МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,

д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV

Спеціальність 123 Комп'ютерна інженерія

123 – КМР.ПЗ.00 – 605.21810516

Студент _____М. М. Матюшок
підпис
«__» _____ 202__ р.

Керівник канд. фіз.-мат. наук, доц. _____ С. В. Пузирьов
підпис
«__» _____ 202__ р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної магістерської роботи

Видано студенту групи 605м факультету комп'ютерних наук

Матюшку Максиму Миколайовичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV

Затверджена наказом по ЧНУ ім. Петра Могили від 08.11.2023 № 226.

2. Строк представлення кваліфікаційної роботи «____» ___ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є: апаратне та програмне забезпечення моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV.

4. Перелік питань, що підлягають розробці
1) огляд сучасних підходів та моніторингових систем контролю дорожнього руху;
2) аналіз переваг та недоліків існуючих систем;
3) розробка апаратної частини;
4)розробка програмної частини імплементації розпізнавання рухомих об'єктів засобами OpenCV;

5. Перелік графічних матеріалів
слайди презентації

6. Завдання до спеціальної частини _

Проведення оцінки умов праці фахівців підприємства, а саме аналіз виробничого приміщення, оцінка природного освітлення. Визначення рівня електричної та пожежної безпеки підприємства

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи
_ канд. фіз.-мат. наук, доц Пузирьов Сергій Володимирович _____
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_Матюшок Максим Миколайович_____
(прізвище, ім’я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема: Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2023	15.09.2023	*Виконано*
2.	Огляд літератури за темою роботи	15.09.2023	01.10.2023	*Виконано*
3.	Складання календарного плану КМР	01.10.2023	03.10.2023	*Виконано*
4.	Аналіз предметної області	05.10.2023	25.10.2023	*Виконано*
5.	Розробка проєктних рішень	25.10.2023	10.11.2023	*Виконано*
6.	Моделювання	10.11.2023	15.11.2023	*Виконано*
7.	Конструювання АПЗ	15.11.2023	20.12.2023	*Виконано*
8.	Перевірка працездатності, тестування та апробація розробленого АПЗ	20.12.2023	05.01.2024	*Виконано*
9.	Аналіз результатів тестування	05.01.2024	10.01.2024	*Виконано*
10.	Відгук керівника КМР	10.01.2024	05.02.2024	*Виконано*
11.	Оформлення КМР та презентації	07.02.2024	08.02.2024	*Виконано*
12.	Попередній захист	31.01.2024	14.02.2024	*Виконано*
13.	Рецензування	14.02.2024	18.02.2024	*Виконано*
14.	Захист кваліфікаційної роботи	26.02.2024	26.02.2024	*Виконано*

Розробив здобувач ВО Матюшок Максим Миколайович_____ 

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи Пузирьов Сергій Володимирович____ _____

(підпис)

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV»

Студент гр. 605м Матюшок Максим Миколайович

Керівник: доцент, канд. фіз.-мат. наук Пузирьов Сергій Володимирович

Актуальність теми кваліфікаційної роботи обумовлена швидкими темпами збільшення кількості транспортних засобів на дорогах та збільшення кількості ДТП у результаті порушення правил дорожнього руху.

Концепція полягає у створенні моніторингової мережі контролю дорожнього руху на руху на основі Raspberry Pi та OpenCV, для контролю дорожнього руху та фіксації порушень.

Для реалізації даної системи потрібно проаналізувати сучасні підходи детектування рухомих об’єктів та обчислення їх швидкості. Серед цих методів слід відмітити найбільш сучасні та поширені, а саме методи, які використовують нейронні мережі для детектування об’єктів. Найбільш поширені – це R-CNN або Fast R- CNN, YOLO, Single Shot Detector (SSDs).

Метою даної роботи є розробка моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Завданням кваліфікаційної магістерської роботи є:

– розробити систему моніторингу дорожнього руху, яка базується на Raspberry Pi та OpenCV;

– розробити алгоритми виявлення автомобілів, пішоходів та інших учасників дорожнього руху;

– реалізувати збір та передачу відеоданих з камер до Raspberry Pi;

– розробити зручний інтерфейс для користувача для перегляду результатів моніторингу та взаємодії з системою;

– провести експерименти для оцінки ефективності та точності розроблених алгоритмів.

Об’єкт дослідження (розробки): процеси моніторингу та контролю дорожнього руху за допомогою Raspberry Pi та OpenCV.

Предмет дослідження (розробки): методи та засоби моніторингу й обробки даних для системи контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Мета: розробити моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Для досягнення поставленої мети було поставлено такі завдання:

– провести аналіз сучасних систем, визначити їх переваги та недоліки;

– проаналізувати сучасні методи та алгоритми детектування об’єктів;

– підібрати згідно з вимогами апаратну складову;

– розробити програмну частину детектування рухомих об’єктів і обчислення їх швидкості;

– провести експерименти для оцінки ефективності та точності розробленої системи.

Кваліфікаційна робота містить: перелік скорочень, вступ, чотири розділи, висновок, перелік джерел посилання та два додатки.

У вступі подані основні аргументи, що розкривають актуальність вибраної теми, а також визначено об'єкт, предмет дослідження, мету і завдання, які необхідно виконати для досягнення цієї мети.

В першому розділі проведено огляд існуючих моніторингових систем контролю дорожнього руху. Визначені які технології використовують дані системи та які мають переваги і недоліки.

Другий розділ містить детальний огляд методів детектування об’єктів, їх переваги та недоліки.

У третьому розділі міститься опис процесу розробки програмної частини та опис обраної апаратної частини.

У четвертому розділі розглядається процес тестування розробленого програмного та апаратного забезпечення, перевіряється коректність роботи інтегрованого алгоритму детектування об’єктів та визначення рівня навантаження на систему під час виконання коду.

У висновку описано результати виконання кваліфікаційної роботи.

Додатки містять блок схему розробленої програмної частини та інформацію про апробацію кваліфікаційної роботи.

У розділі, присвяченому охороні праці, розглянуті заходи, що гарантують виконання вимог охорони праці на робочому місці.

Кваліфікаційна робота містить 64 сторінки (без додатків), 43 рисунків, 30 джерел посилання, 3 додатки.

ASTRACT

to the qualifying master's thesis

"Traffic control monitoring system based on Raspberry Pi and OpenCV"

Student: Matiushok Maksym Mykolayovych

Supervisor: associate professor, candidate physics and mathematics Science
Puzyryov Serhii Volodymyrovych

The relevance of the thesis topic is driven by the rapid increase in the number of vehicles on roads and the rising number of road accidents due to traffic rule violations. The concept involves creating a monitoring network for traffic control based on Raspberry Pi and OpenCV to oversee traffic and record violations.

To implement this system, a thorough analysis of modern approaches to detecting moving objects and calculating their speed is necessary. Notably, methods utilizing neural networks for object detection, such as R-CNN, Fast R-CNN, YOLO, and Single Shot Detector (SSDs), are widely recognized.

The goal of the thesis is to develop a monitoring network for traffic control based on Raspberry Pi and OpenCV. The tasks include:

Developing a traffic monitoring system based on Raspberry Pi and OpenCV.

Designing algorithms for detecting cars, pedestrians, and other traffic participants.

Implementing data collection and transmission from cameras to Raspberry Pi.

Creating a user-friendly interface for viewing monitoring results and interacting with the system.

Conducting experiments to evaluate the effectiveness and accuracy of the developed algorithms.

The object of research (development) is the processes of monitoring and controlling traffic using Raspberry Pi and OpenCV. The subject of research (development) is the methods and means of monitoring and data processing for a traffic control system based on Raspberry Pi and OpenCV.

The thesis contains a list of abbreviations, introduction, four chapters, conclusion, a list of references, and two appendices. The introduction presents the main arguments justifying the relevance of the chosen topic and outlines the object, subject of research, as well as the goal and tasks necessary to achieve it.

In the first chapter, an overview of existing traffic monitoring systems is conducted, examining the technologies used and discussing their advantages and disadvantages.

The second chapter provides a detailed review of object detection methods, highlighting their pros and cons.

The third chapter describes the process of developing the software part and the chosen hardware component.

The fourth chapter covers the testing process of the developed software and hardware, verifying the correctness of the integrated object detection algorithm, and determining the system's load during code execution.

The conclusion summarizes the results of the thesis.

The appendices include a block diagram of the developed software and information about the thesis's validation.

The section on occupational safety discusses measures to ensure compliance with safety requirements in the workplace.

The thesis consists of 64 pages (excluding appendices), includes 43 figures, 30 references, and 3 appendices.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

OS	Operating system
CSI	Camera Serial Interface
SSH	Secure Shell
VNC	Virtual Network Computing
DNN	Deep Neural Network
ОС	Операційна систена
з/п	за порядком
РМ	робоче місце
АПЗ	апаратно-програмне забезпечення

ВСТУП

Сучасний світ стрімко розвивається, збільшуючи об'єм транспортного руху та ставлячи перед суспільством виклики в області безпеки на дорогах та оптимізації управління транспортним потоком. У цьому контексті розробка та впровадження ефективних систем моніторингу дорожнього руху стають актуальною проблемою, яка вимагає інноваційних рішень та передових технологій.

Однією з перспективних областей у цьому контексті є використання моніторингової мережі, побудованої на базі Raspberry Pi та OpenCV. Raspberry Pi, як відома вбудована платформа, та OpenCV, як бібліотека комп'ютерного зору з відкритим вихідним кодом, утворюють потужний інструментарій для реалізації системи, спроможної виявляти, аналізувати та реагувати на різноманітні сценарії дорожнього руху.

Ця технологічна ініціатива не лише має на меті забезпечення безпеки учасників дорожнього руху та оптимізації транспортного потоку, але й відкриває широкі можливості для розвитку «розумних» систем управління дорожнім простором. Впровадження моніторингової мережі на базі Raspberry Pi та OpenCV може стати кроком уперед у напрямку створення інтелектуальних та високоефективних рішень для сучасних викликів у галузі транспортної інфраструктури.

Актуальність теми «Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV» визначається рядом факторів, що стосуються покращення безпеки на дорогах та оптимізації управління транспортним рухом.

По-перше, забезпечення безпеки дорожнього руху. Зростання об'єму транспортного потоку та ризику дорожньо-транспортних пригод створює необхідність в розробці ефективних систем моніторингу для попередження аварій та вжиття заходів безпеки.

По-друге, оптимізація транспортного потоку. Використання моніторингової мережі на основі Raspberry Pi та OpenCV може допомогти в реальному часі виявляти перешкоди, затори та інші проблеми на дорогах, що дозволяє управлінцям оптимізувати рух транспорту та запобігати заторам.

По-третє, зменшення кількості ДТП. Система може виявляти порушення правил дорожнього руху, такі як перебігання на червоне світло, неправильний обгін тощо, і сприяти вжиттю заходів для зменшення кількості аварій.

Метою даної роботи є розробка моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Об’єктом дослідження є моніторинг та контроль дорожнього руху за допомогою Raspberry Pi та OpenCV.

Предметом дослідження є методи моніторингу та обробки даних для системи контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Завданням кваліфікаційної магістерської роботи є:

– розробити систему моніторингу дорожнього руху, яка базується на Raspberry Pi та OpenCV;

– розробити алгоритми виявлення автомобілів, пішоходів та інших учасників дорожнього руху;

– реалізувати збір та передачу відеоданих з камер до Raspberry Pi;

– розробити зручний інтерфейс для користувача для перегляду результатів моніторингу та взаємодії з системою;

– провести експерименти для оцінки ефективності та точності розроблених алгоритмів.

Апробація роботи відбулася в рамках XXVI Всеукраїнської щорічної науково-практичної конференції «Могилянські читання – 2023 Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (м. Миколаїв, ЧНУ ім. Петра Могили) [16].

1 АНАЛІТИЧНИЙ ОГЛЯД МОНІТОРИНГОВИХ МЕРЕЖ КОНТРОЛЮ ДОРОЖНЬОГО РУХУ

Моніторингові системи контролю дорожнього руху набирають все більшої популярності та все частіше використовуються не тільки у великих мегаполісах, а й у невеликих містах. Такі системи можуть бути як стаціонарні так і мобільні. Проте вартість таких систем залишається ще доволі високою. З розвитком технологій значно покращились можливості таких систем, підвищилась якість та швидкість розпізнавання, швидкість фіксацій порушень правил дорожнього руху та значно підвищився відсоток правильних розпізнавань номерних знаків автомобілів.

Мета даної роботи – розробка моніторингової мережі контролю дорожнього руху.

Для розпізнавання транспортних засобів та фіксації порушень можна використати бібліотеку OpenCV.

OpenCV [1] (Open Source Computer Vision) - це відкрита бібліотека комп'ютерного зору, призначена для обробки зображень та комп'ютерного зору в реальному часі. Розроблена для підтримки алгоритмів машинного зору, OpenCV надає ряд функцій для виявлення об'єктів, розпізнавання облич, вимірювання об'єктів та багато іншого. Завдяки своїй багатоплатформності та відкритому вихідному коду, OpenCV став широко використовуваною технологією в галузі комп'ютерного зору та обробки зображень.

Для запису відеопотоку буде використовуватися камера для Raspberry Pi [2], яка підключатиметься до мікрокомп’ютера Raspberry Pi, на якому виконуватиметься обробка даних за допомогою методів бібліотеки OpenCV, потім дані передаватимуться на сервер, в разі виявлення порушення сервер відправлятиме відповідне сповіщення.

1.1 Дослідження методів моніторингу дорожнього руху

В даному розділі будуть докладно розглянуті та проаналізовані різноманітні підходи та технології, використовувані для моніторингу дорожнього руху, зокрема з використанням Raspberry Pi та OpenCV [3]. Метою цього дослідження є з'ясування переваг та недоліків кожного методу, їх ефективності в різних умовах та можливості їхнього використання для покращення систем безпеки та управління дорожнім рухом.

Розділ також висвітлить актуальні тенденції у галузі моніторингу дорожнього руху, включаючи нові розробки та інновації, які можуть впливати на ефективність та широке впровадження таких систем у міста та регіони. Аналіз і порівняння різних методів нададуть чітке уявлення про найкращі практики та можливі напрямки для подальших досліджень у цій важливій галузі.

Методи моніторингу дорожнього руху умовно поділяються на методи з використанням однієї камери та методи з використанням двох та більше камер. Деякі з цих методів було розглянуто та виділено їх недоліки та переваги.

Метод визначення швидкості за допомогою відомої дистанції.

Принцип роботи цього метода полягає в наступному. Камера Raspberry Pi фіксує об'єкт (автомобіль), коли він проходить точки А та Б на дорозі. Відома фіксована дистанція між точками А та Б дозволяє обчислити середню швидкість автомобіля за допомогою формули.

$$V = \frac{S}{t} \tag{1.1}$$

Серед переваг слід відзначити простоту реалізації. Цей метод не потребує складних алгоритмів обробки зображення. Також низька обчислювальна складність. Використання лише простих арифметичних операцій.

Вагомим недоліком цього методу є обмежене використання на рівних ділянках. Ефективно працює лише на прямих та рівних ділянках дороги з відомою дистанцією між точками.

Метод використання оптичного потоку.

Даний метод працює за наступним принципом. Камера фіксує зміни положення пікселів між послідовними кадрами. Засновуючись на змінах положення пікселів та часі між кадрами, обчислюється швидкість автомобіля.

До переваг цього методу слід віднести можливість роботи в реальному часі та відстеження руху об’єкта.

Серед недоліків слід виділити чутливість до шуму. На точність вимірювань даного методу впливає рівень шумів на кадрах, чим рівень шумів вищий, тим точність вимірювань нижча. Окрім цього недоліку, є ще один доволі вагомий – метод піддається акумулюванню помилок. Накопичуються помилки з часом, особливо при тривалому використанні.

