Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Програмне забезпечення комп'ютерного зору для розпізнавання множини об'єктів на зображенні

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-4
Кирил МАКЄЄВ
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Наталія БОЛОГОВА
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Макєєву Кирилу Костянтиновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмне забезпечення комп'ютерного зору для розпізнавання множини об'єктів на зображенні

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) мову програмування : Java; 2) алгоритм: Віоли-Джонса: 3) інтегроване середовище розробки: IntelliJ IDEA;4) бібліотека: OpenCV.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблеми та огляд існуючих рішень;
2) аналіз існуючих модифікацій алгоритму Віоли-Джонса ;
3) аналіз вимог до створення програми;
4) розробка рішення реалізації застосунку;
5) програмна реалізація застосунку;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 16 слайдів презентації

6. Консультанти розділів роботи (заповнюється 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

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	27.05.25-30.05.25	
2	Аналіз модифікацій алгоритму Віоли-Джонса	31.05.25-03.06.25	
3	Розробка рішення реалізації алгоритму	04.06.25-07.06.25	
4	Програмна реалізація програми	08.06.25-09.06.25	
5	Оформлення матеріалів кваліфікаційної	10.06.25-11.06.25	
	роботи		
6	Подання кваліфікаційної роботи керівникові та	12.06.25-13.06.25	
	її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання "26" травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Наталія БОЛОГОВА
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 57с., 24 рис., 1 табл., 1 дод., 10 джерел.

АЛГОРИТМ ВІОЛИ-ДЖОНСА, SIFT, ДЕТЕКТОР ХАРРІСА, БІНАРИЗАЦІЯ ЗОБРАЖЕНЬ, INTELLIJ IDEA, JAVA, ЗАСТОСУНОК, БІБЛІОТЕКА OPENCV.

Метою кваліфікаційної роботи є створення програмного продукту для комп’ютерного зору, здатного виявляти різнотипні об’єкти на зображенні з використанням алгоритму Віоли-Джонса.

У ході виконання кваліфікаційної роботи було поставлено такі завдання:

- проаналізувати принципи роботи алгоритму Віоли-Джонса та його особливості в контексті комп’ютерного зору;
- реалізувати програму, що демонструє локалізацію кількох об’єктів із використанням механізмів багатопоточності;
- провести порівняльне тестування реалізації алгоритму в однопотоковому та багатопотоковому виконанні.

ABSTRACT

Bachelor’s thesis: 57 pages, 24 figures, 1tables, 1 appendices, 10 sources.

VIOLA – JONES ALGORITHM, SIFT, HARRIS DETECTOR, IMAGE BINARIZATION, INTELLIJ IDEA, JAVA, APPLICATION, OPENCV LIBRARY.

The major goal of this thesis is the qualification work is to create a software product for computer vision capable of detecting different types of objects in an image using the Viola-Jones algorithm.

In order to during the qualification work, the following tasks were set:

- analyze the principles of the Viola-Jones algorithm and its features in the context of computer vision;
- implement a program that demonstrates the localization of several objects using multithreading mechanisms;
- conduct comparative testing of the algorithm implementation in single-threaded and multithreaded execution.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 7
ВСТУП 8
1 АНАЛІЗ СТАНУ ПИТАННЯ ЗАСТОСУВАННЯ АЛГОРИТМУ ВІОЛИ-ДЖОНСА 10
1.1 Дослідження предметної галузі 10
1.2 Опис роботи алгоритму Віоли-Джонса 11
1.3 Розгляд модифікацій алгоритму Віоли-Джонса 16
1.4 Постановка задачі 22
2 ПРОЕКТУВАННЯ ЗАСТОСУНКІВ З РОЗПІЗНАВАННЯ МНОЖИННИХ ОБ'ЄКТІВ НА ЗОБРАЖЕНІ 24
2.1 Концептуальна модель 24
2.2 Логічна модель 26
2.3 Фізична модель 32
3 ЗПРОГРАМНА РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ 34
3.1 Програмна реалізація програми 34
3.2 Тестування програми 41
ВИСНОВКИ 46
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 48
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 49

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЗКЗ – застосунок комп’ютерного зору (англ., Computer Vision Application)

app – мобільний застосунок (англ., application)

IDEF0 – методологія функціонального моделювання (англ., Integration Definition for Function Modeling)

RASW – метод розпізнавання об'єктів (англ., Recognition Algorithm based on Sliding Window)

RGB – формат кольорового зображення (англ., Red, Green, Blue)

SIFT – алгоритм інваріантного масштабного перетворення ознак (англ., Scale-Invariant Feature Transform)

SURF – прискорене стійке виявлення ознак (англ., Speeded-Up Robust Features)

WDS – бездротова розподільча система (англ., Wireless Distribution System)

ВСТУП

З кожним роком обсяг інформації у світі стрімко зростає. Технології постійно еволюціонують, і з'являються нові підходи до обробки та систематизації даних. Сьогодні навіть для виконання базових завдань, таких як аналіз дорожнього трафіку або виявлення порушень правил, використовуються спеціалізовані алгоритми, які замінюють ручну працю людини. Однак навіть для великих корпорацій обробка таких даних залишається непростим завданням. Сотні фахівців працюють над тим, щоб навчити комп’ютер відтворювати звичні для людини дії в реальному житті, і вже сьогодні можна спостерігати вагомі результати цієї роботи.

Актуальність обраної теми пояснюється тим, що розпізнавання об’єктів на зображеннях є одним із найпоширеніших напрямів у сфері штучного інтелекту. Ця технологія має широкий спектр застосувань: від автоматичного зчитування номерних знаків транспортних засобів до аналізу міміки людини для визначення її емоційного стану.

У кваліфікаційній роботі розглядається алгоритм розпізнавання об'єктів, відомий як метод Віоли-Джонса. Проаналізовано його ключові переваги, обмеження та сфери застосування. Алгоритм буде використано для виявлення об’єктів на цифрових зображеннях, і додатково оцінено його ефективність у межах конкретних практичних завдань. Слід зазначити, що існує велика кількість альтернативних підходів до локалізації об'єктів, таких як SIFT, детектор Харріса, методи бінаризації зображень, а також алгоритми математичної морфології. Кожен із них має як сильні сторони, так і певні недоліки.

Завдання об'єктного розпізнавання може мати велике значення в екстрених ситуаціях. Наприклад, у випадку аварії та виникнення пожежі система комп’ютерного зору здатна автоматично повідомити рятувальні

служби для оперативного реагування [1].

Методи комп’ютерного зору зазвичай створюються для вирішення окремих прикладних задач, зосереджених на виділенні характерних ознак об’єктів. Універсальних рішень у цій галузі не існує: навіть при спробі зосередитись на одній конкретній задачі та звести похибку до мінімуму, завжди залишатимуться певні обмеження. Однією з головних проблем є значна трудомісткість навчання моделей для виявлення конкретних об'єктів, що потребує великих часових витрат, хоча і підвищує точність. Чим більше ресурсів витрачено на тренування системи, тим вища ймовірність точного розпізнавання, однак повністю виключити хибні спрацьовування не вдається – можна лише зменшити їхню кількість.

Наукова новизна даної роботи полягає у застосуванні алгоритму Віоли-Джонса для розпізнавання кількох різнотипних об’єктів на одному зображенні.

Об'єкт дослідження – система комп’ютерного зору, що виконує розпізнавання кількох об’єктів на зображенні. Предмет дослідження – застосування алгоритму Віоли-Джонса для вирішення задачі багатоб’єктного розпізнавання.

Мета дослідження – розробити програму на основі комп’ютерного зору, яка здатна виявляти об’єкти різних типів на одному кадрі з використанням методу Віоли-Джонса.

Для досягнення мети необхідно реалізувати такі завдання:

- дослідити принципи функціонування алгоритму Віоли-Джонса;
- створити програмний прототип для демонстрації роботи алгоритму під час локалізації об'єктів із використанням технологій паралельних обчислень;
- провести тестування роботи алгоритму у двох варіантах – на однопоточному та багатопоточному програмному забезпеченні.

1 АНАЛІЗ СТАНУ ПИТАННЯ ЗАСТОСУВАННЯ АЛГОРИТМУ ВІОЛИ-ДЖОНСА

1.1 Дослідження предметної галузі

Нині є безліч різних підходів до спрощення життя людства. Але не всі вони досить ефективні на перший погляд. Уявимо таку ситуацію, на дорозі було скоєно транспортне порушення, камера зафіксувала все, що відбувається, і відправила звіт до відділу з обробки фото-реєстраційного матеріалу для винесення вердикту – чи винна людина і чи потрібно виписати йому штраф чи ні [2]. Це робиться для того, що штучний інтелект має поріг помилки набагато вищий, ніж людський погляд на речі. Якщо програма сама виносила висновок і становила санкції на правопорушення, це давало б можливість гнучко розподіляти людські ресурси і направити їх на важливіші речі.

Перед автором поставлено завдання – розробити програму комп'ютерного зору для розпізнавання безлічі об'єктів на зображенні.

Існує безліч різноманітних алгоритмів з розпізнавання об'єктів на зображенні, наприклад: Eigenface заснований на поданні зображень як суму базисних компонентів, ISODATA використовує кластеризацію при заданій кількості груп, ERDAS Imagine максимізуючий спектральний відгук цільового вектора і мінімізуючий вплив фону, заснованого на фоні тощо [3].

На розгляд буде взято алгоритм Віоли-Джонса розроблений Полом Віолою та Майклом Джонсоном у 2001 році. Спочатку він був розроблений для розпізнавання людських осіб, але також знайшов своє застосування для інших об'єктів. Невід'ємним плюсом даного алгоритму є те, що він здатний досить швидко працювати на слабкі пристрої. Через це навіть у роки він показував досить непогані результати з розпізнавання об'єктів [4].

В результаті додаток отримає широку сферу застосування завдяки її

можливостям, якщо взяти за основу алгоритм Віоли-Джонса.

1.2 Опис роботи алгоритму Віоли-Джонса

Точність аналізу даних зростає з кожним днем, як і швидкість їх обробки, а поріг помилки йти до нульового значення. Ось і своє застосування у цій сфері знайшов алгоритм Віоли-Джонса. Застосунок, що розробляється, буде засновано на даному алгоритмі. Раніше було сказано, що алгоритм відмінно показав себе на апаратурі 2001 року, оскільки він включає досить негроміздкі обчислення, завдяки чому можна швидко отримати бажаний результат [3].

Алгоритм дозволяє знаходити об'єкти на зображеннях з високою точністю та низькою кількістю помилкових спрацьовувань.

Алгоритм Віоли-Джонса працює наступним чином: на вхід надходить зображення, представлене у вигляді двомірної матриці, елементи якої – це пікселі з яскравістю, розташованої в діапазоні від 0 до 255. Якщо зображення на вході є чорно-білим, то достатньо однієї двовимірної матриці, якщо зображення має кольоровий, то таке зображення формат RGB [4].

