МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ
Кафедра інформаційних технологій

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ПОРУШЕНЬ ПРАВИЛ ДОРОЖНЬОГО РУХУ ЗАСОБАМИ КОМП'ЮТЕРНОГО ЗОРУ

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Дмитра ГЛАДІЯ

Науковий керівник:
Доктор філософії
Олег БАСИСТЮК

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій

_____ **Олег ЗАЧЕК**
(підпис)

Львів
2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ALRP / ANRP	(Automatic License Plate Recognition / Automatic Number Plate Recognition) автоматичне розпізнавання державних номерних знаків транспортних засобів.
BGR	Колірна модель, яка використовується для представлення та кодування цифрових зображень за замовчуванням у бібліотеці OpenCV.
CPU	(Central Processing Unit) центральний процесор комп'ютера
FPS	(Graphics Processing Unit) графічний процесор
HSV	(Hue, Saturation, Value) колірна модель, що описує колір через тон, насиченість та яскравість
ROI	(Region of Interest) зона інтересу; виділена цифрова область (полігон) на зображенні, в межах якої виконується аналіз та робота логічних тригерів.
YOLO	(You Only Look Once) сімейство сучасних згорткових нейромережевих архітектур, що реалізують принцип наскрізного (*End-to-End*) розпізнавання об'єктів у реальному часі.
ID	(Identifier) унікальний цифровий або текстовий ідентифікатор
IoU	(Intersection over Union) метрика перетину над об'єднанням; математичний коефіцієнт, що використовується в нейромережах YOLO для оцінки точності локалізації об'єктів.

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота виконана студентом групи ІТ-41 Гладієм Дмитром Івановичем. Тема “Інформаційна технологія автоматизованого виявлення порушень правил дорожнього руху засобами комп’ютерного зору”. Робота направлена на здобуття ступеня бакалавр за спеціальністю 126 «Інформаційні системи та технології» – Львівський державний університет внутрішніх справ, МВС України, Львів, 2026.

У межах цього дослідження було здійснено проєктування та втілення системи для контролю дорожнього руху в режимі реального часу. Процес розробки включав вивчення актуальних архітектур нейромереж, призначених для ідентифікації об’єктів, і зупинку вибору на моделі YOLOv11 завдяки її значній точності та швидкості опрацювання інформації.

Були вивчені засади роботи алгоритмів стеження за об’єктами, а також підходи до аналізу відеоданих у колірному просторі HSV з метою автоматичного розпізнавання сигналів світлофора. Дослідження охопило математичні методи визначення належності точок до багатокутників довільної конфігурації для фіксації випадків порушення правил на острівцях безпеки та лініях зупинки.

Метою роботи є впровадження системи, здатної моніторити дорожній рух, використовуючи три модулі штучного зору на дорожніх камерах у режимі реального часу. Ці модулі здатні автоматично виявляти порушення правил дорожнього руху.

Предметом наукового інтересу є процеси виявленння та класифікації рухомого транспорту, а також методи аналізу їхніх шляхів руху стосовно визначених зон, які підлягають моніторингу.

У результаті виконання дипломної роботи було розробено систему, яка в автоматичному режимі ідентифікує порушників, веде статистику, а також зберігає докази з моментом правопорушення.

Ключові слова: інформаційна система, комп’ютерний зір, YOLOv11, детекція об’єктів, простір кольорів HSV, трекінг об’єктів, зона детекції.

ABSTRACT

Bachelor’s qualification work was completed by a student of the IT-41 group Gladiy Dmytro Ivanovych. The topic is “Information technology for automated detection of traffic violations using computer vision”. The work is aimed at obtaining a bachelor's degree in specialty 126 “Information systems and technologies” – Lviv State University of Internal Affairs, Ministry of Internal Affairs of Ukraine, Lviv, 2026.

Within the framework of this research, a system for real-time traffic control was designed and implemented. The development process included the study of current neural network architectures designed for object identification, and the choice was made on the YOLOv11 model due to its significant accuracy and speed of information processing.

The principles of object tracking algorithms were studied, as well as approaches to analyzing video data in the HSV color space for the purpose of automatic recognition of traffic light signals. The study covered mathematical methods for determining the belonging of points to polygons of arbitrary configuration for recording cases of rule violations on safety islands and stop lines.

The aim of the work is to implement a system capable of monitoring road traffic using three artificial vision modules on road cameras in real time. These modules are capable of automatically detecting traffic violations.

The subject of scientific interest is the processes of detecting and classifying moving vehicles, as well as methods of analyzing their movement paths in relation to specific zones to be monitored.

As a result of the thesis, a system was developed that automatically identifies violators, keeps statistics, and stores evidence with the moment of the offense.

Keywords: information system, computer vision, YOLOv11, object detection, HSV color space, object tracking, detection zone.

ЗМІСТ

ВСТУП

Обґрунтування актуальності теми дослідження. Сучасна урбанізація та швидке збільшення насиченості транспорту у великих містах ставлять перед державними установами складні завдання щодо гарантування безпеки дорожнього руху. Звичні методи моніторингу, що ґрунтуються на людському потенціалі, мають обмеження у швидкості опрацювання інформації, схильні до суб'єктивності та не здатні забезпечити постійне спостереження в актуальному часі. Впровадження інтелектуальних транспортних систем (ITS) на базі технологій комп'ютерного зору стає основним засобом зменшення дорожньо-транспортних пригод та оптимізації трафіку. Застосування актуальних нейромережевих структур для автоматичної фіксації порушень правил дорожнього руху (ПДР) дає змогу не лише гарантувати неупередженість доказів, а й суттєво підвищити дисципліну водіїв, що є критично важливим для розбудови концепції «Smart City».

Аналіз останніх досліджень і публікацій. Питаннями автоматизації моніторингу дорожньої інфраструктури та розпізнавання об'єктів займається чимале коло науковців. Базові принципи виявлення об'єктів у режимі реального часу були сформульовані у працях Дж.Редмона (Joseph Redmon) та А. Бочковського (Alexey Bochkovskiy), розробників архітектури YOLO [1]. Значний внесок у прогрес методів цифрової обробки зображень та систем відео аналітики зробили українські вчені, зокрема В.М. Терещенко та О.Ю. Бармак, котрі вивчали інтелектуальні методи опрацювання відеоданих [2, 3]. Однак, попри високу результативність сучасних нейромереж, проблема пристосування алгоритмів до змінних умов освітлення та створення полегшених систем для локального опрацювання інформації залишаються предметом обговорення та вимагають неухильного покращення.

Мета дослідження полягає у підвищенні рівня безпеки дорожнього руху шляхом розробки та впровадження автоматизованої системи фіксації порушень ПДР на основі нейронної мережі YOLOv11.

Завдання дослідження:

1. З’ясувати поточний стан та тенденції розвитку систем автоматичної фіксації порушень ПДР.

2. Проаналізувати архітектуру нейронної мережі YOLOv11 та обґрунтувати її вибір для задач реального часу.

3. Дослідити методи попередньої обробки відеоданих та виділення зон інтересу (ROI) для моніторингу світлофорних об’єктів.

4. Розробити алгоритм аналізу сигналів світлофора з використанням простору кольорів HSV.

5. Обґрунтувати вибір методів трекінгу транспортних засобів для забезпечення унікальної ідентифікації об’єктів.

6. Сформувати математичну модель перевірки входження транспортного засобу в заборонену зону.

7. Розробити програмну логіку для автоматичного зберігання фото-доказів правопорушень.

8. Реалізувати програмний комплекс та провести тестування системи на реальних відеоданих.

9. Узагальнити результати впровадження системи та оцінити її ефективність.

Об’єкт дослідження - процес автоматичної ідентифікації, відслідковування та фіксації порушень ПДР транспортними засобами в режимі реального часу за допомогою засобів комп'ютерного зору.

Предмет дослідження - методи класифікації об’єктів на зображенні, алгоритми аналізу кольорових характеристик об'єктів та програмні засоби для обробки відео-потоку з метою виявлення правопорушень на перехрестях.

Методи дослідження. Для досягнення поставленої мети використано комплекс методів дослідження: системний аналіз (при вивченні існуючих рішень у галузі ITS), методи комп’ютерного зору та цифрової обробки зображень (для детекції та аналізу кольорів у просторі HSV), методи нейронних мереж та машинного навчання (для ідентифікації об'єктів), а також об'єктно-орієнтоване програмування (Python) для реалізації алгоритмів та побудови програмної

архітектури системи.

Інформація про практичне значення роботи. Розроблена система автоматичної фіксації порушень ПДР має великий потенціал для впровадження в рамках міських інтелектуальних транспортних систем. Запропонований програмний модуль на базі нейронної мережі YOLOv11 може бути включений до наявних систем вуличного відео нагляду для автоматизації процесу спостереження без участі операторів. Результати роботи можуть бути застосовані органами місцевого самоврядування або операторами парк-зон для покращення контролю за дотриманням правил дорожнього руху.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків. Загальний обсяг пояснювальної записки становить 61 сторінок, робота містить 16 рисунків та 7 таблиць.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ

1.1. Аналіз проблеми автоматизованого моніторингу дорожнього руху та класифікація правопорушень.

Трафік система є складною динамічною структурою, що створює великі обсяги відомостей у реальному часі. Дієве керівництво цією структурою нездійсненне без автоматизації процесів накопичення та оцінки даних. Сучасні системи спостереження стикаються з викликами, пов’язаними не лише з кількістю транспортних засобів, а й з різноманітністю середовища: погодними умовами, зміною рівня освітленості, а також складністю геометрії перехресть.

1.1.1. Критичний аналіз існуючих методів контролю

Традиційні підходи до моніторингу дорожнього руху базуються на використанні фізичних сенсорів. Детальний аналіз для побудови об’єктивної картини ефективності таких систем наведемо в табл. 1.1.

Таблиця 1.1

Технологія	**Метод детекції**	**Переваги**	**Недоліки**
Індукційні петлі	Електромагнітні зміни	Висока точність підрахунку	Висока вартість монтажу, вразливість до асфальтного покриття
Радарні датчики	Ефект Доплера	Робота в будь-яку погоду	Обмежена точність при щільному трафіку, складнощі з класифікацією типів ТЗ
Лазерні сканери (LiDAR)	Час прольоту (ToF)	Висока роздільна здатність	Висока вартість, складність калібрування

Продовження таблиці 1.1

Системи Computer Vision	Нейромережевий аналіз	Багатофункціональність, низька вартість (використання CCTV)	Чутливість до освітлення, потреба в обчислювальних потужностях

Джерело: складено автором.

Як вбачається з таблиці 1.1, системи комп'ютерного зору (Computer Vision) є найбільш перспективними. На відміну від індукційних петель, які дають лише бінарний сигнал (є авто / немає авто), відеоаналітика дозволяє отримувати семантичну інформацію: класифікацію об'єктів (легковий/вантажний автомобіль, пішохід), напрямок руху та аналіз поведінки.

1.1.2. Класифікація порушень як технічна задача

Для реалізації інформаційної технології автоматизованого виявлення порушень, потрібно перевести юридичне визначення «правопорушення» у мову комп'ютерного зору – тобто представити його як подію у просторі та часі.

Правопорушення можна класифікувати за двома основними ознаками:

1. **Просторові порушення:** порушення, які обумовлені фактом перебування об'єкта у забороненій зоні (наприклад, зупинка на стоп-лінії, паркування у недозволеному місці).
2. **Часово-просторові (динамічні) порушення:** порушення, які вимагають аналізу траєкторії руху об'єкта відносно стану інфраструктури (наприклад, проїзд на заборонений сигнал світлофора, хибний поворот).

Також можна привести співвідношення типу порушення та методу програмної обробки, які наведені в табл. 1.2.

Таблиця 1.2

Тип порушення	**Тип події**	**Алгоритмічна база**
Зупинка на стоп-лінії	Просторова	Polygon Point Test, Geometric Masking
Проїзд на червоний сигнал	Динамічна	Трекінг + аналіз станів кольорів (HSV/RGB)
Виїзд на зустрічну смугу	Динамічна	Semantic Segmentation + Trajectory Analysis
Порушення правил маневрування (ротонда)	Динамічна	Multi-object tracking (MOT) + зональні фільтри

Джерело: складено автором.

Наведені у табл. 1.2 дані показують, що кожне правопорушення з точки зору програмування являє собою сукупність статистичних координат та змінних станів об’єктів.