Метод використання маркерів або об'єктів.

Принцип роботи даного метода полягає в наступному. Камера виявляє та відстежує спеціальні маркери чи об'єкти на дорозі. На основі координат маркерів та часі між кадрами, визначається швидкість руху автомобіля.

Серед переваг слід віднести можливість відстеження руху об'єктів в реальному часі. Та можливість роботи в різних умовах, а саме висока ефективність у різних умовах освітлення та на різних типах доріг.

Проте наявні вагомі недоліки цього методу. Даний метод потребує встановлення спеціальних маркерів чи об'єктів на дорозі. Окрім цього дуже вразливий до виникнення перешкод перед об’єктами-маркерами. В разі виникнення перешкод, які закривають маркери, знижується точність вимірів.

Метод використання алгоритмів відстеження об'єктів.

Використовуються алгоритми відстеження, наприклад алгоритми OpenCV, які визначають об'єкти на кадрах та відстежують їх рух. На основі відомих координатах та часі між кадрами, обчислюється швидкість руху автомобіля.

Вагомою перевагою даного методу – це можливість відстеження руху багатьох об'єктів одночасно.

Окрім вище переглянутих методів є декілька методів з використанням

двох камер.

Стерео-пара для визначення швидкості.

Принцип цього методу полягає в наступному. Використовуються дві камери Raspberry Pi, розташовані на відомій відстані одна від одної (створюючи стереопару). Вимірюється переміщення об'єкта відносно лівої та правої камери. На основі відомої відстані між камерами, та часу обчислюється швидкість об'єкта.

Даний метод забезпечує високу точність вимірювань, оскільки враховує просторовий зсув об'єкта. Можливість визначення глибини. Окрім швидкості можна визначити глибину об'єкта.

Серед недоліків слід виділити складність в отримані точних результатів. Даний метод вимагає точного калібрування та синхронізації камер для отримання точних вимірювань. Окрім цього, вартість та складність обладнання. Вимагає використання двох камер, що може збільшити вартість та рівень складності проєкту.

Використання оптичного потоку з двох камер.

Для роботи цього методу необхідно використати дві камери для фіксації руху об'єкта. За допомогою оптичного потоку вимірюється зміна позицій пікселів на кожній з камер. За допомогою стереозору обчислюється глибина та швидкість руху об'єкта.

Даний метод забезпечує точні вимірювання швидкості в реальному часі. Також метод враховує тривимірний рух об'єкта, що підвищує точність.

Проте точність вимірювань цього методу залежить від умов освітлення, та цей метод вимагає високу обчислювальну потужність, щоб забезпечити високу точність вимірювань в режимі реального часу.

Використання двох камер в системі моніторингу дорожнього руху забезпечує вищу точність вимірювань, ніж системи з однією камерою, також надають можливість враховувати просторові аспекти руху об'єкта, та забезпечує додаткові можливості для аналізу та обробки зображення. Проте

системи, що використовують дві камери мають значно більшу вартість та вимагають точної синхронізації камер для отримання точних результатів вимірювання.

Стисле порівняння методів визначення швидкості об'єкту наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння методів визначення швидкості об'єкту

Метод визначення швидкості	Опис	Переваги	Недоліки
Визначення за допомогою відомої дистанції	Вимірює час, який транспортний засіб проходить між двома точками з відомою відстанню	Простота в реалізації, оптимальне співвідношення ціна-продуктивність	Залежність від точності вимірювання відстані, може бути менш точним при великій швидкості
Використання оптичного потоку	Вимірює різницю в положенні об'єкта між послідовними кадрами	Можливість визначення швидкості навіть за умов обмеженої видимості	Вразливість до зміни освітлення та тіней, може бути менш точним при великих швидкостях
Використання маркерів або об'єктів	Використання розпізнавання маркерів або унікальних об'єктів для визначення руху	Ефективність в умовах обмеженої видимості, висока точність	Потребує правильного калібрування та врахування перешкод або змін в середовищі
Використання алгоритмів відстеження об'єктів	Застосовує алгоритми для відстеження рухомих об'єктів в кадрах	Ефективність у відстеженні руху, висока точність	Може вимагати значних обчислювальних ресурсів, особливо при великій кількості об'єктів або високій роздільності зображення

1.2 Переваги та недоліки моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV

Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV вимагає відносно невеликих витрат, адже використовується мікрокомп’ютер Raspberry Pi та Raspberry Pi Cam, які мають оптимальне співвідношення ціна-продуктивність. Така система забезпечує високий рівень функціональності та продуктивності.

Окрім цього, використання OpenCV надає розробнику багатий набір

методів та алгоритмів для обробки зображень та комп'ютерного зору, а відкритий вихідний код сприяє гнучкості та можливості модифікацій залежно від потреб проєкту.

Мікрокомп’ютер Raspberry Pi має вбудовані GPIO-піни, що дозволяють легко інтегрувати різноманітні датчики та модулі, такі як камери, датчики руху чи датчики відстані. Що в свою чергу дозволяє легко модифікувати систему для певних потреб.

Мікрокомп’ютер Raspberry Pi має низький рівень споживання енергії, що дозволяє економити ресурси та забезпечує стабільну роботу системи на довгий термін, окрім цього, це надає можливість створити мобільну систему моніторингу дорожнього руху, використавши спеціальний для сімейства мікрокомп’ютерів Raspberry Pi модуль живлення та акумулятор.

Проте, сімейство даних мікрокомп’ютерів може не надавати потрібної обчислювальної потужності в досить великих та більш складних системах моніторингу, де вимагається велика кількість обчислень та необхідно обробляти велику кількість відеоданих. Окрім цього, Raspberry Pi та OpenCV можуть вимагати стабільних умов роботи, і ефективність системи може страждати від екстремальних погодних умов чи атмосферних впливів. Також слід зазначити що Raspberry Pi Cam, як і всі інші камери, чутлива до змін умов освітлення. З точки зору кібербезпеки, необхідно реалізувати належний захист такої системи, задля запобігання втрати конфіденційних даних та порушення їх цілісності.

1.3 Огляд конкурентних моніторингових систем контролю дорожнього руху

На сьогодні існують численні сучасні рішення, які використовують камери для визначення швидкості автомобіля на основі відеозаписів. Ці технології використовують різноманітні алгоритми комп'ютерного зору та

глибокого навчання для аналізу відео та визначення швидкості руху автомобіля.

У даному розділі проводиться огляд конкурентних моніторингових систем контролю дорожнього руху, що вже існують на ринку. Аналізуються їхні технічні характеристики, функціональні можливості та переваги, а також виявляються можливі недоліки та обмеження. Цей огляд спрямований на визначення найкращих практик та ідентифікацію прогалин, які можуть бути вирішені або покращені у власній моніторинговій мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV.

Враховуючи швидкий темп розвитку технологій та зростання інтересу до розумних систем управління транспортним потоком, огляд конкурентних рішень є важливим етапом у створенні ефективної та конкурентоспроможної системи моніторингу дорожнього руху.

SPECS3 (Speed Check Services) [4] огляд системи моніторингу швидкості.

Дана система моніторингу швидкості використовує індуктивні петлі, розташовані на дорозі, для визначення руху транспортних засобів. Додатково використовуються відеокамери для відстеження та ідентифікації транспортних засобів.

В даній системі визначення швидкості автомобілей визначається методом вимірювання часу руху об’єкта між індуктивними петлями.

Основне завдання SPECS3 – це вимірювання швидкості транспортних засобів на певній ділянці дороги. Додатково система здатна фіксувати та документувати випадки перевищення швидкості для подальшого використання цієї інформації. Система SPECS3 має вбудовані заходи безпеки, що роблять її менш вразливою до радіочастотного впливу.

Основні переваги системи SPECS3:

– використання індуктивних петель дозволяє отримувати точні та надійні вимірювання швидкості транспортних засобів;

– система може бути успішно використана на важкодоступних ділянках дороги, де інші технології можуть бути менш ефективними;

– інтеграція системи SPECS3 може сприяти зменшенню аварій та порушень швидкісного режиму, завдяки реалізації превентивних заходів та контролю.

Основні недоліки системи SPECS3:

– впровадження системи SPECS3 може бути витратним, зокрема через потребу в інфраструктурних роботах для встановлення індуктивних петель;

– ефективність системи може залежати від стану та правильності встановлення індуктивних петель на дорозі;

– використання індуктивних петель та обмежене використання відеокамер може обмежувати можливості відстеження транспортних засобів в різних сценаріях.

VITRONIC PoliScan Speed [5] огляд системи моніторингу швидкості.

В даній системі використовується цифрова камера високої роздільної здатності для фіксування порушень та ідентифікації транспортних засобів. Для точного вимірювання швидкості транспортних засобів використовується лазерна технологія.

Використовується ефект Доплера для визначення швидкості об'єкта. Вимірюється зміна частоти сигналу, відбитого від рухомого транспортного засобу.

Основні переваги системи VITRONIC PoliScan Speed:

– використання лазерної технології та ефекту Доплера забезпечує високу точність вимірювань швидкості транспортних засобів;

– система ефективно працює в різних погодних умовах та при різних рівнях освітленості;

– здатна виявляти та фіксувати порушення швидкісного режиму на великій відстані, що робить її ефективною для великих ділянок доріг;

– може служити як ефективний інструмент для зниження аварійності та порушень швидкісного режиму через страх водіїв від можливості отримання штрафів.

Основні недоліки системи VITRONIC PoliScan Speed:

– лазерні та високоякісні фотокамери можуть збільшити вартість впровадження системи;

– недостатня чистота лінз чи елементів системи може впливати на якість фотофіксацій та точність вимірювань.

Redflex Speed Enforcement огляд системи моніторингу швидкості.

Система Redflex Speed Enforcement використовує двосторонній радар для вимірювання швидкості транспортних засобів та відеокамеру високої роздільної здатності для фотофіксації порушень та ідентифікації транспортних засобів.

Redflex Speed Enforcement [6] надає можливість миттєво фіксувати транспортні засоби, що перевищують швидкісний режим, за допомогою відеокамери та радару. Також наявна здатність автоматично ідентифікувати та фіксувати номери транспортних засобів.

Основні переваги системи Redflex Speed Enforcement:

– використання радарної технології забезпечує високу точність вимірювань швидкості, навіть на великих відстанях;

– здатність функціонувати на великих відстанях та в різних погодних умовах, забезпечуючи стабільну роботу в ускладнених умовах;

– система має високий ступінь автоматизації, що дозволяє ефективно фіксувати порушення та ідентифікувати транспортні засоби без додаткового втручання операторів;

– система забезпечує можливість інтеграції з іншими системами безпеки для створення комплексного підходу до моніторингу дорожнього руху.

Основні недоліки системи Redflex Speed Enforcement:

– велика вартість встановлення та обслуговування системи може бути фінансово витратною для певних організацій та урядових структур;

– питання щодо приватності можуть виникати через високоточне фіксування та зберігання даних про транспортні засоби та їх рух;

– як і більшість систем моніторингу, ефективність може залежати від стану дорожньої інфраструктури та обладнання.

Порівнюючи вище перелічені системи моніторингу швидкості Redflex Speed Enforcement, VITRONIC PoliScan Speed і SPECS3, слід зазначити, що кожна з них має свої унікальні характеристики та переваги.

Стисле порівняння моніторингових систем дорожнього руху наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння моніторингових систем дорожнього руху

Назва	**Переваги**	**Недоліки**
Redflex Speed Enforcement	Висока точність вимірювань, ефективність на великих відстанях, інтеграційна гнучкість	Висока вартість впровадження та питання щодо приватності
VITRONIC PoliScan Speed	Висока точність вимірювань, ефективність у різних умовах, можливість роботи на великих відстанях.	Висока вартість впровадження та залежність від чистоти елементів системи
SPECS3	Точність вимірювань, ефективність у важкодоступних місцях, можливість реалізації превентивних заходів	Висока вартість впровадження та обмежена можливість відстеження

У кожної системи є свої контекстозалежні переваги та обмеження, і вибір між ними повинен бути здійснений на основі конкретних потреб та умов проєкту. Важливо враховувати вартість впровадження, вимоги до точності вимірювань, а також питання приватності та етичні аспекти при виборі системи моніторингу швидкості для конкретного застосування

1.4 Вимоги до апаратно-програмного забезпечення

Для розробки моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV [7] необхідний модуль камери високої роздільної здатності [8], який надає можливість знімати відео в мінімальній якості 720p, щоб мати змогу розпізнавати номерні знаки для фотофіксації порушень правил дорожнього руху, але щоб збільшити відсоток успішних розпізнавань номерних знаків необхідна якість відео 1080p, проте для поставленої задачі необхідно щоб модуль камери мав змогу робити якомога більше кадрів за секунду, наприклад 60fps та більше, щоб не втрачати точність вимірів.

Щоб забезпечити достатню швидкість обробки даних, треба щоб мікрокомп’ютер мав чотирьох’ядерний центральний процесор, мінімум 1 ГБ оперативної пам’яті, та підтримувати стандарт IEEE 802.11 b/g/n/ac чи мати Gigabit Ethernet порт, щоб забезпечити належну швидкість передачі даних до серверу, чи отримувати дані з серверу. Окрім цього потрібний надійний блок живлення, оригінального виробництва та потрібно реалізувати ефективну систему охолодження, особливо при тривалому використанні. Для збереження операційної системи та іншого програмного забезпечення необхідна мікро-SD карат об’ємом мінімум 16 ГБ. Також наявність додаткових USB-портів для підключення додаткових пристроїв, які можуть бути необхідні для розширення функціонала.

Щодо програмного забезпечення, то операційна система має бути Raspbian [10] або інша оптимізована для Raspberry Pi операційна система. Необхідно встановити Python та бібліотеки OpenCV для обробки зображень та відео. Окрім цього, необхідно оновити та встановити всі потрібні бібліотеки та залежності. Також треба реалізація алгоритми обробки зображень та аналізу дорожнього руху за допомогою OpenCV [11]. Для зручності користування та для відображення отриманих даних, необхідно розробити простий інтерфейс користувача, завдяки якому користувач зможе переглядати інформацію, що надає система.

1.5 Висновки до розділу 1

В даному розділі розглянуті та проаналізовані різноманітні підходи та технології і методи визначення швидкості об’єкта (транспортного засобу) за допомогою Raspberry Pi та бібліотеки OpenCV. Окрім цього були розглянуті вже існуючі моніторингові системи контролю дорожнього руху, а саме SPECS3 (Speed Check Services), VITRONIC PoliScan Speed та Redflex Speed Enforcement, визначені їх переваги та недоліки, розглянуті які методи використовує кожна система та наведені певні обмеження та можливості цих систем. Також проаналізовані переваги та недоліки моніторингової мережі контролю дорожнього руху на основі Raspberry Pi та OpenCV, серед переваг слід зазначити невелику вартість, високий рівень продуктивності, легкість інтеграції та модифікації системи. Крім цього, були сформовані вимоги до апаратно-програмного забезпечення, для забезпечення відповідного рівня продуктивності системи.

Був проведений аналітичний огляд, методів та алгоритмів, які надає бібліотека OpenCV, для роботи з відеопотоком та розпізнавання транспортних засобів та визначення їх швидкості.

2 МЕТОДИ ВИЯВЛЕННЯ РУХОМИХ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ OPENCV З ВИКОРИСТАННЯМ ВИЗНАЧЕННЯ КОНТУРІВ ТА ВІДНІМАННЯ ФОНУ

Виявлення рухомих об'єктів в сучасному світі відіграє ключову роль у різноманітних областях, від забезпечення безпеки через охоронне спостереження до ефективного моніторингу руху в різних сценаріях. У контексті постійного розвитку галузі комп'ютерного зору, викликаним швидким темпом технологічного прогресу, виявлення рухомих об'єктів стає необхідною складовою для забезпечення надійного та ефективного функціонування систем.

