарної роботи програмами є хиткою.

2.3 Застосування вагових коефіцієнтів

Цей метод є модернізацією алгоритму застосування центрального дескриптору тим, що у кожного біта центрального дескриптора додатково з'являється свій ваговий коефіцієнт. Дана модернізація дозволяє зменшити вплив нечітких значень центрального дескриптора. Під нечітким значенням мається на увазі, один з 512 бітів дескриптора, для якого при побудові нулів та одиниць було майже порівну. Таким чином ми вказуємо, що при класифікації дескриптора к центру, полягати на даний біт не варто. Також ми робимо більш значущим біт(значення), який зустрічається частіше. Якщо якесь з значень дескриптора при побутові центрального дескриптора зустрінеться у 100% випадків, то було би логічно, якщо би цей біт впливав на класифікацію більш значуще ніж інші.

Для кожного значення вектора m_i у (2.10) будуємо ваговий коефіцієнт W_i.

$$W_i(\mathrm{b}) = \begin{cases} \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n}, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) \geq \mathrm{s(i)}/2, \\ 1 - \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n}, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) < \mathrm{s(i)}/2, \end{cases} \quad x_d \in \mathrm{Z}^i, \quad b = 1,...,512, \quad (2.12)$$

де $x_d(\mathrm{b})$ – біт з номером b для дескриптора з номером d в описі еталона, n - кількість бітів дескриптора (512).

Таким чином для кожного біту дескриптора ми отримуємо значення коефіцієнту від 0.5 до 1 в незалежності від значення біту, 0 чи 1. Нам

важливо тільки чи співпадає біт чи ні. А цей ваговий коефіцієнт якраз вказує на вірогідність появи біту.

Етап аналізу відрізняється більш суттєво, ми аналізуємо елементи $x \in \mathrm{Z}$ загального вмісту структурних описів бази еталонів (навчальна вибірка) шляхом віднесення їх до відповідного центру (класу) з використанням конкурентного способу (2.14).

$$\upsilon = \arg \min_{j=1,\ldots,\mathrm{J}} \rho(\mathrm{b},\mathrm{m}_j), \tag{2.13}$$

Тут у якості $\rho(\mathrm{x},\mathrm{m}_i,\mathrm{W}_i)$ застосуємо алгоритм оцінки за вагою:

$$\rho(\mathrm{x},\mathrm{m}_i,\mathrm{W}_i)=\sum_{b=1}^{512}\begin{cases} W_i m\ , & x(b)=m_i(b) \\ -W_i m, & x(b)\neq m_i(b) \end{cases}, \tag{2.14}$$

що визначає кількість розбіжних бітів для двох двійкових послідовностей однакової довжини (детекторів ОТ).

У результаті оброблення для кожного еталона отримуємо його подання $\mathrm{H}[Z^i]=(\mathrm{h}_1^i,\ldots,\mathrm{h}_J^i)$, де h_a^i – ціле число, яке відповідає розподілу елементів множини Z^i за класами еталонів.

Різновидністю розглянутого підходу є метод «відмови від слабких». Цей метод є близьким до алгоритму з ваговими коефіцієнтами. Але замість діапазону значень від 0.5 до 1 вагові коефіцієнти приймають дискретні значення 0 або 1.

Для кожного значення вектора m_i (2.13) обчислюємо ваговий коефіцієнт W_i.

Аналізуємо елементи $x \in \mathrm{Z}$ загального вмісту структурних описів бази еталонів (навчальна вибірка) шляхом віднесення їх до відповідного центру (класу) з використанням конкурентного способу (2.14).

У якості $\rho(\mathrm{x},\mathrm{m}_i,\mathrm{W}_i)$ застосовуємо алгоритм оцінки за вагою (2.15).

$$W_i(\mathrm{b}) = \begin{cases} 1, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) \geq (\mathrm{s(i)}/2), \quad \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n} > 0.5 + r \\ 1, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) < (\mathrm{s(i)}/2), \quad 1 - \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n} > 0.5 + r \\ 0, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) \geq (\mathrm{s(i)}/2), \quad \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n} < 0.5 + r \\ 0, & \sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b}) < (\mathrm{s(i)}/2), \quad 1 - \dfrac{\sum_{d=1}^{s(\mathrm{i})} x_d(\mathrm{b})}{n} < 0.5 + r \end{cases} \tag{2.15}$$

де r - це радіус слабких. Ми будемо брати значення 0.05.

Для проаналізованих алгоритмів під час класифікації ми брали до уваги два випадки. Перший випадок, коли в нас біт дескриптора однаковий з відповідним бітом вхідного дескриптора. Другий випадок – коли вони різні. Тепер беремо до уваги тільки ті випадки, коли біти в нас не однакові, аналогічно підрахунку відстані Хемінга. Для вагових коефіцієнтів та відмови для слабких замість формули (2.13) тепер будемо використовувати (2.1). А формула підрахунку вагових коефіцієнтів (2.14) буде приймати наступний вигляд:

$$\rho(\mathrm{x},\mathrm{m}_i,\mathrm{W}_i) = \sum_{b=1}^{512} \begin{cases} 0, & x(b) = m_i(b) \\ W_i m, & x(b) \neq m_i(b) \end{cases}, \tag{2.16}$$

Способи отримання самих вагових коефіцієнтів не змінюються. Дана модифікація частково спрощує підрахунок коефіцієнтів й повністю імітую відстань Хемінга.

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках атестаційної роботи розроблено алгоритми бінарної обробки даних для структурної класифікації зображень. Для реалізації обрана бібліотека з відкритим доступом OpenCV. Це обумовлено можливістю безкоштовного використання бібліотеки та можливості вільного продажу програмного забезпечення, написаного з її використанням, а також висока швидкість, що обумовлюється високопродуктивною мовою C++, й великий обсяг готових рішень, зв’язаних з обробленням зображень.

OpenCV (Open Source Computer Vision Library) [4] - це бібліотека програмного забезпечення з відкритим кодом для комп'ютера та машинного навчання. OpenCV був побудований, щоб забезпечити загальну інфраструктуру для комп'ютерних програм зору та прискорити використання сприйняття машин у комерційних продуктах.

Бібліотека має понад 2500 оптимізованих алгоритмів, що включає в себе повний комплект як класичного, так і сучасного комп'ютерного бачення та алгоритмів машинного навчання. Ці алгоритми можуть бути використані для виявлення та розпізнавання облич, визначення об'єктів, класифікації людських дій у відео, рухання руху камери, відстеження рухомих об'єктів, витягування 3D-моделей об'єктів, створення 3D-точкових хмар із стереокамер, зшивання зображень разом для отримання високої роздільної здатності зображення цілісної сцени, знайти схожі зображення з бази даних зображень, видалити червоні очі з зображень, створених за допомогою спалаху, стежити за рухами очей, визнати декорації та встановити маркери для накладання його на додану реальність тощо. Бібліотека широко використовується в компаніях, дослідницьких групах та урядових структурах [4].

OpenCV має інтерфейси C++, Python, Java та MATLAB і підтримує Windows, Linux, Android та Mac OS. OpenCV спрямовується в основному на додатки в режимі реального часу та використовує інструкції MMX та SSE, коли це можливо. В даний час активно розвиваються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написано спочатку на C++ і має шаблонний інтерфейс, який працює без проблем з контейнерами STL [4].

Причини вибору мови програмування C++ [27]:

- підтримуються різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси);
- передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Весь код, неявно генерується компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної до іншого типу), визначений в стандарті. Також строго визначені місця програми, в яких цей код виконується. Це дає можливість заміряти або розраховувати час реакції програми на зовнішнє подія;
- висока сумісність з мовою Сі, що дозволяє використовувати весь існуючий Сі-код (код на Сі може бути з мінімальними переробками скомпільовано компілятором C ++; бібліотеки, написані на Сі, зазвичай можуть бути викликані з C ++ безпосередньо без будь-яких додаткових витрат, в тому числі і на рівні функцій зворотного виклику, дозволяючи бібліотекам, написаним на Сі, викликати код, написаний на C ++);
- мова підтримує поняття фізичної (const) і логічної (mutable) константності. Це робить програму надійніше, так як дозволяє компілятору, наприклад, діагностувати помилкові спроби зміни значення змінної. Оголошення константності дає програмісту, що

читає текст програми додаткове уявлення про правильне використання класів і функцій, а також може бути підказкою для оптимізації. Перевантаження функцій-членів за ознакою константності дозволяє визначати зсередини об'єкта мету виклику методу (константний для читання, неконстантний для зміни). Оголошення mutable дозволяє зберігати логічну константность при використанні кешей і ледачих обчислень;

- використовуючи шаблони, можливо створювати узагальнені контейнери і алгоритми для різних типів даних, а також спеціалізувати і обчислювати на етапі компіляції;
- можливість імітації розширення мови для підтримки парадигм, які не підтримуються компіляторами безпосередньо. Наприклад, бібліотека Boost.Bind дозволяє пов'язувати аргументи функцій;
- можливість створення вбудованих предметно-орієнтованих мов програмування. Такий підхід використовує, наприклад бібліотека Boost.Spirit, що дозволяє задавати EBNF-граматику парсерів прямо в коді C ++;
- ефективність. Мова спроектована так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми. Жодна з мовних можливостей, що призводить до додаткових накладних витрат, не є обов'язковою для використання - при необхідності мова дозволяє забезпечити максимальну ефективність програми;
- використовуючи шаблони і множинне спадкування можна імітувати класи-домішки і комбінаторних параметризацію бібліотек. Такий підхід застосований в бібліотеці Loki, клас SmartPrt якої дозволяє, керуючи всього декількома параметрами часу компіляції, згенерувати близько 300 видів «розумних покажчиків» для управління ресурсами;
- є можливість роботи на низькому рівні з пам'яттю, адресами.

3.2 Особливості програмної реалізації

Для роботи програми побудовано декілька баз зображень. Для роботи алгоритму потрібна вибірка з еталонними зображеннями, на основі яких будуть побудовані центри дескрипторів, друга база зображень, це зображення які порівнюються з центрами кластеризації еталонних зображень, тобто зображення, що будуть класифіковані до одного з класів (зображень), що знаходяться у вибірці еталонних зображень. За допомогою бібліотеки OpenCV ми отримуємо дескриптори BRISK для кожного зображення (як для еталонів так й для вхідних зображень). Але перед тим як побудувати вектори дескрипторів ОТ за допомогою алгоритму BRISK, треба позбавити зображення від шумів, та провести перетворення зображення у чорно біле при необхідності. Як показано на рисунку 3.1, це робиться для покращення виділення коректних ОТ на зображенні. Для зображень з камер різної якості, та при різному освітленні, первинна обробка зображань скоріше за все буде різною.

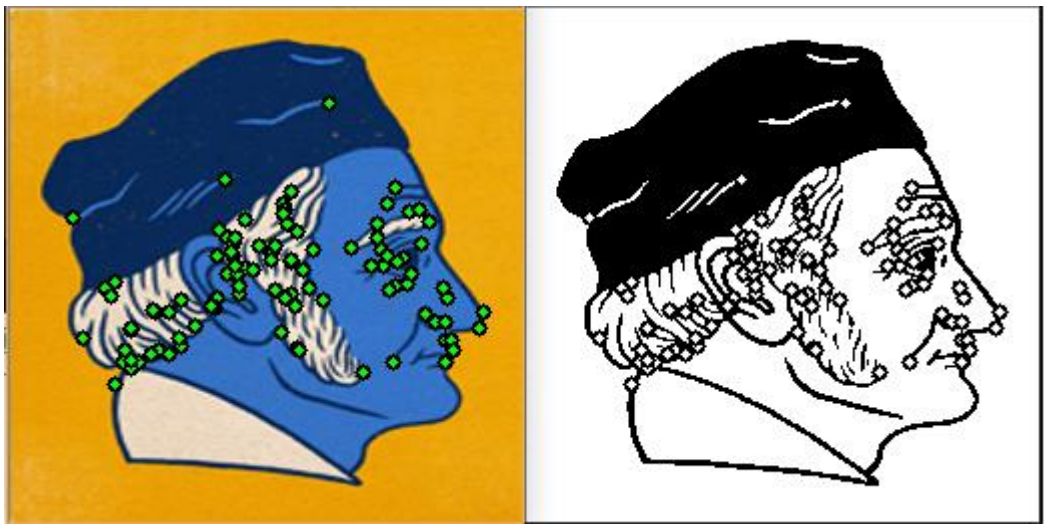


Рисунок 3.1 – Початкова обробка зображення

Наступним кроком після побудови дескрипторів, є вирівнювання кількості ОТ між зображеннями. Тобто, щоб усіх зображень була однакова кількість ОТ, як у зображень вхідних так й у еталонів. Для цього було

побудовано функцію sortDescriptorsAndKP, яка редагує кількість ОТ для кожного зображення.