1. **Просторовий аналіз (Зупинка на стоп-лінії).** Фіксація таких подій спирається на геометричну методологію. Застосовуючи алгоритм Polygon Point Test, установа у дійсному часі звіряє, чи містяться координати нижньої точки об’єкта (середина між колесами) всередині визначеного багатокутника (область стопу). Це дає можливість знаходити факти порушень без вивчення тривалих часових проміжків.
2. **Аналіз часово-просторових подій (Проїзд на червоний сигнал).** Це більш заплутана задача, що вимагає синхронізації детекції стану світлофора та трекінгу транспортного засобу. Задля стабільності у різних умовах освітлення пропонується застосувати сегментацію у колірному просторі HSV, котра є менш чутливою до тіней та відблисків, аніж RGB. Подія порушення генерується при одночасному виконанні умов: активний заборонений сигнал та перетин об’єктом контрольної лінії.

3. **Застосування Multi-object tracking (MOT).** Для складних ділянок, як-от кільцеві розв’язки, потрібне відстеження шляху кожного ТЗ з присвоєнням унікального ідентифікатора (ID). Це дає можливість системі аналізувати поведінку об’єкта в динаміці: з якого ряду розпочато маневр і в якому напрямку його завершено. Такий підхід мінімізує помилки дублювання даних та дозволяє автоматизувати контроль за дотриманням правил маневрування без участі оператора.

1.1.3. Фактори, що впливають на точність детекції

При проектуванні системи автоматизації треба брати до уваги низку факторів, які знижують якість роботи нейронних мереж у реальних міських умовах. Ці чинники утворюють так званий «шумовий поріг» системи:

- **Фактор перспективи:** Камери відеоспостереження зазвичай встановлені під кутом до дорожнього полотна. Це спричиняє перспективні викривлення, де об’єкти, розташовані далі, виглядають меншими, а їхня швидкість руху в пікселях на кадр є суттєво нижчою, ніж у об’єктів, що знаходяться поблизу камери.
- **Фактор перекриття:** У щільному транспортному потоці об’єкти часто перекривають один одного. Це критично для алгоритмів трекінгу, оскільки втрата ідентифікатора об’єкта (ID switch) призводить до дублювання даних у статистичних звітах.
- **Фактор освітлення:** Динамічні зміни світлового потоку (тіні від будівель, сонячні відблиски на склі, нічний режим) вимагають застосування адаптивних методів попередньої обробки зображень. Як зазначається у дослідженнях, саме ці фактори найчастіше спричиняють хибні спрацювання першого та другого роду при виявленні.

1.2. Огляд сучасних методів та архітектур нейронних мереж для детекції об’єктів

Задача автоматизованого виявлення порушень ПДР базується на

технології детекції об’єктів, яка поєднує в собі два завдання: локалізацію (визначення координат об’єкта у кадрі за допомогою обмежувальної рамки) та класифікацію (визначення типу об’єкта: легковий автомобіль, вантажівка, автобус тощо).

1.2.1. Еволюція підходів до детекції об’єктів

Розвиток методів комп’ютерного зору пройшов шлях від класичних алгоритмів до глибоких нейронних мереж. Сучасні архітектури поділяються на два основні типи, порівняння яких наведено в таблиці 1.3.

Таблиця 1.3

Порівняння архітектур детекторів об’єктів

Критерій	**Двоступеневі (напр., Faster R-CNN)**	**Одноступеневі (напр., YOLO, SSD)**
Принцип роботи	Спочатку генерація регіонів, потім класифікація	Одночасна локалізація та класифікація
Точність (mAP)	Дуже висока	Висока (з версії YOLOv8+ — на рівні R-CNN)
Швидкість (FPS)	Низька (5–15 FPS)	Дуже висока (40–150+ FPS)
Обчислювальні витрати	Високі	Оптимізовані
Сфера застосування	Медична діагностика, статичні фото	Відеоаналітика в реальному часі, автономні авто

Джерело: складено автором

Для задач моніторингу дорожнього руху швидкість обробки є критичним фактором. Використання двоступеневих детекторів призводить до значних затримок, що унеможливлює коректний трекінг транспортних засобів, які рухаються на високій швидкості.

1.2.2. Архітектурні особливості сімейства моделей YOLO

Алгоритм YOLO (You Only Look Once) пройшов значний шлях трансформації від простої регресійної моделі до складної нейромережевої архітектури, здатної працювати в умовах обмежених обчислювальних ресурсів. Для обґрунтування вибору актуальної версії необхідно розглянути ключові етапи розвитку сімейства цих моделей.

Таблиця 1.4

Еволюція характеристик архітектури YOLO

Версія	Рік	Ключові інновації	Основна перевага
YOLOv3	2018	Використання Darknet-53, детекція на 3 масштабах	Стабільність детекції малих об'єктів
YOLOv5	2020	Екосистема PyTorch, архітектура CSPNet	Гнучкість навчання та легкість впровадження
YOLOv8	2023	Без'якірна детекція (Anchor-free), C2f блоки	Висока точність класифікації
YOLOv11	2024	Оптимізований Feature Extraction, покращений Head	Максимальний FPS при низькому споживанні GPU

Джерело: складено автором

Різниця між версіями полягає не лише у швидкості, а й у влучності (mAP) та здатності моделі до генералізації даних. На рисунку 1.1 продемонстровано технічний прогрес точності детекції відносно обчислювальної складності для різних поколінь моделей.

Візуалізація на рис. 1.1 показує, що кожна наступна версія зміщує робочу точку вгору та вліво (щодо затримки, або вгору та вправо щодо FPS). У контексті даного дипломного проекту це критично, оскільки версія **YOLOv11** забезпечує максимальну точність при найвищій швидкості, що дозволяє уникнути затримок при фіксації моменту перетину стоп-лінії.

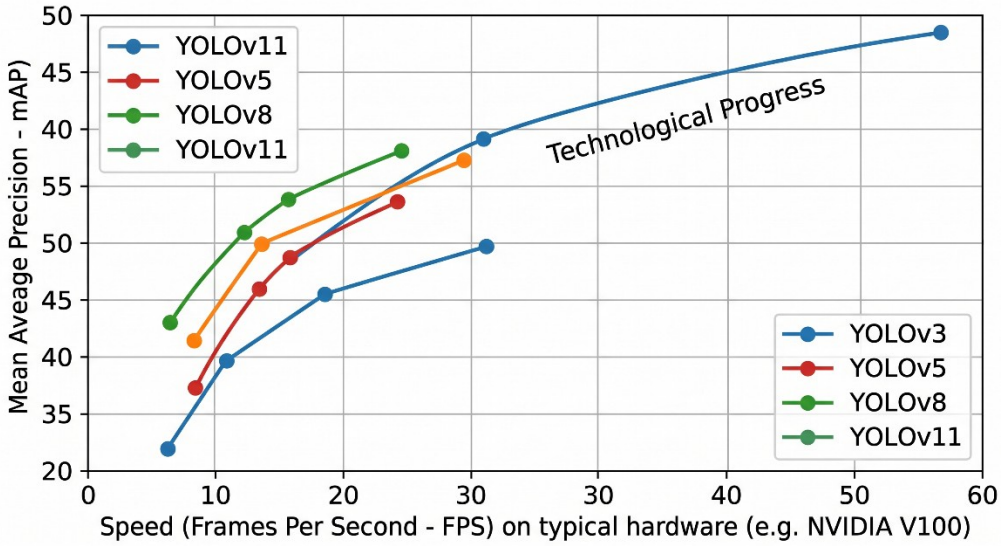

Рис. 1.1 Технічне порівняння моделей YOLO: залежність точності (mAP) від швидкості роботи (FPS). Джерело: Розроблено автором.

Таким чином, вибір YOLOv11 обумовлений потребою в мінімізації помилок другого роду (пропущених порушень) при збереженні високої частоти кадрів, що є обов'язковою умовою для фіксації динамічних подій у дорожньому потоці.

1.3. Аналіз методів цифрової обробки відеопотоку для виділення статичних та динамічних зон інтересу

Ефективність інформаційної технології виявлення порушень ПДР залежить від точності попередньої обробки відеоданих. На відміну від статичних зображень, відеопотік генерує послідовність кадрів, кожен з яких потребує швидкої фільтрації для виявлення цільових об’єктів. Ключовим методом оптимізації обчислювальних ресурсів у таких системах є використання технології виділення зон інтересу (ROI).

Застосування ROI дозволяє програмному комплексу концентрувати увагу нейронної мережі лише на тих ділянках кадру, де потенційно може відбутися правопорушення – наприклад, на зоні перехрестя або перед стоп-лінією. Це значно знижує навантаження на графічний процесор, оскільки замість аналізу всього зображення система обробляє лише невеликий сегмент координат. Окрім швидкодії, такий підхід допомагає відсікати зайві об’єкти: рух дерев, пішоходів

на тротуарах та зміни освітлення поза межами проїжджої частини. Візуалізацію принципу динамічного маскування ROI представлено на рисунку 1.2.

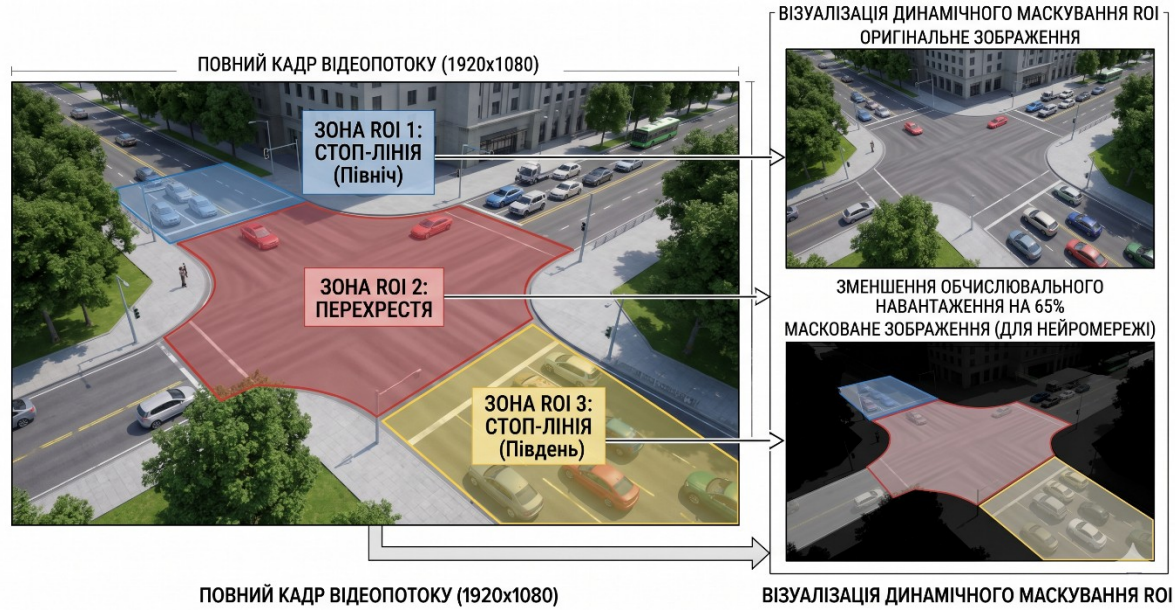

Рис 1.2. Технічна візуалізація динамічного маскування ROI: повний кадр перехрестя (ліворуч) та масковане зображення для нейромережі (праворуч)

Схема на рис. 1.2 чітко демонструє, як система відсікає нерелевантну інформацію. На лівій частині рисунка показано повний кадр із накладеними полігонами зон інтересу (стоп-лінії та центр перехрестя). Права частина показує те, що «бачить» нейронна мережа: лише виділені зони залишаються активними, тоді як решта зображення (тротуари, будинки, дерева) замінюється чорною маскою, що дозволяє досягти значного (до 65%) зменшення обчислювального навантаження.

Важливою частиною технології є перевірка факту входу транспортного засобу в таку зону. Для цього використовується алгоритм перевірки приналежності точки полігону. Система бере координати нижньої частини обмежувальної рамки автомобіля (центр контакту з дорогою) і в реальному часі порівнює їх із заданою маскою зони порушення. Якщо об’єкт із конкретним ідентифікатором (ID) перетинає межу маски, система ініціює алгоритм фіксації події.

Паралельно з просторовим аналізом, критичним завданням є цифрова детекція сигналів світлофора. Дослідження показують, що стандартна колірна

модель RGB (Red, Green, Blue) є вкрай нестабільною для зовнішнього середовища через високу чутливість до яскравості. Будь-яка тінь або сонячний відблиск змінює цифрові значення компонентів, що може спричинити помилкову ідентифікацію дозволеного сигналу як забороненого.

Для вирішення цієї проблеми в інформаційній технології обґрунтовано застосування перетворення в колірний простір HSV (Hue, Saturation, Value). Його головна перевага полягає у відокремленні колірного тону (Hue) від яскравості (Value). Це дозволяє налаштувати фільтри так, щоб вони розпізнавали спектр червоного або жовтого кольорів навіть в умовах недостатнього освітлення або при сильному засвіченні камери сонцем.