Бібліотека OpenCV, відома своїм відкритим кодом та розширеним набором інструментів для комп'ютерного зору, є важливим інструментом в даній сфері. Забезпечуючи надійні рішення для виявлення рухомих об'єктів, OpenCV дозволяє впроваджувати інноваційні технології в різноманітних сферах, відзначаючись високою точністю та широким спектром застосувань.

Нижче розглянута комбінація виявлення контурів і віднімання фону, які можна використовувати для виявлення рухомих об’єктів за допомогою OpenCV.

Використання методу Contour Detection разом із технікою віднімання фону, відомою як Background Subtraction, утворює ефективний засіб для визначення та обведення об'єктів в реальному часі. Ця комбінація створює потужний механізм для виявлення рухомих об'єктів, де Background Subtraction відокремлює рухомі об'єкти від непорушного фону. Такий підхід відрізняється практичністю та обчислювальною ефективністю, що робить його ідеальним для застосувань, які вимагають швидкого та точного виявлення об'єктів.

2.1 Contour Detection за допомогою методів OpenCV

В даному розділі докладно розглянутий процес визначення контуру об’єкта за допомогою методів OpenCV.

Контури можна пояснити як криву, що з’єднує всі безперервні точки – граничні пікселі, що мають однаковий колір або інтенсивність. Контури корисні для аналізу форми, виявлення та розпізнавання об’єктів. OpenCV дозволяє легко знаходити та наносити контури на зображеннях.

Для визначення контуру об’єкта та нанесення його на зображення OpenCV надає наступні методи: findContours() та drawContours(). Крім цих методів дана бібліотека надає можливість використовувати алгоритми апроксимації контуру з різними параметрами, два найбільш використовуваних CHAIN_APPROX_SIMPLE та CHAIN_APPROX_NONE.

CHAIN_APPROX_SIMPLE:

Цей алгоритм використовується для апроксимації контурів та виключає зайві точки, які лежать на одній прямій. Він робить контур більш компактним, використовуючи лише крайні точки, які є необхідними для представлення форми об'єкта. В основному, цей алгоритм використовується для спрощення геометричного представлення контурів.

Основні переваги даного алгоритму:

– зменшення кількості точок на контурі, зберігаючи його геометричну структуру;

– використання економія пам'яті та обчислювальних ресурсів.

Проте слід зазначити, що вагомим недоліком цього алгоритму є можливість втратити деяку точність при апроксимації контуру.

CHAIN_APPROX_NONE:

Цей алгоритм включає всі точки контуру, не виконуючи жодної апроксимації. Всі точки, які утворюють контур, залишаються незмінними. Такий підхід дозволяє точно представити форму об'єкта, але може призвести до великої кількості точок, особливо у випадку складних форм.

Вагомою перевагою цього алгоритму, в порівнянні з попереднім, є точне відображення форми об'єкта.

Проте слід відзначити деякі недоліки цього алгоритму, а саме:

– може бути створена велика кількість точок, особливо під час створення контуру для складних форм;

– даний алгоритм вимагає більше ресурсів для зберігання та обробки точок.

Щодо швидкості виконання, обидва алгоритми зазвичай працюють досить швидко, але швидкість може залежати від конкретної ситуації, складності формі та обсягу даних.

Якщо необхідно обробити великий потік даних та обчислювальна потужність обмежена, то слід використовувати алгоритм CHAIN_APPROX_SIMPLE для зменшення навантаження на систему.

Для того щоб визначити та нанести контури об’єктів на зображення, використовуючи методи OpenCV, треба виконати декілька кроків.

Перш за все, треба зчитати зображення та конвертувати його у градації сірого. Цей процес є критичним, оскільки підготовлює зображення для наступного етапу обробки. Перетворення в одноканальне зображення у відтінках сірого є необхідним етапом для визначення порогу, який, в свою чергу, є важливим для ефективної роботи алгоритму виявлення контурів.

Слід зазначити що більшість методів бібліотеки OpenCV, для коректної роботи, вимагають саме чорно-білі зображення. Виконувати будь-які конвертації та перетворення зображень потрібно відповідно до документації бібліотеки.

Зазвичай, під час пошуку контурів на чорно-білому зображені використовується бінарний алгоритм порогової обробки або алгоритм Кенні.

Для прикладу використаний бінарний алгоритм порогової обробки. Даний алгоритм перетворює зображення в чорно-біле, виділяючи область інтересу для полегшення процесу виявлення контурів. Порогове значення перетворює межі об’єкта на зображенні в повністю білі, щоб всі пікселі мали однакову інтенсивність. Після цього алгоритм може виявляти межі об’єктів за цими білими пікселями.

Рисунок 2.1 – Результат виконання бінарного алгоритму порогової обробки

Важливо пам’ятати, що чорні пікселі, які мають значення 0, сприймаються як пікселі фону та ігноруються.

В результаті виконання даного алгоритму отримується двійкове представлення вихідного зображення RGB. Взявши білі пікселі по периметру кожного об’єкта як пікселі подібної інтенсивності, алгоритм об’єднає їх, щоб сформувати контур на основі показника подібності.

Після обробки вихідного зображення, можна почати процес пошуку та відображення контурів об’єктів. Для пошуку контурів OpenCV надає метод findContours(). Даний метод має три обов’язкових параметри, серед яких:

– image: бінарне зображення отримане в результаті виконання алгоритму порогової обробки;

– mode: це режим пошуку контурів. Для прикладу використовується RETR_TREE, даний режим означає, що алгоритм знайде всі можливі контури з бінарного зображення;

– method: даний параметр визначає метод контурної апроксимації, в якості прикладу використовується CHAIN_APPROX_NONE.

Варто підкреслити, що mode-параметр стосується типу контурів, які будуть отримані, тоді як method-параметр стосується того, які точки всередині контуру зберігаються.

Після виконання попереднього кроку, потрібно використати функцію drawContours(), яка дозволяє накласти контури об’єктів на вхідне зображення. Ця функція має чотири обов'язкові та кілька додаткових аргументів. Нижче перелічені обов'язкові аргументи:

– image: це вхідне зображення RGB, на яке будуть нанесені контури об'єктів;

– contours: вказує на контури, отримані за допомогою функції findContours();

– contourIdx: координати пікселів точок контуру перераховані в отриманих контурах. За допомогою цього аргументу можна вказати індекс позиції в цьому списку, вказуючи точно, яку точку контуру потрібно нанести на вхідне зображення. Вказання від'ємного значення нанесе всі точки контуру;

– color: даний аргумент вказує на колір точок контуру, які будуть відображені;

– thickness: цей аргумент відповідає за товщину точок контуру.

Рисунок 2.2 – Результат виконання функцій пошуку та нанесення контурів на вхідне зображення

2.2 Визначення контурів об’єктів, з використанням одного окремого каналу червоного, зеленого чи синього

Була досліджена поведінка алгоритму пошуку контурів об’єктів з використанням окремого каналу, R (червоного), G (зеленого) та B (синього) замість зображення у градації сірого. Даний алгоритм виявлення контуру шукає порогові пікселі, які мають подібну інтенсивність та колір, то у випадку використання одного із RGB каналу, алгоритм відпрацьовуватиме некоректно і отримані результати не будуть відповідати вимогам.

Рисунок 2.3 – R (червоний) канал

Рисунок 2.4 – G (зелений) канал

Рисунок 2.5 – B (синій) канал

На поданому вище зображеннях можна відзначити, що алгоритм виявлення контурів не здатний правильно визначити межі об'єктів. Це пояснюється тим, що він не може належним чином розпізнати контури, використовуючи RGB, а також через слабко виражену різницю інтенсивності між пікселями. Саме тому необхідно конвертувати вхідне зображення в бінарне для ефективного виявлення контурів.

2.3 Реалізація фонового віднімання засобами OpenCV

Спочатку потрібно зафіксувати всі відеокадри та передати їх до кодового конвеєра. Тому був створений об'єкт захоплення відео за допомогою методу cv2.VideoCapture(). Для віднімання фону був створений інший об'єкт за допомогою методу cv2.createBackgroundSubtractorMOG2. Далі потрібно перевірити, чи наш об'єкт «cap» працює належним чином. Після перевірки починається захоплення всіх відеокадрів один за одним за допомогою методу cap.read(). Тепер необхідно застосувати віднімання фону до кожного відеокадру за допомогою методу backSub.apply().

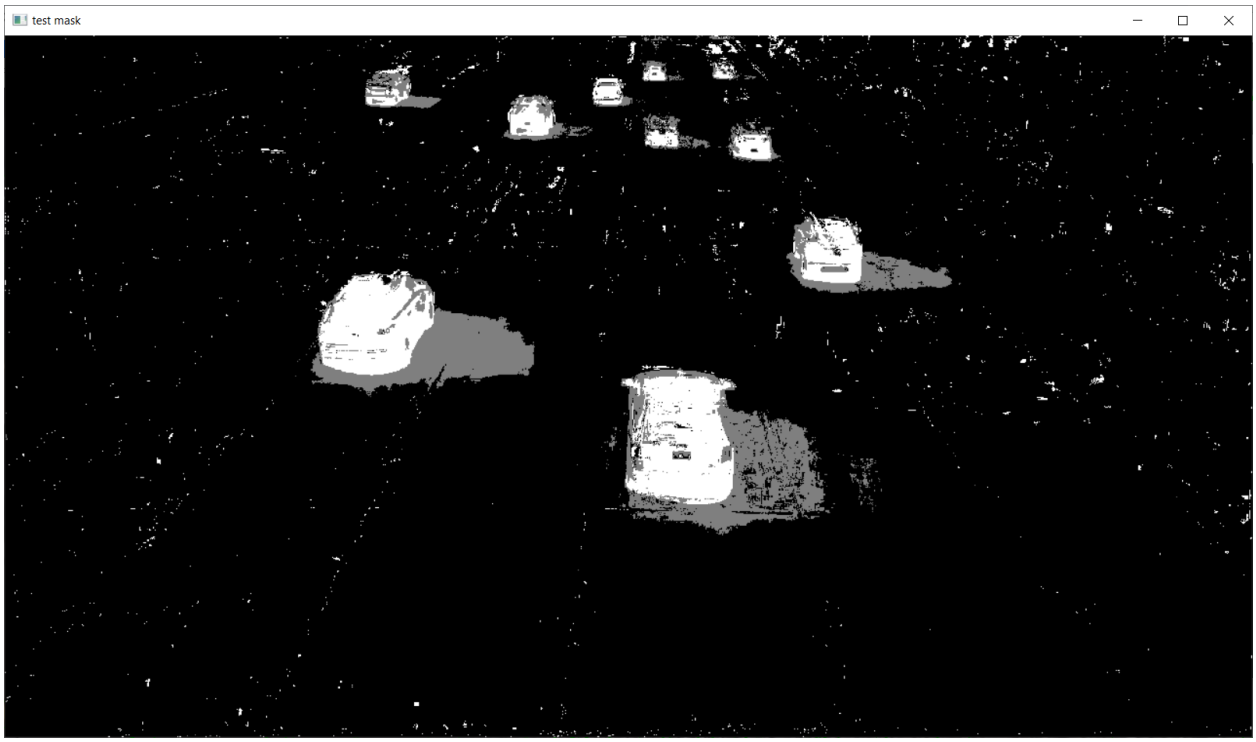


Рисунок 2.6 – віднімання фону засобами OpenCV

Як видно з рисунка, виявлені всі рухливі пікселі між двома послідовними кадрами. Це включає автомобілі, тіні та деякі дрібні білі краплі. Тому що, при взаємодії з реальними сценаріями різноманітні фактори можуть впливати на відеозапис системи відеоспостереження. Це може включати в себе як погодні умови, такі як сильний вітер.

2.4 Виявлення та нанесення контурів рухливих об’єктів засобами OpenCV

Після віднімання фону тепер є чітка двійкова маска потрібної області інтересу (Region of Interest, ROI). Тож можна перейти до виявлення контурів рухомих об'єктів.

Для цього необхідно передати передній план (mask) у функцію cv2.findContours, щоб знайти всі можливі точки контуру. Це стосується всіх виявлених об'єктів на поточному кадрі. Після цього треба передати всі точки контуру у функцію cv2.drawContours, щоб нанести контур для кожного виявленого контуру об'єкта.

Рисунок 2.7 – Виявлення та нанесення контурів рухливих об’єктів засобами OpenCV

Як видно з рисунку нанесення всіх контурів також включає в себе багато непотрібного. Ці дрібні шуми на контурах в подальшому негативно вплинуть на результати.

2.5 Покращення виявлення контурів за допомогою порогового визначення зображення та морфологічних операцій

Шумні контури виникають через рух тіней та камери. Потрібно видалити весь шум, щоб отримати чіткий кадр із виявленим контуром потрібної області інтересу (автомобіль).

Для вирішення даної проблеми можна виконати наступні кроки:

– Thresholding;
– Erosion and Dilation;
– Contour Filtration.

Для порогового перетворення необхідно скористатися функцією cv2.threshold та передати маску переднього плану як вхідні дані. Встановити

порогове значення 180 та максимальне значення 255, це просто замінить всі значення менше 180 на 0 і видалить всі тіні.

Рисунок 2.8 – Результат виконання порогового перетворення

Після застосування порогового перетворення була отримана бінарна маска зображення. Це маска автомобілів без будь-яких тіней. Однак у масці все ще залишаються невеликі білі краплі. Наступним кроком є видалення контурів навколо цих невеликих крапель.

Щоб видалити ці невеликі білі краплі, треба застосувати ерозію і потім розширювати зображення для отримання кращої та ширшої маски. Для цього слід скористатися функцією cv2.morphologyEx, використовуючи cv2.MORPH_OPEN як тип морфології, крім цього треба передати порогову маску. Нижче перелічені параметри, які потрібно передавати у функцію cv2.morphologyEx():

– src: вихідне зображення. Кількість каналів може бути довільною. Глибина має бути однією з cv.CV_8U, cv.CV_16U, cv.CV_16S, cv.CV_32F або cv.CV_64F;

– dst: цільове зображення того самого розміру та типу, що й вихідне зображення;

– op: визначає тип морфологічної операції;

– kernel: структуруючий елемент;

– anchor: позицію прив'язки kernel елементу. Від’ємні значення означають, що прив’язка знаходиться по центру;

– iterations: задає кількість застосування dilation;

– borderType: метод піксельної екстраполяції.

