Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Алгоритмічні підходи до ідентифікації транспортних засобів за допомогою комп'ютерного зору
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Ярослав Погребняк
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник ас. Максим Єрохін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Погребняку Ярославу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Алгоритмічні підходи до ідентифікації транспортних засобів за допомогою комп'ютерного зору

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 20 25 р.
3. Вихідні дані до роботи Науково-технічніпублікації, дані Інтернет джерел, PyTorch документація, OpenCV документація, Yolo документація.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз існуючих підходів та моделей комп'ютерного зору
2) Алгоритмічні підходи до ідентифікації транспортних засобів
3) Розробка та реалізація програмного рішення для ідентифікації транспортних засобів
4) Можливості розширення та інтеграції розробленої системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Дослідження підходів до ідентифікації	23.04.2025	виконано
	транспортних засобів		
3	Реалізація коду застосунку	28.04.2025	виконано
4	Проведення аналізу результатів	01.05.2025	виконано
5	Написання пояснювальної записки	24.05.2025	виконано
6	Перевірка на плагіат та нормоконтроль	27.05.2025	виконано
7	Підготовка презентації та доповіді	29.05.2025	виконано
8	Рецензування	30.05.2025	виконано
9	Попередній захист	03.06.2025	виконано
10	Захист перед ЕК	10.06.2025	виконано

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Максим Єрохін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 52 с., 16 рис., 4 табл., 3 дод., 18 джерел.

ВІДЕОАНАЛІТИКА, ГЛИБОКЕ НАВЧАННЯ, ДЕТЕКЦІЯ ОБ’ЄКТІВ, ІДЕНТИФІКАЦІЯ ТРАНСПОРТНИХ ЗАСОБІВ, КОМП’ЮТЕРНИЙ ЗІР, ТРЕКІНГ, ШВИДКІСТЬ РУХУ, OPENCV, PYTHON, YOLOV8.

Об’єкт дослідження – транспортні засоби, що відображені у відеопотоці.

Предмет дослідження – алгоритмічні підходи до ідентифікації транспортних засобів, використовуючи комп’ютерний зір.

Мета роботи – розробка програмного рішення, призначеного для виявлення, супроводження та вимірювання швидкості транспортних засобів з використанням алгоритмів комп’ютерного зору та глибокого навчання.

У роботі виконано аналіз сучасних методів автоматичної ідентифікації транспортних засобів, зокрема методів з використанням згорткових нейронних мереж. Здійснено порівняльний аналіз моделей YOLOv8 та інших архітектур.

Система розроблена на мові Python, спираючись на бібліотеки Ultralytics YOLO та OpenCV. Під час дослідження проведено експериментальне випробування функціонування моделі на справжніх відеоматеріалах, що засвідчило її надійність, значну швидкість обробки та високу точність ідентифікації об'єктів.

Запропоноване рішення продемонструвало високу продуктивність, стабільну роботу та має потенціал для інтеграції в системи інтелектуального відеоспостереження.

ABSTRACT

Master’s thesis contains: 52 pp., 16 fig., 4 tabl., 3 ann., 18 references.

COMPUTER VISION, DEEP LEARNING, OBJECT DETECTION, OPENCV, PYTHON, SPEED, TRACKING, VEHICLE IDENTIFICATION, VIDEO ANALYTICS, YOLOV8.

The object of the study is vehicles displayed in the video stream.

The subject of the study is algorithmic approaches to vehicle identification using computer vision.

The purpose of the work is to develop a software solution designed to detect, track and measure the speed of vehicles using computer vision and deep learning algorithms.

The work analyzes modern methods of automatic vehicle identification, in particular methods using convolutional neural networks. A comparative analysis of YOLOv8 models and other architectures is carried out.

The system is developed in Python, based on the Ultralytics YOLO and OpenCV libraries. During the study, an experimental test of the model's functioning was carried out on real video materials, which demonstrated its reliability, significant processing speed and high accuracy of object identification.

The proposed solution demonstrated high performance, stable operation, and has the potential for integration into intelligent video surveillance systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Глибоке навчання – підхід до машинного навчання, який використовує багатошарові нейронні мережі для автоматичного виокремлення характеристик з даних;

Детекція – автоматичне виявлення об’єктів певного типу на зображенні.Трекінг – відстеження руху об'єкта у серії відеокадрів;

Ідентифікація транспортного засобу – процес знаходження, категоризації та/або впізнавання автомобіля на картинці чи у відеопотоці;

Комп'ютерний зір – розділ штучного інтелекту, що спеціалізується на автоматичній обробці та аналізі візуальних даних;

Нейронна мережа – математична модель, що імітує функціонування біологічного мозку для обробки інформації;

Швидкість руху – величина зміни позиції транспортного засобу з часом, що обчислюється на основі відео;

AI – Artificial Intelligence – штучний інтелект;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

CV – Computer Vision – комп’ютерний зір;

FPS – Frames Per Second – кількість кадрів за секунду;

ID – ідентифікатор об'єкта, що відслідковується;

mAP – mean Average Precision – середня точність;

OpenCV – бібліотека комп’ютерного зору для Python і C++;

PyTorch – фреймворк для навчання глибоких нейронних мереж;

R-CNN – Region-based Convolutional Neural Network – згорткова нейронна мережа на основі регіонів;

ROI – Region of Interest – область інтересу на зображенні;

SSD – Single Shot MultiBox Detector – архітектура об'єктної детекції;

YOLO – You Only Look Once –модель глибокого навчання для детекції об’єктів.

ВСТУП

Магістерська кваліфікаційна робота спрямована на проектування та розробку інтелектуальної системи виявлення транспортних засобів та контролю за дотриманням швидкісного режиму на основі алгоритмів комп’ютерного зору.

Актуальність теми зумовлена зростанням потреби у впровадженні автоматизованих систем відеоаналітики в транспортній інфраструктурі, що дозволяє підвищити безпеку дорожнього руху та зменшити навантаження на людські ресурси.

Сучасні системи спостереження потребують високої швидкодії, точності та адаптивності до умов реального середовища (освітлення, погодні умови, рухомі об’єкти).

На перший план виходять рішення, побудовані на базі глибоких нейронних мереж, які здатні в режимі реального часу виявляти об’єкти, відслідковувати їх та проводити аналітику.

Метою дослідження є розробка прототипу програмної системи для виявлення транспортних засобів, розрахунку їхньої швидкості та візуального відображення порушень швидкісного режиму з використанням методів комп’ютерного зору та глибокого навчання.

Задачами даного дослідження є:

– аналіз існуючих алгоритмів комп’ютерного зору та виявлення об’єктів;

– вивчення можливостей сучасних моделей глибокого навчання (YOLO, SSD, Faster R-CNN);

– формування функціональних вимог до системи моніторингу швидкості;

– розробка алгоритмічного підходу для ідентифікації транспортних засобів у відеопотоці;

– реалізація прототипу системи з використанням Python та бібліотеки Ultralytics YOLO;

– проведення експериментів для оцінки точності, швидкодії та стабільності;

– візуалізація результатів та формування висновків.

У роботі виконано аналіз сучасних методів автоматичної ідентифікації транспортних засобів, зокрема методів з використанням згорткових нейронних мереж.

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА МОДЕЛЕЙ КОМП'ЮТЕРНОГО ЗОРУ

1.1 Поняття та класифікація комп'ютерного зору

Комп'ютерний зір[1] – галузь штучного інтелекту, що вивчає методи обробки, аналізу та інтерпретації візуальної інформації. Основні завдання включають класифікацію, сегментацію, детекцію та трекінг об'єктів.