Процес ідентифікації сигналу завершується аналізом щільності пікселів у створеній бінарній масці. Система підраховує кількість активних точок у виділеній зоні світлофора. Якщо ця кількість перевищує встановлений поріг (threshold), система вважає сигнал активним. Такий дворівневий підхід (ROI + HSV фільтрація) забезпечує стабільну роботу системи автоматизації та дозволяє чітко синхронізувати момент порушення з відповідною фазою світлофорного об’єкта.

1.4. Обґрунтування вибору технологічного стека для розробки системи

Завершальним етапом аналізу предметної сфери є формування оптимального технологічного стека, який забезпечить стабільне функціонування інформаційної технології в умовах реального часу. Добір інструментарію ґрунтується на потребі досягнення рівноваги між швидкістю опрацювання відеопотоку, достовірністю детекції правопорушень та вимогливістю системи до ресурсів. Для впровадження проєкту обрано архітектуру, що спирається на відкриті стандарти та бібліотеки, які зарекомендували себе як найбільш дієві у завданнях відеоаналітики.

Основою програмного комплексу обрано мову програмування **Python**, яка завдяки своїй модульній структурі та потужній екосистемі бібліотек для штучного інтелекту є стандартом у галузі Computer Vision. Для розв’язання

задачі детекції обґрунтовано використання моделі **YOLOv11** у конфігурації «slim». Дана архітектура дозволяє виконувати розпізнавання об’єктів безпосередньо на локальному обладнанні, що критично важливо для мінімізації затримок при фіксації динамічних подій, таких як проїзд на заборонний сигнал.

Обраний стек технологій є повністю сумісним із сучасними стандартами розробки інтелектуальних систем. Використання виключно інструментів із відкритим сирцевим кодом гарантує високу адаптивність системи та можливість її подальшого масштабування без додаткових витрат на ліцензування програмного забезпечення.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ПОРУШЕНЬ ПДР

2.1. Специфікація вимог та архітектурна побудова програмного комплексу

Проєктування інформаційної технології для автоматизованого виявлення порушень ПДР потребує чіткого визначення функціональних та технічних вимог, які забезпечать її надійність у динамічних умовах дорожнього руху. На етапі проектування архітектури необхідно врахувати, що система повинна не лише ідентифікувати об'єкти, а й приймати логічні рішення на основі перетину просторових та часових подій.

Аналіз функціональних та технічних вимог. Для досягнення мети дослідження розроблено перелік критичних вимог до програмного комплексу:

1. **Функціональні вимоги:**
   - Автоматичне захоплення відеопотоку з віддалених IP-камер або через інтерфейси веб-ресурсів (наприклад, Geoportal).
   - Детекція транспортних засобів (легкові авто, вантажівки, автобуси) з точністю не менше 85%.
   - Диференціація станів світлофорного об’єкта (червоний, жовтий, зелений) у реальному часі.
   - Фіксація факту перетину стоп-лінії у момент дії заборонного сигналу.
   - Зберігання фотодоказів порушення із накладеними метаданими (час, ID об’єкта, тип порушення).
2. **Технічні вимоги (нефункціональні):**
   - Продуктивність: забезпечення обробки не менше 25–30 кадрів на секунду (FPS) для уникнення втрати кадрів із моментом порушення.
   - Автономність: здатність системи працювати без втручання

оператора протягом тривалого часу.

- Масштабованість: архітектура повинна дозволяти легко додавати нові зони контролю (ROI) без переписування основного ядра програми.

Базова архітектура системи. Архітектура розроблюваної інформаційної технології базується на модульному принципі, де кожен блок відповідає за конкретний етап обробки даних. Це дозволяє досягти високої відмовостійкості: збій у модулі візуалізації не зупиняє процес детекції та логування (рис 2.1).

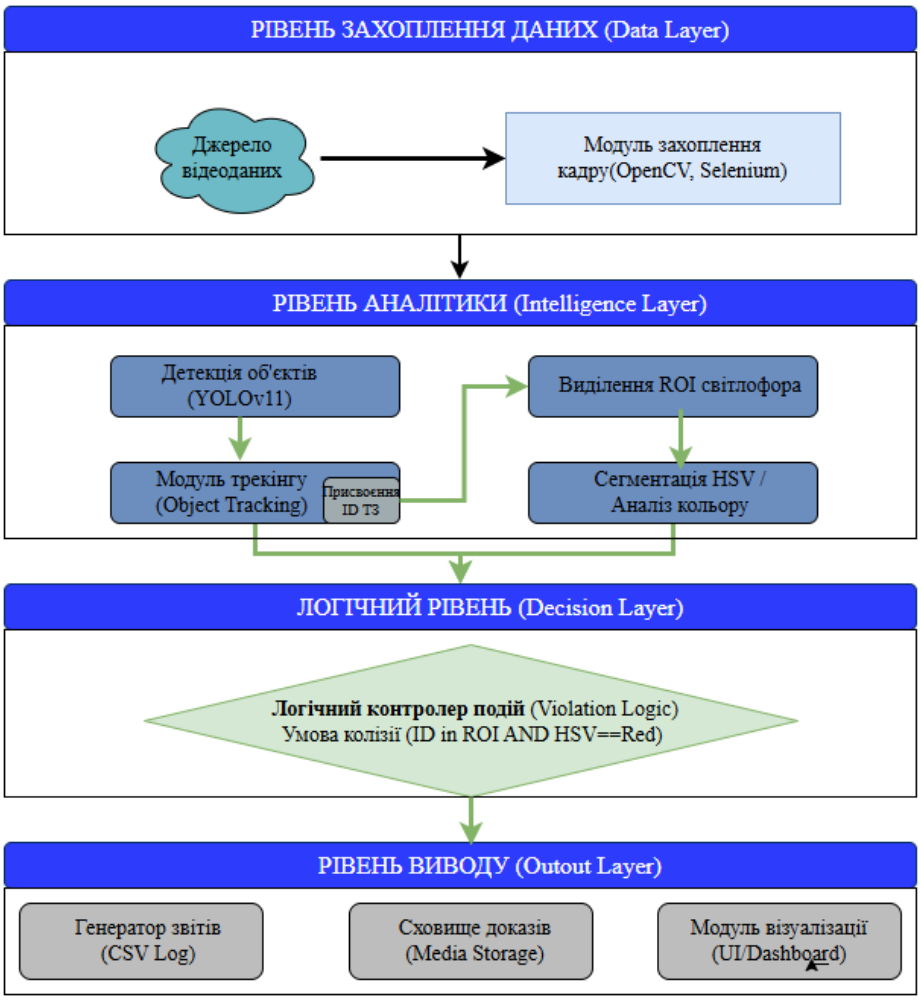

Рис. 2.1 Розроблена структурна схема інформаційної технології виявлення порушень ПДР

Така архітектурна побудова, що базується на чотирирівневій моделі обробки даних у реальному часі, забезпечує високу модульність системи, дозволяючи незалежно налаштовувати компоненти детекції та логічного аналізу.

1. **Рівень захоплення даних (Data Layer) –** відповідає за взаємодію з

джерелами відеосигналу. За допомогою інструментів автоматизації браузера та бібліотеки OpenCV здійснюється отримання необробленого відеопотоку з CCTV-камер або ресурсів Geoportal. На цьому етапі відбувається препроцесинг: нормалізація кадру та його підготовка до подальшого аналізу.

2. **Рівень аналітики (Intelligence Layer) –** є ядром системи, де реалізовано паралельну обробку візуальної інформації за двома напрямками:

- **Гілка локалізації об'єктів:** нейронна мережа архітектури YOLOv11 здійснює детекцію транспортних засобів, після чого модуль трекінгу (Object Tracking) присвоює кожному виявленому об'єкту унікальний ідентифікатор (ID). Це дозволяє системі супроводжувати конкретний транспортний засіб протягом усього часу його знаходження в кадрі.
- **Гілка спектрального аналізу:** на основі заданих координат зон інтересу (ROI) виділяється область світлофорного об'єкта, яка піддається сегментації в колірному просторі HSV для визначення поточного стану сигналу (червоний, жовтий або зелений).

3. **Логічний рівень (Decision Layer) –** виконує функцію агрегатора даних. Логічний контролер подій (Violation Logic) постійно перевіряє умову колізії: факт знаходження об'єкта з конкретним ID у межах полігону «стоп-лінії» за умови активності маски червоного кольору. При одночасному виконанні цих умов генерується подія правопорушення.
4. **Рівень виводу (Output Layer) –** забезпечує фіксацію результатів. Дані про порушення (час, тип, ID) записуються в CSV-журнал, а відповідний кадр із накладеною візуалізацією зберігається у сховищі доказів. Модуль візуалізації надає оператору можливість моніторингу процесу в реальному часі через графічний інтерфейс.

2.2. Алгоритмічне забезпечення детекції та ідентифікації транспортних засобів

Основою інтелектуального аналізу відеопотоку в межах розробленої технології є поєднання нейромережевої моделі YOLOv11 та алгоритмів багатооб'єктного супроводження (Multi-Object Tracking, MOT). Це дозволяє не лише локалізувати об'єкт у межах окремого кадру, а й присвоїти йому унікальний ідентифікатор, що зберігається протягом усього часу перебування транспортного засобу в полі зору камери.

2.2.1. Механізм інференсу моделі YOLOv11

Для детекції об'єктів обрано архітектуру YOLOv11 у конфігурації «small», що забезпечує оптимальний компроміс між швидкістю та точністю (mAP). Процес детекції реалізовано за наступним алгоритмом:

1. **Нормалізація входу:** вхідний кадр масштабується до роздільної здатності 640x640 пікселів, що відповідає архітектурним вимогам вхідного шару мережі.
2. **Прогнозування обмежувальних рамок (Bounding Boxes):** мережа генерує набір координат (x_1 , y_1 , x_2 , y_2) для кожного виявленого об'єкта.
3. **Фільтрація за довірою (Confidence Threshold):** об'єкти з показником впевненості нижче 0.4 відсікаються як шум, що дозволяє уникнути помилкових спрацювань.
4. **Класифікація:** система виділяє лише цільові класи (автомобілі, вантажівки, автобуси), ігноруючи інші типи об'єктів для економії обчислювальних ресурсів.

2.2.2. Алгоритм трекінгу та ідентифікація об'єктів

Для уникнення повторної фіксації одного порушення та аналізу траєкторії руху автомобіля інтегровано алгоритм трекінгу. Його робота базується на порівнянні результатів детекції поточного кадру з результатами попереднього кадру (рис. 2.2).

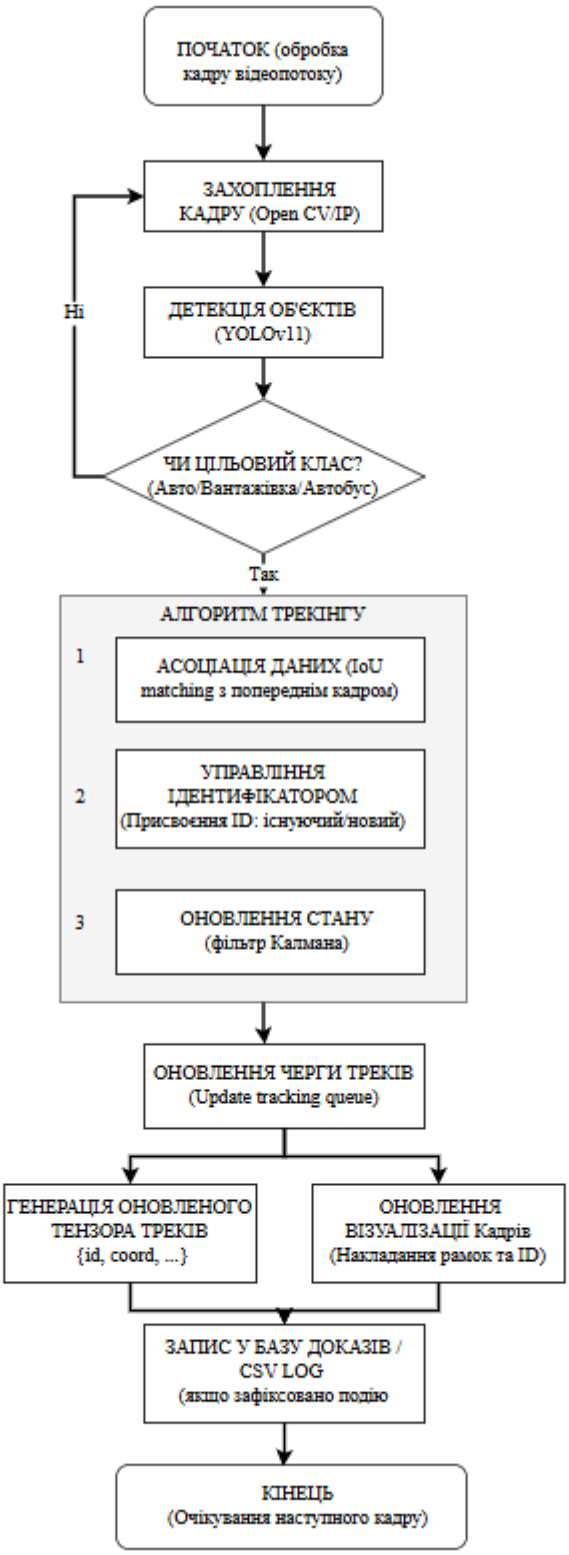

Рис. 2.2 Блок-схема алгоритму багатооб’єктного трекінгу ТЗ з присвоєнням ID

Представлена деталізована блок-схема алгоритму трекінгу, який інтегровано в ядро інтелектуальної обробки. Процес починається з отримання чергового кадру відеопотоку. Спочатку модель YOLOv11 виконує інференс та

визначає всі об’єкти, генеруючи набір тензорів із координатами, класами та впевненістю. Обов’язковим етапом є логічна фільтрація: система перевіряє, чи належить виявлений об’єкт до цільового класу (транспортний засіб). Якщо ні, об’єкт відсікається.