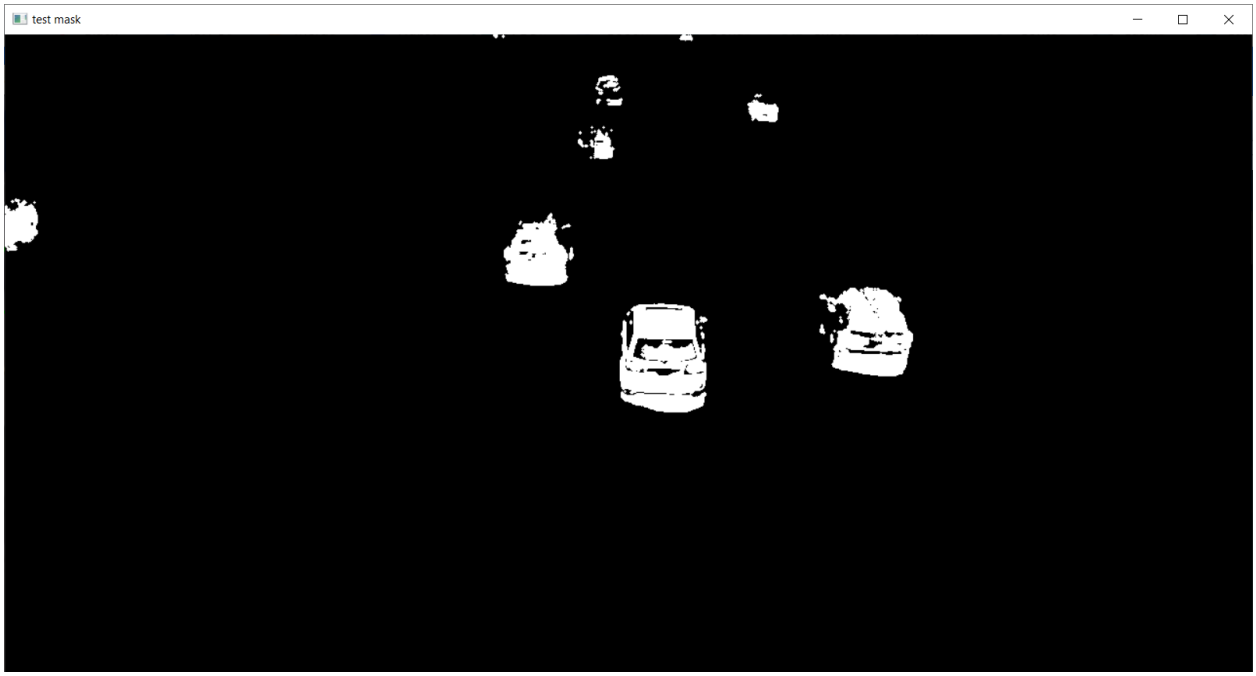


Рисунок 2.9 – Результат виконання морфологічних операцій

В результаті виконання попередніх функцій було отримано значно чистіше зображення. Решту очищення можна виконати, просто фільтруючи контури з невеликими площами.

Метод фільтрації контурів іноді може бути методом проб та помилок. Для кожної ситуації потрібно окремо підбирати площу контуру.

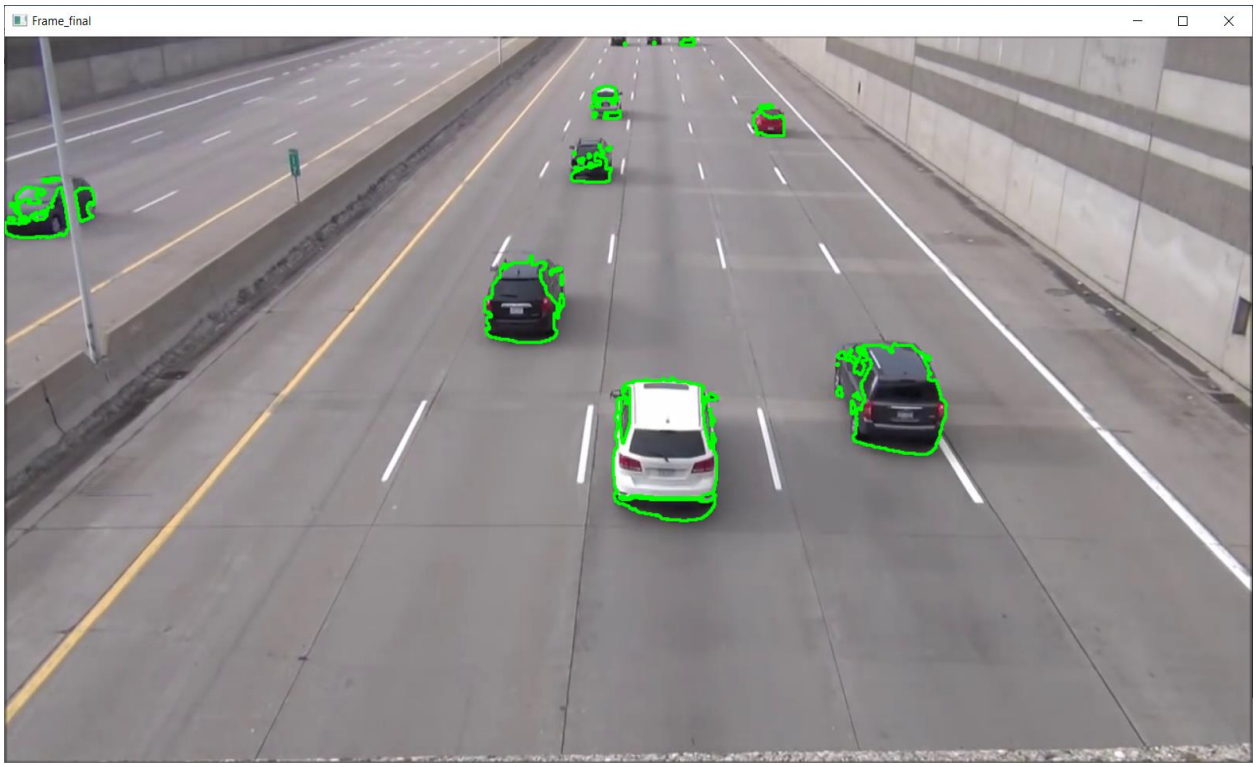


Рисунок 2.10 – Результат виконання фільтрації контурів

2.6 Визначення об’єктів на статичному зображені та на відео

Визначення об’єктів на статичному зображені та на відео відрізняється за деякими ключовими аспектами.

Перш за все, різні вхідні дані. Для статичного зображення вхідні дані зазвичай представлені у вигляді одного зображення, а для відео вхідні дані - це послідовність кадрів, де кожен кадр подається на обробку в певній послідовності.

Для статичного зображення ініціалізація та обробка починається із завантаження та аналізу одного статичного зображення. Для відео – ініціалізація включає в себе відкриття відеопотоку та налаштування параметрів, таких як ширина та висота кадру.

У випадку статичного зображення, його обробка відбувається лише один раз, а у випадку відео – обробка повторюється для кожного кадру у відеопослідовності. Оптимізація тут важлива для забезпечення швидкої обробки в реальному часі.

Для виявлення об’єктів на статичному зображені можна використовувати більш ресурсоємні та точні алгоритми, такі як Faster R-CNN або SSD.

Алгоритм Faster R-CNN використовується для об'єктного виявлення в зображеннях та відео і є досить популярним методом у сферах комп'ютерного зору та машинного навчання. Використовується для виявлення обличчя, транспортних засобів або інших предметів. Даний алгоритм використовує CNN (конволюційну нейронну мережу) для обробки та аналізу зображень, це надає високу точність виявлення об’єктів, але має доволі високу складність обчислення та вимагає високий рівень обчислювальної потужності щоб забезпечити швидку роботу даного алгоритму.

Алгоритм SSD часто використовується в ситуаціях, де важлива швидкість виявлення об'єктів, таких як моніторингові системи в реальному часі або в системах відеоспостереження чи автономних автомобілях та інших додатках з обмеженими ресурсами. Даний алгоритм надає високу швидкість, але має значно нижчу точність в порівнянні з алгоритмом Faster R-CNN, особливо коли на зображені присутня велика кількість різноманітних об'єктів.

Для виявлення об’єктів на відео використовуються оптимізовані алгоритми, як YOLO (You Only Look Once) або ж Single Shot MultiBox Detector (SSD), вони часто використовуються для забезпечення швидкої роботи в режимі реального часу.

YOLO застосовується в системах відеоспостереження, автономних автомобілях, системах безпеки та інших систем, де важлива швидкість та робота в режимі реального часу. Даний алгоритм забезпечує достатню швидкість виявлення об'єктів в системах реального часу та має доволі низький рівень складності реалізації, проте алгоритм YOLO може мати проблеми з точністю виявлення невеликих об'єктів через грубе розбиття на сітку та низьку роздільну здатність.

Загалом, використання OpenCV для обробки та аналізу статичних зображень та відео відрізняється управлінням послідовністю кадрів, оптимізацією для роботи в режимі реального часу та виводом результатів. Отже, при виборі методу важливо враховувати специфіку завдання та ресурсні обмеження.

2.7 Математична модель визначення швидкості автомобіля

Для визначення швидкості рухомого об'єкта (автомобіля), можна використовувати різні підходи і методи, та різні математичні формули відповідно.

Нижче розглянуто визначення пройденої відстані об’єктом за допомогою Евклідової метрики (відстані) [17]. Цей підхід полягає у визначенні координат об’єкта на попередньому кадрі та їх порівнянні з координатами цього ж об’єкта у наступному кадрі.

Основа розрахунку швидкості полягає у визначенні пройденої відстані за секунду. Але у випадку із зображеннями відсутня інформація у метрах, тому необхідно конвертувати пройдені пікселі за секунду (px/s) у метри за секунду (м/с), а потім у кілометри за годину (км/год).

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.1}$$

де (x_2, y_2) та (x_1,y_1) координати транспортного засобу поточного та попереднього кадрів відповідно.

Наступний крок це обчислення швидкості об'єкта S (m/s).

$$S = \frac{d\ in\ pixel}{pixel\ per\ meter} \times frame\ per\ second \tag{2.2}$$

Після цього, необхідно обчислити швидкість об’єкта S (km/h), для цього треба скористатись наступною формулою.

$$S(km/h) = S(m/s) \times 3.6 \tag{2.3}$$

Пікселі на метр можна розрахувати за допомогою об'єкта-референсу. Наприклад, ширина автомобіля становить 2 метри, і на зображені ця відстань

становить 100 пікселів, отже, один метр відстані становить 50 пікселів на зображені. Проте, слід зазначити що, об'єкт буде зменшуватися, коли він віддаляється від камери. В такому випадку треба визначити область, в якій буде розраховуватись кількість пікселів на метр, і оцінка швидкості буде виконуватися в даній області.

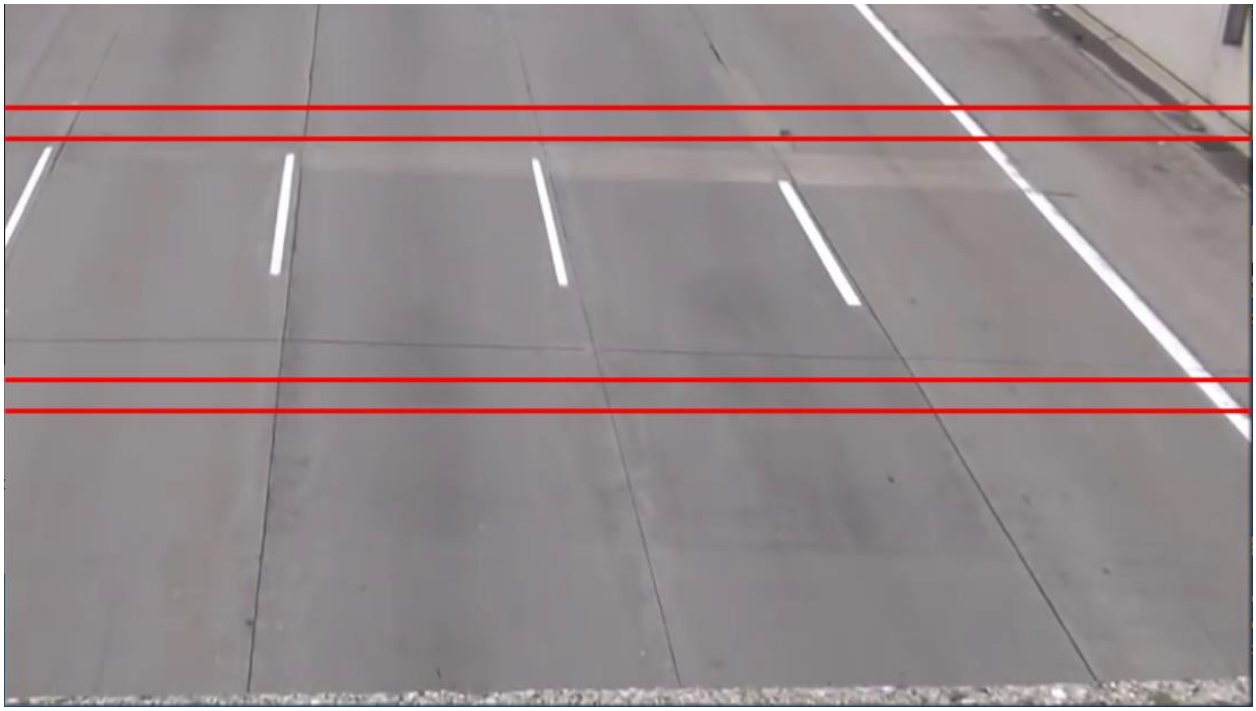


Рисунок 2.11 – Обчислення швидкості об’єкта

Для обчислення швидкості об’єкта потрібно розмістити на фреймі зазвичай дві, інколи чотири лінії, коли об’єкт перетинає першу лінію запускається таймер, а коли автомобіль перетне другу лінію – таймер вимикається.

Наступний крок це безпосередньо обчислення швидкості. Відстань між двома лініями заздалегідь визначена, наприклад 10 метрів. Використання чотирьох ліній замість двох обумовлене неточністю алгоритму, виникають випадки коли об’єкт перетнув лінію, а таймер не запустився, тому при використанні двох ліній підряд, підвищується точність обчислень, а саме якщо таймер не запустився при перетині об’єктом першої лінії, то він запуститься при перетині другої, реалізація цього підходу може бути різною для кожної ситуації.

2.8 Object detection та object tracking з використанням нейронних мереж та машинного навчання

Комп’ютерне бачення – це галузь, яка набула величезної популярності в останні роки. Невід'ємною частиною комп'ютерного зору є виявлення об'єктів. Виявлення об’єктів допомагає у виявленні транспортних засобів та будь-яких інших об’єктів. Різниця між алгоритмами виявлення об’єктів і алгоритмами класифікації полягає в тому, що в алгоритмах виявлення наноситься обмежувальна рамка навколо об’єкта, щоб знайти його на зображенні. Крім того, не обов’язково наносити лише одну обмежувальну рамку у випадку виявлення об’єктів, може бути багато обмежувальних рамок, що представляють різні об’єкти.

Для забезпечення високої швидкості та точності виявлення об'єктів у системах реального часу, таких як системи відеоспостереження, різноманітні моніторингові системи, використовуються нейронні мережі та машинне навчання.

Коли справа доходить до виявлення об’єктів на основі deep machine learning, використовуються три найбільш популярні детектори об’єктів:

– R-CNN або Fast R- CNN;

– YOLO;

– Single Shot Detector (SSDs).

Алгоритм R-CNN [18] працює наступним чином:

– знаходження потенційних об'єктів на зображенні та розділення їх на регіони за допомогою методу selective search;

– визначення ознак кожного отриманого регіону за допомогою згорткової нейронної мережі;

– класифікація оброблених ознак за допомогою методу опорних векторів (SVM, Support Vector Machine) та уточнення меж регіонів за допомогою лінійної регресії;

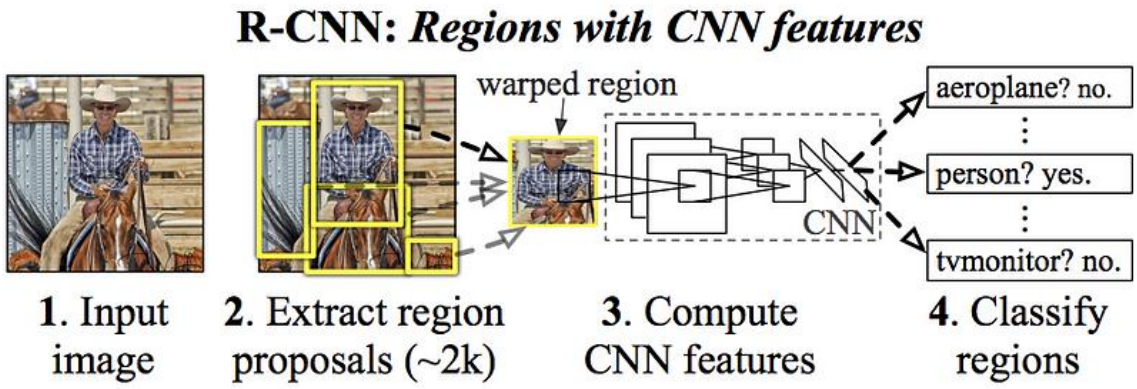


Рисунок 2.12 – Алгоритм роботи R-CNN

В результаті виконання даного алгоритму отримуються окремі регіони з об’єктами та їх класи. Згорткова нейронна мережа забезпечує високий рівень точності детектування об’єктів у випадку обробки та аналізу статичних зображень. Серед вагомих недоліків варто відмітити:

– потрібно витрати значну кількість часу для навчання мережі, оскільки потрібно класифікувати близько 2000 знайдених регіонів для кожного зображення;

– даний алгоритм не використовується в системах реального часу, тому що обробка зображення займає 30-50 секунд.