У якості параметрів функція отримує вектор ОТ та дескрипторів й число особливих точок, що повинно залишитися. Функція позбавляється від найбільш близьких ОТ на зображенні у піксельній відстані. Дозволяє покрити більшу площу на зображенні покриту ОТ, але зменшити їх нагромадження в одній точці. Нижче приведена її програмна реалізація.

```
void sortDescriptorsAndKP(vector<KeyPoint> &kp, Mat &descriptor,int MaxcolElem) {
      int *            Masmindist     = new int[kp.size()-1];
      Mat              descriporSizeN = descriptor.clone();
      vector<KeyPoint> newKP;
      int              k              = 0;
      int delElem = descriporSizeN.rows - MaxcolElem;
      descriporSizeN.pop_back(delElem);
      for (int i = 0; i < kp.size() - 1; ++i) {
            Masmindist[i] = calcHammingDistance(kp[i].pt.x + kp[i].pt.y, kp[i + 1].pt.x
+ kp[i + 1].pt.y);
      }
      for (int ss = 9; ss >= 0; --ss) {
            for (int i = 0; i < kp.size() - 1; ++i) {
                  if (ss == Masmindist[i]) {
                        descriporSizeN.row(k) = descriptor.row(i);
                        newKP.push_back(kp[i]);
                        ++k;
                        if (k == MaxcolElem) { goto a; }
            }}}
a:
      kp         = newKP;
      descriptor = descriporSizeN; }
}
```

Рисунок 3.2 – Програмна реалізація функції sortDescriptorsAndKP

Це робиться заради кращого порівняння наших центрів еталонних зображень й кластерів, що розпізнаються. Бо після побудові векторів ОТ з однаковими параметрами для алгоритму BRISK, для кожного зображення знаходиться різна кількість ОТ. Для зображення у якого буде 215 ОТ, центр буде знайдений більш чітко, але коли у інших зображень буде по 110 ОТ, їх порівняння буде більш чітке між собою ніж з тим же зображенням, що має більшу кількість ОТ, через нерівні умови побудов кластерів, тому ми жертвуємо точністю наших центрів кластерів, але тим самим покращуємо сам алгоритм розпізнавання відносно еталонних зображень. На зображенні

нижче показано, з правої частини - зображення з 250 ОТ до використання алгоритму й з лівої - 105 після.

Рисунок 3.3 – Показ роботи функції sortDescriptorsAndKP

```
int findCenterForKlaster(Mat descriptorsImage, int n){
	int AVG = 0;
	int mas[8] = { 0,0,0,0,0,0,0,0 };
	bitset< 8 > a;
	for (int i = 0; i < descriptorsImage.rows; ++i) {
		a = bitset< 8 >(descriptorsImage.at<uchar>(i, n));
		for (int j = 0; j < 8; ++j)
			mas[j] += a[j];
	}
	for (int j = 0; j < 8; ++j) {
		if (descriptorsImage.rows / 2 > mas[j])
			a[j] = 0;
		else
			a[j] = 1;
	}
	AVG = a.to_ullong();
	return (int)AVG;
}
```

Рисунок 3.4 – Програмна реалізація функції findCenterForKlaster

Наступним кроком будується центральний дескриптор для кожного з еталонних зображень. Для цього було реалізовано декілька алгоритмів: метод класифікації за допомогою центрального дескриптора, метод побудови центрального дескриптора з вагою та метод відмови від слабких.

Оскільки дескриптори мають бінарний вид. То для всіх алгоритмів ми порівнюємо сумарну кількість нулів та одиниць, для кожного з 512 бітів, та обираємо найбільш зустрічаючись для усього вектора в цілому, отриманий дескриптор є нашим центром. На рисунку 3.4 ми бачимо програмну реалізацію методу класифікації за допомогою центрального дескриптора.

```
int findCenterForKlaster(Mat descriptorsImage, int n, double
*mas_frequency_of_occurrence)
  {

    double mas[8] = { 0,0,0,0,0,0,0,0 };
    bitset< 8 > a;

    for (int i = 0; i < descriptorsImage.rows; ++i) {
      a = bitset< 8 >(descriptorsImage.at<uchar>(i, n));
      for (int j = 0; j < 8; ++j)
        mas[j] += a[j];
    }
    for (int j = 0; j < 8; ++j) {
      if (descriptorsImage.rows / 2 > mas[j]) {
        a[j] = 0;
        mas_frequency_of_occurrence[n * 8 + j] = 1.0 -  mas[j]/ descriptorsImage.rows;
      }
      else {
        a[j] = 1;
        mas_frequency_of_occurrence[n * 8 + j] = mas[j] /
(double)descriptorsImage.rows;
      }
    }
    return (int)a.to_ullong();
  }
```

Рисунок 3.5 – Програмна реалізація функції findCenterForKlaster з вагою

На рисунку 3.5 показано реалізацію знаходження центру зображення з вагою. Для цього ми передаємо до нашої функції вектор в якому будемо зберігати вагу для кожного біту побудованого нами центра. Сам центр будується так само як у випадку з попереднім алгоритмом. Таким чином отримавши біт переможець 1 чи 0, ми під вагою отримуємо його співвідношення до біту переможеного. Тобто, якщо ми маємо 100 ОТ, й перший біт був одиниця у 70-ти випадках, а нулем у 30-ти випадках. То перший біт нашого центру буде ‘1’, а його вага буде ‘0.7’. Перейдемо до другого біту, він тепер у 10-ти випадках дорівнював одинці, а у 90-то

випадках був нулем. Тобто другий біт нашого центру буде ’0’, а його вага буде дорівнювати ‘0.9’.

```
int findCenterForKlasterWithoutWeek(Mat descriptorsImage, int n, double
*mas_frequency_of_occurrence)
  {

    double mas[8] = { 0,0,0,0,0,0,0,0 };
    bitset< 8 > a;

    for (int i = 0; i < descriptorsImage.rows; ++i) {
      a = bitset< 8 >(descriptorsImage.at<uchar>(i, n));
      for (int j = 0; j < 8; ++j)
        mas[j] += a[j];
    }
    for (int j = 0; j < 8; ++j) {
      if (descriptorsImage.rows / 2 > mas[j]) {
        a[j] = 0;
        mas_frequency_of_occurrence[n * 8 + j] = floor((1.0 - mas[j] /
descriptorsImage.rows)+0.45);
      }
      else {
        a[j] = 1;
        mas_frequency_of_occurrence[n * 8 + j] = floor((mas[j] /
(double)descriptorsImage.rows)+0.45);
      }
    }
    return (int)a.to_ullong();
  }
```

Рисунок 3.6 – Програмна реалізація функції findCenterForKlasterWithoutWeek

На рисунку 3.6 показано реалізацію знаходження центру зображення за допомогою алгоритму відмови від слабких. Він схожий на алгоритм з ваговими коефіцієнтами але в даному випадку під вагою ми отримуємо значення 1 чи 0. А також ми маємо поріг слабкості, що дорівнює ‘0.05’. Приведемо приклад, ми знову маємо 100 ОТ. Перший біт був нулем у 58 випадках, а нулем у 42 випадках. Таким чином перший біт буде ‘0’, а його вага буде ‘1’. Для другого біту, 0 у нас зустрівся у 46 випадках, а один у 54. Таким чином другий біт буде ‘1’, а його вага ‘0’, через те, що його переважна кількість менша ніж поріг слабкості. Цей алгоритм можна використовувати для оцінки якості збудованого центру на даній вибірці ОТ, чим більше нулів, то тим більш не якісний центр ми отримали. Його слабкість це центри, що є не якісними. В цьому випадку навіть при переможній кількості ОТ, що

приєдналися до нього, його вага може бути меншої ніж у іншого центра у якого вага всіх бітів одиниця, про те тим самим ми звільняємося від ОТ, що приєдналися би через не якісний біт. Тому треба бути обережним при підборі порогу слабкості.

Отримавши центри, ми можемо надіслати до нашої програми вхідні зображення які будуть віднесені до одного з зображень що зараз знаходиться в нашій виборці еталонів.

Першим кроком ми так само як й для еталонних зображень, будуємо векторі ОТ. Після цього кожен дескриптор порівнюється з усіма центрами за допомогою розрахунку відстані по Хемінгу й відноситися до найближчого за відстанню, для методу класифікації за допомогою центрального дескриптора. Для алгоритму з ваговими коефіцієнтами ми перед початком порівнювання дескриптора, кожен центр має значення ваги 0. Після цього ми починаємо порівнювати наш вхідний дескриптор з кожним центром по черзі. Уявимо, що в нас 3 центри. Спочатку ми порівнюємо з 1-шого по 512 біт нашого першого центра з вхідним дескриптором. Якщо біт співпадає, то ми збільшуємо значення на відповідну вагу, якщо не співпав, то зменшуємо. Таким чином ми можемо отримати такі значення: -7.48, 130.67, 15.8 для першого, другого та третього дескрипторів. Переможе той центр у якого значення найбільше. Для відмови від слабких, даний етап такий самий як й у вагових коефіцієнтів.

Коли усі дескриптори віднесені до своїх центрів, обирається центр до якого віднесено найбільша кількість ОТ. Даний евристичний метод не змінює центр зображення, а проводить порівняння дескрипторів з центром, що був збудований у самому початку, що дає перевагу в швидкості над тим самим Кохоненом, або іншими алгоритмами, у яких центр кластеризації постійно зміщається. На цьому етапі наше розпізнавання завершується.

Також було програмно реалізовано додаткові функції такі як побудова графіків та діаграм дескрипторів, конвекторів даних як внутрішнього використання у алгоритмах так й для виводу інформації для користувача, що

дозволяє оцінити роботу алгоритмів й порівняти їх (на рисунку 3.17 ви побачите одну з додаткових класів MySort). Нижче приведено діаграму послідовності роботи програми.