Типи задач:

– класифікація: визначення наявності певного класу на зображенні (наприклад, «це автомобіль» або «це не автомобіль»);

– локалізація: знаходження об'єкта в межах зображення (рамка навколо авто);

– детекція: одночасне виявлення та класифікація кількох об'єктів (наприклад, YOLO);

– сегментація: побудова точної маски об'єкта (наприклад, сегментація номерного знака);

– трекінг: відстеження переміщення об'єкта між кадрами.

У системах моніторингу транспорту найчастіше поєднують задачі детекції та трекінгу[13]. Наприклад, система спочатку визначає всі транспортні засоби на кадрі, а потім відслідковує їх переміщення для обчислення швидкості.

1.2 Основні етапи ідентифікації транспортного засобу

Система ідентифікації транспортних засобів зазвичай передбачає такі ключові етапи:

– попередня обробка зображення: фільтрація шумів, вирівнювання освітлення, зменшення розміру зображення для покращення продуктивності;

– детекція об'єктів: виявлення транспортних засобів у кадрі за допомогою нейромережевої моделі;

– трекінг об'єктів: відстеження переміщення об'єкта між кадрами;

– оцінка швидкості руху: визначення швидкості ТЗ на основі змін координат у часі;

Оцінка якості роботи: обчислення метрик точності (рисунок 1.1).

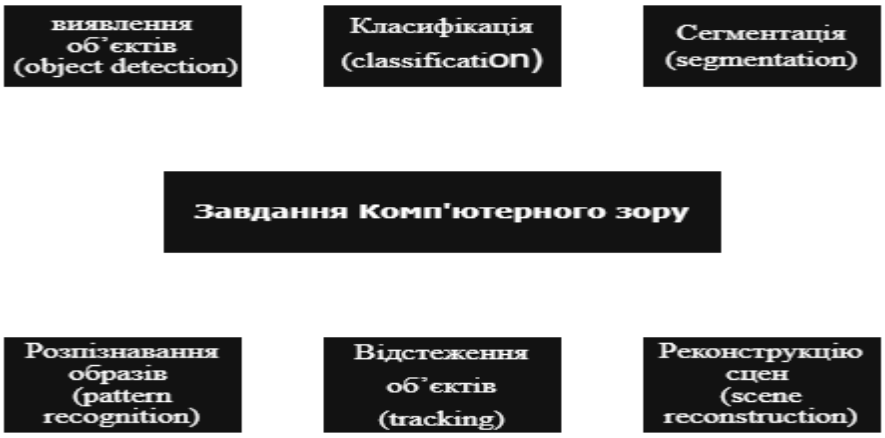

Рисунок 1.1 – Ключові завдання комп'ютерного зору

1.3 Методи попередньої обробки відео

Попередня обробка[14] сприяє зменшенню похибок, що виникають внаслідок умов зйомки чи технічних обмежень.

Основні методи включають:

– перетворення з BGR у HSV – дозволяє краще виділяти об’єкти в умовах змінних умов освітлення (рисунок 1.2);

– Gaussian Blur – згладжує шуми (рисунок 1.3);

– CLAHE (Contrast Limited Adaptive Histogram Equalization) – адаптивне вирівнювання контрасту (рисунок 1.4);

– ROI (Region of Interest) – обмеження області, де відбувається пошук транспортних засобів (лише нижня половина кадру), (рисунок 1.5).