Окрім цих недоліків слід зазначити, що даний алгоритм застарів та не використовується для детектування об’єктів на відео через свою складність обчислень та низьку швидкість обробки фреймів.

На відміну від попереднього алгоритму, принцип роботи якого полягає у виявленні об’єктів з використанням областей (регіонів) для локалізації об’єкта на зображенні, алгоритм YOLO суттєво відрізняється від регіон-орієнтованих алгоритмів. У YOLO одна конволюційна мережа передбачає межі обрамлення та ймовірності класів для цих обрамлень.

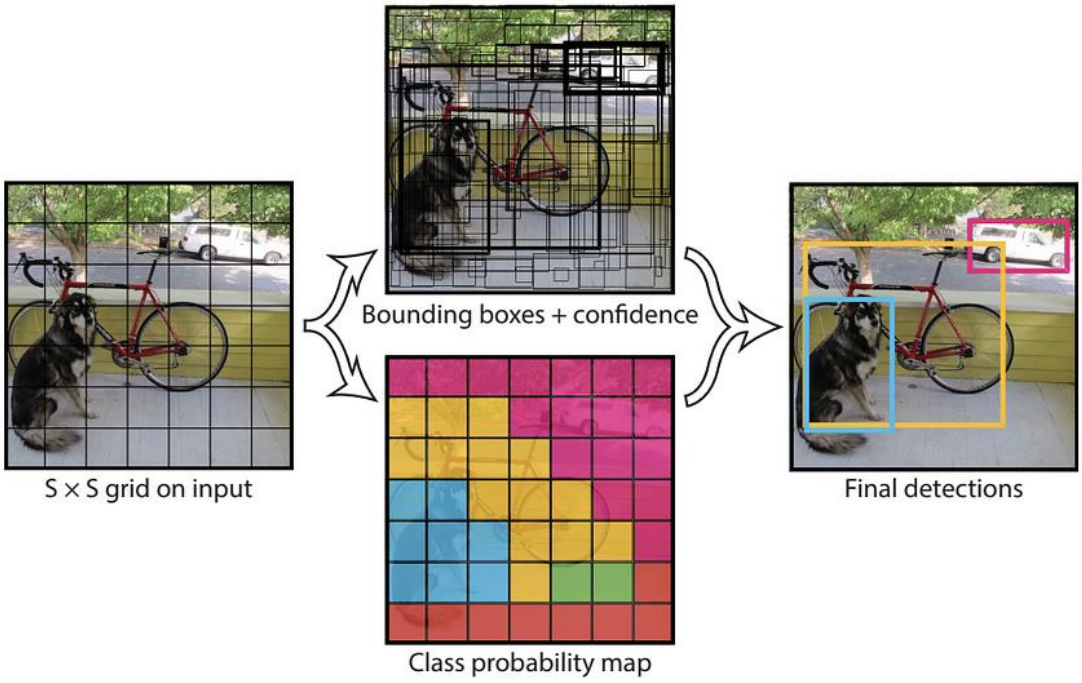


Рисунок 2.13 – Принцип роботи алгоритму YOLO

Принцип роботи YOLO полягає в тому, що даний алгоритм зчитує зображення і розділяє його на сітку розміром SxS, всередині кожної ячейки наносить m обрамлень. Для кожного обрамлення мережа виводить ймовірність класу та значення зсуву для обрамлення. Вибираються обрамлення, які мають ймовірність класу вище порогового значення, і використовуються для визначення положення об'єкта на зображенні.

YOLO [19] значно швидший (45 кадрів в секунду) за інші алгоритми виявлення об'єктів. Однак обмеженням алгоритму YOLO, є те що виникають проблеми з виявленням невеликих об'єктів на зображенні, наприклад, труднощі з визначенням птахів. Це пов'язано з просторовими обмеженнями алгоритму.

2.9 Висновки до розділу 2

В даному розділі був проведений аналіз методів та алгоритмів виявлення об'єктів на статичному зображені та на відео, які надає бібліотека OpenCv. Розглянуті методи та їх комбінації для покращення якості нанесених контурів навколо об'єктів без зайвих шумів та тіней. Було визначено, що обов'язково

потрібно виконувати перетворення зображення на бінарне, для коректного визначення контурів об'єктів, якщо не виконувати такого перетворення, а використовувати зображення в RGB форматі, алгоритми не зможуть точно знаходити края об’єктів та коректно визначити контури. Проаналізовані морфологічні оператори та їх комбінації для усунення зайвих шумів та тіней з зображення, а саме thresholding, erosion and dilation та contour filtration. Дані методи значно підвищують точність виявлення об’єктів.

Окрім цього було наведено ключові відмінності між виявленням об’єктів на статичному зображені та на відео. Основна відмінність, що у випадку обробки та аналізу статичного зображення можна використовувати повільні, але точні алгоритми детектування об’єктів. А у випадку обробки відео – потрібно підбирати алгоритм який забезпечує роботу в режимі реального часу.

Також була наведена математична модель для обчислення швидкості, а саме був розглянутий спосіб обчислення швидкості об’єкта за допомогою Евклідової метрики (відстані).

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Апаратне забезпечення

У сучасному світі розмаїття мінікомп'ютерів відкриває широкий спектр можливостей для ентузіастів та розробників. Видатним представником цього класу пристроїв є Raspberry Pi, але на ринку з'являється все більше інших вражаючих альтернатив, наприклад сімейство мінікомп’ютерів Orange Pi. Ці мінікомп'ютери, незважаючи на свою компактність, володіють значущим впливом у різних сферах, таких як освіта, розваги, робототехніка та вбудовані системи.

Завдяки невеликому розміру та доступності, мінікомп'ютери стали платформою для втілення ідей у життя. Вони дозволяють користувачам

експериментувати з програмуванням, створювати власні проєкти та реалізовувати творчі концепції, розширюючи межі можливостей технології.

Ці пристрої забезпечують різноманіття можливостей, такі як, розвинені проєкти Інтернету речей (IoT), створення мультимедійних центрів та ігрових систем. Разом із розвитком мережевих технологій, мінікомп'ютери можуть використовуватися для віддаленого моніторингу, автоматизації домашнього середовища та вирішення ряду завдань в області науки та досліджень.

Для реалізації моніторингової мережі контролю дорожнього руху був обраний одноплатний мінікомп’ютер Raspberry Pi 3 B+ [20].

Рисунок 3.1 – Зовнішній вигляд мінікомп’ютера Raspberry Pi 3 B+

Перш за все, перед початком розробки, необхідно встановити операційну систему. Для цього потрібно встановити карту пам’яті microSD як мінімум Class10. Для мінікомп’ютерів сімейства Raspberry Pi існує велика кількість операційних систем, такі як Ubuntu чи Raspberry Pi OS. В якості операційної системи була встановлена Raspbian.

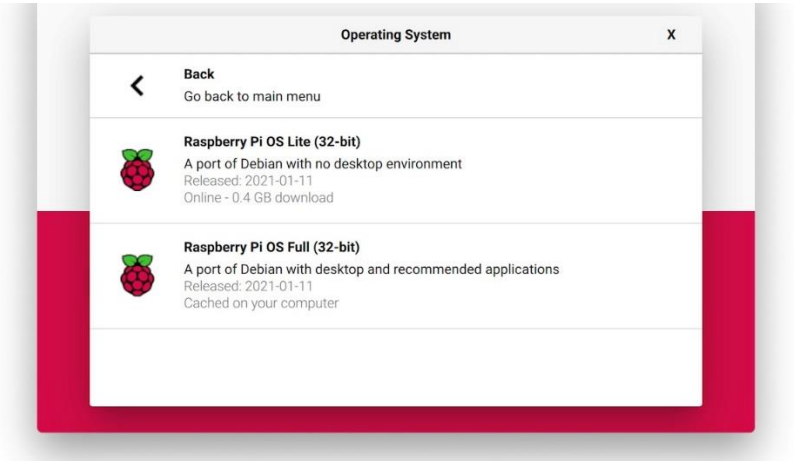


Рисунок – 3.2 – Встановлення Raspbian

Raspbian – це операційна система на основі Linux, спеціально призначена для використання на мінікомп'ютерах серії Raspberry Pi. Вона розроблена спеціально для оптимальної роботи на апаратному забезпеченні Raspberry Pi та включає в себе низку компонентів та інструментів для зручного налаштування та використання цих пристроїв.

Raspbian базується на операційній системі Debian і надає засоби для виконання різних завдань, включаючи програмування, створення різноманітних проєктів Інтернету речей (IoT). Дана операційна система містить стандартні інструменти для Linux, такі як термінал, редактор тексту, бібліотеки та додатки, а також інтерфейс користувача з графічним оточенням для зручності використання.

Даний мінікомп’ютер має чотирьох ядерний ARMv8 64-бітний процесор з робочою частотою 1,4 ГГц, об’єм оперативної пам’яті становить 1 Гб формату LPDDR2 з частотою 900 МГц. Для запобігання перегріву, була покращена система охолодження, тепло рівномірно розподіляється по області нагріву, в порівнянні з попередньою моделлю.

Raspberry Pi 3 B+ має 40 GPIO портів входу – виходу, також наявний порт підключення дисплея Display Serial Interface (DSI) та порт для підключення камери MIPI Camera Serial Interface (CSI-2). Крім цього, присутні інтерфейси бездротового зв’язку WiFi 2,4 ГГц та 5 ГГц IEEE 802.11.b/g/n/ac, та Bluetooth: 4.2 Classic і Low Energy (BLE).

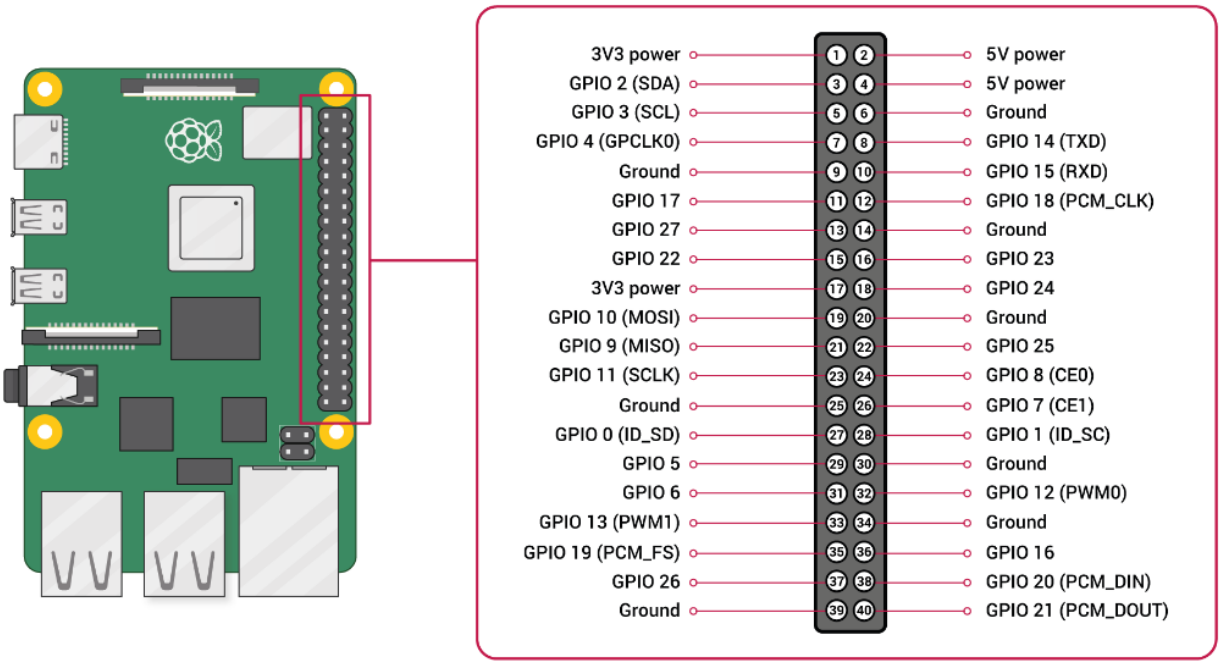


Рисунок 3.3 – Raspberry Pi B+ pinout

Окрім мінікомп’ютера необхідна спеціальна камера, яка підключається до Raspberry Pi, використовуючи CSi інтерфейс, який спеціально спроєктований для підключення камер до мінікомп’ютера.

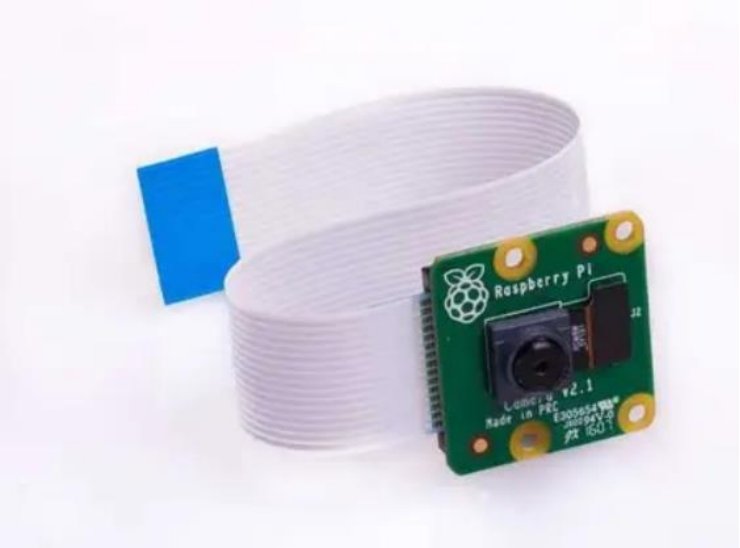


Рисунок 3.4 – Raspberry Pi Cam

Raspberry Pi Camera Module v2 - має високоякісний 8-мегапіксельний датчик зображення Sony IMX219 з фіксованим фокусом, спеціально розроблений для плати Raspberry Pi. Датчик підтримує зображення з роздільною здатністю 3280 x 2464 пікселів, а також відео 1080p30, 720p60 і 640x480p60/90. Він підключається до плати за допомогою невеликих роз'ємів

у верхній частині плати та використовує спеціальний інтерфейс CSi, призначений спеціально для роботи з камерами.

Сам датчик має невеликі розміри - всього 25 мм x 23 мм x 9 мм. Вага становить приблизно 3 г, що ідеально підходить для проєктів де вага та розмір є важливими критеріями.