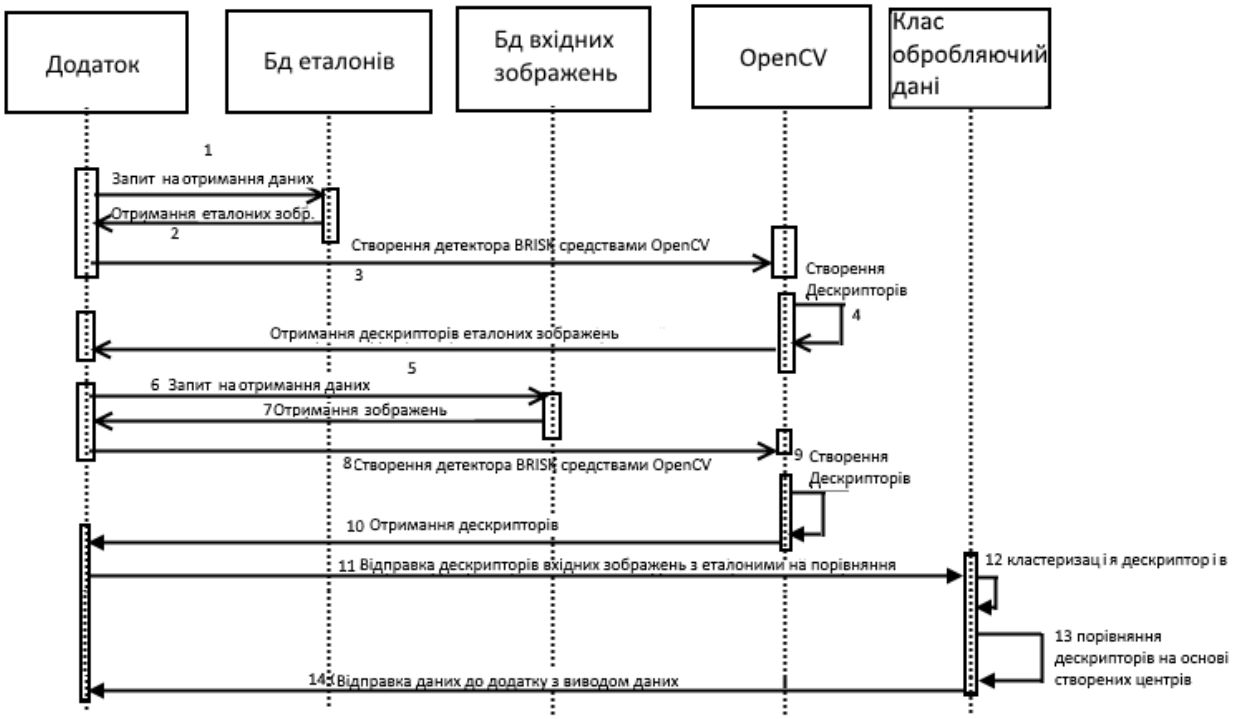


Рисунок 3.7 – Діаграма послідовності

3.3 Інструкція користувача

Оскільки програма представляє собою алгоритмічну реалізацію вирішення задач класифікації, то будемо розглядати її як бібліотеку. Для її використання, спочатку до проекту треба підключити бібліотеку OpenCV додатком до якої є дана робота. Виконуємо наступну послідовність дій:

Крок 1. Завантаження та установка OpenCV

Насамперед необхідно завантажити інсталяційний файл OpenCV (Доступний на офіційному сайті проекту: http://opencv.org/*). Нас цікавить версія для Windows.

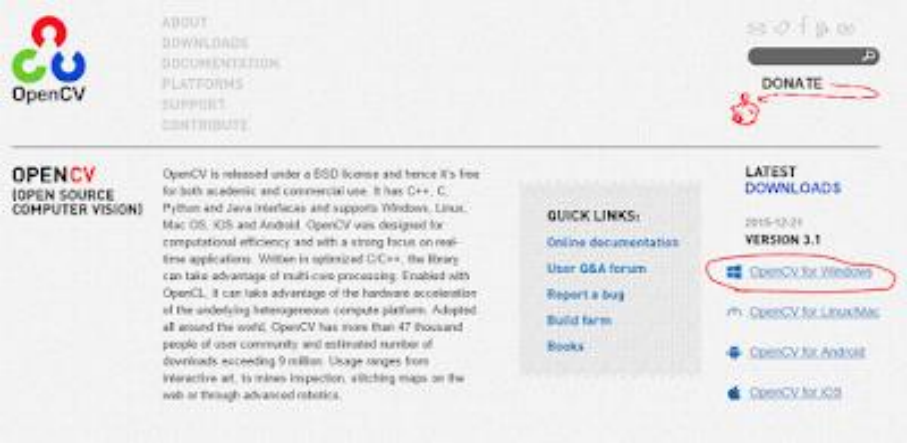


Рисунок 3.8 – Сторінка офіційного сайту OpenCV

Після того, як файл завантажиться, запускаємо його і вибираємо каталог для установки: встановити можна куди вам захочеться. Виберемо наприклад диск D: \ (Створювати папку не треба: при розпакуванні в обрану директорію додасться папка "opencv").

Після розпакування отримуємо досить важку папку:

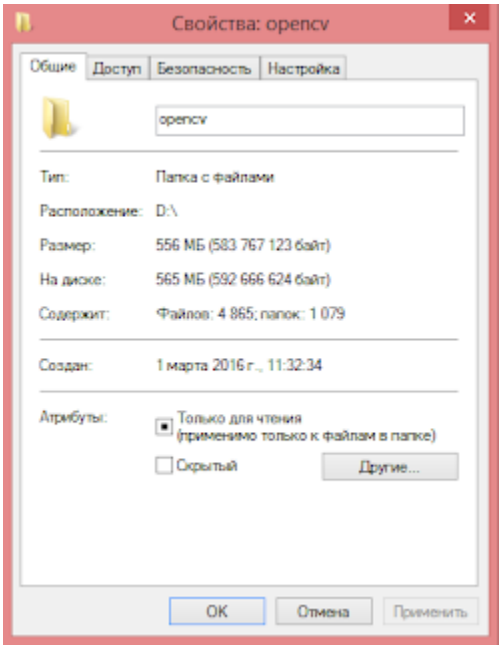


Рисунок 3.9 – Розпакована бібліотека OpenCV у розділ D

Крок 2. Налаштування змінних середовища

Для комфортного використання рекомендується налаштувати змінні середовища.

Відкриваємо: "Панель управління" > "Система" > "Додаткові параметри системи" > Додатково > "Змінні середовища ..."

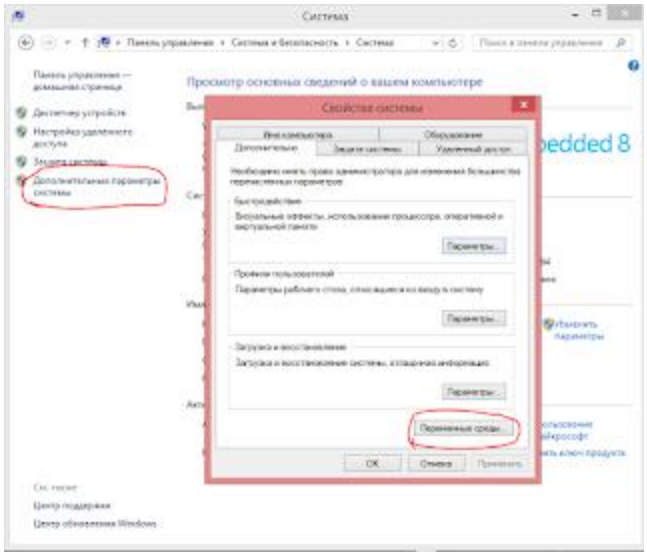


Рисунок 3.10 – Налаштування системи

Створимо таку системну змінну:

- Змінна: OPENCV_DIR.
- Значення: [Місце установки opencv] \ opencv \ build \ x64 \ vc14 (В нашому випадку: D: \ opencv \ build \ x64 \ vc14).
- Далі виправляємо змінну Path: Додаємо в кінець її значення наступний рядок:;% OPENCV_DIR% \ bin.
- Після додавання змінної середовища необхідно перезавантажити комп'ютер;

Крок 3. Налаштування проекту Visual Studio

Якщо під час попередніх кроків Visual Studio була запущена, її необхідно перезапустити

У проекті, де необхідно використовувати OpenCV відкриваємо властивості проекту (Проект > Властивості або Alt + F7)

Перед внесенням змін виберіть конфігурацію "Усі зміни", а платформу "x64".

У закладці C / C ++ Вказуємо додаткові каталоги включення файл: $ (OPENCV_DIR) \ .. \ .. \ include

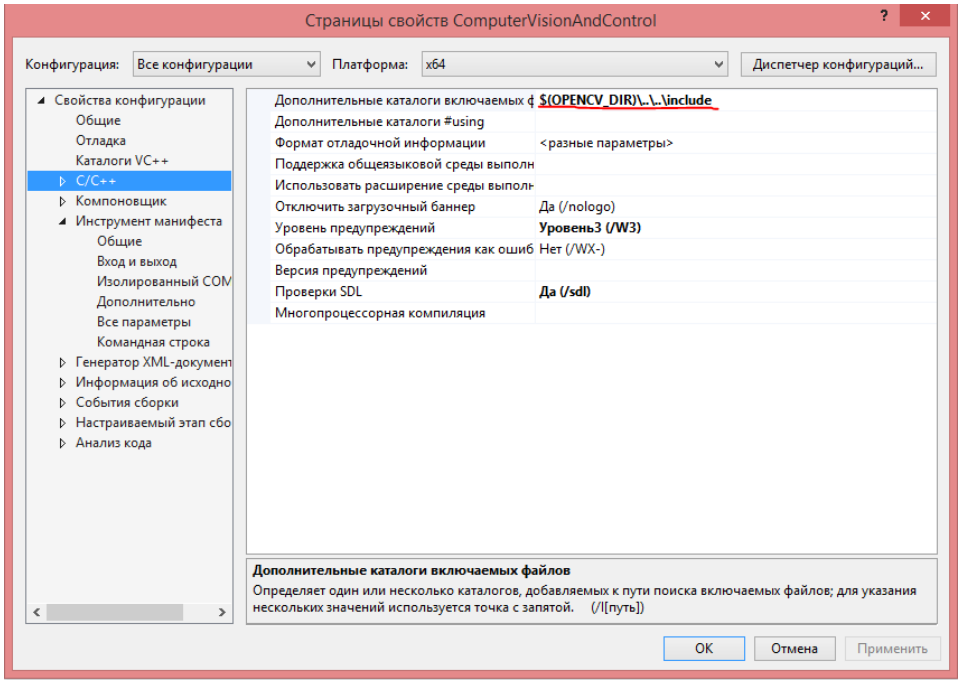


Рисунок 3.11 – Конфігурація проекту. C/C++ , Додаткові каталоги включень

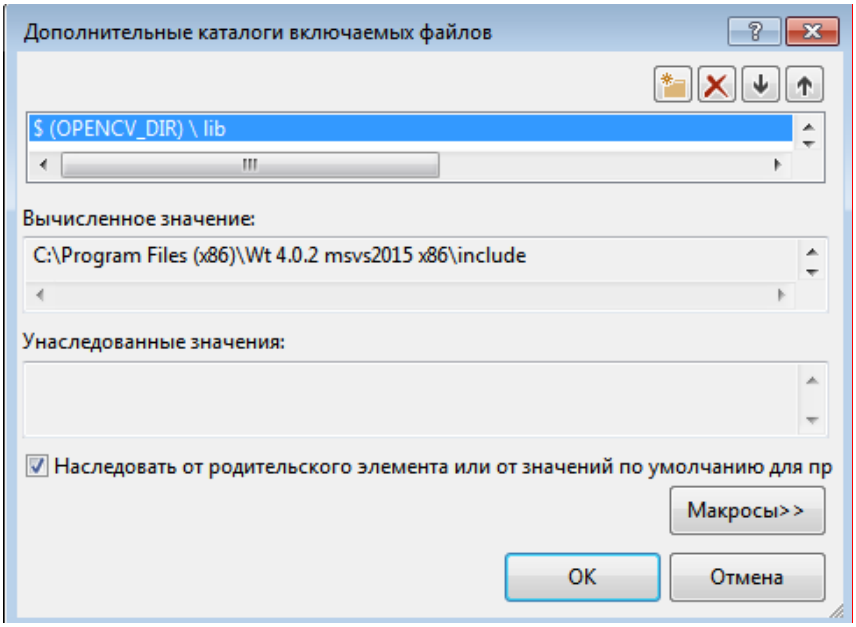


Рисунок 3.12 – Додаткові каталоги включень

У закладці Лінкер > Загальні вказуємо додаткові каталоги бібліотек: $ (OPENCV_DIR) \ lib