Одним із головних принципів методу є перетворення чорно-білого зображення на інтегральне уявлення (рисунок 1.1).

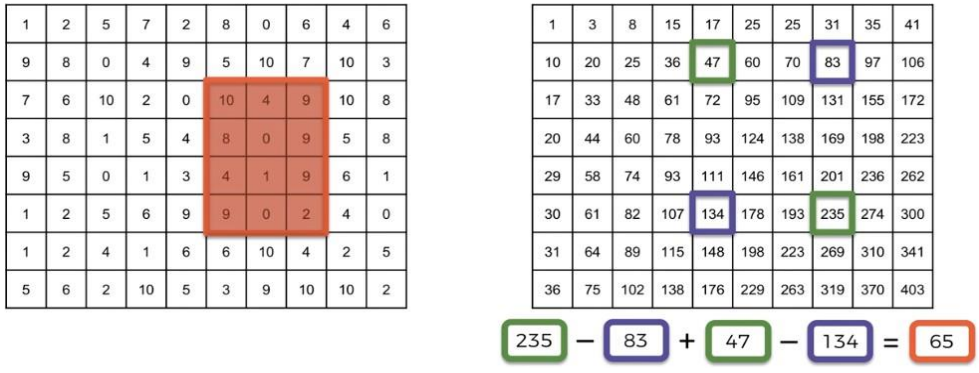

Рисунок 1.1 – Приклад обчислення інтегрального зображення

Дане уявлення використовується найчастіше й інших алгоритмах, наприклад, вейвлет-перетвореннях, SURF та багатьох інших розібраних алгоритмах [5]. У кожному елементі інтегральної матриці зберігається сума інтенсивностей всіх пікселів, що знаходяться ліворуч і вище за цей елемент. Ця дія дозволяє порахувати сумарну яскравість довільного прямокутника, незалежно від його розмірів на зображенні, що добре підходить для застосування ознак Хаара. Інтегральне уявлення за розміром збігається із вихідним. Якщо значень матриці немає, вони вважаються рівними нулю. Підрахунки яскравості у довільному прямокутнику проводиться за формулою:

$$II(x,y) = \sum_{i=0,j=0}^{i\leq x,\leq y} I(i,j), \tag{1.1}$$

де I(i,j) – це яскравість пікселя вихідного зображення.

Елемент матриці II(x,y) є сумою пікселів у прямокутній області від елемента з індексами (0,0) до елемента (x,y). Час підрахунку такої матриці – лінійно. Для того, щоб обчислити суму яскравості пікселів, що потрапляють в область прямокутника, достатньо всього чотири операції щодо звернення до масиву та три арифметичні операції [6].

Розглянемо прямокутник ABCD (рисунок 1.2).

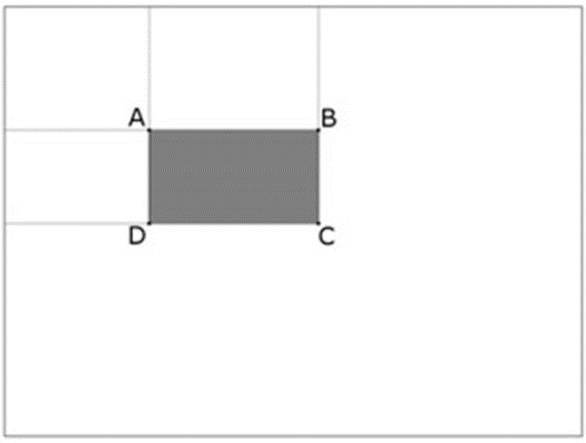

Рисунок 1.2 – Підрахунок сумарної яскравості пікселів, що потрапляє у прямокутну область

Підсумувати значення яскравості пікселів, потрапили в прямокутну область можна за формулою [7]:

$$\sum ABCD = II(A) + II(C) - II(B) - II(D), \quad (1.2)$$

де II(A), II(B), II(C), II(D) – суміжні прямокутники щодо виділення.

Робота алгоритму полягає у пошуку деяких ознак на зображенні. примітиви, тобто кожен ознака є двійковою маскою – чорно-біле зображення (рисунок 1.3).

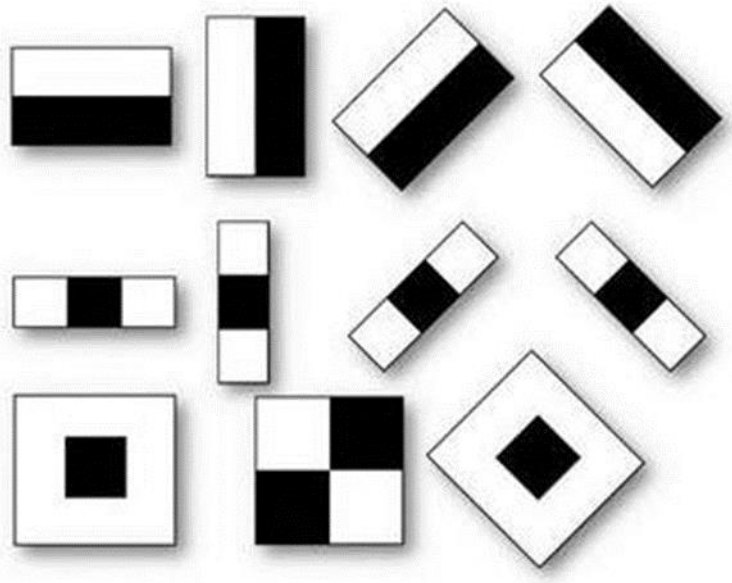

Рисунок 1.3 – Примітиви Хаара

За допомогою даних примітивів виноситися висновок чи є в цій галузі об'єкт, що розпізнається, або цю область слід прибрати з розгляду [6]. Вердикт виносити завдяки обчислюваній ознаки за формулою:

$$F = X - Y, \quad (1.3)$$

де X - сума яскравостей пікселів світлою частиною ознаки, що

закривається;

Y – сума яскравостей пікселів під чорною областю ознаки.

Дане значення ознаки порівнюється зі значенням в каскаді-класифікаторі, навченому на розпізнавання певного об'єкта, якщо це значення більше певного порога, отже, у цій галузі є об'єкт, що розпізнається, в іншому випадку область відкидається з розгляду.

Навчання каскадів-класифікаторів здійснюється за допомогою алгоритму AdaBoost [6]. Це алгоритм машинного навчання, розроблений такі як Йоава Фройнд і Роберт Шапір. Відмінною рисою є те, що під час навчання будує цілу конструкцію з базових алгоритмів навчання з метою покращення ефективності. AdaBoost є досить адаптивним алгоритмом, оскільки кожен наступний класифікатор будується по об'єктах, які попередні класифікатори не змогли класифікувати правильно. Кожному обраному на навчання об'єкту задаються деякі числові характеристики, звані терезами. Ці ваги характеризують важливість алгоритму. Кожну ітерацію, коли класифікатор не зміг локалізувати об'єкт, дані ваги збільшуються, що змушують класифікатор загострити увагу.

Розглянемо переваги алгоритму AdaBoost:

- здатність до узагальнення. Алгоритм будує композиції, які дають приріст до точності навчання, перевищуючи за якістю базові алгоритми, які закладені у його основі. Узагальнення можна збільшувати, доповнюючи алгоритм AdaBoost новими алгоритмами, що базуються;
- обчислювальні ресурси, що споживаються алгоритмом AdaBoost, не значні. Час побудови композиції залежить від часу навчання основних алгоритмів;
- простота реалізації;
- можливість обробити об'єкти, які є «шумом».

Так само, як і всі придумані алгоритми людиною, у AdaBoost є свої недоліки:

- алгоритм перенавчає каскади-класифікаторів при значних рівнях шуму у вихідних даних. Є експоненційна функція втрат, яка призначає високі ваги важким об'єктам, на яких осічку дають більшість базових алгоритмів. Нерідко трапляється ситуація, коли ці трудомісткі об'єкти є шумом і AdaBoost починає навчатися на нього, що веде до перенавчання. Це вирішується застосування м'якших функцій втрат;

- алгоритм має отримувати на вхід велику кількість вибірок;

- метод послідовного додавання може спричинити створення неоптимального набору базових алгоритмів. Вирішити цю проблему можна з міццю відкату до раніше побудованих алгоритмів;

- AdaBoost іноді будує громіздкі композиції з базових алгоритмів навчання. Через це підвищується час навчання каскаду-класифікатора, також потрібно більше пам'яті для зберігання побудованих композицій.

На рисунку 1.4 наведено приклад роботи методу Віоли-Джонса з навченим каскадом-класифікатором для розпізнавання осіб.

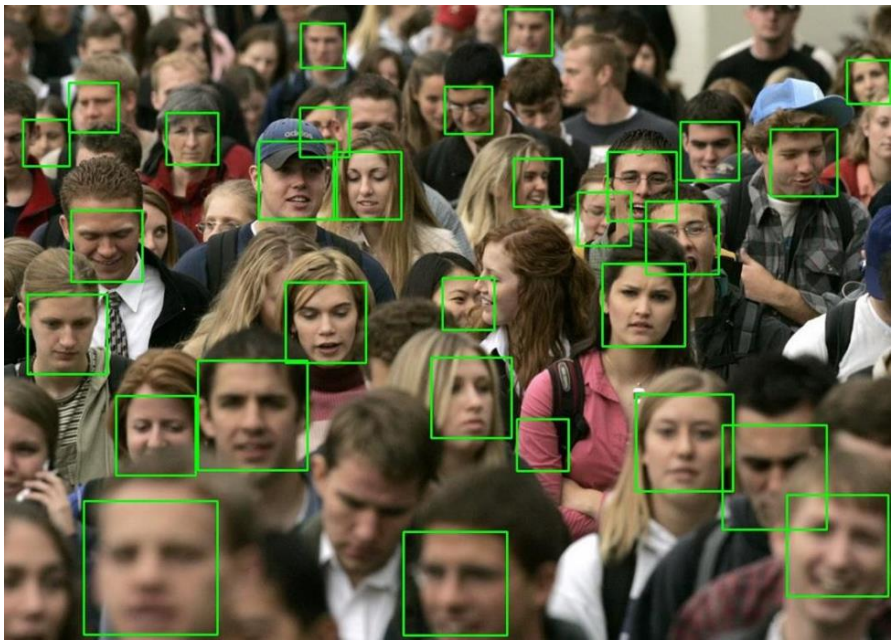

Рисунок 1.4 – Приклад роботи методу Віоли-Джонса

Як видно на рисунку, алгоритм не може розпізнати деяких людей у

натовпі, оскільки вони загороджені іншими людьми. Але він добре показує себе при розмитті зображення. Це з тим, що каскад був добре навчений на об'ємній вибірці. З усього складового можна виявити деякі переваги та недоліки алгоритму Віоли-Джонса.