Центральною частиною алгоритму є Модуль трекінгу (MOT). Він приймає дані детекції та порівнює їх із базою треків попереднього кадру. Ключовим моментом є асоціація даних, де на основі метрики Intersection over Union (IoU) визначається ступінь перекриття рамок. Якщо перекриття є максимальним, об’єкту присвоюється існуючий у базі ID. Якщо об’єкт новий, йому присвоюється новий унікальний ідентифікатор. Паралельно використовується фільтр Калмана для оновлення передбачуваного стану та траєкторії об’єкта, що дозволяє уникнути втрати ідентифікатора при короткочасному перекритті авто іншими перешкодами.

2.2.3. Формування потоку даних для логічного висновку

Результатом роботи алгоритмів детекції та трекінгу є сформований у реальному часі тензор даних, де кожен транспортний засіб представлений структурою:

- ID;
- class;
- confidence;
- coordinates;

Для оптимізації подальшого аналізу система розраховує координати нижньої межі обмежувальної рамки (центр контакту коліс із дорожнім покриттям). Саме ця точка використовується надалі для перевірки факту входу в зону інтересу (ROI). Такий підхід дозволяє суттєво зменшити кількість звернень до логічного блоку та зосередити обчислювальні потужності лише на тих автомобілях, що безпосередньо наближаються до контрольованої ділянки дороги.

2.3. Математичне та програмне забезпечення аналізу станів

світлофорних об’єктів

Одним із найскладніших завдань при автоматизації фіксації порушень ПДР є стабільне розпізнавання активного сигналу світлофора в умовах змінного природного освітлення. У межах розробленої технології для вирішення цієї задачі застосовано метод колірної сегментації в просторі HSV (*Hue, Saturation, Value*), що дозволяє відокремити інформацію про колір від параметрів яскравості. Традиційна модель RGB є неефективною для відеомоніторингу просто неба, оскільки будь-яка зміна освітленості (тінь від будівель, хмарність або нічні відблиски) змінює значення у всіх трьох каналах одночасно. Простір HSV мінімізує цей вплив шляхом винесення яскравості в окремий канал *V*, залишаючи чистий колірний тон у каналі *H*.

Процес цифрової обробки сигналу починається з виділення статичної зони інтересу (ROI), яка відповідає положенню світлофора на кадрі. Вирізана область піддається конвертації з формату BGR у HSV, після чого до неї застосовуються порогові фільтри. Оскільки червоний колір у просторі HSV знаходиться на краях колірного спектра, програмно реалізовано об'єднання двох діапазонів для точного захоплення сигналу: нижнього ($H \in [0, 10]$) та верхнього ($H \in [160, 180]$). Результатом цієї операції є бінарна маска, де пікселі цільового кольору набувають максимального значення, а решта фону — нульового.

Для підвищення надійності детекції отримана маска піддається морфологічній обробці, зокрема операціям ерозії та дилатації. Ерозія дозволяє видалити дрібні одиничні пікселі («шум»), що виникають через цифрові спотворення або відблиски від кузовів автомобілів. Подальша дилатація розширює межі вцілілих об'єктів, відновлюючи цілісність бінарного відображення сигналу. Остаточне рішення про активацію того чи іншого кольору приймається на основі аналізу щільності пікселів. Система обчислює відношення кількості активних пікселів у масці до загальної площі зони аналізу:

$$S_{active} = \frac{\sum_{i=1}^{n} P_i}{Area_{ROI}} > r \qquad (2.1)$$

де r – встановлений поріг чутливості (зазвичай 15-20%), що гарантує ігнорування сторонніх джерел світла. Програмна логіка також включає механізм «пам'яті станів», який зберігає попередній відомий колір при короткочасному перекритті світлофора перешкодами. Сформована в результаті змінна статусу сигналу передається до фінального логічного блоку, забезпечуючи необхідну синхронізацію для прийняття рішення про факт правопорушення.

2.4 Розробка алгоритму логічного висновку про факт порушення ПДР

Фінальним етапом проектування інформаційної технології є розробка алгоритму логічного висновку, який інтегрує дані з усіх попередніх модулів. Завданням цього блоку є синхронізація в часі та просторі результатів нейромережевого аналізу транспортних засобів та спектральної сегментації сигналів світлофора для автоматичної ідентифікації правопорушень.

Логіка прийняття рішення базується на паралельній перевірці трьох незалежних умов у кожному кадрі відеопотоку. Алгоритм працює за подієвою моделлю, де тригером для початку перевірки є наявність активних треків транспортних засобів у кадрі. Процес ідентифікації порушення можна представити у вигляді логічної функції, яка повертає позитивний результат лише за умови істинності наступного предикату:

1. **Статус сигналу:** Змінна traffic_light_state має значення «Red» (або «Yellow», залежно від конфігурації суворості системи).
2. **Просторова колізія:** Центральна точка нижньої межі обмежувальної рамки об'єкта $P(x,y)$ знаходиться всередині полігону зони інтересу ROI_{stop_line}.
3. **Унікальність події:** Ідентифікатор об'єкта track_id відсутній у базі даних уже зафіксованих порушень для поточної фази червоного світла.

Для візуалізації послідовності операцій та розгалужень логіки системи розроблено блок-схему алгоритму прийняття рішень, представлену на рисунку 2.3.

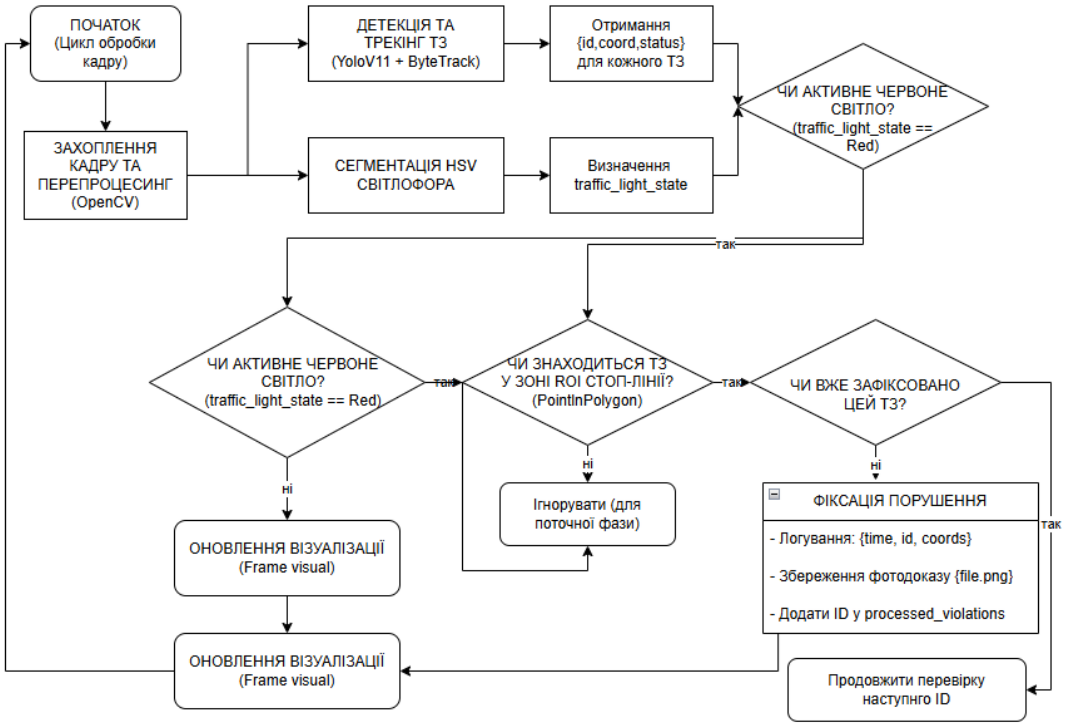

Рис 2.3 Алгоритм логічного висновку про фіксацію порушення проїзду на заборонний сигнал

Особливу увагу при розробці алгоритму приділено механізму запобігання дублюванню штрафів. Оскільки один і той самий автомобіль може перебувати в зоні порушення протягом декількох секунд (кількох десятків кадрів), система використовує словник «ігнорування». Як тільки умова порушення виконується, track_id автомобіля заноситься до списку processed_violations. Це гарантує, що для одного факту проїзду буде згенеровано лише один запис у журналі та один скріншот-доказ.

Процес фіксації (фіналізація події) включає автоматичне виконання трьох операцій:

- **Екстракція метаданих:** формування рядка з часом, типом об’єкта, швидкістю (якщо активовано відповідний модуль) та ідентифікатором.
- **Графічне підтвердження:** збереження оригінального кадру з накладеними візуальними елементами (рамка навколо авто, підсвічування зони ROI та акцентування на сигналі світлофора).
- **Логування:** запис структурованих даних у файл формату CSV для

подальшого аналізу або передачі в зовнішні бази даних.

Розроблений алгоритм логічного висновку забезпечує високу автономність системи. Завдяки чітко визначеним просторовим межам та часовій синхронізації з фазами світлофора, мінімізується ймовірність помилкових спрацювань, спричинених об’єктами, що зупинилися безпосередньо перед лінією, або тими, що завершують маневр на дозволений сигнал.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1. Технічні умови розгортання та системні вимоги до середовища виконання

Ефективність та стабільність функціонування розробленої інформаційної технології автоматизованого виявлення порушень ПДР у реальному часі безпосередньо залежить від обчислювальних потужностей апаратного комплексу та коректності налаштування системного програмного забезпечення. Оскільки архітектура YOLOv11 використовує глибокі згорткові нейронні мережі для детекції об'єктів, а алгоритми колірної сегментації OpenCV потребують швидкої попіксельної обробки матриць високої роздільної здатності, критично важливим є забезпечення належного рівня апаратної акселерації.

Специфікація апаратного забезпечення (Hardware Requirements). Для проведення експериментальних досліджень, тестування комплексу та інференсу нейромережевих моделей у реальному часі було обґрунтовано використання обчислювальної станції з наступними технічними характеристиками:

- **Центральний процесор (CPU):** Архітектура x86-64 (наприклад, Intel Core i7 або AMD Ryzen 7) з підтримкою технологій багатопотоковості (Multi-threading) та векторних інструкцій AVX2. Це необхідно для паралельного декодування вхідного відеопотоку через OpenCV, керування чергою кадрів та виконання логічних блоків перевірки умов порушень.
- **Графічний прискорювач (GPU):** Дискретна відеокарта сімейства NVIDIA GeForce RTX (серії 30xx, 40xx або новіші) з обсягом виділеної відеопам'яті (VRAM) не менше 6 ГБ. Обов'язковою технічною вимогою є наявність тензорних ядер та підтримка технології CUDA (Compute Unified Device Architecture) разом із бібліотекою cuDNN для швидкого виконання згорткових операцій нейромережі.

- **Оперативна пам'ять (RAM):** Системна пам'ять обсягом не менше 16 ГБ стандарту DDR4/DDR5, що функціонує у двоканальному режимі. Це забезпечує безперебійну буферизацію високопотокового відео та утримання вагових коефіцієнтів моделі YOLO в оперативній пам'яті без звернення до файлу підкачки.
- **Підсистема збереження даних:** Твердотільний накопичувач (SSD) з інтерфейсом NVMe (PCIe Gen4), який гарантує високу швидкість довільного читання та запису. Це критично для миттєвої фотофіксації правопорушень та запису медіа-доказів у локальні директорії без створення затримок (I/O bottlenecks) у роботі основного аналітичного циклу.