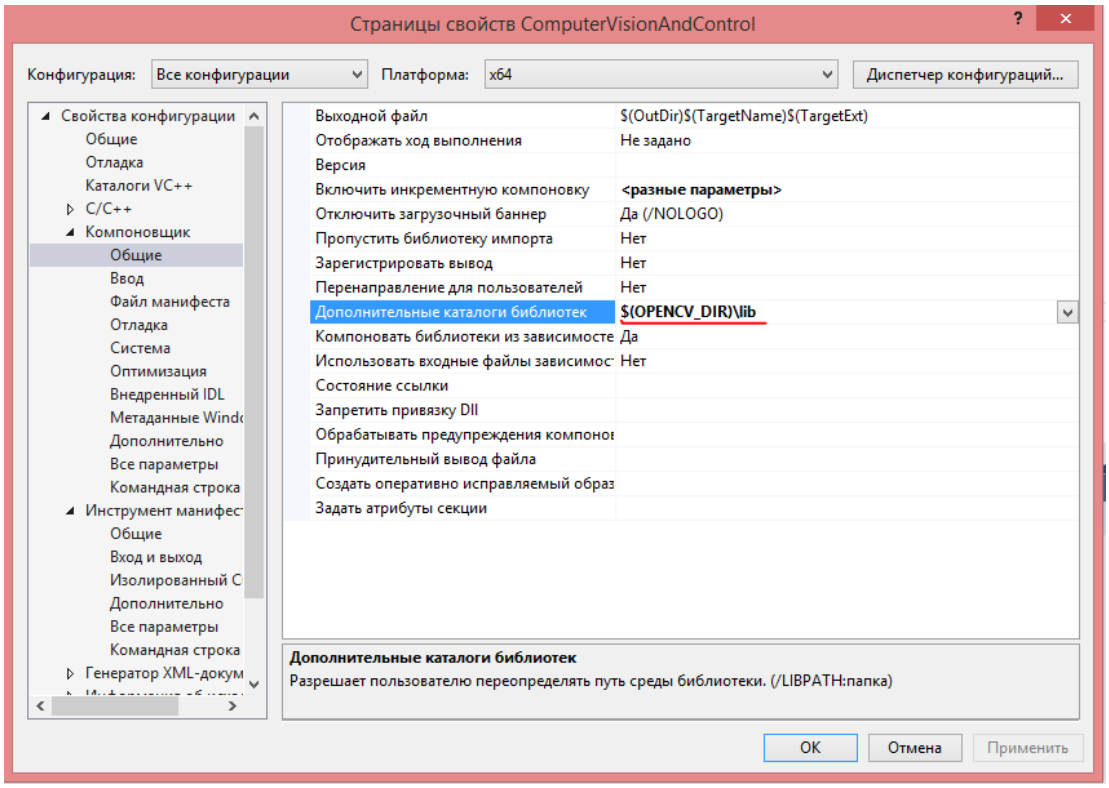


Рисунок 3.13 – Конфігурація проекту. Лінкер , Додаткові каталоги бібліотек

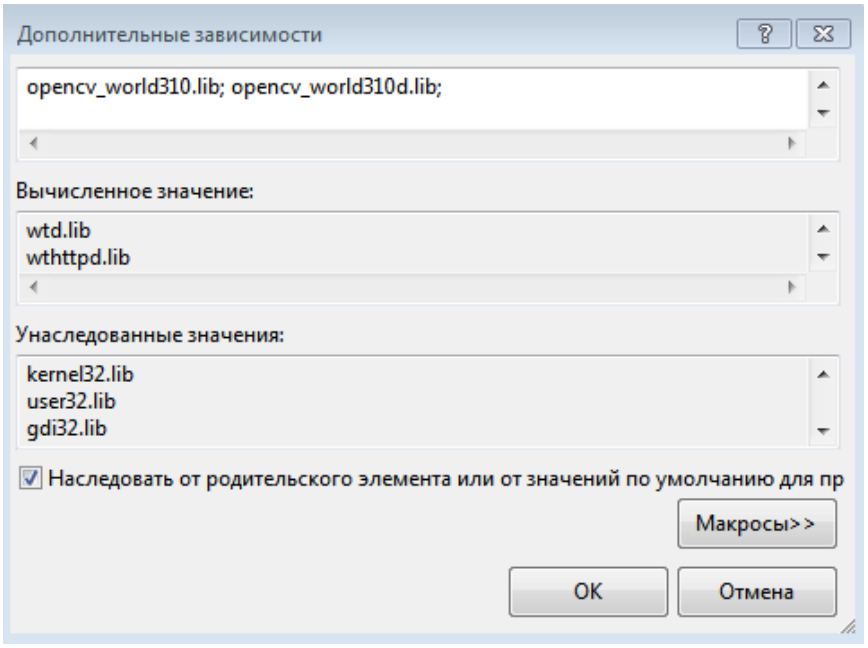


Рисунок 3.14 – Додаткові каталоги бібліотек

У закладці Лінкер > Введення вказуємо додаткові залежності: opencv_world310.lib; opencv_world310d.lib; Та копіюємо їх у головну папку проекту, а також до папки з виконуючим файлом.

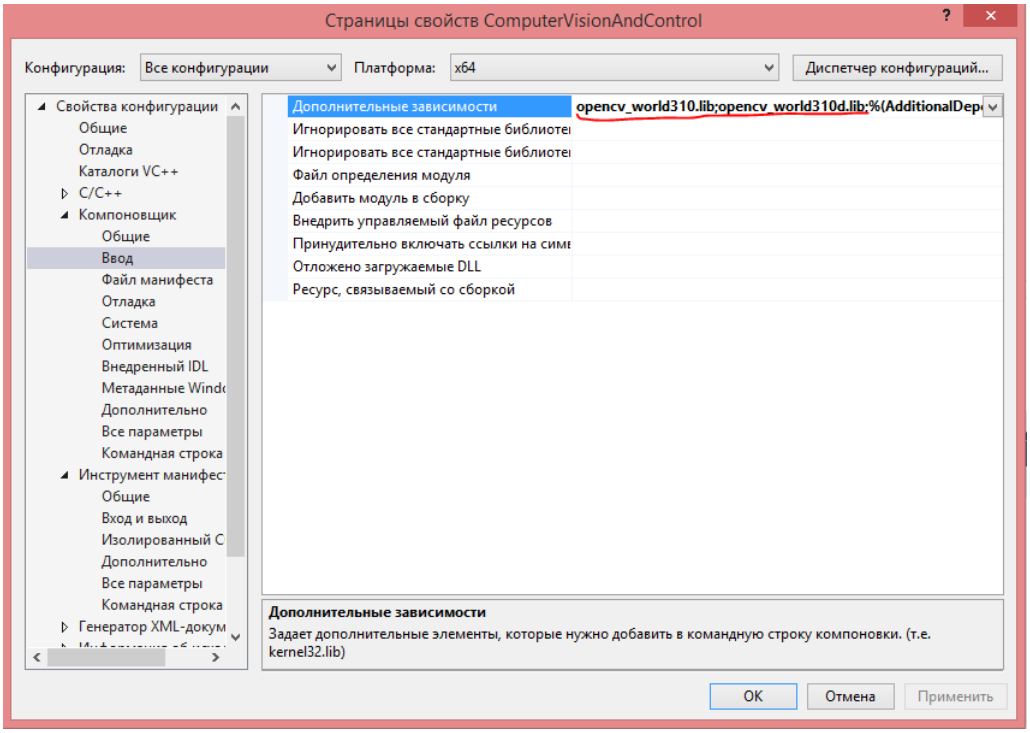


Рисунок 3.15 – Конфігурація проекту. Лінкер

```
Create_Central_descriptors centers(detectorBRISK, img, number_of_image/2, 51, 'H');
Image_definition::definition(detectorBRISK, centers, img[i], 51, 'H');
```

Рисунок 3.16 – Побудова дескрипторів BRISK

Тепер ми можемо приступати до підключення файлу, його можливо узяти як відкритий код, або як бібліотеку. В нашому випадку ми просто скопіюємо відкритий код до нашого проекту. Три файли kasterization.h, Work_With_Brisk.h та myMatche.h. Як користувач, ми маємо два класи Create_Central_descriptors, та Image_definition. Перший реалізую алгоритми створення центрів. Для цього, п’ятим параметром передається ‘H’,’W’ або ’S’ – метод побудови центрального дескриптора, вагових коефіцієнтів та відмови

від слабких відповідно. А других клас відносить вхідне зображення до одного з побудованих нами центрів, що був переданий до програми попереднім кроком. В даній конфігурації програма розрахована на використання з дескрипторами BRISK.

Можна шостим параметром передати true, або false для друку даних, з їх обробкою. На рисунку 3.16 приведено приклад в якому до бази було внесено, зображення цифр. Метод Klast::distributionDescriptors проводить розподілення дескрипторів ОТ по збудованим нами центрам й повертає бінарний результат. Також ми маємо внутрішній клас MySort завдяки якому ми проводимо обробку даних й можемо виводити наступні данні, що показані на рисунку 3.18.

```
for (int i = 0; i < n2; ++i) {
            MySort                    mySort;
            vector<vector<vector<bitset< 8 > > > > klaster =
Klast::Convert_to_vector_bits(klasterDescripIm, n);
            mySort.matr = Klast::distributionDescriptors(klaster,
Klast::Convert_to_vector_bits(&descriptorsImage2[i]));
            mySort.sort();
            mySort.show();
            std::cout << "This is number: " << mySort.maxId() << std::endl;
            imshow("matches", imgMatch2[i]);
            while (true) {
                  if (cv::waitKey(10) == 27)
                        break;
            }
      }
```

Рисунок 3.17 – Обробка дескрипторів

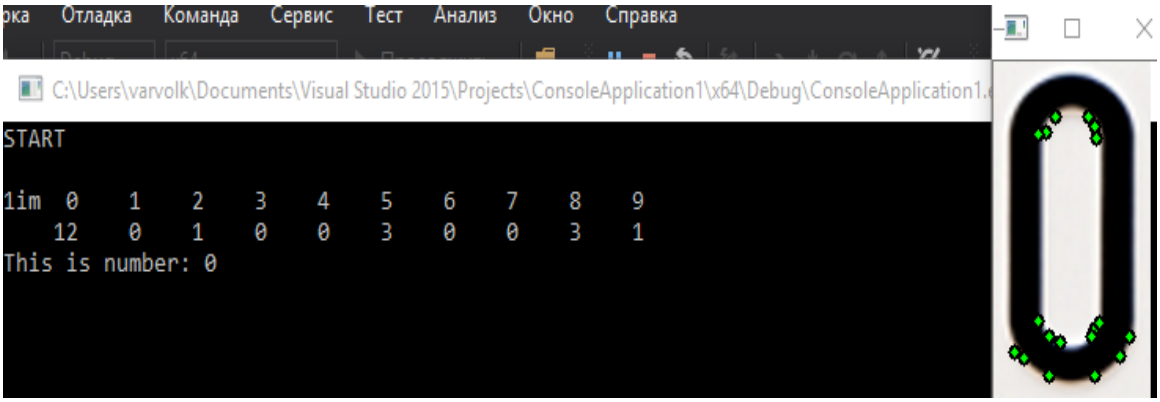


Рисунок 3.18 – Приклад роботи програми

3.4 Аналіз результатів дослідження

У середовищі C++ з використанням бібліотеки програм OpenCV [4] нами розроблено програмну модель для класифікації зображень на підставі описів, сформованих детектором BRISK.

Проведено ряд дослідів. Для першого, у якості бази еталонів було узято 15 зображень (рис. 3.19). Усі зображення мають однаковий розмір 256x256, та обмежені кількістю особливих точок на момент передачі дескрипторів до алгоритмів, кожне зображення має 51 дескриптор. Через обчислення центрів класів у відповідності до логіки (3) сформовано вектори $h[Z^i]$ для кожного із еталонів. Приклади центрів показано на рисунку 3.20.