Переваги:

- досить швидка робота алгоритму;
- здатність алгоритму розпізнавати будь-які об'єкти, слід лише навчити каскад-класифікатора;
- мале споживання обчислювальних потужностей.
- висока точність розпізнавання. Чим більша вибірка, тим краще точність.

Недоліки:

- алгоритм погано реагує на сторонні речі, на об'єкт розпізнавання. Наприклад, на людині одягнені сонце захисні окуляри;
- погано позначається розпізнавання зміни ракурсу на об'єкт.
- чутливість до освітлення;
- тривалий час навчання каскадів-класифікаторів.

Деякі недоліки алгоритму наважуються з допомогою його модифікацій, а також вони збільшують продуктивність.

1.3 Розгляд модифікацій алгоритму Віоли-Джонса

Алгоритми створені на вирішення прикладних завдань. Кожен із них відрізняється своїми особливостями і в кожного є свої недоліки, от і метод Віоли-Джонса не став винятком. Але людство не стоїть на місці, вони вигадали, як можна модернізувати таке привабливе рішення.

Існують кілька способів модернізації алгоритму Віоли-Джонса:

- передобробка вхідного зображення;
- зміна алгоритму роботи Віоли-Джонса

До обробки вхідного зображення належать такі методи, як згладжуючий фільтр Гаусса, пошук кордонів на зображенні з використанням оператора Преввіту, Собеля або Лапласа і метод Канни [7].

Зміна алгоритму роботи Віоли-Джонса включає метод RASW.

Познайомимося з деякими способами модифікацій методу Віоли-Джонса, представленими вище.

Обсяг роботи алгоритму Віоли-Джонса був із дозволом вхідного зображення. Якщо розмір зображення збільшиться, то збільшиться кількість обчислень, а це не дуже добре при роботі методу в реальному часі. Для цього був придуманий метод RASW – Runtime Adaptive Sliding Window. Його ще просто називають вікно, що сканує. Де розмір сканування визначається деяким числом. Крок сканування сильно впливає на точність об'єкта, що розпізнається, а також характеризує пропускну здатність. Якщо задати занадто великий крок, то пропускна здатність збільшиться, але в свою чергу, точність розпізнавання зменшиться, тому що не всі об'єкти можуть потрапити у вікно, що розпізнається, і будуть пропущені [8].

Для переміщень скануючого вікна слід знайти особливу закономірність, щоб вікно рухалося швидше в областях, які не містять об'єкта, що розпізнається, і повільніше в безпосередній близькості від об'єкта. Виходячи з цього, можна було підвищити швидкість сканування без погіршення якості. Також особливим плюсом було б те, що шанс помилки класифікатора знизилася виявити об'єкт, що не шукав, в іншому місці, де його взагалі немає.

У стандартній реалізації способу Віоли-Джонса крок задається невеликим значенням, наприклад один або два. Оскільки високе значення призводить до погіршення якості розпізнавання.

Розробники алгоритму RASW провели аналіз та виявили взаємозв'язок

між наявністю об'єкта в області зображення і номером класифікатора каскаду, у якому це вікно вважається відкинутим. В областях, які містять лише фон – вікно відкидається раніше, ніж перебуваючи в безпосередній близькості

до об'єкта, що розпізнається. В результаті дійшли такого висновку, що чим ближче об'єкт, що шукається, тим вище ступінь виходу і навпаки.

Алгоритм роботи методу RASW буде представлений рисунку 1.5.

Require: scaled input image X, classifier cascade S
Ensure: vector V of detected faces (bounding boxes)

1: **for** $y \leftarrow 0$ to $X.height - subwindow.height$ **do**
2: $\Delta_x \leftarrow 1$
3: **for** $x \leftarrow 0$ to $X.width - subwindow.width$ **do**
4: $\Delta_x \leftarrow \Delta_x - 1$
5: $\Delta_y[x] \leftarrow \Delta_y[x] - 1$
6: **if** $\Delta_x = 0$ AND $\Delta_y[x] = 0$ **then**
7: *exit-stage* $\leftarrow S(x, y)$
8: **if** *exit-stage* $= n$ **then**
9: $R \leftarrow (x, y, width, height)$
10: push R into V
11: **end if**
12: **if** *exit-stage* $< \Delta_{x,t1}$ **then**
13: $\Delta_x = \Delta_{x,max}$
14: **else if** $\Delta_{x,t1} \leq$ *exit-stage* $< \Delta_{x,t2}$ **then**
15: $\Delta_x = \Delta_{x,nom}$
16: **else**
17: $\Delta_x = \Delta_{x,min}$
18: **end if**
19: **else if** $\Delta_x = 0$ **then**
20: $\Delta_x \leftarrow \Delta_{x,min}$
21: **else if** $\Delta_y[x] = 0$ **then**
22: $\Delta_y[x] \leftarrow \Delta_{y,min}$
23: **end if**
24: **end for**
25: **end for**

Рисунок 1.5 – Алгоритм методу RASW

Кожен зсув скануючого вікна запускає каскад-класифікатор, який повертає ступінь виходу (рядок 7). Після цього йде перевірка умови, що, якщо ступінь виходу дорівнює n – числу класифікаторів у каскаді, отже об'єкт, що розпізнається, знайдений (рядок 8) і координати поточного вікна містяться у вектор V (рядок 10). Крок вікна може набувати різних значень по осі абсцис і ординат, залежно від ступеня виходу попереднього положення вікна. Для того,

щоб зсув вікна був правильним, слід зберігати інформацію про рівень виходу попереднього положення вікна. На кожній ітерації алгоритму крок по осі абсцис і осі ординат зменшуються на одиницю, коли обидві змінні дорівнюватимуть нульовому значенню, то вікно буде оброблено каскадом-класифікаторів [9]. Якщо тільки одне з цих значень дорівнюватиме нулю, то це значення приймається за мінімальну величину кроку (рядки 20 і 22) [10].

Перевагою даного методу є прискорення алгоритму Віоли-Джонса без втрат якості розпізнавання, а саме швидке переміщення вікна, що розпізнається, по областях, де не міститься об'єкт, що розпізнається, і повільне переміщення в його близькості.

Недоліки методу:

- складність реалізації;
- вплив шуму область поліпшення даного методу. При великій кількості шуму швидкість роботи буде зменшуватися.

Наступною модифікацією буде особливо відомий алгоритм, розроблений Джоном Канні на зорі комп'ютерного зору в 1986 році. Він називається «Метод виділення кордонів Канни».

Алгоритм складається з п'яти кроків:

- розмиття зображення для видалення шуму (згладжування);
- пошук градієнтів. Виявлення максимальних значень градієнтів на зображенні та відтворення меж;
- пригнічення не максимальних градієнтів. Максимуми, що є локальними, відзначаються як градієнти;
- фільтрація кордонів.

Створюючи цей алгоритм за основу, було взято такі критерії:

- стійкість до шуму;
- якісна локалізація меж об'єктів;
- одна межа об'єкта повинна викликати лише один відгук алгоритму.

Завдяки даним критеріям було визначено функцію вартості помилок,

внаслідок мінімізації значень якої було знайдено лінійний оператор, що робить згортки зображення.

Виходячи зі сказаного вище алгоритм меж Канні працює наступним чином: насамперед відбувається видалення шуму, а саме згладжування, яке можна подати у вигляді формули Гауса для двовимірного випадку:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}\, e\frac{-(x^2+y^2)}{2\sigma^2}, \tag{1.4}$$

де σ – це коефіцієнт згладжування, що він більше, то сильніше згладжується вхідне зображення.

Після процесу видалення шумів (згладжування) здійснюється обчислення градієнтів згладженого зображення. Воно обчислюються у двох напрямках, у вертикальному та горизонтальному поданні. Ця операція розраховується з урахуванням першої похідної. Значення, отримані в результаті подібної обробки, поєднуються за формулою:

$$E_{ij} = \sqrt{g_{vij}^2 + g_{hij}^2}\,, \tag{1.5}$$

де g_v – вертикальна вистава;

g_h – горизонтальне уявлення.

Напрямок градієнта обчислюється за формулами:

$$\theta_{ij} = tan^{-1}\ \frac{g_{vij}}{g_{hij}} \tag{1.6}$$

$$E_{Tij} = \begin{Bmatrix} E_{ij}, \text{якщо } E_{ij}, > \text{T}, \\ 0, \text{інакше} \end{Bmatrix},$$

де θij – кут напряму вектора;

T- Поріг видалення шумів.

Коефіцієнт T підставляється таким чином, що межі об'єкта, що розпізнається, будуть виділені і залишаться в результаті без змін, після видалення більшої частини шумів.

Наступна операція алгоритму Кані придушення не максимальних градієнтів. Щоб це зробити, слід здійснювати обнулення градієнтів по двох порогах T1 і T2, причому з виконанням умови. Розглянутий градієнт не можна обнулювати, поки його значення у вибраній точці менше ніж поріг T1 і більше, ніж поріг T2 [9]. Дані дії здійснюються для виявлення найбільш значущих контурів на зображенні.

Далі буде наведено приклад роботи алгоритму Канни (рисунок 1.6).

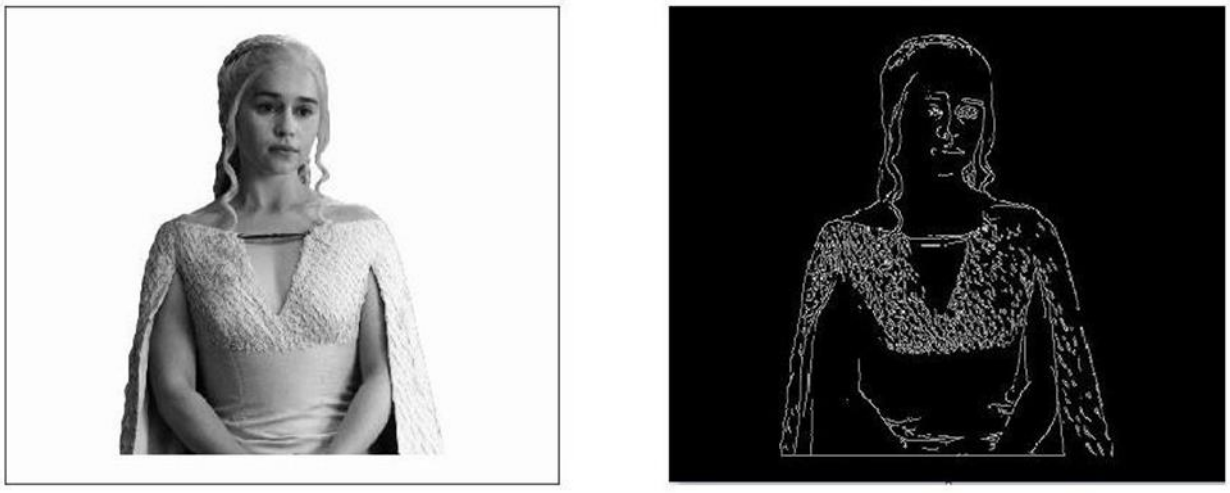

Рисунок 1.6 – Приклад роботи методу виділення меж Канни

Перевагою даного покращення алгоритму Віоли-Джонса є низький коефіцієнт помилки під час пошуку об'єкта на зображенні [9].