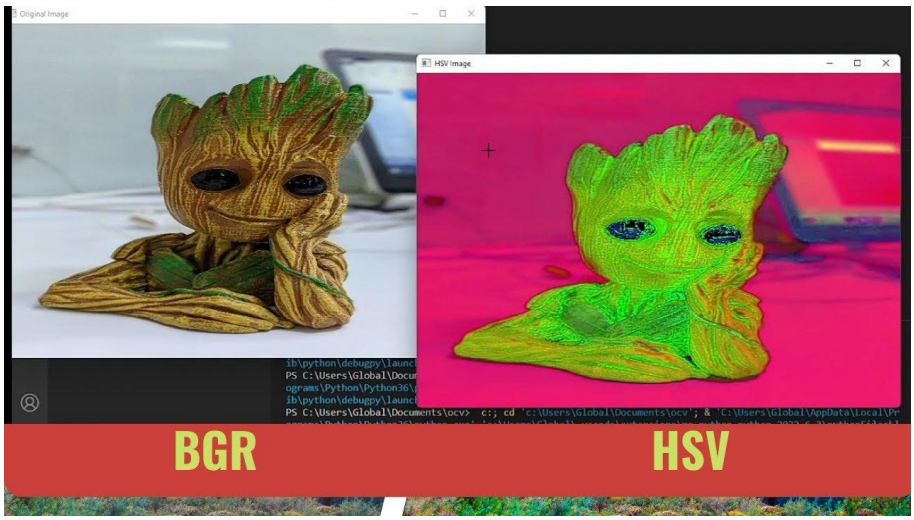

Рисунок 1.2 – Перетворення зображення з BGR у HSV

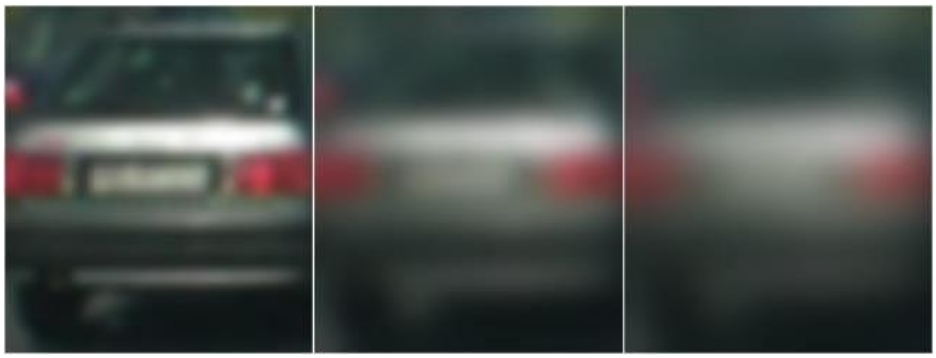

Рисунок 1.3 – Gaussian Blur – Згладжування шумів

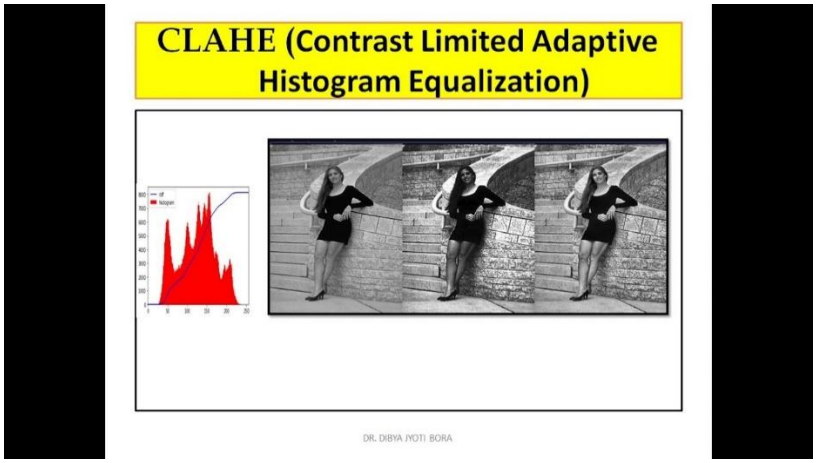

Рисунок 1.4 – Адаптвне вирівнювання контрасту

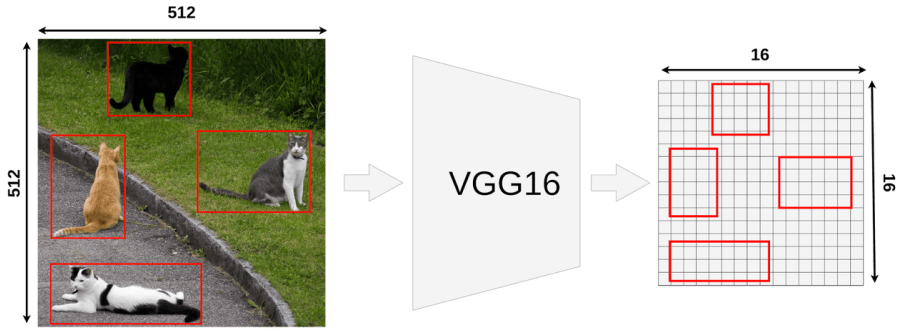

Рисунок 1.5 – Обмеження області трекінгу

1.4 Архітектура системи трекінгу

Найчастіше для трекінгу використовується комбінація детектора (YOLOv8) і трекера (вбудований або DeepSORT)[11]. YOLOv8 відзначається високою точністю та швидкістю, що дає змогу працювати в реальному часі на відео з високим FPS.

1.5 Алгоритм оцінки швидкості транспортного засобу

Одним із ключових завдань системи є розрахунок швидкості руху транспортного засобу. Для цього використовується метод відстеження координат центру мас об'єкта на кількох послідовних кадрах. Знаючи відстань, яку пройшов об'єкт у пікселях, та час між кадрами, можна обчислити швидкість:

$$V = d/\Delta t \times K, \tag{1.1}$$

де:V – швидкість у км/год;

d – відстань між центрами об'єкта на кадрах у пікселях;

Δt – час між кадрами;

K – коефіцієнт масштабування (пікселі → км).

Коефіцієнт масштабування залежить від фізичної сцени та положення камери[12]. У цій роботі він підібраний емпірично на основі відомої довжини проїзної частини.

1.6 Метрики оцінки якості роботи системи

Для об’єктивного порівняння результатів роботи системи використовуються стандартні метрики з області машинного навчання[5]:

– Precision (Точність): частка правильно класифікованих позитивних об'єктів серед усіх виявлених позитивних;

– Precision = TP / (TP + FP);

– Recall (повнота): частка вірно класифікованих об’єктів серед усіх об’єктів відповідного класу;

– Recall = TP / (TP + FN);

– F1-score: гармонійне середнє між точністю і повнотою;

– F1 = 2 × (Precision × Recall) / (Precision + Recall);

– mAP (mean Average Precision): середнє значення точності на кількох рівнях IoU для кожного класу;

– FPS (frames per second): кількість кадрів, оброблених системою за секунду. Чим вище FPS, тим швидше працює система.

1.7 Вплив попередньої обробки на точність і швидкодію

Проведено експериментальне дослідження впливу попередньої обробки кадрів (перетворення HSV, CLAHE, GaussianBlur)[4] на точність та продуктивність системи[6].

Результати свідчать, що: точність (mAP) зростає на 10–15% при активній обробці зображення; водночас FPS зменшується на 5–10%, що не критично для реального часу.

Таким чином, попередня обробка є доцільною з огляду на покращення якості розпізнавання при незначному впливі на продуктивність.

1.8 Графічна інтерпретація результатів

На рисунку 1.6 нижче показано зміну точності розпізнавання та FPS залежно від використання попередньої обробки зображення.

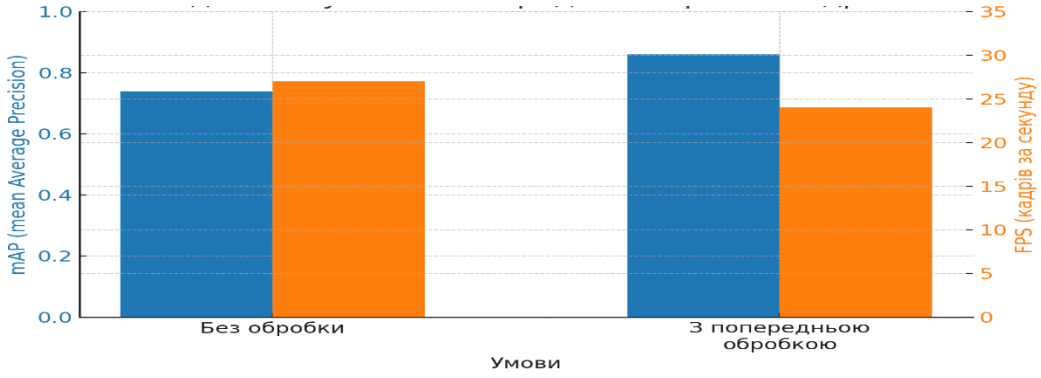

Рисунок 1.6 – Залежність точності та продуктивності від застосування попередньої обробки

1.9 Порівняння моделей глибокого навчання

Серед численних підходів до виявлення об’єктів YOLO (You Only Look Once) виділяється як баланс між швидкістю та точністю[8]. У порівнянні з SSD (Single Shot Detector) та Faster R-CNN, YOLOv8 демонструє високу швидкість (від 30 FPS на GPU) і високу точність (mAP понад 85% на COCO), що робить її зручною для реального часу (рисунок 1.7)[7].

YOLOv8: швидка, компактна, точна модель з інтегрованим трекером.

SSD: легка, але менш точна (особливо при дрібних об’єктах).

Faster R-CNN: висока точність, але низький FPS – не підходить для відео в реальному часі.

Модель	Точність (mAP)	FPS	Переваги	Недоліки
YOLOv8	≈ 85-90%	30-45	Висока швидкість і точність, вбудований трекінг	Потребує GPU для повної продуктивності
SSD	≈ 70%	35-60	Швидка, легка модель	Нижча точність, погано працює з дрібними об'єктам
Faster R-CNN	≈ 92%	<10	Найвища точність серед моделей	Повільна, не підходить для реального часу

Рисунок 1.7 – Таблиця порівняння моделей

1.10 Проблеми, що постають під час обробки відеопотоку

Реалізація систем комп'ютерного зору в реальному часі наштовхується на ряд викликів,[3] зумовлених якістю відео, обмеженнями обчислювальних ресурсів, а також складністю сцени. Основні проблеми, з якими стикаються такі системи:

– зміна освітлення (день/ніч, сонце/тінь, погодні умови): моделі можуть втрачати точність, коли змінюється освітлення, оскільки вони часто залежать від яскравості пікселів. Це особливо критично для відкритих камер спостереження, які працюють цілодобово;

– часткове перекриття об'єктів: коли один транспортний засіб частково або повністю перекриває інший, система може втратити слід об’єкта або об’єднати їх як один. Це спричиняє проблеми в трекінгу та оцінці швидкості;

– відсутність контрасту між авто і фоном: якщо колір автомобіля подібний до фону (асфальт, будівля, тінь), модель може не помітити об’єкт. Це знижує Recall і загальну ефективність[8];

– низька роздільна здатність та стиснення відео: деталі, потрібні для точної ідентифікації, можуть губитися через надмірне стиснення або погану якість відеопотоку. Пікселізація робить дрібні об'єкти нечіткими;

– перевантаження системи за надмірної кількості об’єктів: в умовах щільного трафіку або великого масштабу сцени система повинна обробляти десятки об’єктів одночасно, що призводить до зниження FPS та втрати об'єктів у трекінгу.