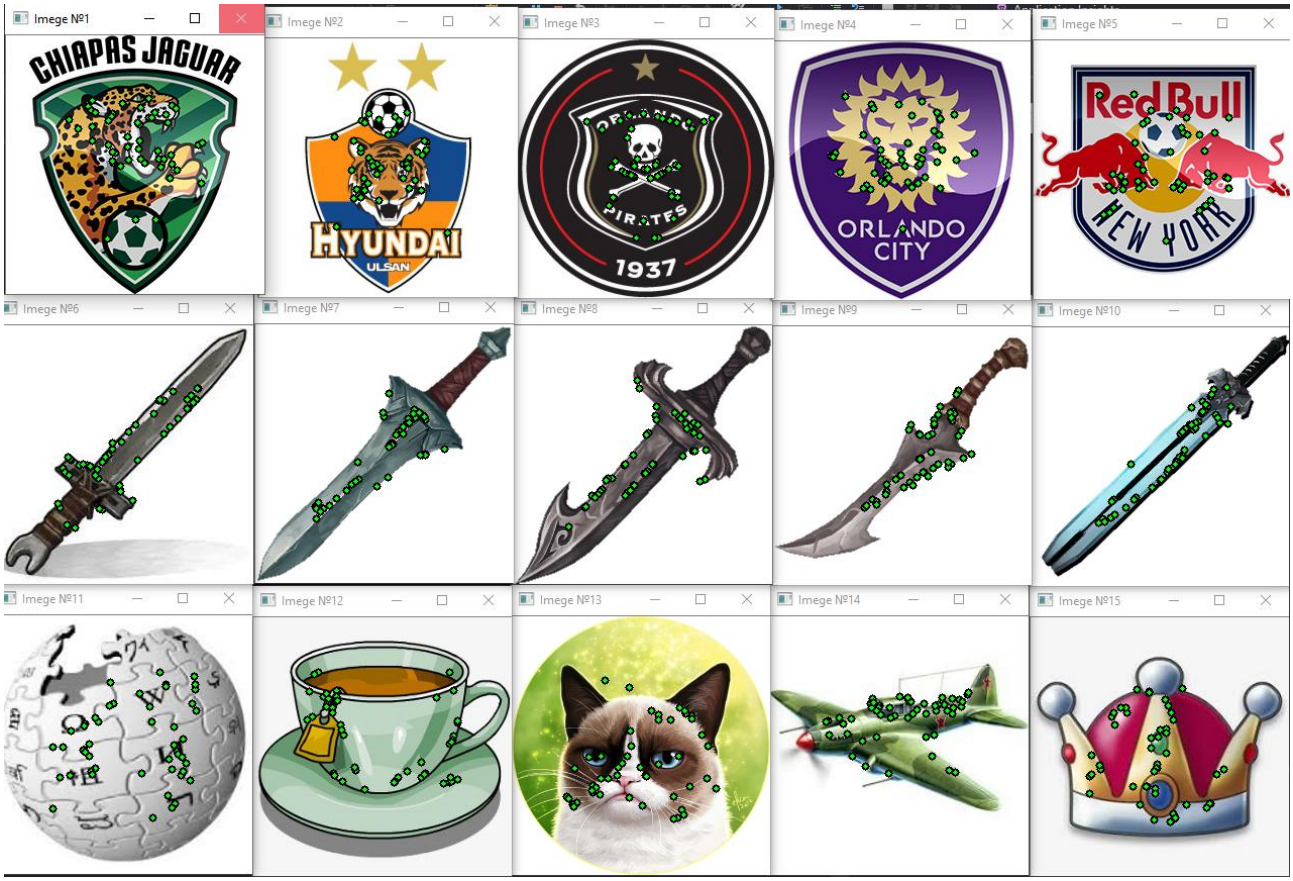


Рисунок 3.19 – Тестова вибірка зображень №1

Рисунок 3.20 – Бітове подання (2.3) для центрів зображень рисунку 4.1 для 7-мого, та 8-мого зображень відповідно

В першому досліді порівнюється швидкодія роботи алгоритмів та їх працездатність. Швидкодію ми будемо вимірювати за допомогою Runtime у режимі від лагодження програми, тому час будемо вимірювати умовними одиницями (УО). А результативність відображається кількістю правильно віднесених ОТ зображення X до зображення X.

	з.1	з.2	з.3	з.4	з.5	з.6	з.7	з.8	з.9	з.10	з.11	з.12	з.13	з.14	з.15
з.1	28	5	4	0	0	0	6	0	1	1	0	2	0	2	2
з.2	5	25	0	8	0	1	0	1	4	0	2	0	2	2	1
з.3	1	0	43	2	5	0	0	0	0	0	0	0	0	0	0
з.4	0	0	0	48	0	0	0	1	0	0	0	0	1	1	0
з.5	1	0	1	2	27	1	1	4	1	0	4	1	5	3	0
з.6	0	1	0	1	0	25	1	2	6	7	0	0	2	3	3
з.7	13	0	0	0	0	6	16	7	1	0	0	4	0	3	1
з.8	6	1	0	1	0	6	2	20	3	2	0	4	1	3	2
з.9	0	0	0	0	1	2	0	7	25	3	0	5	1	6	1
з.10	0	0	0	0	1	7	2	2	12	25	0	1	0	1	0
з.11	0	0	2	2	8	0	0	0	2	1	23	3	7	3	0
з.12	3	2	0	1	6	1	4	2	1	3	0	20	0	7	1
з.13	0	0	1	4	0	0	0	4	0	1	1	0	33	0	7
з.14	0	2	0	2	1	13	0	4	4	0	0	0	0	23	2
з.15	2	1	1	4	2	2	0	0	2	1	0	3	2	0	31

Рисунок 3.21 – Результат класифікації

На рисунку 3.21 по діагоналі, що переважно темно зеленого кольору, ми можемо побачити скільки дескрипторів зображення 1-15 віднеслись з 51-го до свого же зображення. На обробку 15-ти зображень було затрачено 6000 УО. Найбільш ефективно було розпізнане зображення 3 та 4, вони мають більше 40-ка ОТ віднесених вірно. Проблеми були виявлені з зображенням 7. В цьому випадку всього лиш 16 ОТ були віднесено вірно, а ще й 13-ть ОТ було віднесено до 1-шого центру. В цілому алгоритм відніс усі зображення вірно за доволі швидкий час.

	з.1	з.2	з.3	з.4	з.5	з.6	з.7	з.8	з.9	з.10	з.11	з.12	з.13	з.14	з.15
з.1	29	1	3	0	0	0	8	2	1	2	0	1	0	2	2
з.2	2	26	0	12	0	1	0	2	2	0	0	0	1	4	1
з.3	1	0	44	2	2	0	0	0	0	0	0	0	0	1	1
з.4	0	0	0	49	0	0	0	1	0	0	0	0	0	1	0
з.5	1	0	1	8	23	2	0	3	1	0	2	1	6	3	0
з.6	0	1	0	2	0	32	0	1	6	2	0	0	1	3	3
з.7	13	0	0	0	0	15	16	1	1	2	0	2	0	1	0
з.8	5	1	0	2	0	17	2	9	3	1	0	4	1	5	1
з.9	0	0	0	0	0	14	0	2	22	1	0	4	0	6	2
з.10	0	0	0	0	0	10	0	0	14	24	0	1	0	1	1
з.11	0	0	2	7	5	0	0	0	4	1	19	3	7	2	1
з.12	3	2	0	3	0	3	5	2	0	3	0	16	0	12	2
з.13	0	0	0	19	0	3	0	3	0	1	0	0	24	0	1
з.14	0	0	0	4	0	18	0	0	1	0	0	0	0	28	0
з.15	0	1	1	8	0	5	0	0	0	3	0	0	1	1	31

Рисунок 3.22 – Результат класифікації з ваговими коефіцієнтами

Метод вагових коефіцієнтів показів не зовсім вдалі результати. Крім того, що було затрачено більше часу, це – 8200 УО, так й загальний працездатність класифікації зменшилась. Зображення 8-мь, зовсім не було розпізнане, а для зображень 7,12,13,10 результати погіршились доволі значуще. Проте для 1-4 та 6 результати трохи покращилися, але доволі не значуще.

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
S1	29	3	3	0	0	0	8	2	1	1	0	1	0	2	1
S2	4	29	0	6	0	0	0	2	4	2	1	0	2	0	1
S3	1	0	44	2	4	0	0	0	0	0	0	0	0	0	0
S4	0	0	0	49	0	0	0	1	0	0	0	0	0	1	0
S5	1	0	2	5	27	1	0	5	0	0	3	1	4	2	0
S6	0	1	0	1	0	25	0	3	9	4	0	0	2	3	3
S7	12	0	0	0	0	12	19	1	1	1	0	3	0	2	0
S8	8	0	0	0	0	7	2	21	4	2	0	3	0	2	2
S9	0	0	0	0	0	5	0	6	29	3	0	4	0	2	2
S10	0	0	0	0	1	8	0	2	12	26	0	1	0	1	0
S11	0	0	3	2	7	0	0	0	2	1	25	3	5	2	1
S12	3	1	0	1	5	0	5	4	0	6	0	21	0	5	0
S13	0	0	0	4	0	0	0	7	0	1	0	0	34	0	5
S14	0	1	0	1	1	13	0	4	2	0	0	0	0	27	2
S15	3	1	1	4	1	2	0	0	2	4	0	2	2	0	29

Рисунок 3.23 – Відмова від слабких

Метод відмови від слабких (рисунок 3.23), для всіх зображень окрім 15-го, відніс правильно більшу кількість ОТ у порівнянні з методом центрального дескриптора. Він є більш вдалою модифікацією, проте час який він витратив – 10062 УО, є доволі значним у співвідношенні з часом методу центрального дескриптора.

Тепер перейдемо до методів з дистанцією (рисунок 3.24), на методі вагових коефіцієнтів дана модифікація впоралася за 8200 ОУ. Та показала куди більш цікавий результат. Значно покращилась працездатність, проте 7-ме зображення було класифіковано не вірно. Хоча треба зазначити, що для частини зображень детектування стало кращим ніж для методу побудови центрального дескриптора (МПЦД), на основі якого ми й проводимо модифікації. Так, для 1,2,5,11 та 12-го зображень кількість ОТ була значно більша від МПЦД. Треба зазначити, що усі дані мали однакову первину обробку, що для методу вагових коефіцієнтів могло сказатися в погану сторону, але інакше були б порушені умови порівняння.

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
s1	35	0	5	0	0	0	5	0	0	0	0	4	0	0	2
s2	6	30	0	0	1	0	0	0	1	0	6	3	0	0	4
s3	1	0	44	0	5	0	0	0	0	0	0	0	0	0	1
s4	0	0	0	43	1	0	0	2	0	0	1	0	2	1	1
s5	1	0	2	1	38	0	1	2	0	0	5	0	1	0	0
s6	0	2	0	0	0	22	2	4	8	3	1	2	1	3	3
s7	16	0	0	0	0	2	15	6	1	0	0	4	0	6	1
s8	10	2	0	0	0	4	1	19	3	1	0	4	0	3	4
s9	0	0	0	0	2	2	0	7	25	3	0	5	0	6	1
s10	0	0	0	0	2	5	2	3	11	21	1	4	0	2	0
s11	0	0	2	0	10	0	0	0	1	0	32	3	2	1	0
s12	4	2	0	0	10	0	2	1	1	1	0	25	0	5	0
s13	1	0	2	0	3	0	0	3	0	0	3	0	31	0	8
s14	0	2	0	0	3	4	0	6	1	3	1	0	0	29	2
s15	3	3	1	2	2	0	0	1	1	0	0	3	0	2	33

Рисунок 3.24 – Метод вагових коефіцієнтів з дистанцією

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
s1	37	1	3	0	0	0	6	0	0	0	0	2	0	1	1
s2	7	30	0	0	1	0	0	0	0	0	8	1	0	0	4
s3	4	0	41	0	5	0	0	0	0	0	0	0	0	0	1
s4	0	0	0	34	5	0	0	2	0	0	4	0	4	1	1
s5	2	0	3	0	38	0	2	1	0	0	5	0	0	0	0
s6	1	2	0	0	0	18	2	9	8	0	2	2	0	3	4
s7	16	0	0	0	0	0	19	8	1	0	0	1	0	5	1
s8	10	3	0	0	0	3	2	23	1	1	0	3	0	1	4
s9	0	1	0	0	1	1	4	9	24	1	2	2	0	5	1
s10	0	0	0	0	1	4	4	3	9	20	3	4	0	3	0
s11	0	0	2	0	9	0	0	0	1	0	36	3	0	0	0
s12	5	0	0	0	13	0	8	2	1	0	0	19	0	3	0
s13	2	0	2	0	9	0	0	3	0	0	6	0	19	0	10
s14	0	2	0	0	3	2	0	9	4	0	4	0	0	24	3
s15	9	3	0	1	3	0	0	0	1	0	0	5	0	2	27

Рисунок 3.25 – Відмова від слабких з дистанцією

Метод відмови від слабких з дистанцією (рис. 3.25) – дана модифікація впоралася за 8200 ОУ. У порівнянні з МПЦД працездатність однакова, проте гірша ніж для звичайного методу Відмови від слабких.