Недоліками є:

- високої обчислювальної складності;
- деякі реалізації Кані схильні до високої чутливості до шуму.

В результаті, наведені модифікації вище показують те, що суті методи

поліпшення алгоритму Віоли-Джонса і кожна модифікація має свої мінуси і плюси, але в розгляд буде взята класична реалізація методу Віоли-Джона.

1.4 Постановка задачі

Відштовхуючи від цього, навіщо створили алгоритм Віоли-Джонса, він також знайшов своє застосування у розпізнавання об'єктів різного роду, незалежно від форми, кольору, розташування просторі. І до цього дня він досить добре показує себе в завданнях розпізнавання об'єктів, що шукаються. Виходячи з цього, напрошується завдання, як покаже себе алгоритм Віоли-Джонса при навантаженні. А також постають питання, яка кількість каскадів-класифікаторів змусить змінити продуктивність методу в найгірший бік і скільки об'єктів за раз можна буде розпізнавати алгоритмом Віоли-Джонса за комфортної роботи? Відповіді на ці питання дасть застосунок , що розробляється з розпізнавання множини об'єктів на зображенні з використанням даного методу.

Як приклад, що має бути результатом роботи програми, що розробляється, наведемо рисунок 1.7.

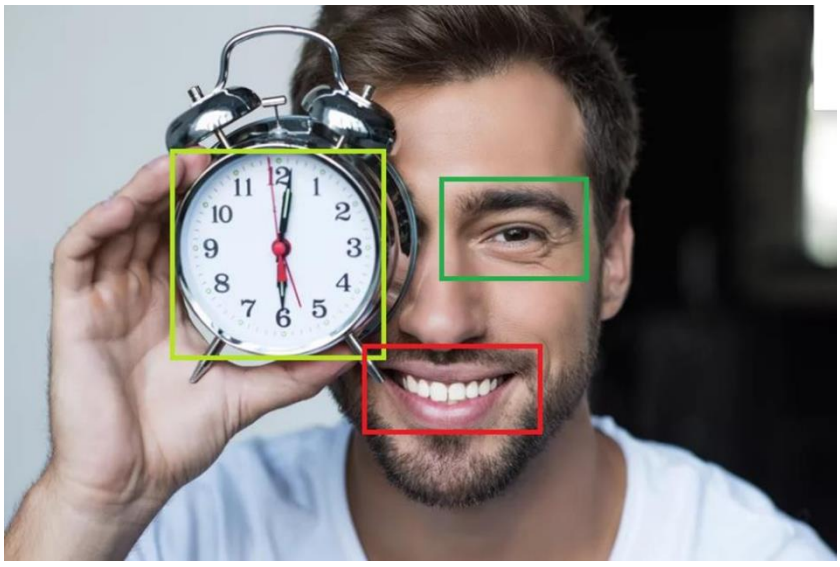

Рисунок 1.7 – Приклад результату роботи програми, що розробляється

Дане зображення є зразковим результатом роботи програми розпізнавання множини об'єктів на зображенні. Як видно, на зображенні було розпізнано кілька об'єктів, таких як око людини, її посмішка та будильник у руці. Щоб досягти цього результату, слід передбачити наступні пункти:

- застосунок з розпізнавання безлічі об'єктів на зображенні має підтримувати, як багатопоточну реалізацію розпізнавання різного роду об'єктів, так і однопоточний підхід;
- підтримка модулів програми розпізнавання об'єктів, як на зображенні, так і на вхідному відео потоці;
- додаток повинен фіксувати результати своєї роботи (вести журнал журнал, зберігати фотографію після обробки);
- підтримка додавання та видалення каскадів-класифікаторів у застосунку;
- впровадження тест кейсів на базі Unit-тестів або Cucumber тестування для підтримки працездатності програми у різноманітних способах взаємодії;
- додаток повинен мати дружній інтерфейс користувача для зручної та інтуїтивної взаємодії;
- в результаті роботи додаток має демонструвати отримані результати в реальному часі.

Таким чином, було проаналізовано предметну область програмного рішення, що розробляється. Проаналізовано алгоритм Віоли-Джонса. Наведено способи модифікації даного алгоритму. Було виявлено, навіщо цей алгоритм може бути корисним у час.

Виходячи з отриманих даних було поставлено вимоги до програмного забезпечення та виявлено очікувані результати.

Зроблені в цьому розділі дії необхідні для успішного проектування програми розпізнавання безлічі об'єктів на зображенні з використанням алгоритму Віоли-Джонса.

2 ПРОЕКТУВАННЯ ЗАСТОСУНКІВ З РОЗПІЗНАВАННЯ МНОЖИННИХ ОБ’ЄКТІВ НА ЗОБРАЖЕНІ

2.1 Концептуальна модель

Людина зазвичай ділить складні завдання на менші підзадачі, що дозволяє глибше проаналізувати суть проблеми, зекономити час та краще зрозуміти окремі, складні для сприйняття аспекти. Для цього було створено низку ефективних методологій і парадигм, які значно полегшують процес розробки програмного забезпечення.

З метою забезпечення розуміння принципу функціонування застосунку, призначеного для розпізнавання множини об'єктів на зображенні, було створено діаграму бізнес-процесу, розроблену на основі методології IDEF0.

IDEF0 є методологією функціонального моделювання та графічною нотацією, що використовується для опису й формалізації бізнес-процесів. Вона вирізняється своєю специфічною рисою — акцентом на ієрархічній підпорядкованості об'єктів. У межах IDEF0 розглядаються логічні взаємозв’язки між процесами, а не їхня послідовність у часі. Методологія також наочно демонструє взаємозв’язки між функціями розроблюваного програмного забезпечення (рисунок 2.1) [10].

Рисунок 2.1 – IDEF0-діаграма розпізнавання об'єкта

Як видно з попередньої ілюстрації, у застосунку комп'ютерного зору (ЗКЗ), який виконує роль оброблювального механізму, на вхід подається кадр зображення, що містить об'єкт, який необхідно ідентифікувати. Для аналізу цього зображення використовуються попередньо завантажений каскад-класифікаторів та алгоритм Віоли-Джонса, що забезпечує розпізнавання об'єкта та формування відповідної області на виході.

Подана діаграма є корисною для розуміння принципів функціонування двох ключових модулів, які необхідно реалізувати в межах розробки:

- модуль розпізнавання об'єктів на статичному зображенні;
- модуль розпізнавання об'єктів у відеопотоці.

Крім того, важливим етапом є завантаження вже навченого каскаду-класифікаторів до додатка для подальшого використання в задачі розпізнавання об'єктів (рисунок 2.2).

Рисунок 2.2 – IDEF0-діаграма завантаження навченого каскаду-класифікаторів у застосунок

На наведеному рисунку продемонстровано процес надходження нового каскаду до застосунку. До нього застосовуються певні елементи управління у вигляді правил валідації. Наприклад, у застосунку повинні бути присутні два однакові каскади-класифікатори з розширенням «.xml» та інші вимоги. Після

успішного проходження перевірки каскад-класифікатор підвантажується в систему комп’ютерного зору та стає готовим до використання.

2.2 Логічна модель

Методика створення логічної моделі передбачає поетапне відображення функціональних аспектів логічної частини програми шляхом побудови діаграми варіантів використання; розкриття елементної структури логічної компоненти на основі діаграми класів предметної області; а також моделювання динамічного аспекту через побудову діаграми послідовності. Вся логічна модель проєктується з використанням засобів мови UML.

UML (Unified Modeling Language) – це мова моделювання, яка слугує інструментом для об'єктно-орієнтованого аналізу та проєктування. Вона базується на трьох ключових концепціях: діаграми, сутності та зв’язки між ними [10].

- сутності – це абстрактні об'єкти, що є базовими елементами моделі;
- зв’язки – це елементи, які встановлюють відношення між сутностями;
- діаграми – це графічні блоки, які групують сутності та зв’язки у структуровану форму подання.

UML широко використовується для візуалізації, побудови та документування програмного забезпечення, що перебуває в процесі розробки.

Одним із ключових інструментів у бізнес-аналізі є діаграма варіантів використання, яка дозволяє визначити коло користувачів програмного забезпечення та окреслити перелік доступних їм дій (рисунок 2.3) [10].

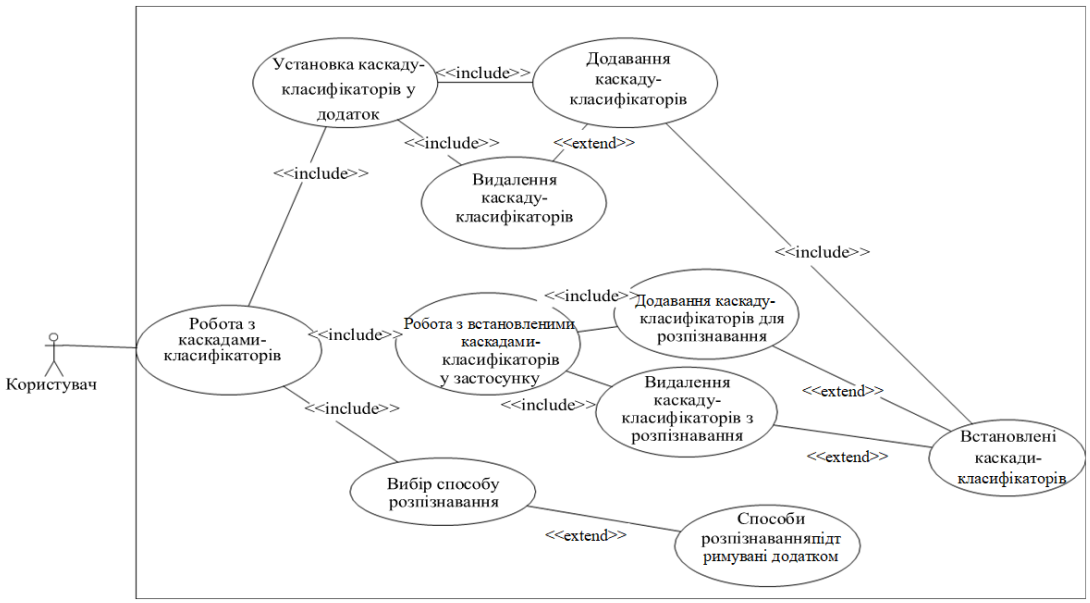

Рисунок 2.3 – Діаграма варіантів використання програми, що розробляється

Основними складовими елементами діаграми є актори, сутності, які виконують дії та варіанти використання, тобто можливі дії, які виконують актори.

На цій діаграмі видно, що є один актор, тобто користувач, який може взаємодіяти із програмою. Користувачу дозволено видаляти та додавати існуючі каскади-класифікаторів для розпізнавання у програмі. Також дозволено додавати нові каскади-класифікаторів у додаток та видаляти їх. Можливий вибір користувачем підтримуваний програмою спосіб розпізнавання об'єкта.