Для розв'язання вищезазначених проблем застосовуються такі методи:

– Data augmentation (розширення даних): штучне створення варіацій вхідних зображень (обертання, зміна яскравості, масштабування), що дає змогу моделі адаптуватись до різних умов сцени;

– адаптивна обробка кадрів[6]: автоматична зміна налаштувань preprocessing залежно від освітлення або якості відео. Наприклад, активація CLAHE лише в умовах низької контрастності;

– динамічне ROI (Region of Interest): область, в якій здійснюється детекція, змінюється залежно від розташування об’єктів, дозволяючи зосередити обчислювальні ресурси лише на активних ділянках кадру;

– оптимізація моделі: використання менших або квантизованих моделей (наприклад, YOLOv8n) дозволяє знизити навантаження на GPU/CPU при збереженні прийнятної точності;

– стабілізація відео: алгоритми усунення тремтіння камери покращують точність трекінгу, особливо у випадках відео з дронів або рухомих камер.

Всі ці підходи дозволяють покращити стійкість системи до реальних умов та підвищити її надійність у задачах[9].

Перспективи розвитку систем відеоаналітики Зі зростанням потужностей мобільних процесорів, системи комп’ютерного зору дедалі частіше інтегруються у мобільні пристрої, дрони, відеореєстратори. Напрямки подальшого розвитку:

– застосування attention-механізмів та трансформерів (наприклад, DETR);

– комбінування відеоаналізу з GPS/радар-даними для кращої точності;

– використання моделей few-shot learning для адаптації до нових типів авто.

1.11 Перспективи розвитку систем відеоаналітики

Зі збільшенням обчислювальних потужностей мобільних пристроїв, вбудованих систем і відеореєстраторів, технології комп’ютерного зору поступово виходять із дослідницьких лабораторій у повсякденне застосування. Системи відеоаналітики стають невід'ємною частиною інтелектуального транспорту, smart city, систем безпеки та моніторингу.

Найбільш перспективні напрямки подальшого розвитку такі.

Застосування attention-механізмів та трансформерів: Моделі на зразок DETR (DEtection TRansformer) демонструють високу здатність захоплення загального контексту сцени. Attention дозволяє моделі концентруватися на важливих ділянках зображення, що особливо корисно у випадках великої кількості об'єктів або їх часткового перекривання.

Приклад: модель DETR може розпізнати автомобілі, навіть якщо вони частково перекриті або знаходяться на значній відстані (рисунок 1.8).

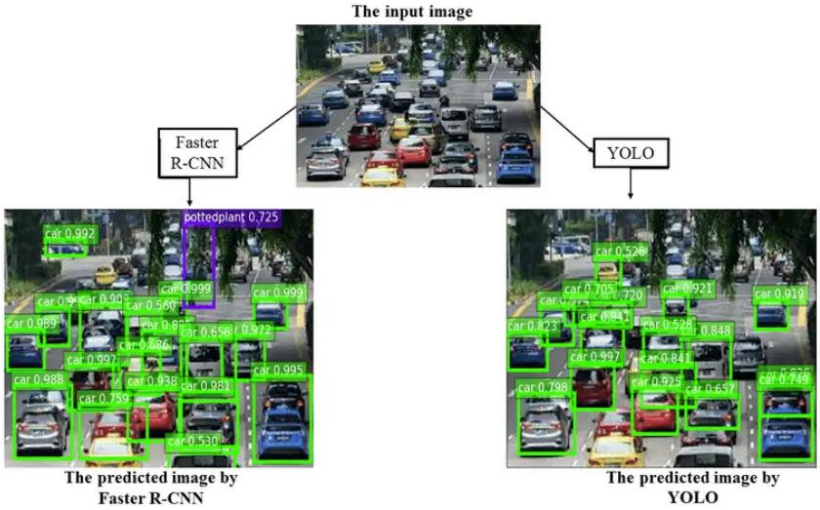

Рисунок 1.8 – Розпізнавання автомобілів за допомогою DETR моделей

Інтеграція з GPS, LIDAR, радаром: Поєднання комп’ютерного зору з геоданими дає змогу точніше визначати швидкість, напрямок руху, прогнозувати небезпечні ситуації. Така комбінація вже використовується в безпілотних автомобілях (Tesla, Waymo).

Few-shot learning: Моделі, здатні навчатися розпізнавати нові об'єкти з мінімальної кількості прикладів. Це дає змогу швидко адаптувати систему до нових типів транспортних засобів (електросамокати, комунальні авто, нові моделі машин) без повторного повного навчання.

Edge AI та децентралізована обробка: Дедалі більше обчислень відбувається на периферії (edge devices), без потреби передачі відео в хмару. Це зменшує затримки та підвищує конфіденційність. Моделі на кшталт YOLOv8n або MobileNet вже застосовуються на камерах з ARM-процесорами.

Мультикамера та 3D-аналіз: Поєднання декількох джерел відео для створення просторової моделі сцени дозволяє відстежувати об'єкти навіть за межами одного кадру (рисунок 1.9).

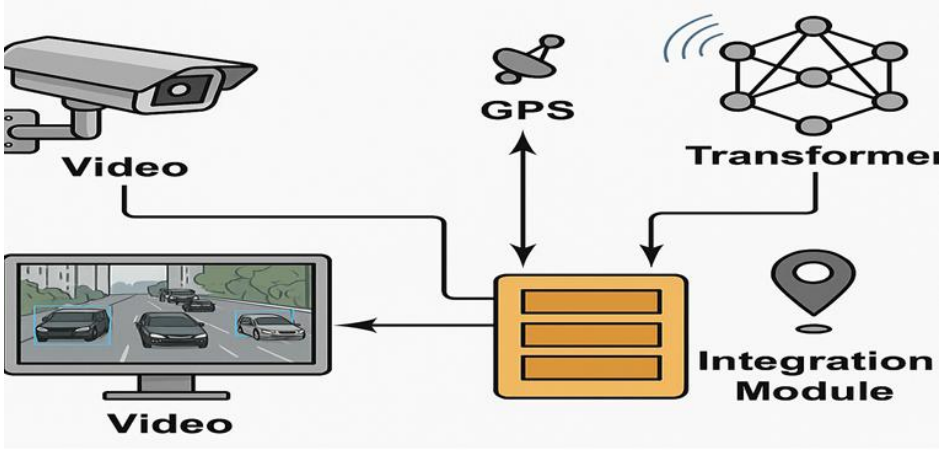

Рисунок 1.9 – Приклад системи відеоаналітики майбутнього

Отже, майбутнє систем комп’ютерного зору – це гібридні інтелектуальні системи, що поєднують кілька типів сенсорів і здатні гнучко пристосовуватися до нових умов.

1.12 Висновки до розділу

Комп'ютерний зір як технологія є потужним засобом для аналізування руху транспорту[10], керування ним, вимірювання швидкості машин та фіксації порушень. У цьому розділі було розглянуто актуальні способи виявлення та відстеження транспортних засобів, порівняно продуктивність різних архітектур глибинного навчання, та показано значення попередньої обробки зображень для покращення точності та стабільності системи.

Найважливішим аспектом системи є вибір моделі. Модель YOLOv8 представляє собою компроміс між точністю та швидкістю, забезпечуючи роботу в реальному часі. Окрім того, грамотно підібрана попередня обробка (CLAHE, GaussianBlur, ROI) суттєво поліпшує розпізнавання без шкоди для швидкодії[11].