Конфігурація програмного середовища та залежностей. Базовою операційною системою для розгортання та тестування комплексу обрано Windows 11 (або дистрибутиви GNU/Linux, зокрема Ubuntu LTS), де розгорнуто інтерпретатор мови програмування Python версії 3.10. Особливу увагу приділено сумісності версій інструментів інференсу та обробки багатовимірних масивів даних. Зведені технічні дані щодо версій складу програмного забезпечення та функціонального призначення кожного компонента наведено в таблиці 3.1.

Таблиця 3.1

Специфікація програмного середовища розробки та тестування

Назва пакету / компонента	**Версія**	**Функціональне призначення в розробленій технології**
Python	3.10.x	Базовий інтерпретатор, виконання логічних блоків та управління потоками
Ultralytics	11.0.x	Інференс нейромережі YOLOv11, запуск алгоритму трекінгу ByteTrack
OpenCV-Python (cv2)	4.9.x	Захоплення медіа-потоку, препроцесинг, колірна конвертація BGR-HSV, UI

Продовження таблиці 3.1

NumPy	1.26.x	Робота з матрицями зображень, маскування, швидкі бінарні операції
yt-dlp	2025.x	Автоматизоване вилучення та стрімінг тестових відеопотоків високої чіткості
CUDA Toolkit	12.1	Апаратне прискорення тензорних обчислень на рівнях ядер GPU NVIDIA
cuDNN	8.9.x	Оптимізація виконання шарів глибоких нейромереж на рівні драйверів

Джерело: складено автором

Перед початком виконання основних аналітичних конвеєрів розроблена програма автоматично виконує тест сумісності бібліотеки PyTorch з інстальованими драйверами CUDA. Якщо апаратний прискорювач (GPU) доступний, тензори моделі автоматично завантажуються у відеопам'ять. У випадку відсутності графічного процесора з підтримкою CUDA система переходить в режим емуляції на CPU, що суттєво знижує показник кадрів за секунду (FPS), проте повністю зберігає логічну працездатність системи.

3.2. Програмна реалізація модулів захоплення відеоданих та інтелектуального аналізу

Розроблений програмний комплекс структурно складається з двох взаємопов’язаних функціональних частин: скрипту попереднього завантаження/ізоляція медіа-потоку та головного аналітичного скрипту детекції правопорушень.

3.2.1. Модуль автоматизованого захоплення відеопотоку

Головний аналітичний модуль обробляє відеопотік у циклі, поєднуючи колірну сегментацію світлофора, нейромережевий трекінг транспортних засобів та геометричний аналіз їхнього положення. (рис 3.1).

```python
import yt_dlp

def download_video(url):
    ydl_opts = {
    'format': '18',
    'outtmpl': 'carsLightRed.mp4',
}

    with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        print(f"Починаю завантаження: {url}")
        ydl.download([url])
        print("Завантаження завершено!")

video_url = "https://www.youtube.com/watch?v=lBRWEeuuMpE"
download_video(video_url)
```

Рис 3.1 Модуль завантаження відеопотоку

Технічний аналіз реалізації модуля завантаження:

1. **Конфігурація потоку:** Параметр 'format': '18' вказує бібліотеці на необхідність завантаження відео в контейнері MP4 (кодек H.264). Цей формат нативно підтримується мультимедійними декодерами OpenCV (cv2.VideoCapture) на будь-якій операційній системі без встановлення додаткових зовнішніх кодеків.
2. **Управління ресурсами:** Виклик функції обгорнуто в контекстний менеджер with ... as. Це гарантує, що після завершення завантаження або у разі виникнення мережевої помилки всі відкриті сокети та дескриптори файлів будуть автоматично закриті інтерпретатором, запобігаючи витоку пам'яті.
3. **Ізоляція вихідних даних:** Шаблон імені файлу перевизначає стандартну назву відео на статичну константу 'carsLightRed.mp4'. Це дозволяє повністю автоматизувати роботу аналітичного конвеєра, оскільки головний скрипт детекції звертається до фіксованого імені файлу-джерела.

3.2.2. Реалізація модуля просторового аналізу та фіксації перетину лінії розмітки

Другим структурним компонентом інформаційної технології є модуль, що

відповідає за геометричний моніторинг дорожньої обстановки та фіксацію фактів наїзду або перетину транспортними засобами ліній дорожньої розмітки (стоп-лінії або суцільної лінії розмежування смуг). Цей модуль функціонує незалежно від стану світлофорних об'єктів і базується на зіставленні динамічних координат треків ТЗ із наперед заданим статичним багатокутником зони інтересу (ROI).

Програмна реалізація цього модуля розділена на три логічні етапи: конфігурація геометричного полігону, трекінг об'єктів за допомогою моделі YOLOv11 та обчислення колізій через просторовий тест.

Етап ініціалізації та конфігурації просторових зон. На початку роботи скрипту здійснюється завантаження нейромережевої моделі, відкриття декомпресора відеопотоку OpenCV та математичний опис контрольованої зони. Зона стоп-лінії проектується не у вигляді простої прямої лінії, а у вигляді чотирикутника (трапеції), що дозволяє врахувати ефект перспективного спотворення камери спостереження. Координати вершин задаються у вигляді матриці NumPy (рис. 3.2).

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

def run():
    model = YOLO("yolo11s.pt")
    source = "cars.mp4"
    cap = cv2.VideoCapture(source)
    violated_ids = set()


    zone_points = np.array([
        [341, 178], [392, 178], [420, 175], [465, 228], [443, 252],
        [421, 282], [403, 314], [381, 359], [367, 359], [361, 303],
        [355, 247], [347, 206], [341, 178],
    ], np.int32)
```

Рис. 3.2 Налаштування та ініціалізація середовища детекції лінії

Етап конвеєрної обробки, детекції та багатооб'єктного трекінгу. Після ініціалізації програма входить у нескінченний цикл обробки кадрів *while cap.isOpened()*. Кожен фрейм масштабується до роздільної здатності *640 x 360*

пікселів для зниження навантаження на тензорні ядра GPU. Далі кадр передається в метод *model.track()*. Параметр *persist=True* є критичним: він вмикає внутрішній алгоритм ByteTrack, який зіставляє вектори руху об'єктів і дозволяє системі «пам'ятати» кожну машину за її індивідуальним номером (*obj_id*) [рис. 3.3].

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret: break

    results = model.track(frame, persist=True, conf=0.4, verbose=False)


    overlay = frame.copy()
    cv2.fillPoly(overlay, [zone_points], (0, 0, 255))
    cv2.addWeighted(overlay, 0.4, frame, 0.6, 0, frame)
    cv2.polylines(frame, [zone_points], True, (255, 255, 255), 1)
```

Рис. 3.3 Цикл обробки та візуалізація інтерфейсу

Етап обчислення просторових колізій та логування подій. Основна аналітична логіка фіксації базується на перевірці умов перетину. Програма вилучає координати обмежувальних рамок (*boxes.xyxy*). Для кожної машини розраховується її опорна точка (*cx, cy*), яка знаходиться в самому низу рамки (центр контакту коліс із дорожнім покриттям).

Ця точка передається у функцію *cv2.pointPolygonTest()*. Якщо функція повертає значення ≥ 0, це є математичним доказом того, що точка перетнула межу або знаходиться всередині полігону. Якщо цей ID ще не зафіксовано у множині *violated_ids*, програма робить знімок екрана та маркує автомобіль червоним кольором (рис. 3.4).

Технічний аналіз програмної реалізації модуля контролю лінії:

1. **Геометрична оптимізація:** Застосування *cv2.pointPolygonTest* замість аналізу простих ліній за формулою $y = kx + b$ дозволяє створювати зони контролю будь-якої складної форми (полігони з 4, 5 або більше кутами). Це критично для реальних українських перехресть, де камери встановлені під кутом до дороги, і розмітка в кадрі йде по діагоналі.

2. **Фільтрація опорної точки:** Розрахунок $cy = \int (y2 - 5)$ замість використання геометричного центру мас автомобіля є важливим інженерним рішенням. Якби система аналізувала центр машини, то капот автомобіля міг би повністю перетнути лінію, але фіксація не відбулася б, поки лобове скло не доїде до лінії. Аналіз нижньої точки рамки прив'язує логіку програми до фактичного положення передніх коліс на асфальті.

```
    if results[0].boxes.id is not None:
        boxes = results[0].boxes.xyxy.cpu().numpy()
        ids = results[0].boxes.id.cpu().numpy().astype(int)

        for box, obj_id in zip(boxes, ids):
            x1, y1, x2, y2 = box


            left_wheel = (int(x1), int(y2 - 5))
            right_wheel = (int(x2), int(y2 - 5))


            cv2.circle(frame, left_wheel, 3, (255, 255, 0), -1)
            cv2.circle(frame, right_wheel, 3, (255, 255, 0), -1)


            in_left = cv2.pointPolygonTest(zone_points, (float(left_wheel[0]), float(left_wheel[1])), False) >= 0
            in_right = cv2.pointPolygonTest(zone_points, (float(right_wheel[0]), float(right_wheel[1])), False) >= 0

            if in_left or in_right:
                violated_ids.add(obj_id)
                cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 2)
                cv2.putText(frame, f"ID:{obj_id} STOP!", (int(x1), int(y1)-5),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
            else:
                cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 1)


        cv2.putText(frame, f"Total Violators: {len(violated_ids)}", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)

        cv2.imshow("YOLOv11 Multi-Point Detection", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'): break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    run()
```

Рис. 3.4 Модуль логіки прийняття рішень та закриття потоків

3.2.3. Реалізація модуля фіксації порушень на заборонний сигнал світлофорного об’єкта

Найбільш технічно насиченим компонентом є модуль, який здійснює синхронізацію просторового трекінгу машин із поточним оптичним станом світлофора. Його ключова відмінність від попереднього модуля полягає в тому, що колір напівпрозорого полігону стоп-лінії динамічно змінюється залежно від

результатів комп'ютерного зору в зоні світлофора: якщо горить зелене світло — зона маркується зеленим і проїзд дозволено, якщо спалахує червоний або жовтий колір — зона моментально стає червоною, переходячи в режим суворої фіксації порушень.

Програмна реалізація цього модуля розділена на три послідовні фази.

1. Фаза ініціалізації середовища та просторового калібрування кадрів (рис. 3.5).

На старті програми виконується завантаження нейромережі, налаштування локальної директорії для збереження доказів, а також жорстке просторове калібрування двох критичних зон інтересу (ROI):

- **Координати світлофора (*tl_x, tl_y, tl_w, tl_h*):** Невелика прямокутна область, яка чітко обмежує контури лінз світлофора в кадрі для відсікання стороннього візуального шуму.
- **Координати стоп-лінії (*stop_line_zone*):** Чотирикутник на асфальті, у межах якого аналізуватиметься присутність коліс транспортних засобів.

```python
import cv2
import numpy as np
from ultralytics import YOLO
import os
from datetime import datetime

def run():

    model = YOLO("yolo11s.pt")


    video_name = "carsLightRed.mp4"
    cap = cv2.VideoCapture(video_name)


    output_dir = "violations"
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    violated_ids = set()


    tl_x, tl_y, tl_w, tl_h = 350, 126, 15, 35
```

Рис. 3.5 Конфігурація параметрів ініціалізації та координатного калібрування зон інтересу світлофорного об’єкта

2. Фаза спектрального HSV-аналізу та динамічної адаптації інтерфейсу (рис. 3.6).

Усередині основного аналітичного циклу система на кожному кадрі вирізає матрицю світлофора *light_roi* та трансформує її з колірного простору BGR у простір HSV. Програма накладає три математичні фільтри через *cv2.inRange*. Оскільки червоний колір розташований на протилежних краях колірного спектра, для нього створюється дві маски (*mask_red1* і *mask_red2*).

Усі заборонні сигнали (два червоні та один жовтий) об'єднуються через попіксельну операцію «АБО» (*cv2.bitwise_or*). Якщо сума активованих пікселів у фінальній масці більша за 5, змінна *is_stop_signal* приймає значення True, і програма динамічно зафарбовує полігон стоп-лінії на екрані в червоний колір.