Невід'ємною частиною логічного моделювання є діаграми класів, які використовуються при моделюванні та проектуванні додатків. Дані діаграми описують додаток у статичному поданні, демонструючи її структуру.

Діаграма класів включає кілька складових елементів, які у свою чергу відображають декларативні знання про предметну область (рисунок 2.4). Декларативність інтерпретуються у класи, інтерфейси та відносини між ними, які є основними поняттями мови UML [10].

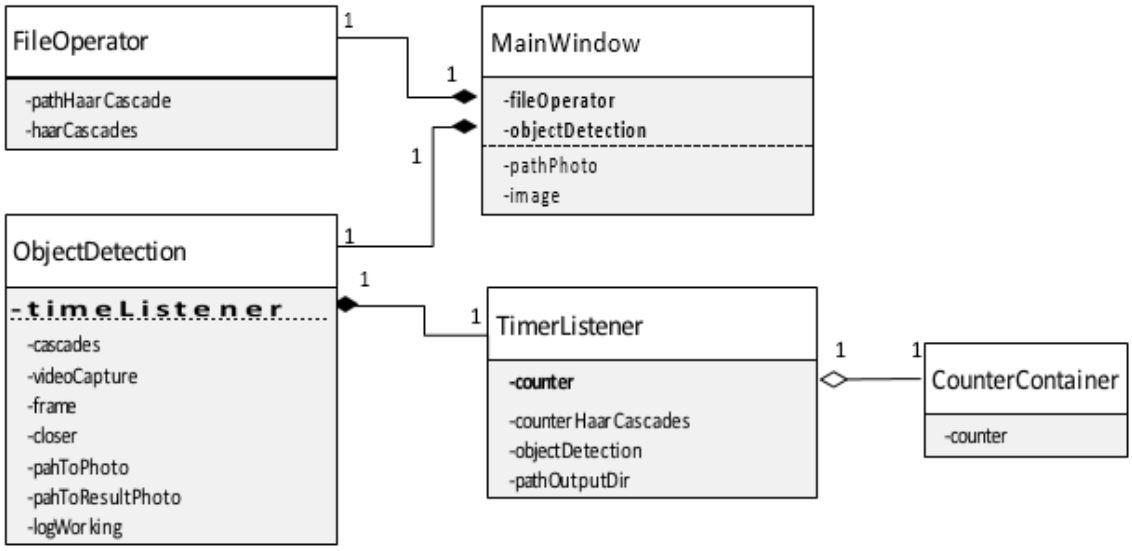

Рисунок 2.4 – Діаграма класів програми, що розробляється

На представленій діаграмі відображено п’ять класів, які взаємодіють між собою через відносини композиції та агрегації. Наведемо короткий опис семантичного призначення кожного з класів:

- MainWindow – клас, що реалізує головне вікно графічного інтерфейсу, об'єднуючи основні компоненти програми. Саме тут реалізовано взаємодію з каскадами-класифікаторами, здійснюється логування результатів роботи та надається можливість вибору методу розпізнавання;

- FileOperator – функціональний модуль для роботи з файлами, відповідальний за зчитування даних про доступні каскади-класифікатори, їхнє додавання або видалення, а також забезпечення допоміжного функціоналу, пов’язаного з обробкою файлів;

- ObjectDetection – клас, що містить основну логіку реалізації застосунку комп’ютерного зору. У ньому імплементовано алгоритм Віоли-Джонса для застосування до статичних зображень і відеопотоку, а також механізм внутрішнього логування результатів розпізнавання;

- TimerListener – клас, який реалізує окремий потік, призначений для обробки кадрів у режимі реального часу (визначення частоти кадрів на

секунду) та збереження результатів до файлу;

- CounterContainer – допоміжний клас, що забезпечує інкрементування лічильника кадрів у межах роботи модуля TimerListener.

Слід зазначити, що розроблюване програмне забезпечення підтримуватиме два основні режими розпізнавання об’єктів на зображеннях, завантажених користувачем та у відеопотоці.

Для кожного з цих режимів буде реалізовано відповідні модулі, які матимуть аналоги з використанням паралельного обчислення, що реалізується засобами мови програмування. Паралелізація стосуватиметься саме компонента розпізнавання об’єктів – кожен окремий потік опрацьовуватиме кадр, використовуючи власний каскад-класифікатор та застосовуючи алгоритм Віоли-Джонса.

Перед реалізацією багатопотокової версії обов’язковим є проектування однопотокового варіанту алгоритму. Це дозволяє виявити критичні зони обчислювального навантаження, які в майбутньому слід оптимізувати. Для цього використовується блок-схема – графічне представлення алгоритмів та процесів, де окремі кроки подано у вигляді фігур (блоків), з'єднаних стрілками, що вказують порядок виконання операцій (рисунок 2.5) [10].

Рисунок 2.5 – Блок-схема узагальненого варіанта однопотокового модуля з розпізнавання об'єкта з відео потоку

На представленому зображенні продемонстровано блок процесу, який відповідає за запуск окремого потоку, що здійснює підрахунок кількості кадрів на секунду. Цей блок виконується до основного циклу, що означає: у рамках однопотокової реалізації саме один потік контролюватиме загальну продуктивність роботи алгоритму під час навантаження. Такий підхід дозволяє оцінити ефективність функціонування алгоритму при зміні кількості підключених каскадів-класифікаторів.

Окремо реалізовано цикл, що відповідає за завершення роботи модуля – він очікує відповідного сигналу від користувача застосунку. Вхідне

зображення для аналізу надходить безпосередньо з камери, після чого обробка передається на алгоритм Віоли-Джонса, який здійснює розпізнавання та формує результат, що виводиться на екран користувача.

Аналізуючи блок-схему, було ухвалено рішення реалізувати паралельну обробку процесу розпізнавання об’єктів – шляхом розподілу навантаження між кількома потоками, кожен з яких використовуватиме власний каскад-класифікатор для обробки кадру (рисунок 2.6).

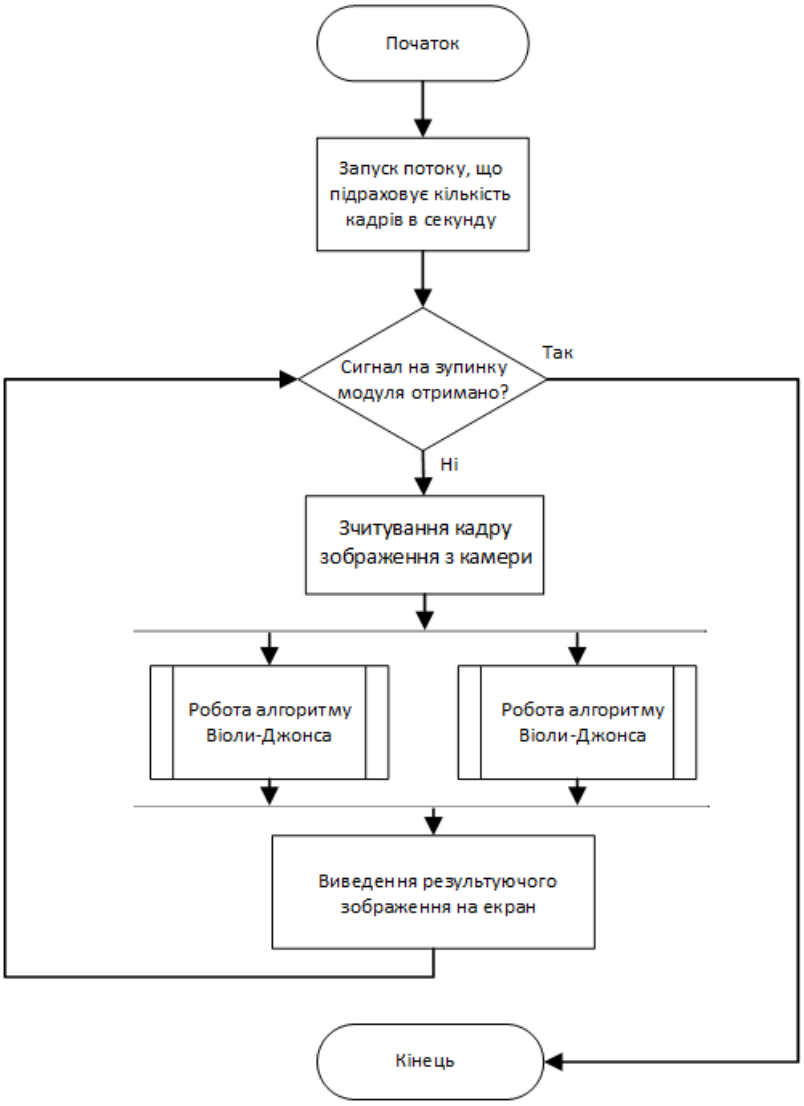

Рисунок 2.6 – Блок-схема узагальненого варіанта багатопоточного модуля

Паралелізація процесу розпізнавання об'єктів на зображенні може суттєво підвищити продуктивність, оскільки ця операція виконуватиметься

одночасно у кількох потоках – відповідно до кількості каскадів-класифікаторів, обраних користувачем.

Раніше були продемонстровані реалізації модуля розпізнавання об’єктів у відеопотоці у двох варіантах: однопотоковому та багатопотоковому. Модуль, що відповідає за розпізнавання об'єктів на статичних зображеннях, матиме подібну архітектуру. Основна відмінність полягає в джерелі надходження даних: у цьому випадку зображення отримуються безпосередньо з операційної системи, оскільки користувач має змогу завантажувати власні фотографії для обробки.

2.3 Фізична модель

Різні компоненти логічної моделі можуть мати матеріальне або фізичне втілення. Вони відображають рівень розуміння структури програмного забезпечення, що розробляється, а також модель поведінки системи. Для створення конкретної фізичної моделі програмного продукту необхідно реалізувати всі логічні елементи у вигляді матеріальних сутностей. Саме для цього використовується аспект фізичного представлення моделі.

Ключовим елементом фізичної моделі виступає діаграма розгортання, яка зазвичай створюється як додаток до технічного завдання. Розробкою такої діаграми займається менеджер проєкту перед початком обговорення архітектурних рішень з командою розробників. Вона часто служить відправною точкою проєкту, оскільки у процесі визначення базових апаратних компонентів формуються критично важливі вимоги до системи.

Діаграма розгортання також є ефективним інструментом для представлення загальної концепції програми зацікавленим сторонам, які не є розробниками. Завдяки мінімальному вмісту технічних деталей, така діаграма є зрозумілою для ширшої аудиторії [10].

На рисунку 2.7 подано діаграму розгортання програми, призначеної для

розпізнавання великої кількості об’єктів на зображенні.