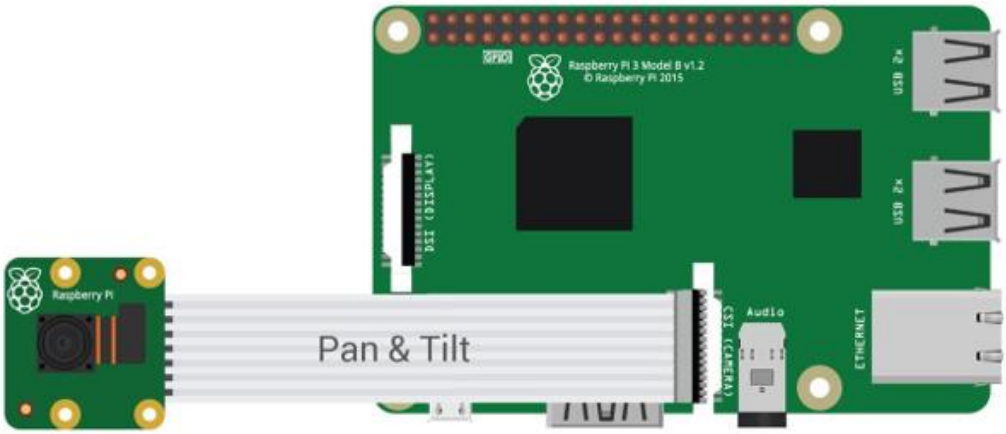


Рисунок 3.5 – Підключення модуля камери до Raspberry Pi

Модуль камери підключається до Raspberry Pi за допомогою інтерфейсу CSI, який забезпечує високу швидкість передачі даних.

Після встановлення операційної системи на мінікомп’ютер Raspberry Pi для керування пристроєм буде використовуватись PuTTy та VNC server. Для цього потрібно на Raspberry Pi увімкнути SSH [21] та VNC [22].

PuTTy [23] – безкоштовний додаток для керування віддаленим вузлом. Це може бути віддалений комп’ютер або сервер. Можуть використовуватись різні протоколи віддаленого доступу, такі як SSH, Telnet, rlogin.

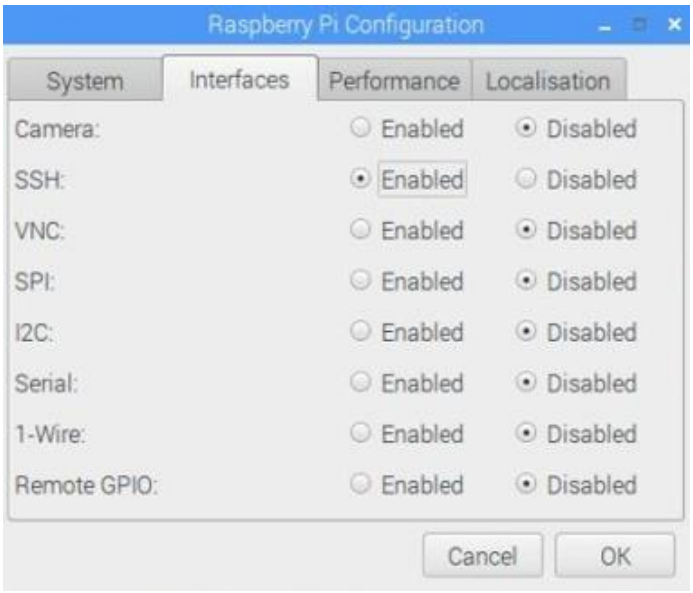


Рисунок 3.6 – Конфігурація Raspberry Pi

3.2 Програмне забезпечення

Програмне забезпечення має виконувати наступні функції:

– зчитування вхідного потоку;

– обробка та аналіз послідовних фреймів;

– детектування рухомих об’єктів (транспортних засобів);

– обчислення швидкості рухомих об’єктів;

– збереження даних.

Для розробки програмного забезпечення була обрана мова програмування Python.

Python є однією з найпопулярніших мов програмування у сфері машинного зору, завдяки можливості розширюваності та відносно низькому рівню складності реалізації. Серед переваг Python у сфері машинного зору треба відмітити:

– зручний синтаксис, адже Python має читабельний та простий синтаксис, що полегшує розробку та зрозуміння коду;

– широкий вибір бібліотек, Python має потужну екосистему бібліотек для машинного зору, таких як OpenCV, TensorFlow, PyTorch;

– активна спільнота, велика так активна спільнота розробників дозволяє легко знаходити рішення та отримувати підтримку.

Незважаючи на досить вагомі переваги, слід також зазначити про недоліки даної мови програмування, а саме:

– швидкодія - у порівнянні з іншими мовами такими як C++ або Java, Python може бути менш продуктивним у виконанні обчислень важливих завдань, особливо під час обробки великих обсягів даних.

3.3 YOLOv8 для виявлення рухомих об’єктів

Для детектування рухомих об’єктів та обчислення їх швидкості був застосований YOLOv8.

YOLO (You Only Look Once) – один із найпопулярніших модулів для виявлення об'єктів в реальному часі та сегментації зображень, що на даний момент вважається як "State-of-The-Art" (SOTA) YOLO є конволюційною нейронною мережею, яка прогнозує межі обрамлення та ймовірності класів зображення за одну оцінку.

Незважаючи на беззаперечну ефективність цього інструменту, важливо мати на увазі, що він був розроблений для загального контексту з метою обслуговування максимально можливої кількості застосувань. Однак для більш конкретних випадків, які вимагають вищої якості, швидкості, роботи з нестандартними зображеннями та інших сценаріїв, рекомендується розуміти архітектуру та, за можливості, налаштовувати її під вимоги завдання.

Перший крок у розумінні архітектури YOLO - це розуміння, що існують три основні блоки в алгоритмі, і все буде відбуватися в цих блоках, а саме: Backbone, Neck та Head.

Backbone

Дана часина відповідає за вилучення значущих ознак з введених даних.

Функції:

– захоплює прості візуальні патерни на початкових шарах, такі як вершини об'єктів та текстури;

– може мати кілька рівнів представлення по мірі просування, захоплюючи ознаки з різних рівнів абстракції;

– забезпечує багатий ієрархічний вигляд введених даних.

Neck

Ця частина виступає як міст між основною частиною та головною частиною, виконуючи операції об'єднання ознак та інтегруючи контекстуальну інформацію. Основна ідея полягає в тому, що дана частина формує піраміди ознак, агрегуючи картки ознак, отримані від основної частини, іншими словами, Neck збирає карти ознак з різних етапів основної частини.

Функції:

– виконує конкатенацію або об'єднання ознак різних масштабів, щоб забезпечити можливість мережі виявляти об'єкти різних розмірів;

– інтегрує контекстуальну інформацію для покращення точності виявлення, розглядаючи більший контекст сцени;

– зменшує просторову роздільність та розмірність ресурсів для полегшення обчислень, що збільшує швидкість, але може також знизити якість моделі.

Head

Головна частина є останньою частиною мережі і відповідає за генерацію виходів мережі, таких як межі обрамлення та імовірності визначення об'єктів.

Функції:

– генерує межі обрамлення, пов'язані з можливими об'єктами на зображенні;

– присвоює імовірності визначення для кожної межі обрамлення, щоб вказати, наскільки ймовірно присутній об'єкт;

– сортує об'єкти в межах обрамлення відповідно до їх категорій.

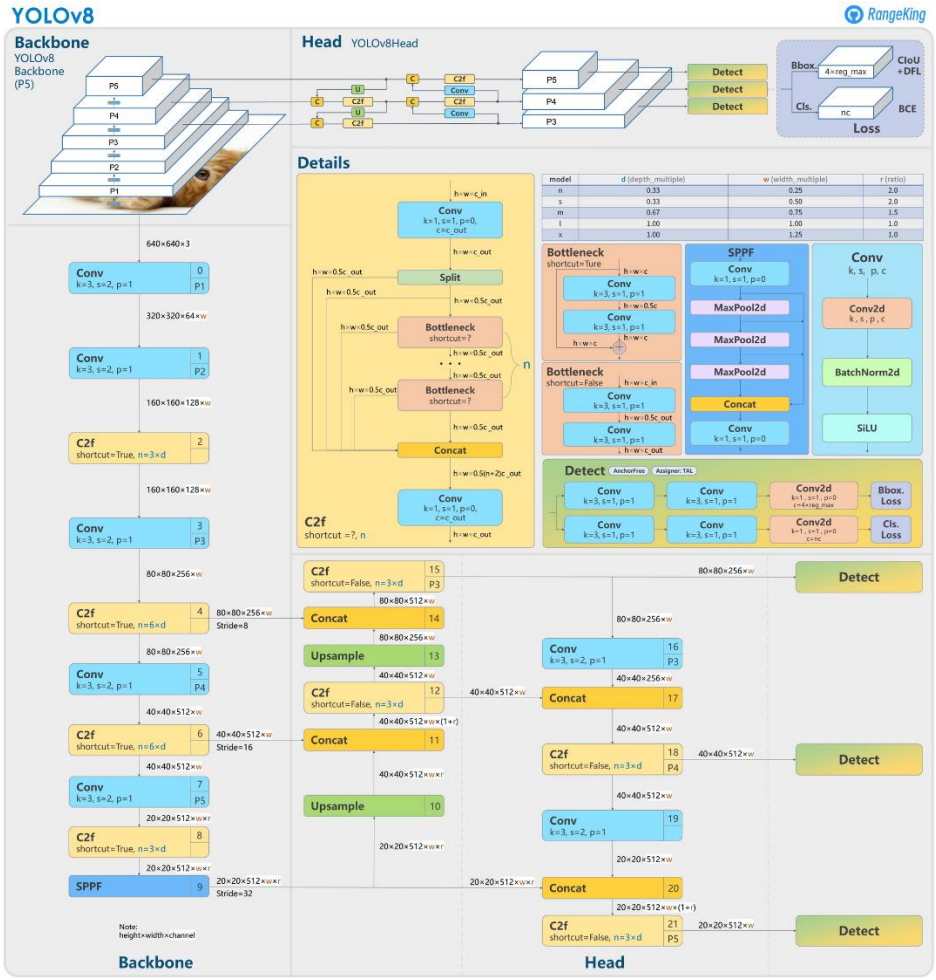


Рисунок 3.7 – Архітектура YOLOv8

Архітектура YOLO [24] використовує підхід локального аналізу ознак замість вивчення зображення як цілого, головною метою цієї стратегії є головним чином зменшення обчислювальних витрат та можливість виявлення в реальному часі. Для вилучення карт ознак в алгоритмі використовуються кілька згорток.

Згортка - це математична операція, яка поєднує дві функції, щоб створити третю. У комп'ютерному зорі та обробці сигналів згортка часто використовується для застосування фільтрів до зображень або сигналів, виділяючи конкретні патерни. У згорткових нейронних мережах (CNN) згортка використовується для вилучення ознак зі входів, таких як зображення. Згортки структуровані за допомогою ядер (K), кроків (s) та додавань (p).

Kernel, також відоме як фільтр, являє собою невеликий масив чисел, який ковзає по вхідних даних (зображення або сигнал) під час операції

згортання. Мета полягає в застосуванні локальних операцій до вхідних даних для виявлення конкретних характеристик. Кожен елемент у ядрі представляє вагу, яка множиться на відповідне значення у вхідних даних під час згортання.

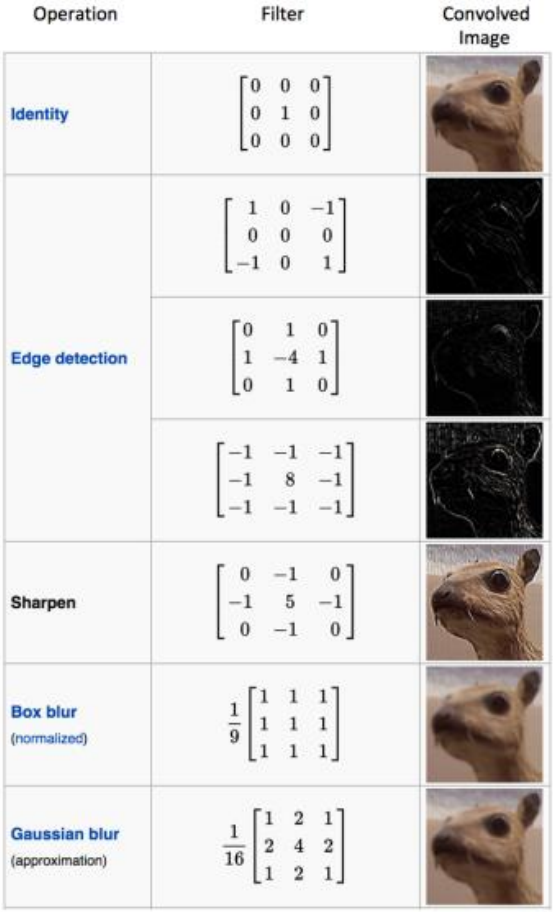


Рисунок 3.8 – Результат застосування різних типів фільтрів

Крок (Stride) — це величина зсуву, якого зазнає ядро, коли воно переміщується по входу під час згортання. Крок 1 означає, що ядро переміщується на одну позицію за раз, тоді як крок 2 означає, що ядро пропускає дві позиції з кожним рухом. Крок безпосередньо впливає на просторові розміри результату згортки. Більші кроки можуть зменшити розмірність результату, тоді як менші кроки зберігають більше просторової інформації. Більші кроки зменшують обчислювальні зусилля, тим самим збільшуючи швидкість операції, що може безпосередньо вплинути на якість.

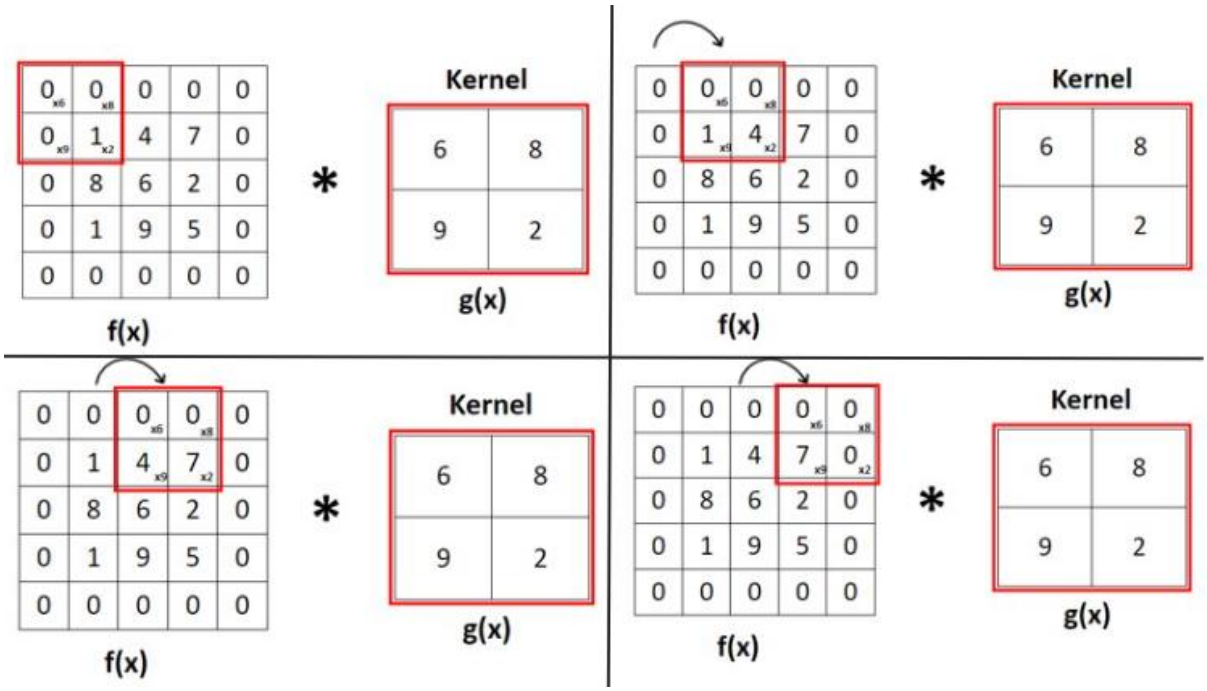


Рисунок 3.9 – Приклад роботи кроку (stride) зі значенням 1

«Доповнення» (padding) означає додавання додаткових пікселів по краях вхідного зображення (зазвичай нулів) перед застосуванням операцій згортання. Це робиться для того, щоб інформація на краях зображення оброблялася так само, як інформація в центрі під час операцій згортання.