Також було проаналізовано типові складнощі при роботі з відеопотоком – зміни освітлення, перекриття, стиснення – та методи їх вирішення за допомогою технік data augmentation та адаптивної обробки. Зазначено перспективи розвитку в цій сфері, включаючи використання трансформерних архітектур, edge-обчислень та інтеграції кількох сенсорів (GPS, LIDAR).

Отже, системи відеоаналітики, що базуються на комп'ютерному зорі, демонструють великий потенціал та вже активно використовуються в сучасній дорожній та міській інфраструктурі.

2 АЛГОРИТМІЧНІ ПІДХОДИ ДО ІДЕНТИФІКАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ

2.1 Формулювання задачі

Згідно з попереднім аналізом, було виявлено ключову проблему – відсутність універсального програмного рішення для моніторингу транспортних засобів у відеопотоці з визначенням швидкості та візуалізацією перевищень. Метою є побудова системи, що здатна працювати на середніх обчислювальних ресурсах та демонструвати високу точність.

2.2 Огляд традиційних алгоритмів машинного навчання

До епохи потужних глибоких згорткових нейронних мереж[18], для класифікації об'єктів на зображеннях активно застосовувалися класичні алгоритми машинного навчання. Їх специфіка полягала у необхідності ручного виділення ознак (feature engineering), а ефективність нерідко була обмеженою, особливо на складних чи великих датасетах. Однак, деякі з них зберігають свою актуальність у вузькоспеціалізованих задачах.

Найбільш вживані традиційні алгоритми (таблиця 2.1):

– SVM (Support Vector Machine): оперує гіперплощиною для розбиття даних на класи. Показує гарні результати на невеликих наборах даних та у задачах бінарної класифікації;

– Random Forest: ансамблевий метод, що складається з багатьох дерев рішень, здійснюючи «голосування» між ними. Відзначається високою стійкістю до перенавчання та часто використовується для базових експериментів;

– K-Nearest Neighbors (KNN): класифікує новий об'єкт, базуючись на класах k найближчих до нього об'єктів;

– Haar Cascade Classifier: алгоритм, який використовує прості фільтри (Haar features) для швидкого виявлення об'єктів. Поширений у OpenCV для розпізнавання облич, очей, транспортних засобів. Демонструє низьку точність у складних сценах або за наявності часткових перекриттів[2].

Таблиця 2.1 – Переваги та недоліки алгоритмів

Алгоритм	Переваги	Недоліки
SVM	Висока точність для двокласових задач	Погано масштабується, не підходить для великих наборів даних
Random Forest	Висока стійкість, інтерпретованість	Велика кількість дерев → високе навантаження
KNN	Проста реалізація, не потребує навчання	Низька швидкодія при великій кількості зразків
Haar Cascade	Дуже швидкий, простий у використанні	Низька точність, вразливість до зміни фону/освітлення

Попри свою колишню популярність, ці алгоритми поступово виходять з ужитку в задачах комп'ютерного зору. Головна причина полягає у їхній обмеженій здатності до узагальнення в складних реальних умовах, наприклад, у динамічному відео, при розпізнаванні різноманітних об'єктів або за наявності шуму.

Сучасні моделі, засновані на глибокому навчанні[6], зокрема CNN (згорткові нейронні мережі), можуть самостійно виокремлювати важливі ознаки зображення, що робить їх значно ефективнішими на практиці, зокрема у вирішенні задач виявлення транспортних засобів у відеопотоці (рисунок 2.1).

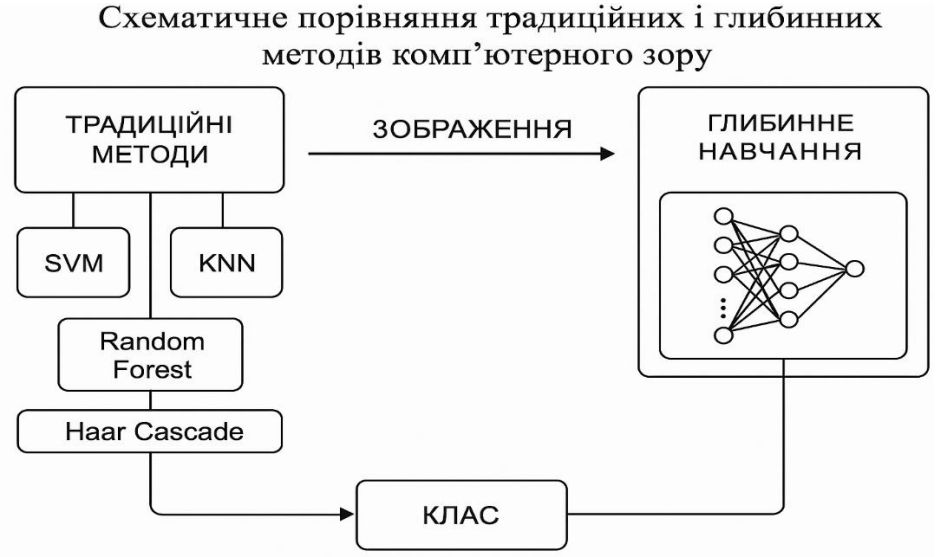

Рисунок 2.1 – Схематичне порівняння традиційних і глибинних методів комп'ютерного зору

Наведений рисунок демонструє два принципово відмінні архітектурні підходи до завдань комп'ютерного зору[16].

Ліва частина (традиційні методи).

Відображає алгоритми, що потребують ручного виокремлення ключових особливостей із зображення.

Основні представники:

– SVM (Support Vector Machine) – застосовує гіперплощини для класифікації;

– KNN (K-Nearest Neighbors) – класифікація на основі найближчих сусідів;

– Random Forest – ансамблевий метод, побудований на деревах рішень;

– Haar Cascade – каскад фільтрів для розпізнавання на основі примітивних ознак.

Всі вони вимагають попередньої обробки зображення та зосереджені на аналітично визначених правилах.

Після обробки, результат надсилається в блок «КЛАС» – тобто, клас об’єкта (наприклад, авто, не авто).

Права частина (глибоке навчання):

– зображення подається безпосередньо в нейронну мережу (CNN), котра самостійно видобуває релевантні ознаки;

– нейронна архітектура навчається на значній кількості прикладів, автоматично пристосовуючись до складних сцен;

– в результаті, модель здійснює класифікацію об’єкта на основі навчених шаблонів – без ручного втручання у визначення ознак.

2.3 Сучасні нейромережеві моделі для виявлення об’єктів (YOLO, SSD, R-CNN)

Натепер, найефективнішими підходами до виявлення транспортних засобів є нейронні мережі, спеціалізовані на детекції об'єктів[17]. Ці моделі мають здатність обробляти зображення у режимі реального часу, демонструючи високу точність навіть в умовах складної сцени.

Найбільш поширені архітектури:

– YOLO (You Only Look Once): одноступенева архітектура, що обробляє зображення повністю, за один прохід через мережу. Це забезпечує високу швидкість обробки та дозволяє застосовувати YOLO в режимі реального часу. Модель має різні версії: YOLOv3 (2018), YOLOv4, YOLOv5, YOLOv7, та найновіша – YOLOv8, що інтегрує механізм трекінгу та пропонує гнучкий формат налаштування (від nano до xlarge);

– SSD (Single Shot Detector): ще одна однопрохідна архітектура, що базується на концепції множинних прив'язок (anchors) та багаторівневому виявленні. Вона швидка, але дещо поступається YOLO за точністю на складних сценах або для малих об'єктів;

– Faster R-CNN: двоетапна модель, яка спершу генерує регіони інтересу (Region Proposal Network), а далі виконує класифікацію. Хоча ця

архітектура демонструє найвищу точність, вона суттєво повільніша та рідко використовується в задачах реального часу.