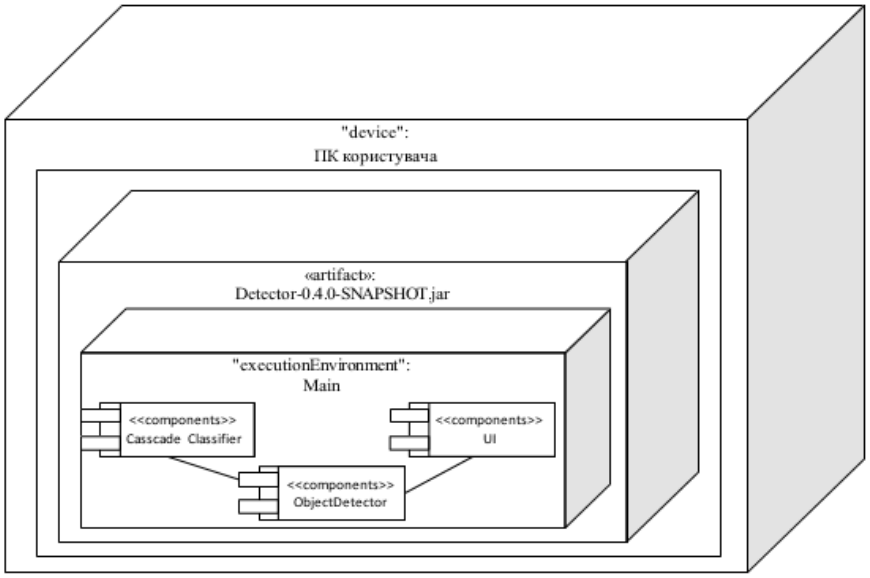

Рисунок 2.7 – Діаграма розгортання програми розпізнавання об'єктів

На персональному комп’ютері користувача встановлено програму розпізнавання об’єктів. Це програмне забезпечення представлено у вигляді архіву Java з розширенням .jar, до складу якого входять основні компоненти: Cascade Classifier, Object Detector та UI. Вони забезпечують ключові функції системи.

У результаті були розроблені концептуальна, логічна та фізична моделі програмного продукту. Такий підхід дозволяє коректно реалізувати систему для розпізнавання множини об'єктів із застосуванням алгоритму Віоли-Джонса, враховуючи всі необхідні аспекти в контексті об’єктно-орієнтованого програмування.

3 ЗПРОГРАМНА РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ

3.1 Програмна реалізація програми

У процесі реалізації дослідницької роботи було використано інтегроване середовище розробки IntelliJ IDEA. Це потужне середовище, розроблене компанією JetBrains, призначене для підтримки різних мов програмування, зокрема Java, JavaScript та Python.

Для створення програмного забезпечення з розпізнавання безлічі об’єктів на зображеннях було обрано мову Java, яка є об'єктно-орієнтованою. Однією з ключових переваг Java є принцип «напиши один раз – запускай скрізь», тобто програма, написана на цій мові, може виконуватись на будь-якій операційній системі, де встановлено Java Virtual Machine. Важливою особливістю також є наявність широкого спектра готових бібліотек, що значно полегшують розробку, зменшуючи потребу у повторному створенні типового функціоналу [4-7].

Для реалізації алгоритму Віоли-Джонса було обрано бібліотеку OpenCV – відкрите програмне забезпечення, що містить велику кількість методів комп’ютерного зору [9].

Для впровадження багатопотокового підходу до ідентифікації об’єктів було прийнято рішення використовувати вбудований клас Thread мови Java. Цей клас інкапсулює основні механізми роботи з потоками та забезпечує зручний інтерфейс для реалізації паралельних процесів [10].

Функціональні можливості програми:

- підтримка одночасного розпізнавання об’єктів різних типів;
- реалізовано два режими роботи: розпізнавання на статичному зображенні та у відеопотоці;
- визначення та виділення об’єктів за допомогою методу Віоли-Джонса;

- можливість одночасного використання декількох каскадів-класифікаторів для виявлення різних об’єктів;

- підтримка підключення користувацьких каскадів-класифікаторів;

- реалізовано валідацію каскадів-класифікаторів для уникнення помилок у роботі застосунку;

- ведення логування продуктивності (наприклад, кадри за секунду) під час обробки відеопотоку в реальному часі;

- можливість збереження обробленого кадру з виділеними об’єктами після завершення процесу розпізнавання.

Проєктна структура застосунку, реалізованого у середовищі IntelliJ IDEA, зображена на рисунку 3.1.

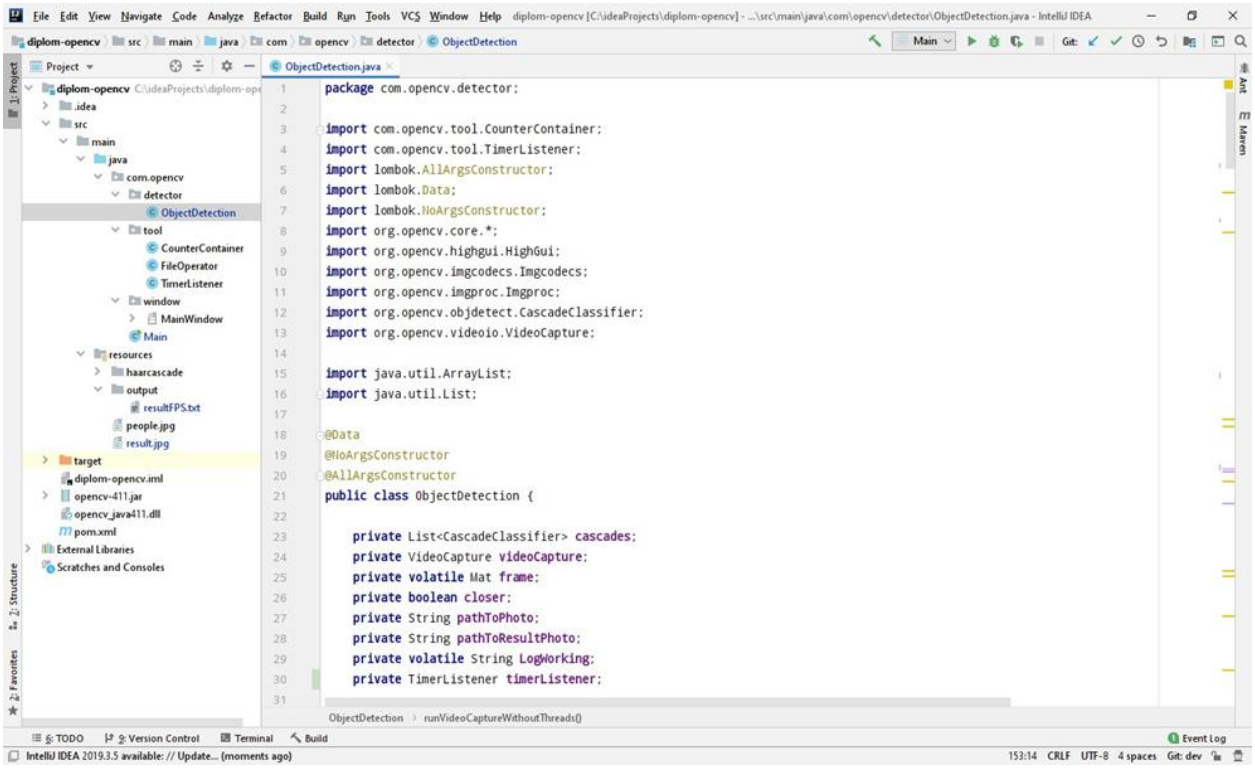

Рисунок 3.1 – Структура проєкту програми у середовищі IntelliJ IDEA

Як систему керування версіями було обрано платформу GitLab. Вона забезпечує централізовану роботу з Git-репозиторіями, підтримує CI/CD-процеси, полегшує тестування, створення та розгортання програмного

забезпечення, сприяє контролю якості коду та дозволяє зосередитись на розробці продукту замість конфігурування інструментів.

На рисунку 3.2 представлено створений репозиторій на платформі GitLab, призначений для розміщення та керування кодом застосунку з розпізнавання безлічі об’єктів на зображенні.

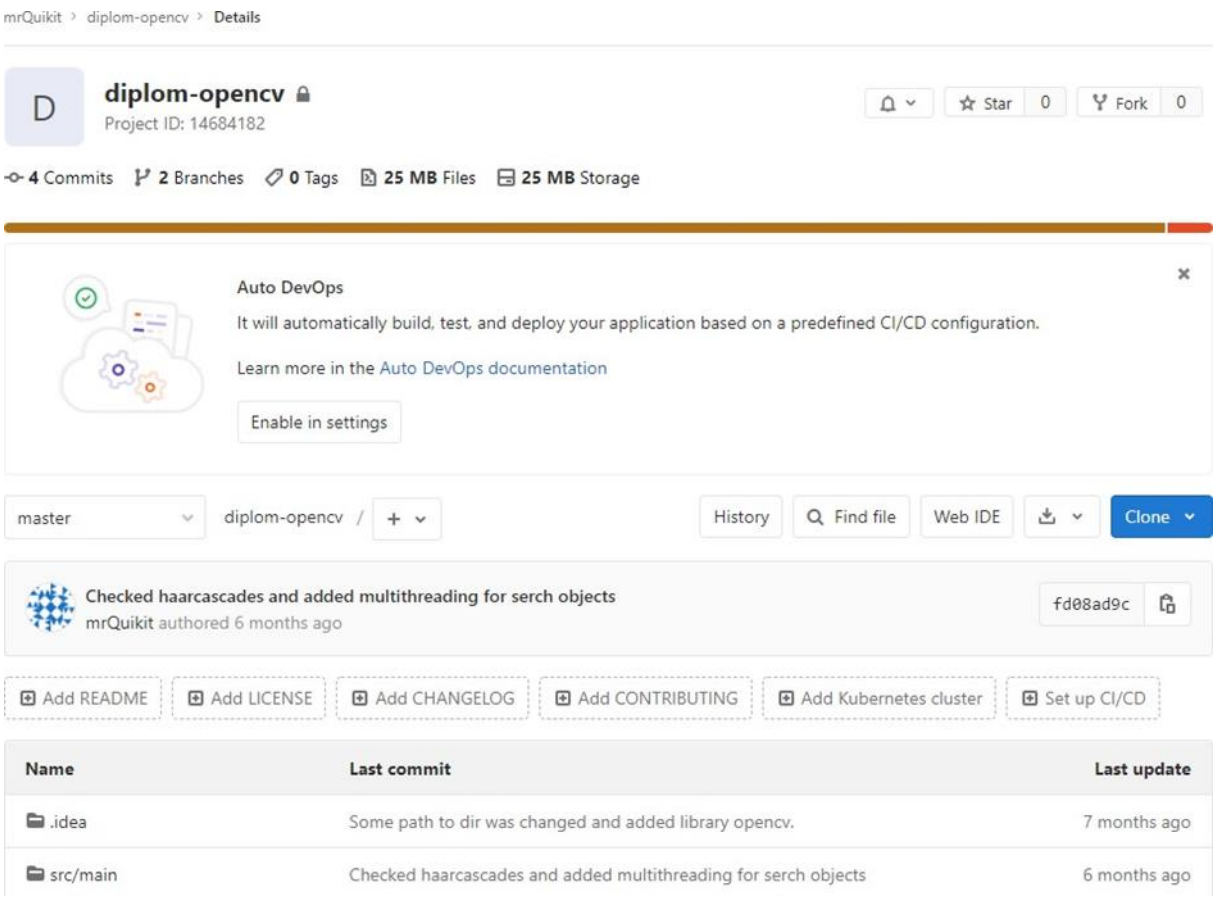

Рисунок 3.2 – Репозиторій для зберігання версій програми, що розробляється

У створеному GitLab-репозиторії було реалізовано дві основні гілки: master і dev, які представляють різні стани розробки програмного продукту. Гілка master містить стабільну версію програми, яка пройшла тестування на наявність критичних помилок. У гілці dev зберігається актуальна версія, що перебуває на стадії активної розробки або оновлення: тут здійснюється виправлення помилок та впровадження нової функціональності. Після завершення етапу розробки зміни з dev інтегруються у master шляхом злиття

(merge), при цьому виконуються дії з вирішення конфліктів між версіями. У підсумку master відображає актуальний, готовий до використання стан продукту.