Коли фільтр (kernel) застосовується до зображення, він зазвичай переглядає зображення піксель за пікселем. Якщо доповнення не застосовано, пікселі по краях зображення мають менше сусідів, ніж пікселі в центрі, що може призвести до втрати інформації в цих областях.

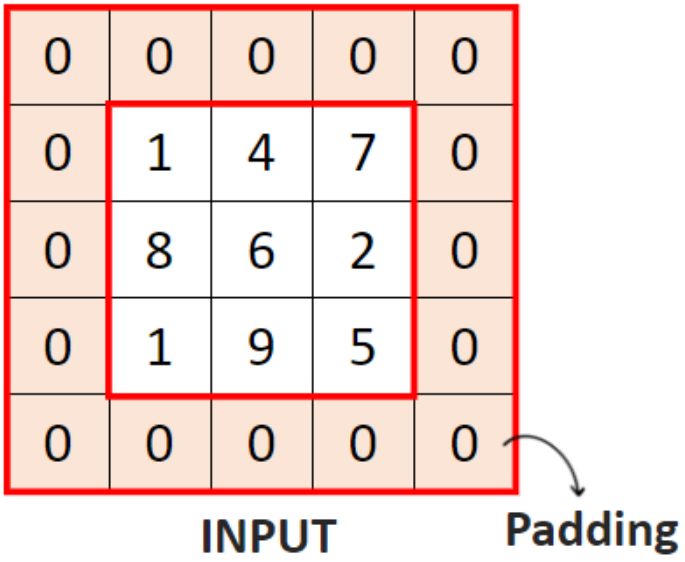


Рисунок 3.10 – Приклад роботи доповнення (padding)

YOLOv8 є моделлю без якорів. Це означає, що вона прогнозує безпосередньо центр об'єкта, а не зсув від відомої якорної рамки.

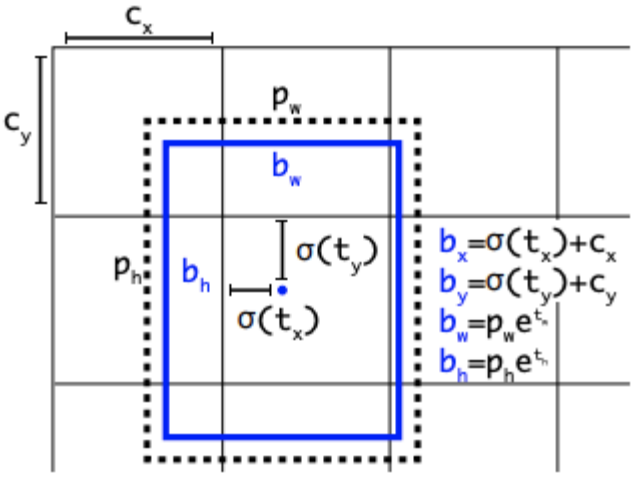


Рисунок 3.11 - Візуалізація anchor box в YOLO

3.4 Реалізація виявлення рухомих об'єктів та обчислення їх швидкості за допомогою YOLOv8

Перед початком розробки необхідно налаштувати середовище розробки, а саме встановити останню версію Python та всі необхідні залежності.

Для того щоб прискорити процес налаштування і не встановлювати всі необхідні залежності окремо, можна скористатись наступною командою.

```
$ pip install -r requirements.txt
```

Рисунок 3.12 – Команда для встановлення залежностей з файлу конфігурації

Спочатку треба створити даний файл конфігурації, в якому зазначено які залежності треба завантажити та встановити.

```
###### Requirements without Version Specifiers ######
nose
nose-cov
beautifulsoup4
#
###### Requirements with Version Specifiers ######
#   See https://www.python.org/dev/peps/pep-0440/#version-specifiers
docopt == 0.6.1             # Version Matching. Must be version 0.6.1
keyring >= 4.1.1            # Minimum version 4.1.1
coverage != 3.5             # Version Exclusion. Anything except version 3.5
Mopidy-Dirble ~= 1.1        # Compatible release. Same as >= 1.1, == 1.*
```

Рисунок 3.13 – Приклад файлу конфігурації

Після встановлення всіх необхідних залежностей та виконання всіх налаштувань, можна переходити безпосередньо до розробки застосунку.

Перш за все, потрібно завантажити модель YOLOv8, а саме попередньо підготовлену модель YOLOv8s.

```
model=YOLO('yolov8s.pt')
```

Рисунок 3.14 – Ініціалізація попередньо підготовленої моделі

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Рисунок 3.15 – Порівняння моделей YOLOv8

Наступний крок це зчитування відео за допомогою OpenCV.

```
cap=cv2.VideoCapture('mockData.mp4')
```

Рисунок 3.16 – Зчитування відео засобами OpenCV

Потім потрібно ініціалізувати початкові дані, а саме лічильник автомобілей, порожній масив, в який додається інформація про об’єкт, який було виявлено. Після чого в області видимості циклу потрібно зчитувати послідовно фрейми, визначати координати об’єктів та наносити bounding box навколо рухомих об’єктів.

```
    for index,row in px.iterrows():
#        print(row)

        x1=int(row[0])
        y1=int(row[1])
        x2=int(row[2])
        y2=int(row[3])
        d=int(row[5])
        c=class_list[d]
        if 'car' in c:
            list.append([x1,y1,x2,y2])
    bbox_id=tracker.update(list)
    for bbox in bbox_id:
        x3,y3,x4,y4,id=bbox
        cx=int(x3+x4)//2
        cy=int(y3+y4)//2

        cv2.rectangle(frame,(x3,y3),(x4,y4),(0,0,255),2)
```

Рисунок 3.17 – Код для визначення координат та нанесення bounding boxes

Також на фрейм треба нанести дві лінії з відомою дистанцією між ними, для обчислення швидкості рухомого об’єкта.

```
cv2.line(frame,(274,cy1),(814,cy1),(255,255,255),1)

cv2.putText(frame,('L1'),(277,320),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)


cv2.line(frame,(177,cy2),(927,cy2),(255,255,255),1)

cv2.putText(frame,('L2'),(182,367),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)
```

Рисунок 3.18 – Нанесення додаткових ліній на фрейм

Після попередніх кроків, можна обчислювати швидкість автомобіля. Це відбувається наступним чином:

– об'єкт перетинає першу лінію L1 – відбувається запуск таймера;

– коли об’єкт перетинає лінію L2 – таймер зупиняється;

– після зупинки таймера в змінну записується час, за який об’єкт пройшов певну відстань, для прикладу відстань становить 10 метрів;

– потім обчислюється швидкість S(m/s) та переведення у величину km/h;

– швидкість автомобіля зберігається у змінну, як і id об’єкта, після чого записується у текстовий файл.

```
if cy1<(cy+offset) and cy1 > (cy-offset):
   vh_down[id]=time.time()
if id in vh_down:

   if cy2<(cy+offset) and cy2 > (cy-offset):
     elapsed_time=time.time() - vh_down[id]
     if counter.count(id)==0:
        counter.append(id)
        distance = 10 # meters
        a_speed_ms = distance / elapsed_time
        a_speed_kh = a_speed_ms * 3.6
        cv2.circle(frame,(cx,cy),4,(0,0,255),-1)
        cv2.putText(frame,str(id),(x3,y3),cv2.FONT_HERSHEY_COMPLEX,0.6,(255,255,255),1)
        cv2.putText(frame,str(int(a_speed_kh))+'Km/h',(x4,y4 ),cv2.FONT_HERSHEY_COMPLEX,0.8,(0,255,255),2)
        filet = open(speed_record_file_location, "a")
        filet.write(str(id) + " \t " + str(int(a_speed_kh))+ ' Km/h' + "\n")
        filet.close()
```

Рисунок 3.19 – Обчислення швидкості об’єкта, виведення інформації на фрейм та запис даних у файл

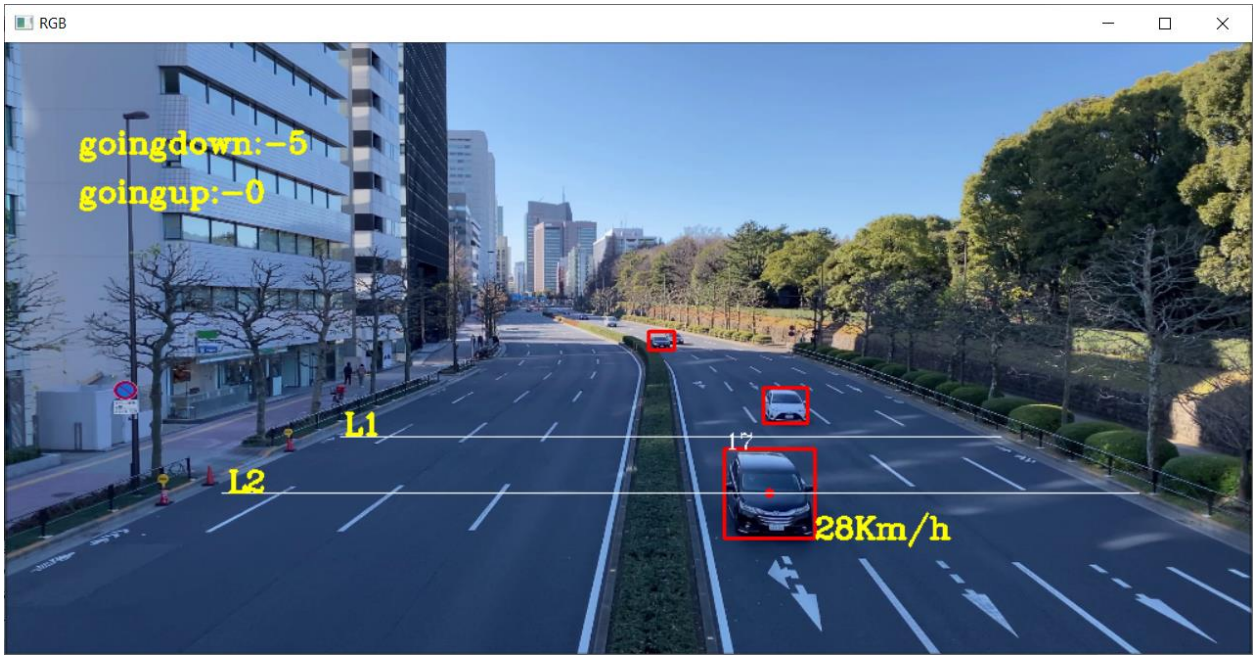


Рисунок 3.20 – Результат виконання коду Win OS

Як видно з рисунка, детектування об’єктів та визначення їх швидкості відбувається коректно. Окрім цього, в лівому куті можна помітити лічильники

автомобілей, вони відображають кількість автомобілей, які рухаються в різних напрямах.

Рисунок 3.21 – Збережені дані під час виконання коду

3.5 Налаштування Raspberry Pi та запуск програмної частини

Перед початком перевірки роботи програмної частини на мінікомп’ютері Rasspberry Pi необхідно провести певні налаштування.

Перш за все, обов’язково треба перевірити щоб встановлена ОС була 64 розрядною, тому що деякі пакети не працюють на 32 розрядній ОС, можуть некоректно встановлюватись (не всі пакети), або ж взагалі спроба встановити завершується помилкою.

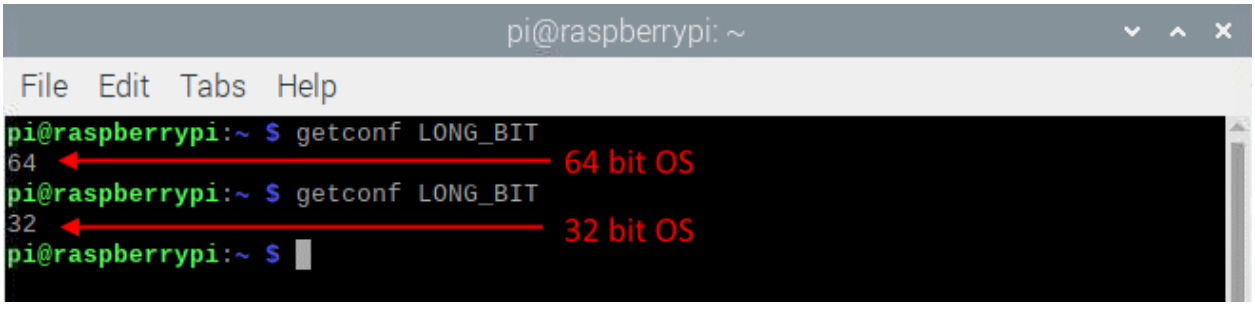


Рисунок 3.22 – Перевірка яка версія ОС встановлена

За допомогою цієї команди можна швидко з’ясувати яка ОС встановлена на Raspbery Pi. Якщо виявилось, що ОС 32 розрядна, то необхідно перевстановити, інакше не запуститься виконання коду.

Наступний крок це встановлення необхідних бібліотек, а саме:

– opencv: бібліотека, що надає набір інструментів та функцій для відкриття, обробки та аналізу зображень і відео;

– numpy: це основний пакет для наукових обчислень на Python;

– pandas: це пакет Python, який забезпечує гнучкі структури даних, розроблені для того, щоб зробити роботу даними достатньо простою та інтуїтивно зрозумілою;

– pyarrow: ця бібліотека надає Python API для функцій, які надаються бібліотеками Arrow C++, а також інструменти для інтеграції Arrow і взаємодії з pandas, NumPy та іншим програмним забезпеченням в екосистемі Python;

– ultralytics YOLOv8: модель виявлення об’єктів у реальному часі та сегментації зображення.

Після встановлення всіх необхідних бібліотек, можна переходити до перевірки роботи програмної частини.

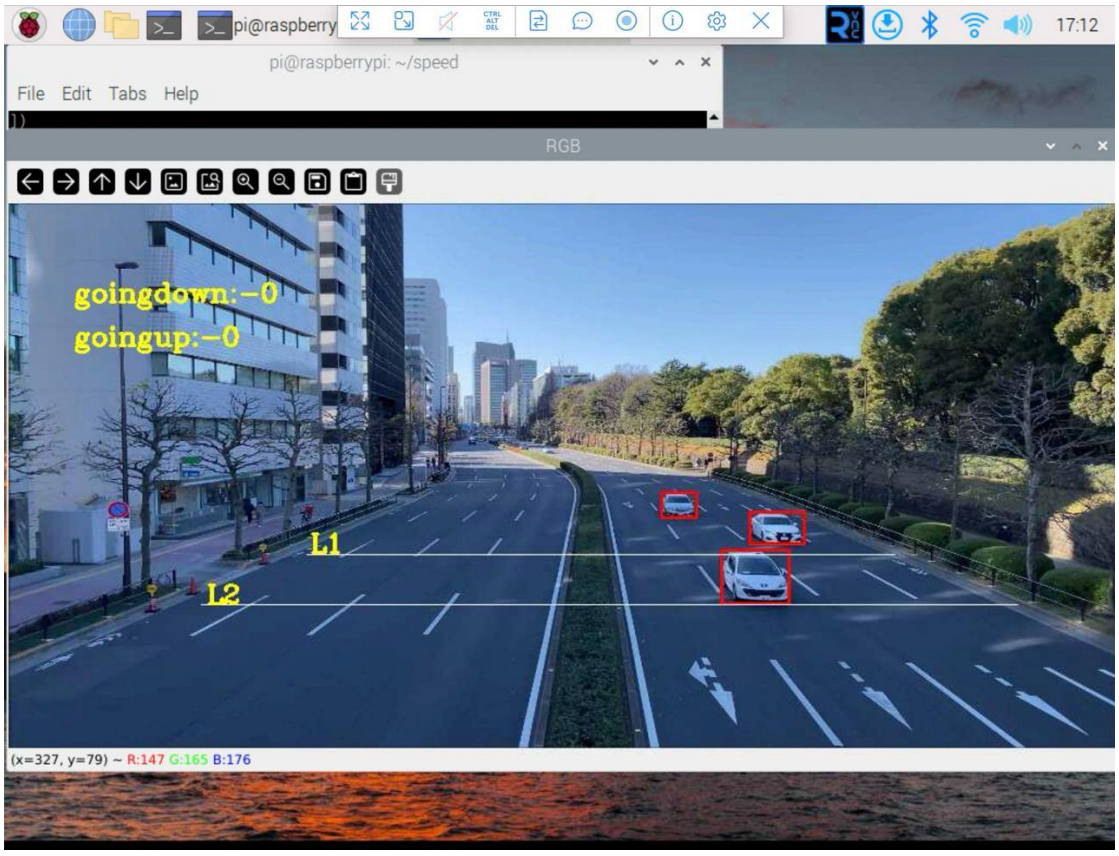


Рисунок 3.23 – Запуск програмної частини на Raspberry Pi

Програмна частина успішно запустилась, проте слід зазначити що потужностей даного мінікомп'ютера не вистачає для довготривалого використання та забезпечення коректної роботи.