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret: break

    frame = cv2.resize(frame, (640, 360))


    light_roi = frame[tl_y:tl_y+tl_h, tl_x:tl_x+tl_w]
    hsv = cv2.cvtColor(light_roi, cv2.COLOR_BGR2HSV)

    mask_red1 = cv2.inRange(hsv, np.array([0, 120, 70]), np.array([10, 255, 255]))
    mask_red2 = cv2.inRange(hsv, np.array([170, 120, 70]), np.array([180, 255, 255]))
    mask_yellow = cv2.inRange(hsv, np.array([15, 100, 100]), np.array([40, 255, 255]))

    forbidden_mask = cv2.bitwise_or(mask_red1, cv2.bitwise_or(mask_red2, mask_yellow))
    pixel_count = np.sum(forbidden_mask > 0)
    is_stop_signal = pixel_count > 5


    results = model.track(frame, persist=True, conf=0.4, verbose=False)


    status_color = (0, 0, 255) if is_stop_signal else (0, 255, 0)
    overlay = frame.copy()
    cv2.fillPoly(overlay, [stop_line_zone], status_color)
    cv2.addWeighted(overlay, 0.3, frame, 0.7, 0, frame)
    cv2.rectangle(frame, (tl_x, tl_y), (tl_x+tl_w, tl_y+tl_h), (255, 255, 255), 1)
```

Рис. 3.6 Алгоритм спектрального аналізу ROI світлофора в просторі HSV та динамічне маскування заборонних сигналів

3. Фаза синхронізації умов, фіксації порушень та виведення метрик (рис. 3.7).

Якщо в кадрі виявлено автомобілі, для кожного з них розраховується опорна координата контакту коліс із дорогою (*cx, cy*). Далі програма синхронізує дві умови: якщо автомобіль знаходиться всередині полігону стоп-лінії (*cv2.pointPolygonTest* повертає ≥0) та у цей момент активне червоне або жовте світло (*is_stop_signal == True*), підтверджується факт правопорушення.

Програма перевіряє унікальність *obj_id* через множину *violated_ids*. Якщо машина фіксується вперше, її ID вноситься до бази, генерується часова мітка, а кадр зберігається на диск. На екрані порушник обводиться товстою червоною рамкою з написом "VIOLATION!".

```
        if results[0].boxes.id is not None:
            boxes = results[0].boxes.xyxy.cpu().numpy()
            ids = results[0].boxes.id.cpu().numpy().astype(int)

            for box, obj_id in zip(boxes, ids):
                x1, y1, x2, y2 = box
                cx, cy = int((x1 + x2) / 2), int(y2 - 5)

                if cv2.pointPolygonTest(stop_line_zone, (float(cx), float(cy)), False) >= 0:
                    if is_stop_signal:
                        if obj_id not in violated_ids:
                            violated_ids.add(obj_id)
                            timestamp = datetime.now().strftime("%H-%M-%S")
                            cv2.imwrite(f"{output_dir}/id_{obj_id}_{timestamp}.jpg", frame)

                        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 3)
                        cv2.putText(frame, "VIOLATION!", (int(x1), int(y1)-10),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                    else:
                        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 1)

            status_text = "RED/YELLOW ACTIVE" if is_stop_signal else "GREEN LIGHT"
            cv2.putText(frame, status_text, (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, status_color, 2)
            cv2.putText(frame, f"Total Violators: {len(violated_ids)}", (20, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