Підведемо результати першого етапу тестування на вибірці з 15 випадкових зображень, побудувавши таблицю оцінювання.

Таблиця 3.1 – Результати першого досліду

Метод	УО	Загальна кількість вірних ОТ	Кількість Зображень у я ких 25+ ОТ вірних	Помилки	Місце AVG
МПЦД	6000	412	10	0	2
Вагових Коефіцієнтів (ВК)	8200	392	7	1	5
Відмова від сдабкого (ВВС)	8200 (8150)	434	12	0	1
ВК з дистанцією	8200 (8100)	442	11	1	3
ВВС з дистанцією	8200 (8150)	409	7	0	4

Через не великий розмір вибірки знайти точну різницю в швидкості не вдалося, бо для одних й тих самих даних комп’ютер видає не детермінований час, цей час то для однієї модифікації кращий, то для другої. В дужках вказано примірний час який мав би бути у швидкості до ВК.

Для другого експерименту було узято вибірку 4 облич 2-ох жінок та 2-ох чоловіків. Кожне з облич має по 2-фотографії зроблені у невеликий проміжок часу. В нашій виборці еталонів знаходяться тільки перші чотири зображення. В виборці зображень, що будуть класифікуватися, будуть знаходитися усі 8-сім зображень. В цьому експерименті ми перевіримо ефективність методів на класифікацію обличчя за одним фото. Треба зазначити, що на зображені фото пари близнюків (брат-сестра).

Для даного експерименту ми брали по 41-ій ОТ для зображення, побачити як наші ОТ розляглися на зображеннях ви можете побачити на (Рис. 3.27).

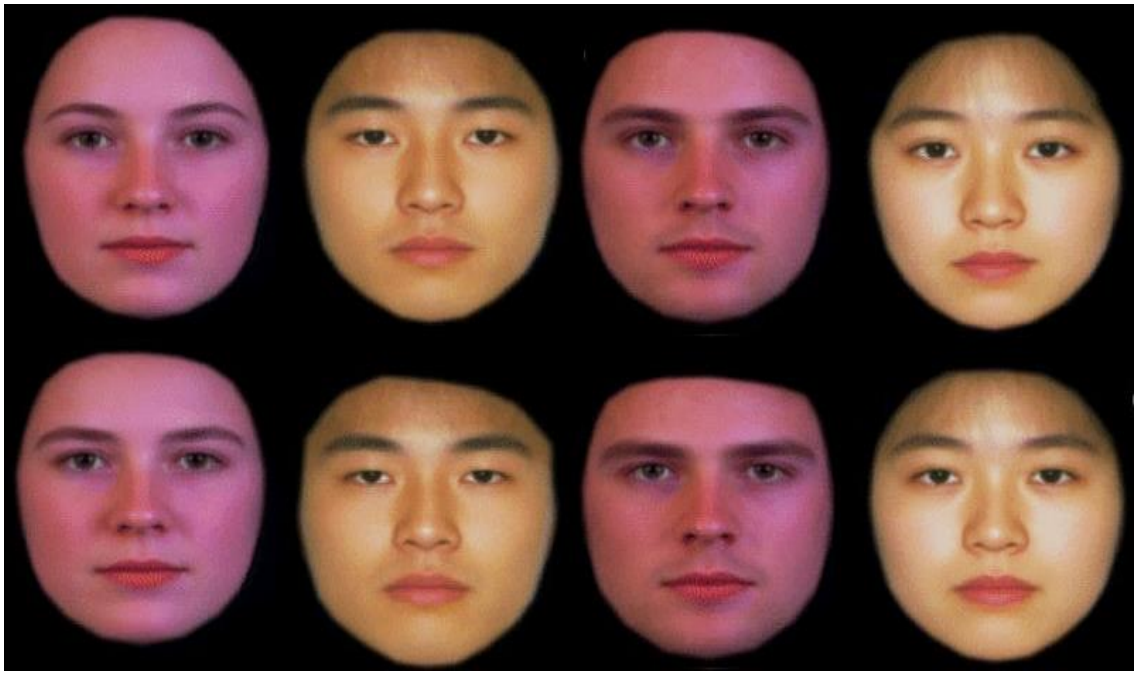


Рисунок 3.26 – Тестова вибірка зображень №2

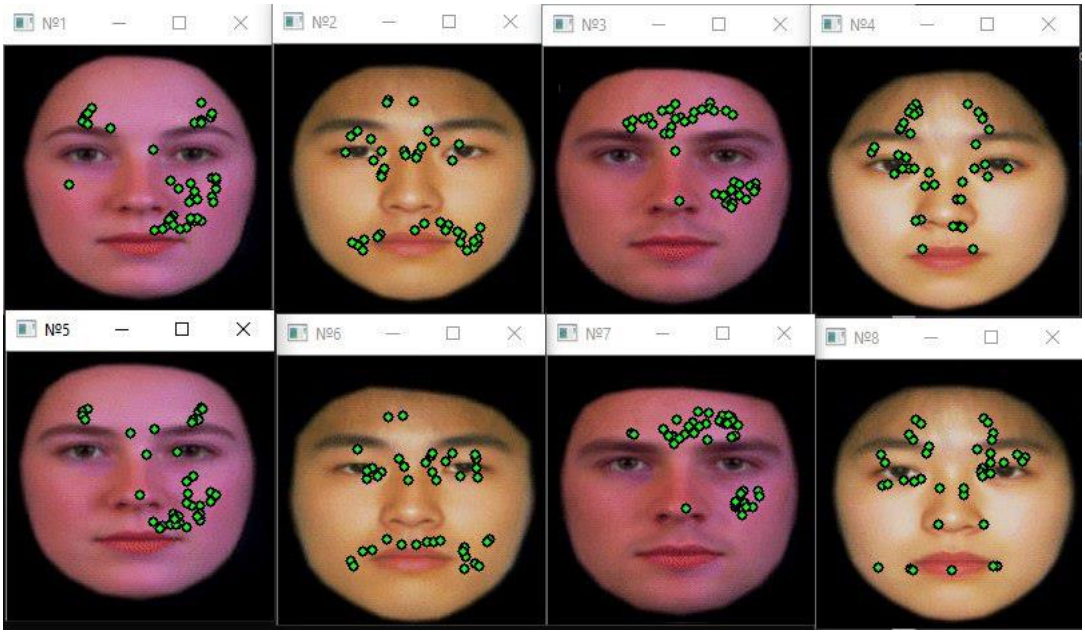


Рисунок 3.27 – Тестова вибірка зображень №2 з їх ОТ

МПЦД показав доволі впевнені результати, класифікувавши усі зображення. Дані для інших алгоритмів будемо порівнювати саме з цим методом.

	s1	s2	s3	s4
s1	31	0	3	7
s2	7	24	5	5
s3	1	2	38	0
s4	1	8	1	31
s5	22	2	2	15
s6	10	19	10	2
s7	8	5	28	0
s8	0	9	6	26

Рисунок 3.28 – Метод центрального дескриптора

Метод вагових коефіцієнтів знову показав не зовсім стійкий результат, не розпізнавши 6-осте зображення. Окрім значно кращого детектування 1-го, та 5-го зображень європеоїдної жінки. У метода вагових коефіцієнтів чітких переваг виявлено не було.

	s1	s2	s3	s4
s1	37	0	1	3
s2	12	18	4	7
s3	3	1	36	1
s4	6	2	2	31
s5	32	0	1	8
s6	17	14	7	3
s7	15	1	25	0
s8	3	6	4	28

Рисунок 3.29 – Метод вагових коефіцієнтів

Метод відмови від слабких показав результат кращий, ніж МПЦД й отримав майже ідеальні результати. У МПЦД були більш погано

класифіковані зображення 5-ть та 6-ть. Для інших зображень класифікація пройшла ідентично МПЦД.

	s1	s2	s3	s4
s1	32	0	2	7
s2	5	24	5	7
s3	2	2	37	0
s4	0	8	1	32
s5	27	1	1	12
s6	8	20	5	8
s7	7	4	28	2
s8	0	10	6	25

Рисунок 3.30 – Відмова від слабких

	s1	s2	s3	s4
s1	31	0	3	7
s2	3	33	1	4
s3	0	6	35	0
s4	0	12	0	29
s5	16	9	3	13
s6	6	29	2	4
s7	0	9	32	0
s8	0	17	2	22

Рисунок 3.31 – Метод вагових коефіцієнтів з дистанцією

Цікавіше ніж в першому експерименті показав себе алгоритм ВК з дистанцією. Показавши гарний результат детектування перших 4-рех зображень. Та трохи гірший результат з другою четвіркою. Цікавим є той факт, що зображення 6-ть розпізнається набагато краще ніж в інших алгоритмах, при чьому значно краще. Проте виникли деякі проблеми з 5-тим зображенням, результати не такі ідеальні, як отілося би. Якщо говорити в цілому, то важко дати одназначну оцінку алгоритму. Проблема в тому, що для данного алгоритму однакова первина обробка вибірки з МПЦД не

гарантує честність їх порівняння. Далі в цьому ж єксперемнті я порівняю ці два алгоритми при різних первиних обробках (Будуть узяті різні групи ОТ для зображень).

Але, поки що перейдемо до ВВС з дистанцією. Дана модифікація ВВС є повністю невдалою, на відміну від ВВС, що показав найкращий результат в другому експерименті, ВВС з дистанцію напроти показав найгірший. Хоча можна й зазначити, що розпізнані погано були тільки брат-сестра близнюки азіатського походження. Вірніше брат був віднесені до сестри, сама сестра була розпізнана добре. Але якщо брати до уваги, що інші алгоритми справилися з цією проблемою, то результати можна вважати на вдалими.

	s1	s2	s3	s4
s1	30	3	3	5
s2	0	41	0	0
s3	0	6	35	0
s4	0	22	0	19
s5	17	13	4	7
s6	3	34	3	1
s7	0	12	29	0
s8	0	26	2	13

Рисунок 3.32 – Відмова від слабких з дистанцією

Тепер для ВК з дистанцією збудуємо за допомогою коректування параметрів алгоритму BRISK та первинної обробки будувати інші ОТ при яких даний алгоритм буде показувати більш гарні результати, дивіться рисунок 3.32.

Це було зроблено, оскільки минулі параметри первинної обробки підбиралися саме під МПЦД. Для достовірності буде проведено порівняння МПЦД з ВК з дистанцією при параметрах побудови ОТ, що підбиралися під сам ВК з дистанцією.