Далі розглянемо реалізацію функціоналу програми для розпізнавання множини об’єктів на зображенні, зокрема у однопотоковій та багатопотоковій конфігурації. На рисунку 3.3 подано фрагмент коду, що демонструє реалізацію локалізації об’єктів у рамках однопотокового підходу.

```
public void runPictureDetectionWithoutThreads(){
    frame = Imgcodecs.imread(pathToPhoto);
    for (CascadeClassifier cascadeClassifier: cascades) {
        detectOnFrame(frame, cascadeClassifier, new Scalar(0, 255, 0));
    }
    Imgcodecs.imwrite(pathToResultPhoto, frame);
    Thread thread = new Thread(() -> {
        HighGui.imshow( winname: "Photo Detector (Threads)", frame);
        HighGui.waitKey( delay: 10);
        while (true) {
            HighGui.imshow( winname: "Photo Detector (Threads)", frame);
            HighGui.waitKey( delay: 10);
            if (closer) {
                HighGui.waitKey(HighGui.n_closed_windows);
                Thread.currentThread().stop();
                break;
            }
        }
    });
    thread.start();
}
```

Рисунок 3.3 – Фрагмент коду однопотокового варіанта функції розпізнавання множини об’єктів

У представленому коді змінна frame є вхідним зображенням, завантаженим користувачем у додаток. За допомогою циклу for здійснюється перебір обраних користувачем каскадів-класифікаторів, які передаються до

функції detectOnFrame, в якій реалізовано алгоритм Віоли-Джонса. Після цього створюється окремий потік, відповідальний за виведення результатів розпізнавання у графічний інтерфейс. Цей потік працює до моменту завершення взаємодії з користувачем.

На основі однопотокового рішення реалізовано багатопоточну версію функції, яка використовує клас Thread для розпаралелювання процесу розпізнавання об'єктів на зображенні (рисунок 3.4).

```
public void runPictureDetection(){
    frame = Imgcodecs.imread(pathToPhoto);
    List<Thread> threads = new ArrayList<>();
    for (int i = 0; i < cascades.size() ; i++) {
        int finalI = i;
        Thread thread = new Thread(() -> {
            int r = (int) (Math.random() * (256));
            int g = (int) (Math.random() * (256));
            int b = (int) (Math.random() * (256));
            System.out.println(Thread.currentThread().getName());
            detectOnFrame(frame, cascades.get(finalI), new Scalar(r, g, b));
        });
        threads.add(thread);
    }
    for (Thread t : threads) {
        t.start();
    }
    boolean flag = true;
    while (flag){
        int counter = 0;
        for (int i = 0; i < threads.size() ; i++) {
            if(threads.get(i).getState() == Thread.State.TERMINATED){
                ++counter;
            }
        }
        if (counter == threads.size()){
            flag = false;
        }
    }
```

Рисунок 3.4 – Фрагмент коду багатопотокового варіанта розпізнавання об’єктів

У цьому варіанті використовується колекція List під назвою threads для збереження створених потоків, кожен з яких оперує одним каскадом-

класифікатором. У межах потоку за допомогою функції random з бібліотеки Math генерується випадковий колір (RGB) для виділення об’єктів на зображенні. Згенеровані значення передаються як аргументи класу Scalar, який забезпечує колірні параметри. Далі відбувається виклик функції detectOnFrame, після чого кожен потік запускається через метод start(). У циклі while здійснюється перевірка завершення роботи потоків (стан Terminated). Результати обробки, як і в однопотоковій реалізації, виводяться в графічний інтерфейс.

Функція detectOnFrame, яка реалізує основну логіку алгоритму Віоли-Джонса, представлена на рисунку 3.5.

```
public void detectOnFrame(Mat frame, CascadeClassifier cascade, Scalar scalar){
    Mat frameGray = new Mat();
    Imgproc.cvtColor(frame, frameGray, Imgproc.COLOR_BGR2GRAY);
    Imgproc.equalizeHist(frameGray, frameGray);
    MatOfRect findedObject = new MatOfRect();
    try {
        cascade.detectMultiScale(frameGray, findedObject, scaleFactor: 1.3, minNeighbors: 1, flags: 0,
                new Size( width: 50, height: 50),new Size( width: 300, height: 300));
    }catch (CvException e){
        e.printStackTrace();
    }
    findedObject.toList().forEach(object->{
        Imgproc.rectangle(frame, new Point(object.x, object.y),
                new Point( x: object.x + object.width, y: object.y + object.height),
                scalar, thickness: 2);
            });
}
```

Рисунок 3.5 – Фрагмент функції detectOnFrame з реалізацією алгоритму Віоли-Джонса

У даному фрагменті аргументи функції мають наступне призначення: frame (об’єкт класу Mat) – вхідне зображення; cascade – переданий каскад-класифікатор; scalar – об’єкт, що містить колірні параметри у форматі RGB. Першим етапом є створення frameGray, тобто зображення у відтінках сірого, яке формується з використанням функції cvtColor класу Imgproc. Далі

застосовується вирівнювання гістограми (equalizeHist) для підвищення контрастності перед обробкою алгоритмом. Сам алгоритм викликається через метод detectMultiScale, що реалізує пошук об'єктів. Його параметри включають: зображення у сірому градієнті, масив знайдених об'єктів (findedObject), коефіцієнт масштабування (scaleFactor), та межі мінімального і максимального розміру об'єктів. Після виявлення об’єктів, їхні межі візуалізуються функцією rectangle.

Модуль розпізнавання у відеопотоці має подібну логіку, проте джерелом зображення у цьому випадку виступає камера. На рисунку 3.6 представлено конструктор класу ObjectDetection, в якому здійснюється ініціалізація ключових об’єктів.

```
public ObjectDetection(List<String> pathCascades){
    cascades = getValidationObjectCascade(pathCascades);
    frame = new Mat();
    videoCapture = new VideoCapture( index: 0);
    closer = false;
    pathToPhoto = "src/main/resources/people.jpg";
    pathToResultPhoto = "src/main/resources/result.jpg";
}
```

Рисунок 3.6 – Фрагмент коду конструктора класу ObjectDetection

Конструктор – це спеціальний метод, призначений для створення екземпляра класу. У цьому випадку аргументом є колекція List, яка містить шляхи до каскадів-класифікаторів. У межах конструктора виконується ініціалізація об’єкта frame (вхідне зображення), об’єкта videoCapture (для взаємодії з камерою), а також змінної closer, що сигналізує про необхідність завершення роботи потоків. Також ініціалізуються шляхи до стандартного зображення та місця збереження результатів.

У результаті реалізовано два функціональні модулі: розпізнавання об’єктів на статичному зображенні та у відеопотоці. Обидва модулі

підтримують однопоточний і багатопоточний режим роботи з використанням можливостей мови Java. Оцінка роботи системи буде розглянута у наступному підрозділі.

3.2 Тестування програми

Для перевірки функціональності застосунку тестування проводилося як із використанням завантаженого зображення, так і за допомогою реального відеопотоку, отриманого з камери з роздільною здатністю 0,3 МП (640×480).

На рисунку 3.7 наведено результат роботи модуля розпізнавання на зображенні в однопоточній реалізації.

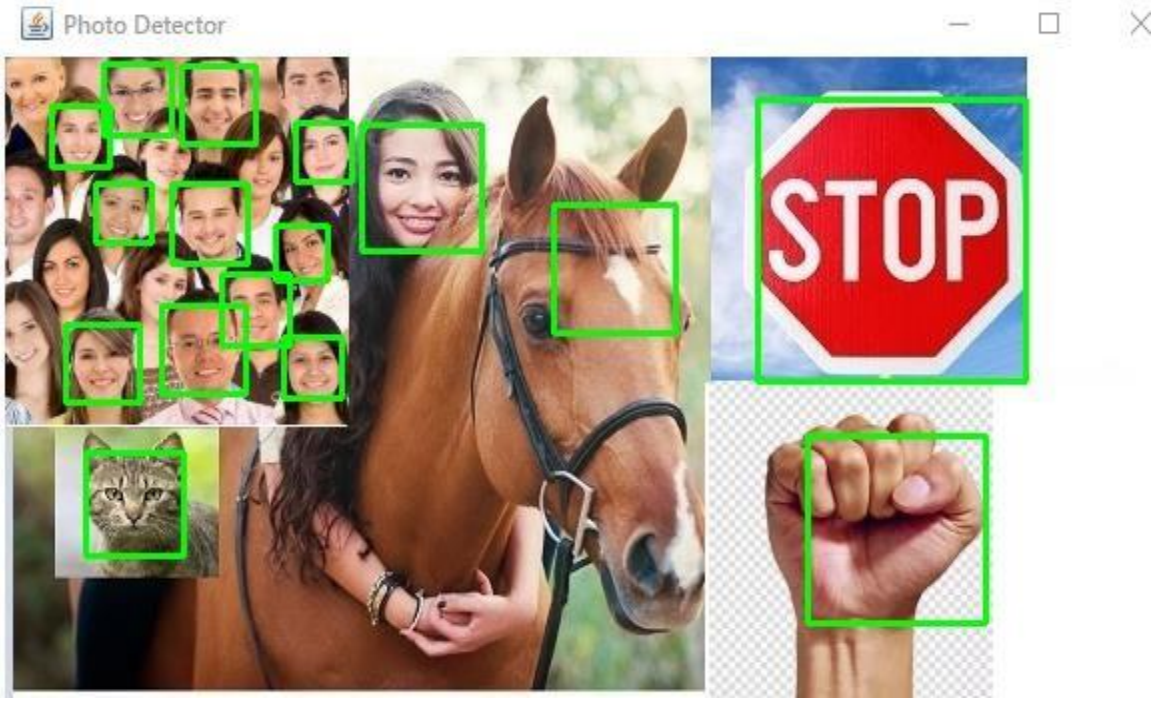

Рисунок 3.7 – Результат однопотокового розпізнавання на завантаженому зображенні

На цьому прикладі алгоритм Віоли-Джонса зміг виявити кілька об’єктів різних типів: обличчя людей, голову кішки, стоп-сигнал та стиснутий кулак. Водночас спостерігається поява артефакту розпізнавання – виявлення області,

де розташований кінь, хоча відповідний каскад-класифікатор не використовувався під час тестування. Подібні артефакти усуваються шляхом підвищення порогового значення параметра відсіювання помилкових спрацювань у функції розпізнавання.