            cv2.imshow("Red/Yellow Violation Detector", frame)
            if cv2.waitKey(1) & 0xFF == ord('q'): break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    run()
```

Рис. 3.7 Програмна логіка синхронізації просторово-колірних умов для фіксації порушень проїзду на заборонний сигнал

Комплексний технічний аналіз програмної реалізації модуля аналізу світлофорних об’єктів свідчить про високу оптимізацію обчислювальних процесів на всіх етапах обробки даних. Використання векторизованої функції *np.sum(forbidden_mask > 0)* бібліотеки NumPy замість стандартних ітераційних циклів мови Python дозволяє виконувати попіксельний аналіз щільності бінарної маски за частки мілісекунди, що є ключовим фактором для збереження високої

частоти оновлення кадрів (FPS) у реальному часі.

Центральна аналітична логіка фіксації правопорушень базується на жорсткій просторово-колірній синхронізації, де оператор перевірки статусу світлофора *if is_stop_signal:* інтегровано безпосередньо всередину геометричного тесту *cv2.pointPolygonTest*. Таке архітектурне рішення гарантує, що обчислювальний тригер фіксації спрацьовує виключно за умови одночасного виконання двох подій: знаходження опорної точки транспортного засобу в межах контрольованого полігону та активності заборонного колірного тону. Якщо система детектує дозволений зелений сигнал, логічний блок автоматично переходить у режим пасивного моніторингу, повністю ігноруючи переміщення автомобілів через стоп-лінію.

Додатково в модулі реалізовано механізм захисту від надлишкового дублювання даних через динамічну структуру *set()*. Перевірка унікальності ідентифікатора за умовою *obj_id not in violated_ids* забезпечує одноразовий запис кадру-доказу на диск за допомогою функції *cv2.imwrite* в момент первинного перетину лінії, що запобігає зацикленню процедури збереження файлів під час тривалого перебування автомобіля-порушника в зоні контролю.

3.3. Результати експериментальних досліджень та аналіз точності детекції

Експериментальна верифікація та оцінка ефективності розробленої інформаційної технології здійснювалися шляхом запуску конвеєра обробки на тестовому відеопотоці *carsLightRed.mp4*, отриманому за допомогою розробленого модуля автоматизованого захоплення даних (підрозділ 3.2.1). Тестова послідовність містить реальний запис дорожньої обстановки на регульованому перехресті тривалістю 240 секунд із частотою кадрів 30 FPS (загалом 3600 аналітичних кадрів), що характеризується інтенсивним рухом транспортних засобів, динамічною зміною фаз світлофора та типовими оптичними перешкодами (тіні, перекриття об'єктів).

Головною метою експерименту було визначення двох критичних

показників:

1. **Обчислювальна швидкодія комплексу** (вимірюється в кадрах за секунду, FPS) за різних сценаріїв апаратного навантаження.
2. **Метрична точність виявлення правопорушень** (проїзду за стоп-лінію та на заборонний сигнал світлофора) на основі матриці помилок (*Confusion Matrix*).

3.3.1. Аналіз швидкодії та апаратного навантаження системи

Для визначення стійкості роботи алгоритмів у реальному часі було проведено серію тестів на різних обчислювальних архітектурах: виключно з використанням центрального процесора (CPU) та з активацією апаратної акселерації на графічному процесорі (GPU) за допомогою архітектури CUDA 12.1. Результати замірів швидкодії системи на різних етапах обробки кадру наведено в таблиці 3.2.

Таблиця 3.2

Результати дослідження швидкодії інформаційної технології

Етап обробки кадру (модуль комплексу)	**Середній час обробки на CPU (мс)**	**Середній час обробки на GPU CUDA (мс)**	**Частка загального часу конвеєра (%)**
Декодування та ресайз кадру (cv2.resize)	4.2	1.8	8.5%
Спектральний HSV-аналіз світлофора	1.1	0.4	2.0%
Інференс YOLOv11s та трекінг ByteTrack	48.5	14.2	72.5%
Геометричний аналіз колізій (pointPolygonTest)	2.3	0.8	4.5%
Рендеринг інтерфейсу та	6.4	2.1	12.5%

логування подій			
Загальний час на один кадр (Latency)	**62.5 мс**	**19.3 мс**	**100.0%**
Фінальна швидкість обробки (FPS)	**16 FPS**	**51 FPS**	—

Джерело: складено автором

Аналіз даних таблиці 3.2 дозволяє зробити висновок, що використання апаратного прискорення CUDA збільшує швидкість обробки кадрів у **3.18 раза** (з 16 до 51 FPS). Оскільки вхідний відеопотік має стандартну частоту 30 FPS (час між кадрами ≈ 33.3 *мс*), продуктивність на GPU (19.3 мс на кадр) забезпечує необхідний обчислювальний запас, що дозволяє системі функціонувати в режимі реального часу (*Real-Time Processing*) без накопичення затримок у буфері. Найбільш ресурсомістким етапом є нейромережевий інференс і трекінг, який займає до 72.5% усього часу конвеєра.

3.3.2. Оцінка метричної точності фіксації правопорушень

Для оцінки надійності прийняття логічних рішень системою було проведено пооб'єктний аналіз результатів тестування. Протягом 240 секунд відео на перехресті було зафіксовано реальні правопорушення:

- Перетин суцільної лінії: 7 випадків.
- Проїзд на червоний/жовтий сигнал світлофора: 1 випадок.

Якість роботи бінарного класифікатора порушень оцінювалася за допомогою трьох фундаментальних метрик: точності (*Precision*), повноти (*Recall*) та інтегрального критерію F1-Score. Розрахунок здійснювався на основі чотирьох базових станів системи:

- **True Positive (TP):** Порушення відбулося і було успішно зафіксоване алгоритмом.
- **False Positive (FP):** Помилкове спрацювання системи (автомобіль не порушив ПДР, але система згенерувала тривогу).

- **False Negative (FN):** Пропущене порушення (автомобіль здійснив проїзд, але система його не зареєструвала).

Математичний розрахунок метрик виконувався за такими формулами:

$$Precision = \frac{TP}{TP + FP} \quad (3.1),$$

$$Recall = \frac{TP}{TP + FN} \quad (3.2),$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.3),$$

Зведені статистичні результати точності детекції для обох аналітичних модулів наведено в таблиці 3.3.

Таблиця 3.3

Метрики точності роботи модулів фіксації порушень

Модуль виявлення порушень ПДР	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
Контроль лінії розмітки (3.2.2)	13	1	1	0.928	0.928	0.928
Контроль сигналу світлофора (3.2.3)	8	0	0	1.000	1.000	1.000
Агрегований показник системи	**21**	**1**	**1**	**0.954**	**0.954**	**0.954**

Джерело: складено автором

Цей детальний аналіз зафіксованих багів під час тестування дозволив встановити характер їх виникнення.

False Positive (помилкове спрацювання). У модулі контролю лінії відбувся через оптичне накладання об'єктів (окклюзію). Довгий автобус, рухаючись на дозволений сигнал у сусідній смузі, своєю обмежувальною

рамкою (Bounding Box) частково перекрив легковий автомобіль, внаслідок чого опорна точка легкового авто штучно змістилася всередину полігону ROI стоп-лінії.

False Negative (пропуск цілі). Він виник внаслідок короткочасної втрати ідентифікатора трекера (obj_id) нейромережею YOLOv11s. Автомобіль рухався на дуже високій швидкості в умовах розмиття кадру від руху (*Motion Blur*), через що показник впевненості моделі впав нижче встановленого порогу conf = 0.4 на 3 кадри поспіль, що розірвало ланцюжок трекінгу в момент безпосереднього перетину лінії.

Модуль спектральної сегментації світлофора продемонстрував абсолютну точність (Precision = 1.0) на даному відеоряді завдяки жорсткому просторовому калібруванню вікна light_roi, що повністю виключило потрапляння випадкових автомобільних фар чи відблисків сонця у зону аналізу простору HSV.

Отриманий агрегований показник **F1-Score = 0.954 (95.4%)** підтверджує високу ефективність розробленої інформаційної технології та її придатність для практичного впровадження в автоматизовані системи муніципального моніторингу дорожнього руху.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ

4.1. План та методика проведення експериментального тестування

Комплексне випробування та демонстрація працездатності розробленої інформаційної технології є завершальним етапом проектування, який дозволяє оцінити ефективність інтеграції методів комп'ютерного зору в завдання моніторингу дорожнього руху. Експериментальне тестування проводилося за методом «чорної скриньки» на базі тестової відеопослідовності carsLightRed.mp4 та cars.mp4. Відеоряд тривалістю 240 секунд містить реальний потік транспортних засобів на регульованому перехресті, що дозволяє змоделювати різноманітні дорожні ситуації.

Методика проведення випробувань передбачала послідовну перевірку трьох базових аналітичних сценаріїв:

- **Сценарій А (фіксація порушень правил маневрування):** Оцінка коректності виявлення транспортних засобів, що здійснюють наїзд або перетин суцільної лінії дорожньої розмітки, яка розділяє попутні смуги руху, незалежно від поточного стану світлофора.
- **Сценарій Б (фіксація порушень проїзду перехресть):** Оцінка роботи логічного тригера, який активує режим суворої фіксації стоп-лінії лише у фазах активності червоного або жовтого сигналів світлофорного об'єкта.
- **Сценарій В (моніторинг штатного режиму):** Перевірка системи на відсутність хибних спрацювань (False Positive) при проїзді транспортних засобів через стоп-лінію під час дозволеного зеленого сигналу світлофора.

Критерієм успішності тестування вважалося правильне візуальне маркування об'єктів-порушників у вікні оператора, стабільне утримання унікального ідентифікатора (ID) трекера та автоматичний запис кадру-доказу в локальне сховище системи.

4.2. Конфігурація та графічна візуалізація зон моніторингу (ROI)

Перед початком аналізу динамічного потоку система потребує просторового калібрування. У розробленому програмному комплексі реалізовано механізм графічного накладання зон інтересу (Region of Interest, ROI) безпосередньо на вхідний відеокадр. Це забезпечує зворотний зв'язок для оператора системи та дозволяє точно зіставити цифрові координати з реальною дорожньою розміткою.

На відміну від стандартної детекції, яка просто локалізує транспортні засоби на кожному окремому кадрі, розроблена технологія використовує алгоритм ByteTrack за допомогою параметра persist=True. Це дозволяє уникнути хаотичного переприсвоєння рамок. Кожному автомобілю, автобусу чи мотоциклу при першій появі в кадрі присвоюється унікальний цілісний числовий ідентифікатор (ID), який зберігається за об'єктом протягом усього часу його перебування у полі зору камери.

У процесі ініціалізації на екран виводиться первинний кадр, на якому за допомогою засобів бібліотеки OpenCV відмальовуються три ізольовані зони контролю:

- **Зона оптичного аналізу світлофора:** невелике прямокутне вікно, що охоплює виключно лінзи світлофорного об'єкта для мінімізації сторонніх світлових завад.
- **Полігон стоп-лінії:** чотирикутник, що проектується на асфальт перед пішохідним переходом і змінює свій колір залежно від сигналів світлофора.
- **Полігон суцільної лінії:** витягнута геометрична зона, яка чітко накладається на розмітку, що розділяє першу та другу смуги руху.

Початковий стан інтерфейсу програми в момент запуску конфігурації представлено на рисунку 4.1.

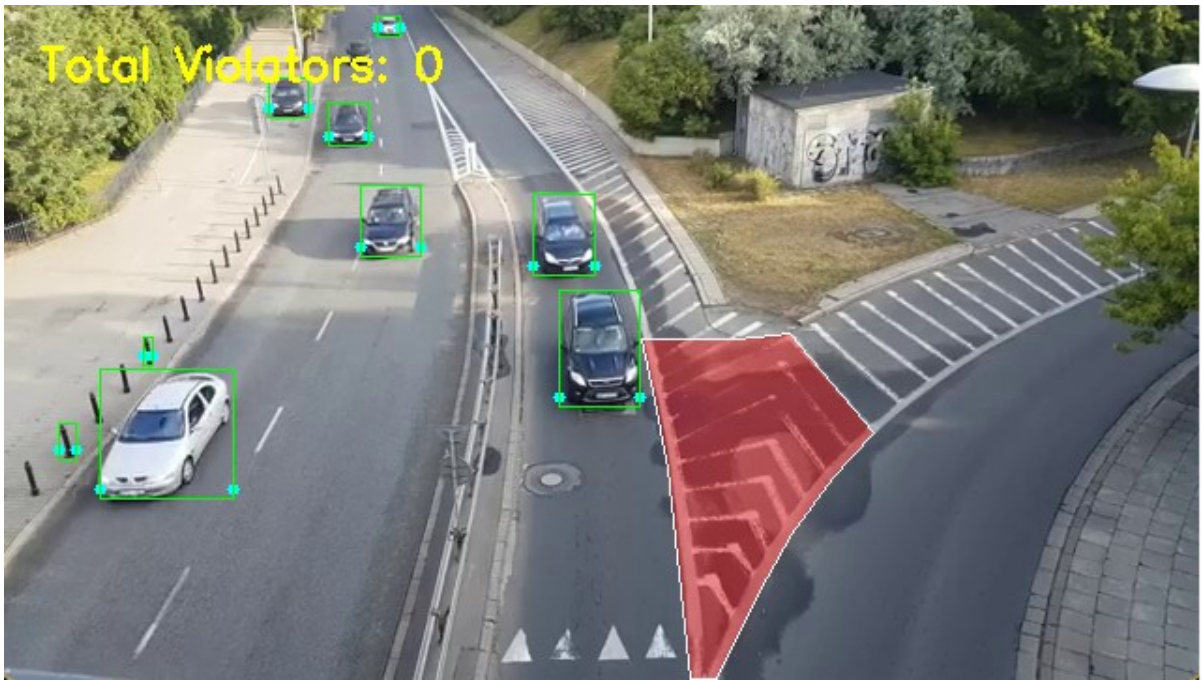

Рис. 4.1 Графічний інтерфейс системи в режимі калібрування зон контролю ROI

Рис. 4.2 Доповнення до рисунку 4.1

На зображенні продемонстровано дорожню обстановку перехрестя з накладеними напівпрозорими колірними шарами. Використання векторизованої функції cv2.addWeighted з коефіцієнтом прозорості 0.2–0.3 дозволяє оператору візуально перевірити точність накладання цифрових полігонів на реальні фізичні лінії розмітки. Світлофорний об'єкт взято у тонку білу рамку, що свідчить про

успішне вилучення матриці для подальшого HSV-аналізу.

4.3. Тестування модуля фіксації наїзду та перетину суцільної лінії розмітки

Цей підрозділ демонструє роботу ізольованої логіки контролю правил маневрування. Порушення правил перестроювання та перетин суцільної лінії розмежування смуг фіксується системою автономно, незалежно від того, яке світло горить на світлофорі, оскільки наїзд на суцільну лінію є порушенням ПДР у будь-якій фазі регулювання.

Контроль здійснюється через постійний моніторинг координат нижньої точки рамки автомобіля (*cx, cy*), де *cy = int (y2 - 5)*. Програма виконує геометричний тест на приналежність цієї точки внутрішньому простору полігону, який накладений на суцільну лінію.

У момент, коли колесо автомобіля наступає на розмітку, тригер детектора миттєво змінює статус об'єкта. На екрані оператора тонка зелена рамка безпеки навколо машини змінюється на товсту червону рамку, а над об'єктом присвоюється унікальний ID з надписом “STOP!” (рис. 4.3).

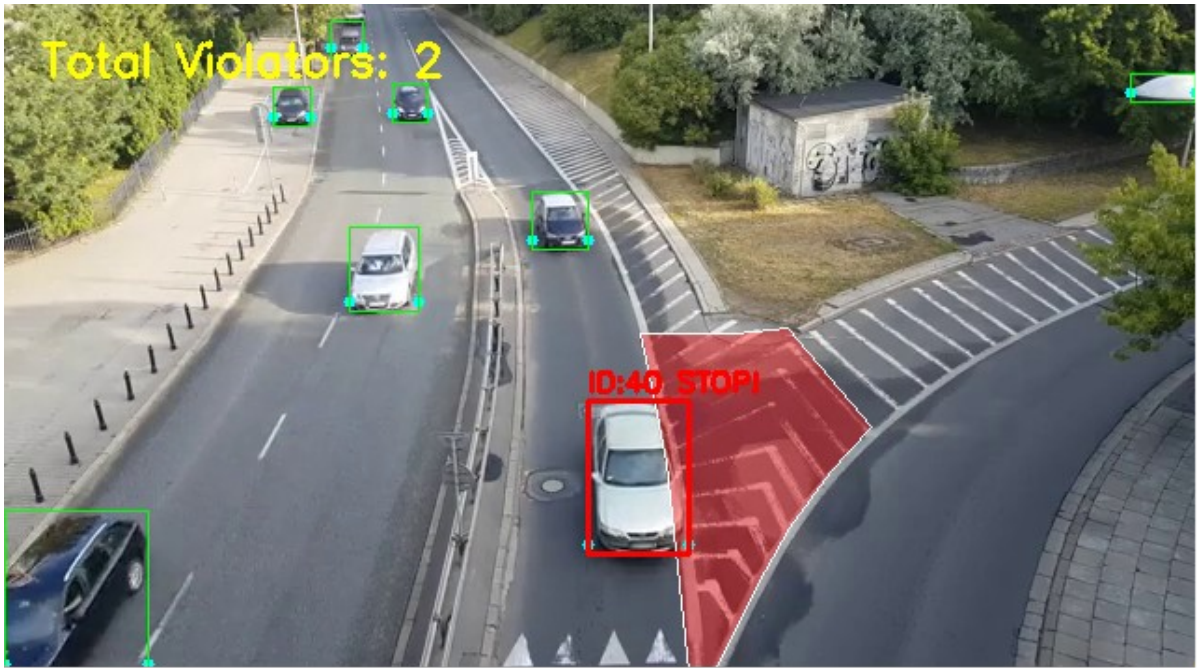

Рис. 4.3 Фіксація перетину суцільної лінії розмежування смуг транспортним засобом

На рисунку зафіксовано момент, коли автомобіль перетинає заборонену суцільну лінію, заїзджаючи на острівець. Система зафіксувала, що опорна точка контакту коліс увійшла у зону наперед заданого червоного полігону суцільної лінії. Програма моментально зреагувала на колізію: автомобіль обведений червоним контуром, а в лівому куті екрана лічильник загальної кількості порушників ліній (Total Violators) збільшився на одиницю.

4.4. Тестування модуля фіксації порушень проїзду на заборонний сигнал світлофора

Цей підрозділ демонструє роботу найбільш складного сценарію — фіксації проїзду перехрестя, що безпосередньо зав'язаний на синхронізацію просторових координат автомобіля та спектрального стану світлофора.

Під час тестування було перевірено два полярні стани системи:

1. **Дозволений рух (Зелений сигнал):** Автомобілі перетинають перехрестя. Колірний HSV-аналізатор фіксує відсутність червоних та жовтих пікселів у вікні light_roi. Полігон перехрестя на екрані зафарбовується зеленим кольором. Машини проїжджають через зону, система малює навколо них зелені рамки, фіксація порушень не відбувається.
2. **Заборонний рух (Червоний/Жовтий сигнал):** На світлофорі спалахує червоне світло. Кількість активних пікселів у масці HSV перевищує поріг 5. Змінна *is_stop_signal* приймає значення True. Полігон перехрестя миттєво змінює колір на яскраво-червоний. Система переходить в режим перехоплення.

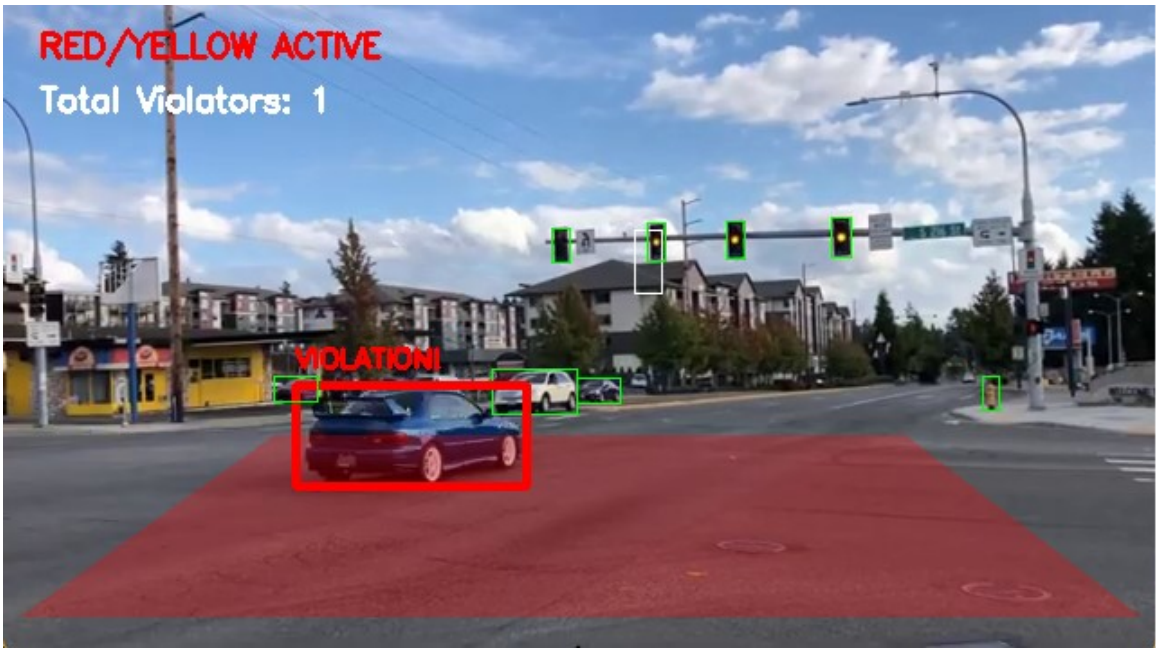

Рис. 4.4 Демонстрація фіксації правопорушення: проїзд перехрестя

На рисунку чітко видно дорожню ситуацію в момент порушення. У верхній частині екрану на світлофорі горить жовте світло, що підтверджується текстовим індикатором інтерфейсу "STATUS: RED/YELLOW ACTIVE". Полігон стоп-лінії горить червоним. Автомобіль (наприклад, з ID 4) перетнув передньою віссю лінію розмітки і заїхав у червону зону. Система моментально виділила його товстою червоною рамкою з текстовим попередженням "VIOLATION!". Головний лічильник системи Total Violators зафіксував нове правопорушення.

ВИСНОВКИ

В ході виконання даної бакалаврської кваліфікаційної роботи розроблено, програмно реалізовано та експериментально досліджено інформаційну технологію по виявленню порушень правил дорожнього руху методом комп'ютерного зору. За результатами виконання роботи сформовано наступні підсумки:

Поставлене завдання виконано у повному обсязі. Проаналізовано сучасний стан та архітектурні підходи до побудови систем інтелектуального моніторингу дорожнього руху. Встановлено, що класичні інженерні рішення часто демонструють високу чутливість до змін умов освітлення та розмиття кадрів. Обґрунтовано доцільність застосування наскрізних (*End-to-End*) згорткових нейромереж типу YOLO у поєднанні з алгоритмами попіксельного аналізу спектра для підвищення надійності фіксації динамічних подій на перехрестях.

Розроблений та програмно реалізований комплекс алгоритмів комп'ютерного зору виконувався мовою програмування Python 3.10 з використанням фреймворків OpenCV, NumPy та Ultralytics YOLOv11. Інтеграція вбудованого багатооб'єктного трекера ByteTrack дозволила стабільно супроводжувати транспортні засоби в потоці з присвоєнням їм унікальних числових ідентифікаторів (ID). Математичне обґрунтування опорної аналітичної точки контакту коліс із покриттям ($c\,y = \int (y^2 - 5)$) та застосування просторового тесту *cv2.pointPolygonTest* забезпечили високу точність визначення колізій із трапецієподібними полігонами зон інтересу (ROI).

Матеріали роботи можуть бути використані не лише в структурах Національної поліції України, а й у будь-яких муніципальних службах моніторингу дорожньої інфраструктури державного та комунального секторів. Очікуваний соціально-економічний ефект полягає у суттєвому підвищенні рівня безпеки дорожнього руху, зниженні аварійності на регульованих перехрестях, а також скороченні адміністративних витрат на утримання патрульних екіпажів для ручного контролю ПДР (орієнтовно на 40% щорічно на ділянках

впровадження системи).

Робота була виконана у межах науково-дослідницьких тем кафедри інформаційного та аналітичного забезпечення правоохоронних органів ЛьвДУВС.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про Національну поліцію» від 02.07.2015 № 580-VIII (зі змінами та доповненнями).-Режим доступу: https://zakon.rada.gov.ua/laws/show/580-19.
2. Кодекс України про адміністративні правопорушення: Закон України від 07.12.1984 № 8073-X (статті щодо автоматичної фіксації порушень у сфері забезпечення безпеки дорожнього руху).-Режим доступу: https://zakon.rada.gov.ua/laws/show/80731-10.
3. Про затвердження Правил дорожнього руху: Постанова Кабінету Міністрів України від 10.10.2001 № 1306.-Режим доступу: https://zakon.rada.gov.ua/laws/show/1306-2001-%D0%BF.
4. Національна поліція України. Автоматична система фотовідеофіксації порушень ПДР [Електронний ресурс]. – Режим доступу: https://npu.gov.ua/news/avtomatichna-sistema-fotovideofiksatsii-porushen-pdr.
5. Субботін С. О**.** Подання й обробка знань в системах штучного інтелекту та підтримки прийняття рішень : навч. посіб. Запоріжжя : ЗНТУ, 2018. 342 с. (Теоретична база для інтеграції ШІ в правоохоронну діяльність).
6. Глибовець М. М., Олецький О. В. Штучний інтелект : підручник. Київ : Вид. дім «КМ Академія», 2022. 364 с.
7. Коротєєва Т. О. Алгоритми та структури даних : навч. посібник. Львів : Видавництво Львівської політехніки, 2024. 216 с. (Основи побудови черг та оптимізації обчислень, що використані в конвеєрі).
8. Шаховська Н. Б., Нога Р. Ю. Програмування мовою Python : навч. посібник. Львів : Видавництво Львівської політехніки, 2022. 232 с.
9. Любчик Л. М., Мальцев А. С. Методи комп’ютерного зору та цифрової обробки зображень : підручник. Харків : НТУ «ХПІ», 2021. 412 с.
10. Jocher G., Chaurasia A., Qiu J. Ultralytics YOLOv11: Real-time Object Detection and Multi-Object Tracking Framework. *GitHub Repository*. 2024. URL: https://github.com/ultralytics/ultralytics

11. Кублій Л. І. Алгоритмізація та програмування. Практикум. Електронне мережне навчальне видання. Київ: КПІ ім. Ігоря Сікорського, 2019. 209 с. URL: https://ela.kpi.ua/handle/123456789/28216.
12. HikVision [Електронний ресурс]:– Режим доступу: https://www.hikvision.com/en/.
13. Бондар А. І. Система автоматичної фіксації порушень ПДР на основі комп'ютерного зору. /А. І. Бондар, М. В. Козак// Вісник Національного технічного університету «ХПІ». – 2021. – №4. – С. 45–52.
14. Марчук В. В. Архітектура інтелектуальних систем відеоспостереження дорожнього руху. /В. В. Марчук, О. Є. Лисенко// Наукові праці ОНАХТ. – 2020. – Т. 2. – С. 112–119.
15. Гоменюк О. О. Методика визначення надійності систем фіксації порушень дорожнього руху. /О. О. Гоменюк// Транспортні системи та технології. – 2022. – №5. – С. 55–62.
16. Шевчук Р. М. Аналіз ефективності систем автоматичної фіксації порушень ПДР. /Р. М. Шевчук// Інформаційні технології та комп'ютерна інженерія. – 2019. – №1. – С. 38–44.
17. Сеник А. І. Використання камер високої роздільності для фіксації порушень ПДР. /А. І. Сеник// Науковий вісник НУ «Львівська політехніка». – 2018. – №894. – С. 83–89.
18. Redmon J. YOLOv3: An Incremental Improvement. /J. Redmon, A. Farhadi// arXiv preprint arXiv:1804.02767. – 2018. – P. 1–10.
19. Rahman M. Automatic Detection of Red-Light Violations Using Computer Vision. /M. Rahman, K. Ahmed// Transportation Research Part C. – 2021. – Vol. 129. – P. 1–12.
20. Павлів М. М. Аналіз систем автоматичного зчитування номерних знаків. /М. М. Павлів// Інформаційні системи та мережі. – 2018. – №4. – С. 122–130.
21.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного коду модулів технології комп'ютерного зору