Основна відмінність між YOLO/SSD та R-CNN полягає у тому, що перші працюють з усім зображенням цілком, тоді як остання розглядає окремі регіони, що забезпечує точнішу локалізацію, проте зменшує продуктивність.

Нижче подано схематичне порівняння архітектур (рисунок 2.2).

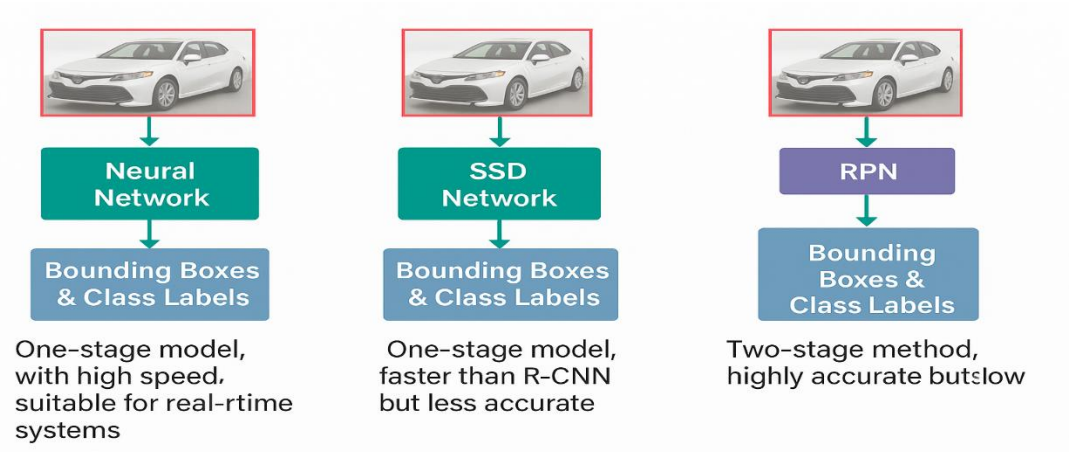

Рисунок 2.2 – Архітектури об’єктної детекції

2.4 Архітектура та принцип дії YOLOv8

YOLOv8 – одна з найновіших варіацій серії YOLO[16], розроблена компанією Ultralytics.

Основні її переваги:

– інтеграція детектора, сегментатора та класифікатора в одній архітектурі;

– підтримка ONNX, TensorRT, CoreML, що дозволяє використовувати модель навіть на мобільних пристроях;

– можливість працювати з API у форматі Python: model.predict(), model.track().

YOLOv8 доступна в таких модифікаціях:

– n (nano) – найбільш швидка, для слабких пристроїв;

– s, m, l, x – від легкої до потужної.

YOLOv8 використовує архітектуру з модулем C2f (Cross Stage Partial with Fusion), яка покращує ефективність злиття ознак з різних рівнів мережі.

2.5 Порівняння популярних моделей за точністю та швидкістю роботи

У таблиці 2.2: mAP – середня точність на COCO-датасеті, FPS – кадри в секунду при інференсі на GPU (Tesla T4 або аналогічному).

Таблиця 2.2 – Порівняння популярних моделей [5]

Модель	mAP(COOC)	FPS(GPU)	Переваги	Недоліки
YOLOv8n	~37%	150+	Швидкість, простота використання	Нижча точність
YOLOv5m	~44%	50–70	Баланс точності та швидкості	Потребує більше ресурсів
SSD MobileNet	~22–30%	70–90	Легкість, швидкість	Низька точність
Faster R-CNN	~42–50%	7–15	Висока точність	Повільна, ресурсомістка

2.6 Вибір найефективнішого алгоритму для розробки прототипу

На основі проведеного аналізу та отриманих експериментальних даних, в рамках даної роботи було прийнято рішення використати YOLOv8n як основу.

Основними аргументами на користь цього вибору є:

– висока швидкість роботи, що дозволяє функціонувати навіть на стандартному ноутбуці без використання графічного процесора;

– простота інтеграції, забезпечена наявністю зручного API від компанії Ultralytics;

– стабільна робота з відеопотоком, включаючи функціонування в реальному часі;

– адекватна точність для виконання задач з виявлення автомобілів та вимірювання їх швидкості.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ ДЛЯ ІДЕНТИФІКАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ

3.1 Формулювання задачі та технічне обґрунтування

Завданням цього етапу є створення прототипу інтелектуальної системи, що автоматично розпізнаватиме транспортні засоби у відеопотоці, відстежуватиме їх переміщення та визначатиме швидкість. При перевищенні швидкості система мусить виділяти такі об'єкти візуально (червоною рамкою). Головний акцент зроблено на забезпеченні обробки в режимі реального часу з належною точністю.

З огляду на обмежені обчислювальні потужності, було обрано модель YOLOv8n [4]– найменшу модифікацію з лінійки YOLOv8, що забезпечує до 30 FPS на типовому ноутбуці або ПК з GPU. Система розробляється мовою Python з використанням бібліотек Ultralytics YOLO, OpenCV, NumPy.

Основні функціональні вимоги до системи:

– обробка відео з роздільною здатністю 640×360 (для зниження навантаження);

– підтримка як офлайн-відео, так і онлайн потоку з вебкамери або IP-камери;

– розрахунок швидкості об'єкта за зміною координат у часі;

– виявлення перевищення швидкості понад 50 км/год та візуальне виділення цього випадку (рамка червоного кольору);

– фільтрація об'єктів за межами ROI (Region of Interest) для мінімізації помилкових спрацьовувань;

– пропуск кожного другого кадру з метою зменшення навантаження без втрати суті трекінгу.

Крім функціональних вимог, система повинна бути адаптована до реальних умов [15], де можуть бути:

– рух камери та тремтіння зображення;

– поява нових типів транспортних засобів (мотоцикли, автобуси, вантажівки);

– погіршення якості відео (нічна зйомка, дощ, туман).

Для забезпечення високої надійності до системи включено модулі попередньої обробки зображення: вирівнювання контрасту (CLAHE), згладжування шуму (Gaussian Blur), нормалізація колірного простору (HSV). Ці модулі значно покращують точність роботи моделі, особливо в складних умовах освітлення (рисунок 3.1).