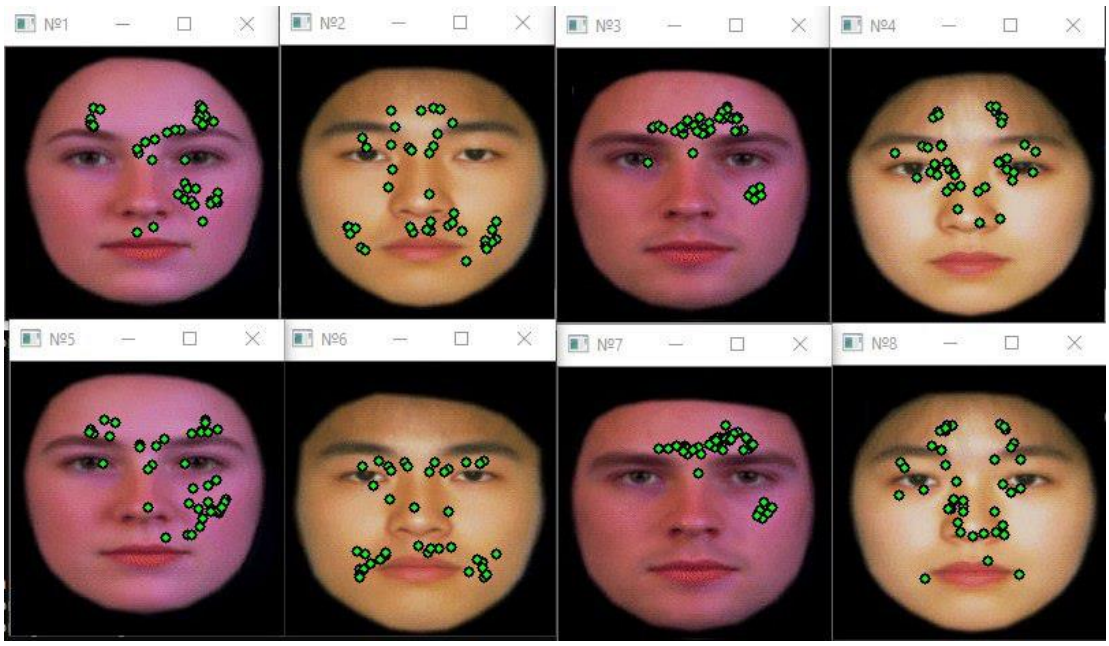


Рисунок 3.33 – Тестова вибірка зображень №2 з перебудованими ОТ

	s1	s2	s3	s4
s1	30	0	0	11
s2	6	29	1	5
s3	1	0	39	1
s4	1	13	0	27
s5	22	10	2	7
s6	12	23	0	6
s7	1	0	40	0
s8	0	14	0	27

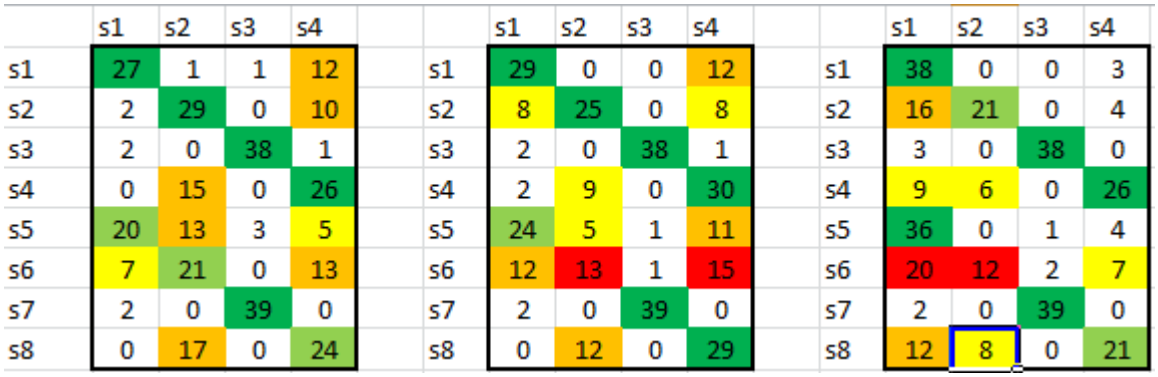
	s1	s2	s3	s4
s1	31	0	0	10
s2	9	25	0	7
s3	4	0	36	1
s4	4	9	0	28
s5	24	4	3	10
s6	15	12	2	12
s7	2	0	39	0
s8	3	12	0	26

Рисунок 3.34 – Порівняння ВК з дистанцією з МПЦД

На цей раз, результати стали протилежними, на рисунку 3.33 з правої частини результат класифікації ВК з дистанцією, а з лівої частини МПЦД. В цьому випадку для зображення номер 6-сть виникли проблеми у МПЦД, коли в минулі рази проблема була якраз у ВК з дистанцією.

Розглянувши рисунок 3.34 ми бачимо доволі віддзеркалений результат для ВВС та ВВС з дистанцією. Тільки метод вагових коефіцієнтів продовжив показувати доволі не втішні результати. Тому для фінального підсумку другого експерименту, будемо брати дані з різними параметрами для

знаходження ОТ й вже на основі двох тестувань обирати найбільш стійкий з алгоритмів.



	s1	s2	s3	s4
s1	27	1	1	12
s2	2	29	0	10
s3	2	0	38	1
s4	0	15	0	26
s5	20	13	3	5
s6	7	21	0	13
s7	2	0	39	0
s8	0	17	0	24

	s1	s2	s3	s4
s1	29	0	0	12
s2	8	25	0	8
s3	2	0	38	1
s4	2	9	0	30
s5	24	5	1	11
s6	12	13	1	15
s7	2	0	39	0
s8	0	12	0	29

	s1	s2	s3	s4
s1	38	0	0	3
s2	16	21	0	4
s3	3	0	38	0
s4	9	6	0	26
s5	36	0	1	4
s6	20	12	2	7
s7	2	0	39	0
s8	12	8	0	21

Рисунок 3.35 – Результати ВВС з дистанцією, ВВС та ВК

Таблиця 3.2 – Результати другого досліду

Метод	Загальна кількість вірних ОТ №1	Загальна кількість вірних ОТ №2	Стійкість №1	Стійкість №2	AVG ОТ	AVG стійкості	Загальна оцінка
МПЦД	219	221	23.5	15.5	220	19.5(195)	415 (4)
Вагових Коефіцієнтів (ВК)	221	231	21.5	20	226	20.75(207)	433 (3)
Відмова від сдабкого (ВВС)	225	222	27.5	18	223	22.75(227)	450 (2)
ВК з дистанцією	227	237	27.5	24	232	25.75(257)	489 (1)
ВВС з дистанцією	218	224	12.5	23	221	17.75(177)	398 (5)

Стійкість підраховується наступнім шляхом. За 25+ вірних ОТ +5 показник стійкості, за просто вірно класифіковане зображення +3. За кожний еталон який переміг вірний еталон. За кожний неправильний еталон який не

переміг, але до нього було віднесено 10+ точок віднімаємо 2 бали стійкості, та -0.5, за кожні невірно віднесені від 5 до 9 точок до одного невірного еталону.

Загальна оцінка вираховується сумою середньої кількості ОТ та загальною стійкістю помноженою на 10-ть. В даному експерименті найкращі результати показали ВВС та ВК з дистанцією набрали найбільшу кількість балів та показали найкращі результати. ВВС з дистанцією на відміну від першого дослідження показав найкращий результат як й при двох різних параметрах.

Останній експеримент проходив на тій же виборці облич, що й другий дослід. Але головною відміною було те, що ми для вхідних зображень також будуємо центр за МПЦД та порівнюємо його за допомогою наших алгоритмів. Ми для кожного алгоритму використовували 2 варіанти первинної обробки з попередніх дослідів. Оскільки кожний алгоритм отримує тільки один дескриптор ОТ, який є центральним дескриптором вхідного зображення, то ми будемо розглядати дані за якими класифікується наш алгоритм, оскільки для одних алгоритмів ми використовуємо відстань, а для інших їх вагу, порівнювати ми будемо тільки за кількістю правильно класифікованих зображень, та тим, наскільки однозначний результат. Результат не однозначний коли різниця між значеннями для вірного центру більше від значення, що набрав не вірний центр на відносно невелике значення.

Для МПЦД використовується відстань Хемінга, тому перемагає центр, до якого відстань найменша. МПЦД у другому варіанті вибірки ОТ, невірно класифікував 6-сте зображення, віднісши його до сестри близнючки. В першому варіанті присутні неякісно класифіковані з 5-го по 7-ме зображення, дивіться рисунок 3.36.

На відміну від МПЦД, ВК підраховує вагу, й перемагає той центр вага якого найбільша. Метод ВК не впорався в двох варіантах,не розпізнавши не

разу 6-сте, й 7-ме у при першому варіанті побудови вибірки ОТ, дивіться рисунок 3.37.

	s1	s2	s3	s4
s1	0	84	54	67
s2	84	0	86	57
s3	54	86	0	79
s4	67	57	79	0
s5	68	110	84	77
s6	70	38	68	41
s7	42	88	38	79
s8	122	78	118	67

	s1	s2	s3	s4
s1	0	85	64	63
s2	85	0	131	54
s3	64	131	0	115
s4	63	54	115	0
s5	28	91	62	71
s6	60	53	96	45
s7	64	127	22	133
s8	76	63	128	33

Рисунок 3.36 – Метод центрального дескриптора

	s1	s2	s3	s4
s1	393	271	358	316
s2	275	373	300	332
s3	323	268	423	300
s4	309	309	315	401
s5	309	233	311	294
s6	303	332	332	353
s7	341	265	380	296
s8	218	280	244	319

	s1	s2	s3	s4
s1	433	284	343	325
s2	307	386	230	342
s3	347	219	424	241
s4	346	324	258	409
s5	399	275	341	311
s6	349	327	291	356
s7	344	222	399	242
s8	325	313	236	372

Рисунок 3.37 – Метод вагових коефіцієнтів

Метод Відмови від слабких показав результат антологічний МПЦД, а переможець підраховується так само як у випадку з методом ВК, дивіться рисунок 3.38.

ВК та ВВС з дистанцією показали трохи кращі результати ніж їх попередники, але дати однозначну відповідь, що результати набагато кращі ми не можемо бо усі алгоритми окрім ВК не змогли класифікувати тільки в одному з випадків.

	s1	s2	s3	s4
s1	483	339	401	382
s2	331	457	337	392
s3	395	335	475	358
s4	375	399	353	474
s5	379	293	345	344
s6	371	427	369	344
s7	423	331	433	356
s8	263	355	273	362

	s1	s2	s3	s4
s1	492	353	391	390
s2	342	459	263	398
s3	394	273	486	282
s4	380	399	297	478
s5	450	339	389	374
s6	390	403	335	420
s7	390	279	459	286
s8	352	379	267	444

Рисунок 3.38 – Відмова від слабких

	s1	s2	s3	s4
s1	0	50	32	42
s2	58	0	61	34
s3	34	52	0	50
s4	42	31	54	0
s5	41	69	56	53
s6	45	20	45	24
s7	25	54	21	52
s8	87	46	89	41

	s1	s2	s3	s4
s1	0	50	40	41
s2	63	0	96	33
s3	42	83	0	84
s4	43	30	83	0
s5	16	55	41	49
s6	41	29	66	26
s7	44	81	12	83
s8	53	36	94	18

Рисунок 3.39 – Метод вагових коефіцієнтів з дистанцією

	s1	s2	s3	s4
s1	0	59	37	46
s2	76	0	69	41
s3	44	61	0	58
s4	54	29	61	0
s5	52	82	65	65
s6	54	12	53	28
s7	30	63	21	59
s8	110	51	101	56

	s1	s2	s3	s4
s1	0	53	47	44
s2	75	0	11	40
s3	49	93	0	98
s4	56	30	94	0
s5	21	60	48	52
s6	51	28	75	29
s7	51	90	13	96
s8	70	40	109	17

Рисунок 3.40 – Відмова від слабких з дистанцією

Для цих же даних було вирішено при різній кількості ОТ та при різних параметрах провести більше 50 дослідів у яких бралось би до уваги тільки кількість не розпізнаних зображень. Ми не брали до уваги наскільки чітке було розпізнавання, та наскільки гарні дані були узяті. Ми перевіряли, як працюють алгоритми при випадкових ситуаціях.