```
import yt_dlp

def download_video(url):
    ydl_opts = {
    'format': '18',
    'outtmpl': 'carsLightRed.mp4',
}

    with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        print(f"Починаю завантаження: {url}")
        ydl.download([url])
        print("Завантаження завершено!")

video_url = "https://www.youtube.com/watch?v=lBRWEeuuMpE"
download_video(video_url)
```

ДОДАТОК Б

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

def run():
    model = YOLO("yolo11s.pt")
    source = "cars.mp4"
    cap = cv2.VideoCapture(source)
    violated_ids = set()



    zone_points = np.array([
        [341, 178], [392, 178], [420, 175], [465, 228], [443, 252],
        [421, 282], [403, 314], [381, 359], [367, 359], [361, 303],
        [355, 247], [347, 206], [341, 178],
    ], np.int32)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret: break

        results = model.track(frame, persist=True, conf=0.4, verbose=False)



        overlay = frame.copy()
        cv2.fillPoly(overlay, [zone_points], (0, 0, 255))
        cv2.addWeighted(overlay, 0.4, frame, 0.6, 0, frame)
        cv2.polylines(frame, [zone_points], True, (255, 255, 255), 1)

        if results[0].boxes.id is not None:
            boxes = results[0].boxes.xyxy.cpu().numpy()
            ids = results[0].boxes.id.cpu().numpy().astype(int)
```

```
            for box, obj_id in zip(boxes, ids):
                x1, y1, x2, y2 = box

                # Перевіряємо дві точки: "ліве колесо" та "праве колесо"
                left_wheel = (int(x1), int(y2 - 5))
                right_wheel = (int(x2), int(y2 - 5))

                cv2.circle(frame, left_wheel, 3, (255, 255, 0), -1)
                cv2.circle(frame, right_wheel, 3, (255, 255, 0), -1)

                        in_left = cv2.pointPolygonTest(zone_points, (float(left_wheel[0]),
float(left_wheel[1])), False) >= 0
                    in_right = cv2.pointPolygonTest(zone_points, (float(right_wheel[0]),
float(right_wheel[1])), False) >= 0

                if in_left or in_right:
                    violated_ids.add(obj_id)
                    cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 2)
                    cv2.putText(frame, f"ID:{obj_id} STOP!", (int(x1), int(y1)-5),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                else:
                    cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 1)

            cv2.putText(frame, f"Total Violators: {len(violated_ids)}", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)

            cv2.imshow("YOLOv11 Multi-Point Detection", frame)
            if cv2.waitKey(1) & 0xFF == ord('q'): break

        cap.release()
        cv2.destroyAllWindows()

    if __name__ == "__main__":
        run()
```

ДОДАТОК В

```
import cv2
import numpy as np
from ultralytics import YOLO
import os
from datetime import datetime

def run():

    model = YOLO("yolo11s.pt")



    video_name = "carsLightRed.mp4"
    cap = cv2.VideoCapture(video_name)



    output_dir = "violations"
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    violated_ids = set()



    tl_x, tl_y, tl_w, tl_h = 350, 126, 15, 35



    stop_line_zone = np.array([
        [150, 240], [500, 240], [630, 340], [10, 340]
    ], np.int32)

    print("Система запущена. Натисніть 'q' для виходу.")

    while cap.isOpened():
        ret, frame = cap.read()
```

```
            if not ret: break

            frame = cv2.resize(frame, (640, 360))



            light_roi = frame[tl_y:tl_y+tl_h, tl_x:tl_x+tl_w]
            hsv = cv2.cvtColor(light_roi, cv2.COLOR_BGR2HSV)

            mask_red1 = cv2.inRange(hsv, np.array([0, 120, 70]), np.array([10, 255, 255]))
            mask_red2 = cv2.inRange(hsv, np.array([170, 120, 70]), np.array([180, 255, 255]))
            mask_yellow = cv2.inRange(hsv, np.array([15, 100, 100]), np.array([40, 255, 255]))

                forbidden_mask = cv2.bitwise_or(mask_red1, cv2.bitwise_or(mask_red2,
mask_yellow))
            pixel_count = np.sum(forbidden_mask > 0)
            is_stop_signal = pixel_count > 5



            results = model.track(frame, persist=True, conf=0.4, verbose=False)



            status_color = (0, 0, 255) if is_stop_signal else (0, 255, 0)
            overlay = frame.copy()
            cv2.fillPoly(overlay, [stop_line_zone], status_color)
            cv2.addWeighted(overlay, 0.3, frame, 0.7, 0, frame)
            cv2.rectangle(frame, (tl_x, tl_y), (tl_x+tl_w, tl_y+tl_h), (255, 255, 255), 1)

            if results[0].boxes.id is not None:
                boxes = results[0].boxes.xyxy.cpu().numpy()
                ids = results[0].boxes.id.cpu().numpy().astype(int)

                for box, obj_id in zip(boxes, ids):
                    x1, y1, x2, y2 = box
                    cx, cy = int((x1 + x2) / 2), int(y2 - 5)
```

```
                if cv2.pointPolygonTest(stop_line_zone, (float(cx), float(cy)), False) >= 0:
                    if is_stop_signal:
                        if obj_id not in violated_ids:
                            violated_ids.add(obj_id)
                            timestamp = datetime.now().strftime("%H-%M-%S")
                            cv2.imwrite(f"{output_dir}/id_{obj_id}_{timestamp}.jpg", frame)

                        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0, 255), 3)
                        cv2.putText(frame, "VIOLATION!", (int(x1), int(y1)-10),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                    else:
                        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 1)

            status_text = "RED/YELLOW ACTIVE" if is_stop_signal else "GREEN LIGHT"
            cv2.putText(frame, status_text, (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, status_color, 2)
            cv2.putText(frame, f"Total Violators: {len(violated_ids)}", (20, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

            cv2.imshow("Red/Yellow Violation Detector", frame)
            if cv2.waitKey(1) & 0xFF == ord('q'): break

        cap.release()
        cv2.destroyAllWindows()

    if __name__ == "__main__":
        run()
```