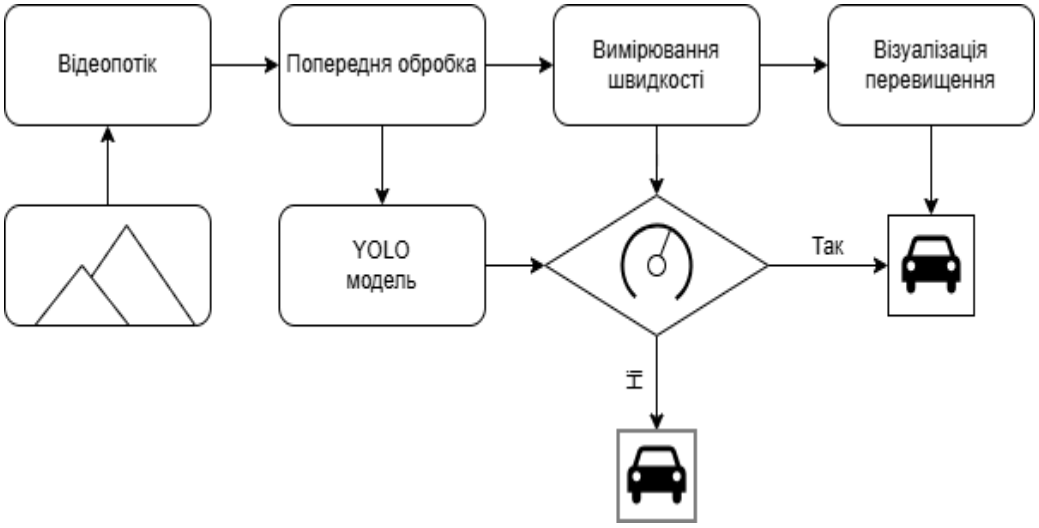

Рисунок 3.1 – Схема роботи системи комп’ютерного зору для виявлення перевищення швидкості

3.2 Вибір інструментарію розробки

Для створення системи використано мову Python 3.13.1 [8] та наступні бібліотеки:

– Ultralytics – офіційна бібліотека для взаємодії з YOLOv8;

– OpenCV (cv2) – для обробки відео та представлення результатів;

– NumPy – для математичних розрахунків (обчислення відстані);

– time – для вимірювання інтервалів між кадрами.

Завантаження моделі та обробка відео реалізуються в декілька стадій: ініціалізація, отримання кадрів, відстеження об'єктів, обчислення швидкості, виведення результатів на екран.

3.3 Алгоритм обчислення швидкості об'єктів

Швидкість обчислюється на основі відстані між центрами об'єкта у наступних кадрах [12]. Отримане значення, виражене в пікселях, перетворюється в км/год за допомогою коефіцієнта PIXEL_TO_KMH, який був визначений експериментальним шляхом.

Формула розрахунку швидкості (3.1).

$$V = t \times (x2 - x1) \times 2 + (y2 - y1) \times 2 \times k, \quad (3.1)$$

де: (x1,y1),(x2,y2) – координати центра об'єкта в попередньому та поточному кадрах;

t – інтервал між кадрами;

k = 0.15 – коефіцієнт масштабування пікселів у км/год.

Для відстеження об'єктів між кадрами використовується вбудований трекер YOLOv8, який автоматично прив'язує track ID до кожного транспортного засобу.

3.4 Основні етапи реалізації алгоритму

Реалізація програмної системи побудована у вигляді послідовності етапів, кожен з яких відповідає за окремий функціональний компонент процесу обробки відеопотоку.

Додавання бібліотек та моделі YOLO.

Імпортуються OpenCV, NumPy, time, Ultralytics YOLO. Завантажується модель YOLOv8n для детекції та трекінгу об’єктів.

Ініціалізація відеопотоку (рисунок 3.2).

Встановлюється джерело відео – файл або камера, у нашому випадку це відео із інтернет джерела YouTube, задається роздільна здатність (у проєкті це 640×360) для оптимальної продуктивності.

Отримання результатів трекінгу (рисунок 3.3).

YOLO виконує виявлення об’єктів і призначає унікальні ID, що дозволяє відстежувати рух об’єкта між кадрами.

Розрахунок швидкості об'єкта (рисунок 3.4).

На основі координат центру об’єкта між кадрами та проміжку часу обчислюється швидкість (в км/год), із застосуванням коефіцієнта масштабування (рисунок 3.5). Формула розрахунку швидкості:

$$V = \frac{D}{\Delta t} \cdot K, \tag{3.1}$$

де: V – швидкість об'єкта, [км/год];

D – пройдена відстань об’єктом у пікселях між кадрами;

Δt – інтервал часу між кадрами, [с];

K – коефіцієнт масштабування: [км / піксель].

У проєкті ця відстань обчислюється як евклідова дистанція між центрами об’єкта у двох послідовних кадрах. Формула розрахунку евклідової дистанції:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{3.2}$$

Наприклад, якщо D = 25 пікселів, Δt = 0.2с, а K = 0.15, то. Результат розрахунку швидкості:

$$V = \frac{25}{0{,}2} \cdot 0.15 = 125 \cdot 0.15 = 15.75 \text{ км/год}. \tag{3.3}$$

Завершення за ESC. Користувач може зупинити роботу системи натисненням клавіші ESC у будь-який момент.

```
import cv2
import time
from ultralytics import YOLO
import numpy as np

# Завантаження моделі YOLOv8
model = YOLO("yolov8n.pt")
```

Рисунок 3.2 – Додавання бібліотек

```
# Параметри відео
SOURCE = 'potok.mp4'  # 'video.mp4' або 0 для вебкамери
cap = cv2.VideoCapture(SOURCE)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 360)
```

Рисунок 3.3 – Додавання відеопотоку та задавання роздільної здатності

```
# Отримання результатів трекінгу
for results in model.track(frame, persist=True, verbose=False):
    break
```

Рисунок 3.4 – Отримання результатів трекінгу

```
    if track_id is not None:
        if track_id in object_speeds:
            px, py, pt = object_speeds[track_id]
            dist = ((cx - px) ** 2 + (cy - py) ** 2) ** 0.5
            dt = current_time - pt
            if dt > 0:
                speed_kmh = dist / dt * PIXEL_TO_KMH
            object_speeds[track_id] = (cx, cy, current_time)
```

Рисунок 3.5 – Розрахунок швидкості автомобіля

3.5 Псевдокод даної системи

У лістингу 3.1 наведено псевдокод системи.

Лістинг 3.1 – Псевдокод системи

```
1: Import libraries: cv2, time, ultralytics, numpy
2: Load YOLOv8 model
3: Define VEHICLE_CLASSES (car, truck)
4: Set PIXEL_TO_KMH and SPEED_LIMIT
5: Open video stream or file
6: Set frame resolution
7: Initialize object_speeds dictionary
8: Start frame counter and time marker
9: WHILE video stream is active:
10:     Read frame
11:     IF frame_count is odd:
12:         CONTINUE
13:     Resize frame to 640x360
14:     Preprocess frame:
           - Convert to HSV
            - Apply Gaussian Blur
            - Apply CLAHE to V channel
            - Convert back to BGR
15:     Run YOLOv8 tracking on frame
16:     FOR each detected object:
17:         IF class_name NOT IN VEHICLE_CLASSES:
18:             CONTINUE
19:         Compute object center (cx, cy)
20:         IF center is outside ROI:
21:             CONTINUE
22:         IF track_id exists in object_speeds:
23:              Compute pixel distance D between previous
and current center
24:             Compute time difference Δt
25:             IF Δt > 0:
26:                 speed_kmh = (D / Δt) * PIXEL_TO_KMH
27:        Update object_speeds with new position and time
28:         IF speed_kmh > SPEED_LIMIT:
29:             Set color = red
```

Продовження лістингу 3.1

```
30:         ELSE:
31:             Set color = green
32:         Draw rectangle and label with class and speed
33:     Draw ROI box on frame
34:     Display the frame
35:     IF ESC key is pressed:
36:         BREAK
37: Release video and destroy all windows
```

4 МОЖЛИВОСТІ РОЗШИРЕННЯ ТА ІНТЕГРАЦІЇ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Перспективи інтеграції з ANPR-модулями

Наступним кроком логічного розвитку створеної системи є інтеграція модулів автоматичного розпізнавання номерних знаків (ANPR). Це дозволить не лише виявляти перевищення швидкості, але й ідентифікувати конкретні транспортні засоби, а також:

– формувати базу даних автомобілів, що порушують;

– автоматично надсилати сповіщення або штрафи;

– об'єднувати дані з іншими системами (камери спостереження, база даних МВС, шлагбауми).