Таблиця 3.3 – Результати третього досліду

Метод	Кількість помилок При 31ОТ	Кількість помилок При 41ОТ	Кількість помилок При 51ОТ	Кількість помилок При 61ОТ	Загальна кількість помилок
МПЦД	8	7	6	2	23
Вагових Коефіцієнтів (ВК)	11	9	10	11	41
Відмова від сдабкого (ВВС)	8	7	5	3	23
ВК з дистанцією	8	5	4	3	19
ВВС з дистанцією	8	4	4	3	18

Результати були доволі цікаві, бо на різноманітних вибірках ОТ для зображень, найкращий результат показав ВВС з дистанцією. Який до цього показував не найкращі дані при більш детальному розгляді самого детектування. Проте ВК показав себе повністю недієздатним в даному досліді й для порівняння центра з центром недієвим варіантом. Зате ВК з дистанцією показав май же такий самий гарній результат як й ВВС з дистанцією зробивши лише на одну помилку більше. МПЦД та ВВС

показали однакові результати, що в даному випадку більше працює у сторону МПЦД, бо він реалізується швидше.

Підведемо підсумок проведених експериментів, але спочатку згадаємо результати при різних дослідах (табл. 3.4).

Таблиця 3.4 – Зведена таблиця результатів дослідження

Метод	Дослід 1	Дослід 2	Дослід 3	Приблизне загальне місце
МПЦД	2	4	3	3 (9)
Вагових Коефіцієнтів (ВК)	5	3	5	4 (13)
Відмова від сдабкого (ВВС)	1	2	4	2 (7)
ВК з дистанцією	3	1	2	1 (6)
ВВС з дистанцією	4	5	1	5 (10)

Найбільш дієвим як й при порівняні центра з центром так й центра з дескрипторами вхідного зображення найркащі результати показ себе ВК з дистанцією. Це обусловлюється тим, що ми придаємо більше значеня тим бітам(значенням) центра, які при його побудові були більш чіткі (мали перевагу над своїм протилежним бітом). А нестійкі значення, навпаки, впливали менше.

Доволі гарні результати показав й метод ВВС, давши слабину лише в третьому досліді. Він при роботі з різноматними даними не показав значної переваги над МПЦД при порівнянні центрів. Це можно обумовити тим, що він погано працює з центрами, які мають багато нестійких значень, бо повністю їх ігнорує. Проте при роботі з центрами, які мають невелику(або

помірну) кількість нестійких значень, алгоритм працює набагато краще від МПЦД. Варто зазначит, що проблему можно нивілювати, якщо при порівнянні замість вагів центру еталону брати ваги центру вхідного зображення.

Метод ВВС з дистанцією показав гарні результати лише в третьому досліді. Можливо при порівнянні замість вагів центру еталону брати ваги центру вхідного зображення, результати би були ще гарніше. Хоча він й показав не найкращі результати в наших дослідах, відкидати його не варто. Останій дослід показав, що можливо були підібрані не найкращі обставини для даного алгоритму.

Алгоритм ВК є повністю невдалим, й показав провальні результати у всіх дослідах. Хоча теоретично даний алгоритм мав би працювати краще всіх, али на практиці він показав найгірші результати.

ВИСНОВКИ

У результаті виконання атестаційної роботи проведено дослідницьку роботу, під час якої розроблено алгоритми покращення детектування зображень за допомогою бінарних центральних дескрипторів. Для детектування ОТ використовувався один з новітніх алгоритмів BRISK. Проведено порівняльний аналіз розроблених на різних вибірках зображень та при різних умовах оброблення.

При практичній побудові правил класифікації візуальних об’єктів із застосуванням детекторів ОТ треба звернути увагу на ефективний вибір та поєднання можливостей як детекторів ОТ, так і апарату для оброблення їх значень для досліджуваних баз зображень.

Серед досліджених варіантів найкращі результати класифікації зображень показав алгоритм «Вагових коефіцієнтів з дистанцією». Це обусловлюється тим, що ми придаємо більше значеня тим бітам (значенням) центра, які при його побудові були більш чіткі (мали перевагу над своїм протилежним бітом).

Методи «Відмови від слабких» та «Відмови від слабких з дистанцією» також є доволі перспективними модифікаціями. Але мають деякі проблеми з вибірками ОТ, де центр має множину настійких значень.

Метод побудови центрального дескриптора хоч й був перевершений у якості класифікування, проте всі аналізовані алгоритми програють йому у швидкодії оброблення даних. Також варто зазначити, що даний метод усе одно класифікує зображення доволі якісно й може використовуватися у ситуаціях, коли потрібна гарна швидкодія та якісне класифікування.

Єдиний з розглянутих алгоритмів, який можна назвати невдалим, – це метод вагових коефіцієнтів, який на перший взгяд мав бути найбільш ефективним, проте практика показала протележні результати.

Наукова новизна дослідження полягає у синтезі нового продуктивного методу класифікації зображень з використанням апарату бітового аналізу та

оброблення даних у просторі дескрипторів ключових точок зображення з використанням вагових коефіцієнтів, що покращують якість класифікації. Головною перевагою над існуючими алгоритмами є швидкодія. Зведення зображення до масиву(дескриптору) з 512 значень, кожне значення може бути або 0, або 1. До центрального дескриптору додається ще один масив з вагами, що дозволяє зменшити фактичний об’їм інформації для класифікації, а по друге, пришвидшує процес класифікація у стільки разів, скільки особливих точок має зображення. Порівняння центра з центром пришвидшує роботу ще у стільки же разів й при тому якість класифікації майже не зменшується, а в деяких випадках навіть покращується.

Практична значущість роботи – отримання практичних програмних моделей для застосування та оцінювання ефективності методів класифікації з бінарним поданням даних та приведенням їх у центри з підтвердженням їх результативності на прикладних базах зображень.

Зважаючи на бінарний вид дескрипторів BRISK, є можливість для побудови результативних правил класифікації скористатися механізмами дерев рішення чи хешування.

Основною проблемою подальшої модернізації е простота алгоритмів, що дарує велику швидкодію. А нагромадження існуючих алгоритмів ваговими коефіцієнтами робить роботу алгоритмів більш довгою.

Результати були апробовані на 22-ому Міжнародному молодіжному форуму [23], у 1-му та 2-гому випусках наукового журналу ‘Системи управління, навігації та зв'язку [16, 17], у збірці наукових праць міжнародної наукової конференції [24]. Також було прийнято участь у всеукраїнському науковому конкурсі наукових робіт за напрямком “Інформатика та кібернетика”, що проходив у Вінниці 24-25 квітня 2019 року, на якому було отримано перемогу третього ступеня, диплом підтверджуючий це знаходиться у додатках.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Duda, R.O., & Hart, P.E., & Stork, D.G. (2000) Pattern classification. Wiley.

2. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G.R. (2011, November). ORB: An efficient alternative to SIFT or SURF. InICCV(Vol. 11, No. 1, p. 2).

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

4. opencv.org. (2019). Available at: https://docs.opencv.org/3.4.1/ [Accessed 5 Nov. 2019].

5. Rad, A.A. & Faez, K. & Qaragozlou, N.(2003). Fast Circle Detection Using Gradient Pair Vectors. Sydney, Australia: Proc. VIIth Digital Image Computing, 879-887.

6. Shapiro, L. (2001). Computer vision, Prentice. Prentice Hall.

7. Leutenegger, S., & Chli, M. & Siegwart, R.Y. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. Computer Vision (ICCV), 2548 – 2555.

8. Анисимов, Б.В. & Курганов, В.Д & Злобин, В.К. (1983). Распознавание и цифровая обработка изображений. Высшая школа.

9. Вапник, В. Н. (1979). Восстановление зависимостей по эмпирическим данным. М.: Наука.

10. Вапник, В. Н. & Червоненкис, А. Я. (1974). Теория распознавания образов. М.: Наука.

11. Глумов, Н.И. & Коломиец, Э.И. & Сергеев, В.В. (1993). Информационная технология распознавания объектов на изображении в режиме скользящего окна. Научное приборостроение, 72-88.

12. Глумов, Н.И. (1994). Яркостная нормализация цифрового изображения в скользящем окне. Деп. в ВИНИТИ. Самара: Самарский государственный аэрокосмический университет, № 1881-В94.

13. Гонсалес, Р. & Вудс, Р. (2005). Цифровая обработка изображений. М.: Техносфера.

14. Гороховатский, В. А., Путятин, Е. П., & Столяров, В. С. (2017). Исследование результативности структурных методов классификации изображений с применением кластерной модели данных.

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

16. Гороховатський, В. О., Гороховатский, В. А., Солодченко, К. Г., & Солодченко, К. Г. (2018). Застосування апарату аналізу та оброблення бітових даних у методах класифікації зображень за множиною ключових точок.

17. Гороховатський, В. О., Гороховатский, В. А., Пупченко, Д. В., Пупченко, Д. В., Солодченко, К. Г., & Солодченко, К. Г. (2018). Аналіз властивостей, характеристик тарезультатів застосування новітніх детекторів для визначення особливих точок зображення.

18. Дуда, Р. & Харт, П. Харт (1976). Распознавание образов и анализ сцен. М.: Мир.

19. Сойфера, В.А. (2003). Методы компьютерной обработки изображений. М.: ФИЗМАТЛИТ.

20. Осовский, С. (2002). Нейронные сети для обработки информации. М.: Финансы и статистика.

21. Прэтт, У.К. (1982). Цифровая обработка изображений. М.: Мир.

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

23. Солодченко, К.Г. & Гороховатський, В.О. (2018). Аналіз бітової інформації у системах структурної класифікації зображень. XXII Міжнародний молодіжний форум Радіоелектроніка та молодь у XXI столітті. Зб. Матеріалів форуму. Т.7. – Харків: ХНУРЕ, 13–14.

24. Солодченко, К.Г. & Гороховатський, В.О. & Путятін, Є.П. (2018). Дослідження структурних методів класифікації зображень з використанням

детектора BRISK. Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту (ISDMCI'2018): збірка наук. праць міжн. наук. конф., с. Залізний Порт, 21–27 травня 2018. – Херсон: ФОП Вишемирський В.С., 279–281.

25. Солонина, А.И. & Улахович, Д.А. & Яковлев, Л.А (2001). Алгоритмы и процессоры цифровой обработки сигналов. СПб.: БХВ-Петербург.

26. machinelearning.ru (2019). Available at: http://www.machinelearning.ru/wiki/index.php?title=Классификация [Accessed 5 Nov. 2019].

27. Страуструп, Б. (1993). Язык программирования С++. Часть 1. К.: ДиаСофт.

28. Фор, А. (1989). Восприятие и распознавание образов. М.: Машиностроение.

29. Форсайт, Д., & Понс, Ж. (2004). Компьютерное зрение. Современный подход (p. 928). М.: ИД Вильямс.

30. Шапиро, Л. (2006). Стокман Дж. Компьютерное зрение. М.: Бином, 120-124.

31. Gorokhovatsky, V.O. and Gadetska, S.V., (2019). Determination of Relevance of Visual Object Images by Application of Statistical Analysis of Regarding Fragment Representation of their Descriptions, Telecommunications and Radio Engineering, 78 (3), pp. 211–220. – doi: 10.1615/TelecomRadEng.v78.i3.20.

32. Kortunov ,V.I., Molchanov, A. O., Gorokhovatsky V.O., (2019). Method of Optical Flow Estimation Based on Image Block Weighting, Telecommunications and Radio Engineering, 78 (9), pp. 783-792. – doi: 10.1615/TelecomRadEng.v78.i9.40.

33. Gorokhovatskyi, V., Vasylchenko, A., Manko, K., & Ponomarenko, R. (2018). Дослідження модифікацій методу встановлення релевантності зображень об'єктів за описами у вигляді множини дескрипторів ключових

точок. Системи управління, навігації та зв’язку. Збірник наукових праць, 5(51), 74-78.

34. Gorokhovatskyi V., Gadetska S., Ponomarenko R. (2019) Recognition of Visual Objects Based on Statistical Distributions for Blocks of Structural Description of Image. Lecture Notes in Computational Intelligence and Decision Making. Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision Making and Problems of Computational Intelligence” (ISDMCI'2019), Ukraine, May 21–25, 2019, pp. 501-512. – Available online: https://rd.springer.com/chapter/10.1007/978-3-030-26474-1_35

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

36. Гороховатский, В.А. & Полякова, Т.В. (2018) Применение пространственных структур признаков для классификации изображений в компьютерном зрении (монография). Харьков: ФОП Панов А.Н.