Для порівняння результатів однопоточної та багатопоточної реалізацій на тому ж самому зображенні було проведено аналогічний експеримент, але вже у багатопоточному режимі (рисунок 3.8).

Рисунок 3.8 – Результат багатопоточного розпізнавання на завантаженому зображенні

Основна різниця між однопотоковою та багатопотоковою реалізацією полягає в кольоровому маркуванні виявлених об’єктів. У багатопоточному режимі кожен виявлений об’єкт виділяється індивідуальним кольором, що дозволяє краще диференціювати результати розпізнавання.

Для даного тесту було обрано чотири каскади-класифікатори із наявних тридцяти дев’яти (рисунок 3.9).

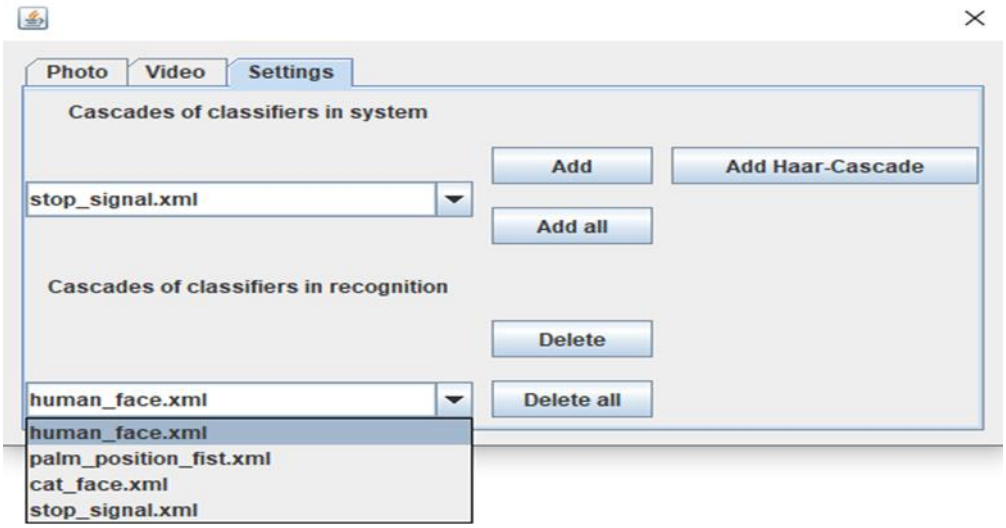

Рисунок 3.9 – Інтерфейс користувача: взаємодія з каскадами-класифікаторами

На зображенні продемонстровано відповідний інтерфейс налаштувань, де видно, що активовано лише чотири каскади зі списку.

Далі проведено тестування модуля розпізнавання відеопотоку в реальному часі за допомогою веб-камери. На рисунку 3.10 показано результат його роботи в однопоточному режимі.

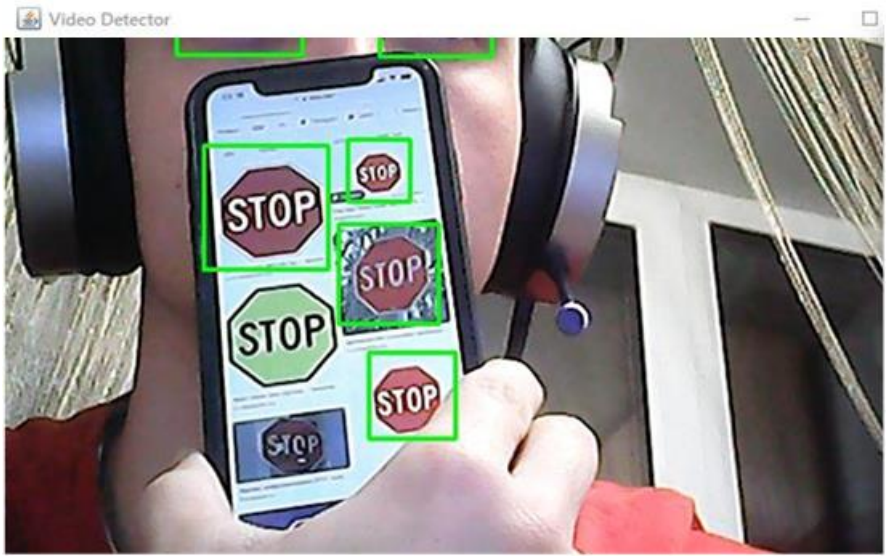

Рисунок 3.10 – Однопоточне розпізнавання у відеопотоці в реальному часі

Програма успішно ідентифікувала два ока людини та стоп-сигнали, що

відображалися на екрані телефону.

Аналіз продуктивності алгоритму Віоли-Джонса. Для відповіді на ключове запитання – наскільки ефективно працює алгоритм Віоли-Джонса при збільшенні навантаження – було проведено серію тестів з різною кількістю підключених каскадів-класифікаторів. Результати наведено у таблиці 3.1.

Таблиця 3.1 – Продуктивність алгоритму Віоли-Джонса при різному навантаженні

Кількість каскадів-класифікаторів	Однопоточна реалізація	Багатопотокова реалізація
	Кадри за секунду (640×480)	
1	26	25
3	15	16
6	9	15
9	5	14
12	4	10

З таблиці видно, що при зростанні кількості каскадів-класифікаторів, однопотокова реалізація поступово втрачає продуктивність. Уже при шести каскадах кількість оброблених кадрів на секунду зменшується до 9, що знижує ефективність системи в реальному часі. Багатопотокова реалізація, навпаки, демонструє стабільніші показники FPS, дозволяючи обробляти до 12 каскадів з прийнятним рівнем продуктивності.

З таблиці видно, що при зростанні кількості каскадів-класифікаторів, однопотокова реалізація поступово втрачає продуктивність. Уже при шести каскадах кількість оброблених кадрів на секунду зменшується до 9, що знижує ефективність системи в реальному часі. Багатопотокова реалізація, навпаки, демонструє стабільніші показники FPS, дозволяючи обробляти до 12 каскадів

з прийнятним рівнем продуктивності.

На цій таблиці видно, як поводиться алгоритм Віоли-Джонса при навантаженні каскадами-класифікаторів різного роду для розпізнавання безлічі об'єктів на зображенні в реальному часі. При однопотоковій реалізації методу розпізнавання об'єктів на відео потоці працювати ставати проблематично при підключенні шести каскадів-класифікаторів, так як кількість оброблюваних зображень сходиться в середньому до дев'яти. Але це можна виправити шляхом застосування багатопоточності. За такого результату можна застосувати для більш-менш комфортної роботи дванадцять каскадів-класифікаторів.

Отже, межі навантаження стандартного алгоритму Віоли-Джонса було виявлено, вважатимуться, що мета роботи досягнуто.

ВИСНОВКИ

Кваліфікаційна робота присвячена дослідженню ефективності функціонування алгоритму Віоли-Джонса в умовах обчислювального навантаження. Метою дослідження була розробка програмного забезпечення для розпізнавання множини об’єктів на зображенні, що визначило основні напрями реалізації.

У процесі дослідження було виконано такі завдання:

- проаналізовано особливості функціонування алгоритму Віоли-Джонса;
- розроблено програмний модуль, що реалізує розпізнавання об’єктів з використанням механізмів розпаралелювання обчислень;
- проведено порівняльне тестування однопотокової та багатопотокової реалізацій алгоритму Віоли-Джонса.

Окрім цього, були ідентифіковані недоліки алгоритму, а також розглянуто потенційні модифікації, спрямовані на їх усунення, із зазначенням можливих наслідків впровадження таких змін.

У результаті реалізовано десктопний додаток мовою програмування Java (у середовищі IntelliJ IDEA) із використанням відкритої бібліотеки OpenCV, який дозволяє розпізнавати об’єкти на:

- статичних зображеннях, завантажених користувачем;
- відеопотоці в реальному часі, що надходить із веб-камери.

Під час тестування встановлено, що продуктивність програми прямо залежить від кількості підключених каскадів-класифікаторів, оскільки при цьому зростає обчислювальне навантаження на центральний процесор через збільшення обсягу оброблюваних математичних операцій.

Алгоритм Віоли-Джонса було протестовано з різною кількістю каскадів-класифікаторів; результати експериментів дозволили визначити граничне

навантаження для забезпечення стабільної та комфортної роботи програми в реальному часі.

Проведене дослідження підтвердило, що алгоритм Віоли-Джонса придатний для локалізації та розпізнавання об’єктів різного типу. Водночас, з огляду на сучасні вимоги до точності та швидкодії, алгоритм не є найбільш перспективним напрямом, оскільки сьогодні існують більш ефективні методи об’єктного розпізнавання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fire and Smoke Detection with Computer Vision [Електронний ресурс] // ImageVision.ai. – Режим доступу: https://imagevision.ai/applications/fire-and-smoke-detection.

2. OpenCV українською: вступ та основи [Електронний ресурс] // IT Master. – Режим доступу: https://itmaster.xyz/opencv-intro.

3. Kleinberg D., Tardos É. Algorithms: Design and Applications. – 2016. – 800 p. – (Classics of Computer Science).

4. Klette R. Concise Computer Vision: An Introduction into Theory and Algorithms. – London: Springer, 2014. – 447 p. – (Undergraduate Topics in Computer Science). – ISBN 978-1-4471-6319-0.

5. Efficient face detection algorithm: using Viola Jones method [Електронний ресурс] / Автор не вказаний. – Режим доступу: https://www.codeproject.com/Articles/85113/Efficient-Face-Detection-Algorithm-using-Viola-Jon.

6. Andrew F. Oracle Certified Professional Java SE Programmer study guide: contains around 300 practice questions with solutions. – 2020.

7. Beyerer J., Fernando P., Christian F. Automated Visual Inspection: Theory, Practice and Applications. – Berlin Heidelberg: Springer, 2016. – 798 p.

8. Horstmann C. S. Core Java Volume I: Fundamentals. – 11th ed. – Prentice Hall, 2018. – 928 p.

9. OpenCV: Cascade Classifier [Електронний ресурс]: документація. – Режим доступу: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (дата звернення: 16.05.2025).

10. Yang H. Java Swing Tutorials – Herong's Tutorial Examples. – HerongYang.com, 2018. – 201 p.