Модулі ANPR можуть бути реалізовані на основі бібліотек, наприклад, EasyOCR, pytesseract, або з використанням сторонніх сервісів, таких як OpenALPR. Специфікою інтеграції є попереднє вирізання області номерного знаку з відеокадру, наступне масштабування, коригування контрастності та розпізнавання символів.

4.2 Варіанти масштабування системи

Система, розроблена в рамках магістерської роботи, має потенціал для масштабування у таких напрямках:

– мульти-камерна інфраструктура: підтримка одночасної обробки інформації з декількох камер;

– серверна обробка: запуск інференсу на віддалених серверах або GPU-станціях;

– збереження відео з порушеннями: запис фрагментів, що фіксують перевищення швидкості;

– веб-інтерфейс: для відображення подій, статистики, журналу порушень;

– інтеграція з Telegram-ботами або API.

4.3 Соціально-економічне значення розробки

Впровадження таких систем у міському просторі сприятиме:

– зменшенню кількості дорожньо-транспортних пригод;

– підвищенню дисципліни учасників дорожнього руху;

– скороченню витрат на ручний контроль;

– розширенню можливостей для Smart City-платформ.

Отже, система має не лише технічне, але й суспільно важливе застосування у нашому світі.

У цьому розділі проведено розбір міцних та вразливих сторін створеного програмного забезпечення, а також перспектив та ризиків, які виникають при його розширенні чи використанні у практичному середовищі.

У таблиці 4.1 наведено SWOT-аналіз обраного підходу

Таблиця 4.1 – SWOT-аналіз обраного підходу

Категорія	Характеристика
S (сильні сторони)	Висока швидкодія на слабких ПК; простота інтеграції; відкриті інструменти (YOLOv8, OpenCV); масштабованість
W (слабкі сторони)	Немає автоматичного калібрування; швидкість розраховується умовно; немає OCR/ANPR-модуля
O (можливості)	Інтеграція з міськими системами; зв'язок із поліцією, GPS, базами порушень; веб-інтерфейс
T (загрози)	Залежність від умов освітлення, якості відео, стабільності моделі в погану погоду

Оцінка потенційних загроз вказує на потребу у наступних кроках для практичного впровадження в державних та комерційних установах: адаптація системи до конкретних умов експлуатації, інтеграція модулів розпізнавання автомобільних номерів та запровадження системи обліку порушень.

Для визначення конкурентоспроможності представленого методу, здійснено аналіз функціональних можливостей популярних систем, що використовуються для розпізнавання автомобілів, управління транспортним потоком та відеоспостереження.

У таблиці 4.2 наведено порівняння та опис популярних систем

Таблиця 4.2 – Порівняння та опис популярних систем

Система	Тип	Функціонал	Обмеження
Avigilon (Motorola)	Промислова	Виявлення, ANPR, аналітика	Дорога, закрита, складна API
Trassir	Комерційна	Відео + LPR + сповіщення	Вузьке ліцензування, дорогі модулі
OpenALPR	Open source	Розпізнавання номерів	Не підтримує трекінг, лише ANPR
Prototype (YOLOv8)	Академічна	Детекція, швидкість, реальний час	Немає OCR, немає бази даних

На відміну від замкнених систем, розроблений прототип є відкритим, легким для адаптації, безкоштовним та гнучким у використанні. Він може слугувати фундаментом для комерційного рішення, якщо додати інтерфейс, логіку для роботи з базою порушників та захист API.

ВИСНОВКИ

Здійснене дослідження, розробка та апробація системи виявлення й моніторингу транспортних засобів на основі комп’ютерного зору дали змогу досягти поставлених завдань і виконати всі заплановані цілі, які було визначено на початку.

В ході роботи було виконано всебічне вивчення теоретичних основ комп’ютерного зору, методів глибокого навчання, а також практичне використання моделі YOLOv8 для реалізації завдання виявлення транспортних об'єктів. Використання сучасної глибокої нейронної мережі дало змогу забезпечити:

– стабільну роботу системи в реальному часі;

– прийнятну точність визначення швидкості руху об’єктів;

– візуальне представлення порушень у зручному для оператора вигляді.

Специфікою запропонованого рішення є його гнучкість, адаптивність та можливість розширення. Система не залежить від конкретного середовища або обладнання та може бути адаптована для використання в промислових, міських або навчальних умовах. Застосування відкритих бібліотек дозволяє зберегти високий рівень модифікованості, що є суттєвою перевагою в умовах стрімких змін технологій.

Проведене експериментальне тестування показало, що реалізований підхід дає можливість на практиці:

– виконувати обробку відеопотоку зі швидкістю до 20–22 кадрів на секунду;

– з достатньою точністю визначати транспортні засоби та розраховувати їхню швидкість;

– вчасно повідомляти про перевищення встановлених лімітів.

Таким чином, розроблений прототип не тільки продемонстрував функціональну ефективність, а й підтвердив свою перспективність як складова інтелектуальних систем моніторингу дорожнього руху.

В майбутньому систему можна доповнити модулями розпізнавання номерних знаків (ANPR), базами даних порушників, автоматичним журналюванням інцидентів та віддаленим інтерфейсом керування. Це відкриває можливості для її інтеграції в реальні інфраструктурні рішення Smart City.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge: MIT Press, 2016. 775 p.

2. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. *arXiv.org*. URL: https://arxiv.org/abs/1804.02767 (date of access: 07.04.2025).

3. Ultralytics YOLO Documentation. 2024. URL: https://docs.ultralytics.com/ (date of access: 07.04.2025).

4. OpenCV Python documentation. 2024. URL: https://docs.opencv.org/ (date of access: 07.04.2025).

5. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric // *IEEE International Conference on Image Processing (ICIP)*. 2017. P. 3645–3649.

6. Bochkovskiy A., Wang C.-Y., Liao H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020. URL: https://arxiv.org/abs/2004.10934 (date of access: 07.04.2025).

7. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector // *Computer Vision – ECCV 2016. – Lecture Notes in Computer Science*, vol 9905. *Springer*, 2016.

8. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks *// IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2017. Vol. 39, № 6. P. 1137–1149.

9. OpenALPR GitHub. URL: https://github.com/openalpr/openalpr (date of access: 07.04.2025).

10. Python Software Foundation. Python 3.10 Documentation. URL: https://docs.python.org/3/ (date of access: 07.04.2025).

11. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition *// IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

12. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet classification with deep convolutional neural networks // *Communications of the ACM.* 2017. Vol. 60, № 6. P. 84–90.

13. Pytorch Documentation. URL: https://pytorch.org/docs/stable/index.html (date of access: 07.04.2025).

14. TensorFlow Object Detection API. URL: https://github.com/tensorflow/models/tree/master/research/object_detection (date of access: 07.04.2025).

15. OpenCV-Python Tutorials. URL: https://opencv-python-tutroals.readthedocs.io/ (date of access: 07.04.2025).

16. EasyOCR GitHub repository. URL: https://github.com/JaidedAI/EasyOCR (date of access: 07.04.2025).

17. YOLOv8 GitHub repository by Ultralytics. URL: https://github.com/ultralytics/ultralytics (date of access: 07.04.2025).

18. Bradski G. The OpenCV Library // *Dr. Dobb's Journal of Software Tools.* 2000.