3.6 Raspberry Pi YOLOv8 stream

Перед початком роботи з YOLO на Raspberry Pi [25], рекомендовано оновити свій мінікомп’ютер.

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get autoremove -y
```

Рисунок 3.22 – Команди для оновлення Raspberry Pi

Також необхідно встановити на мінікомп’ютер останню версію Python та ultralytics Python пакет. Після встановлення всього необхідного треба перезавантажити мінікомп’ютер.

Наступний крок – це запуск TCP-потоку за допомогою Libcamera.

```
libcamera-vid -n -t 0 --width 1280 --height 960 --framerate 1 --inline --listen -o tcp://127.0.0.1:8888
```

Рисунок 3.23 – Команда для запуску TCP-потоку

Щоб перевірити чи все було правильно встановлено та коректно працює, можна скористатися фрагментом коду з офіційної документації.

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
results = model('tcp://127.0.0.1:8888', stream=True)

while True:
    for result in results:
        boxes = result.boxes
        probs = result.probs
```

Рисунок 3.24 – Перевірка коректної роботи

3.7 Висновки до розділу 3

В даному розділі була детально проаналізована робота алгоритму YOLOv8, який використовується під час реалізації додатка детектування об’єктів та обчислення їх швидкості. Був проведений аналітичний огляд трьох основних частин даного алгоритму.

Крім цього були наведені характеристики апаратної частини, та проведений огляд програмної частини. Для реалізації програмної частини була обрана мова програмування Python, тому що Python має потужну екосистему бібліотек для машинного зору, таких як OpenCV, TensorFlow, PyTorch та велику активну спільноту, що дозволяє знаходити відповіді та рішення на різноманітні питання.

Також було проведено попереднє налаштування середовища розробки та підготовка до розробки програмної частини. Проведений огляд програмного коду реалізованого додатку детектування об’єктів та визначення їх швидкості, були розглянуті окремі важливі частини коду, які відповідають за виявлення транспортного засобу та обчислення його швидкості. Перевірені результати виконання коду.

Окрім цього було наведено порівняння продуктивності та швидкості моделей YOLOv8. Також була розглянута можливість використання YOLOv8 з мінікомп’ютером Raspberry Pi, згідно з офіційною документацією було проведено налаштування Raspberry Pi та був запущений TCP-поток, для того щоб замість готового відео використовувати відеопотік в режимі реального часу.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Згідно з офіційною документацією для коректної роботи YOLOv8 та для забезпечення достатнього рівня продуктивності система має відповідати наступним вимогам:

– 4-ядерний центральний процесор;

– для детектування об’єктів мінімум 2 Гб оперативної пам’яті, у випадку тренування моделей рекомендовано мати 16 Гб оперативної пам’яті;

– 64-розрядна операційна система.

Конкретні вимоги до обладнання можуть залежати від різних факторів, у тому числі індивідуальної складності та характеристик вхідного набору даних. Потрібно стежити за використанням ресурсів системи під час навчання мережі чи детектування об’єктів, щоб переконатися, що система працює ефективно. У випадку використання мінікомп’ютерів сімейства Raspberry Pi, рекомендовано використовувати як мінімум Raspberry Pi 4 для оптимальної роботи YOLOv8, проте ця модель Raspberry Pi потребує легких моделей YOLOv8. У випадку використання більш складних моделей можуть виникати проблеми зі стабільністю роботи та продуктивністю.

4.1 Визначення рівня навантаження на систему

Було визначене навантаження на систему, додаток запускався на ноутбуці з центральним процесором восьмиядерним AMD Ryzen 7 з тактовою частотою 3.8 ГГц та об’ємом оперативної пам’яті 16 Гб.

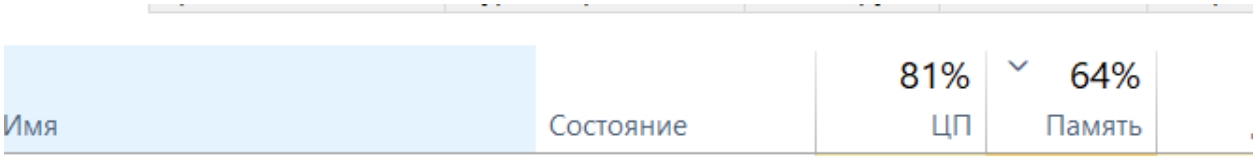


Рисунок 4.1 – Навантаження на систему під час виконання коду

Коли трафік збільшується та збільшується кількість об'єктів навколо яких треба нанести bounding box, то навантаження на ЦП становить 80-85%, а коли трафіку стає менше або взагалі відсутній, то навантаження становить 65-75%.

4.2 Порівняння рівня навантаження в залежності від використаної моделі YOLOv8

Використання моделі yolov8x.pt. Дана модель забезпечує найбільшу точність детектування, але сама повільна серед всіх наявних моделей.

Використання даної моделі навантажує ЦП на 85% при розгоні до 4.1 ГГц тактової частоти.

Крім найбільшої моделі була використана найменша yolov8n.pt. Дана модель забезпечує найбільшу швидкість, проте має низьку точність у порівнянні з іншими. Також слід зазначити що навантаження на ЦП становило 65-68% з частотою 3.8 ГГц.

Найбільш оптимальна модель для даного сценарію – це модель yolov8s.pt. Вона забезпечує достатню швидкість і точність детектування відповідно.

4.3 Визначення рівня навантаження на систему Raspberry Pi

Були проведені дослідження в результаті яких були визначені навантаження на систему.

Спочатку була використана модель yolov8s.pt, одна серед найменших моделей.

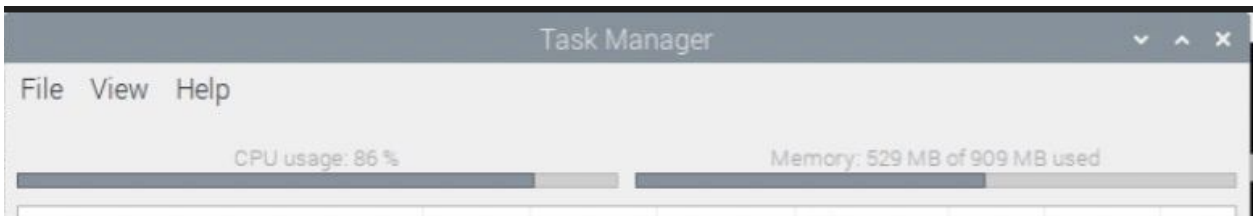


Рисунок 4.2 – Навантаження на систему Rasspberry Pi з використанням yolov8s.pt моделі

Навантаження в середньому становить 85%, проте слід зазначити що такий рівень навантаження лише коли мало трафіку, коли на фреймах присутня велика кількість автомобілей, то навантаження зростає до 95%, місцями становить 100% та виникають фрізи та підвисання, що негативно впливає на отримані результати.

Після чого була використана найменша модель, яка забезпечує найвищу швидкість та найнижчу точність детектування yolov8n.pt.

Рисунок 4.3 - Навантаження на систему Rasspberry Pi з використанням yolov8n.pt моделі

Отриманий результат дуже схожий на попередній, рівень навантаження нижчий лише на декілька відсотків.

Також слід зазначити що, процесор Rasspberry Pi сильно нагрівається під час виконання коду, що не дозволяє використовувати Rasspberry Pi тривалий час.

Для покращення роботи програмної частини на даному мікрокомп’ютері потрібно виконувати додаткову оптимізацію коду або використовувати додатковий пристрій для підвищення рівня продуктивності Edge TPU [27].

Coral USB Accelerator with Edge TPU - USB-прискорювач призначений для підвищення обчислювальної потужності системи для виконання програм машинного навчання. Цей пристрій має стандартний інтерфейс (USB-порт), що робить його сумісним з багатьма системами. Coral USB Accelerator додає до системи співпроцесор Edge TPU, здатний виконувати 4 трильйони операцій (тераопераций) в секунду (TOPS), використовуючи 0,5 Вт на кожен TOPS (2 TOPS на Ват).

Рисунок 4.4 – Зовнішній вигляд Edge TPU

4.4 Висновки до розділу 4

В даному розділі були проведенні дослідження навантаження на систему під час роботи реалізованої програмної частини. Дослідження були проведені на ноутбуці з центральним процесором восьмиядерним AMD Ryzen 7 з тактовою частотою 3.8 ГГц (4.2 ГГц при розгоні) та об’ємом оперативної пам’яті 16 Гб. Виходячи з результатів проведених досліджень, слід зазначити що для виявлення об’єктів з використанням моделі yolov8n.pt чи yolov8s.pt, цілком достатньо потужності процесора AMD Ryzen 7, для забезпечення стабільної роботи програмної частини. Проте у випадку використання моделі yolov8x.pt, що забезпечує найбільшу точність виявлення об’єктів, чи у випадку навчання мережі, чи підготовці власної моделі даних потрібно мати значно потужнішу систему, щоб забезпечити стабільність та швидкість роботи.

Крім цього було визначено, що згідно з офіційною документацією, у випадку використання мінікомп’ютерів сімейства Raspberry Pi рекомендовано використовувати як мінімум модель Raspberry Pi 4 для оптимальної роботи YOLOv8, проте ця модель Raspberry Pi потребує легких моделей YOLOv8. У випадку використання більш складних моделей виникатимуть проблеми зі стабільністю роботи та продуктивністю.

Також було перевірено наскільки коректно працює програмна частина на Raspberry Pi 3 B+. Хоча на офіційному сайті зазначено, що можна використовувати третю модель сімейства Raspberry Pi, але використання даної моделі призводить до нестабільної роботи та результатів низької точності, тому краще використовувати модель Raspberry Pi 4 чи Raspberry Pi 5.

ВИСНОВКИ

В ході виконання кваліфікаційної магістерської роботи був проведений аналіз існуючих моніторингових систем контролю дорожнього руху, технологій які використовуються в цих системах та методів для визначення швидкості автомобіля. Крім цього були проаналізовані переваги та недоліки кожної системи, також проведено їх порівняння.

Був проведений детальний аналіз методів та алгоритмів для детектування об’єктів на зображені або відео, які надає бібліотека OpenCV. Було визначено, що для точного детектування об’єктів необхідно обов’язково конвертувати вихідне зображення в бінарне, адже алгоритми детектування мають дуже низьку точність при використанні кольорового зображення, тому що некоректно визначають граничні пікселі об’єктів в результаті чого будуються неправильні контури навколо об’єктів.

Крім цього був проаналізований новий алгоритм з використанням нейронної мережі для детектування об’єктів YOLOv8, який забезпечує високу точність детектування та високий рівень швидкості детектування, що дозволяє використовувати цей алгоритм в системах реального часу. Також YOLOv8 надає вже підготовленні моделі, на цей час їх доступно п’ять варіантів. Окрім готових моделей є можливість створити свою власну модель та навчити її. Був детально розглянутий принцип його роботи та архітектура даного алгоритму.

Була реалізована програмна частина мовою програмування Python з використанням методів бібліотеки OpenCV та YOLOv8. Створений додаток дозволяє детектувати рухомі об'єкти (автомобілі), обчислювати їх швидкість та зберігати дані у файл. В якості апаратної частини була обрана наявна Raspberry Pi 3 Model B+ та модуль камери Raspberry Pi Cam. Було проведено тестування роботи додатка та визначений рівень навантаження на систему під час виконання коду.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАНЬ

1. OpenCV – Open Computer Vision Library. URL: https://opencv.org/ (Last accessed:14.01.2024).
2. Rosebrock A. OpenCV Vehicle Detection, Tracking, and Speed Estimation. URL: https://pyimagesearch.com/2019/12/02/opencv-vehicle-detection-tracking-and-speed-estimation/ (Last accessed:14.01.2024).
3. Detect Speed with a Raspberry Pi, Camera and OpenCV. URL: https://core-electronics.com.au/guides/detect-speed-raspberry-pi/ (Last accessed:14.01.2024).
4. SPECS3 (Speed Check Services) URL: https://www.gov.uk/government/publications/specs3-speedmeter (Last accessed:14.01.2024).
5. VITRONIC PoliScan Speed URL: https://www.vitronic.com/en-us/traffic-technology/speed-enforcement (Last accessed:14.01.2024).
6. Redflex Speed Enforcement URL: https://redflex.com/solution/radarcam/ (Last accessed:14.01.2024).
7. Install OpenCV on Raspberry Pi URL: https://qengineering.eu/install-opencv-on-raspberry-pi.html (Last accessed:14.01.2024).
8. Raspberry Pi Camera Options. URL: https://www.arrow.com/en/research-and-events/articles/raspberry-pi-camera-options-and-usage (Last accessed:14.01.2024).
9. Raspberry Pi OpenCV image processing. URL: https://maker.pro/raspberry-pi/tutorial/how-to-read-display-and-save-images-with-opencv-on-raspberry-pi (Last accessed:14.01.2024).
10. Raspberry Pi Camera Connection URL: https://www.raspberrypi.com/documentation/accessories/camera.html (Last accessed:14.01.2024).
11. Raspbian URL: https://www.raspbian.org/ (Last accessed:14.01.2024).

12. Raspberry Pi. URL: https://www.raspberrypi.org/ (Last accessed:14.01.2024).
13. Raspberry Pi vehicle detection. URL: https://pyimagesearch.com/2019/12/02/opencv-vehicle-detection-tracking-and-speed-estimation/ (Last accessed:14.01.2024).
14. OpenCV object tracking. URL: https://learnopencv.com/the-complete-guide-to-object-tracking-in-computer-vision/ (Last accessed:14.01.2024).
15. Raspberry Pi OpenCV computer vision. URL: https://shorturl.at/fhszB (Last accessed:14.01.2024).
16. Матюшок М. М., Пузирьов С. В. Потокова передача відео засобами WEBRTC. Могилянські читання – 2023: тези доповідей XXVI Всеукраїнська науково-практична конференція. Миколаїв, 6–10 листоп. 2023 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2023. С. 430.
17. Euclidean Metric. URL: https://mathworld.wolfram.com/EuclideanMetric.html (Last accessed:18.01.2024).
18. R-CNN Algorithm. URL: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html (Last accessed:18.01.2024).
19. Ultralytics YOLO. URL: https://docs.ultralytics.com/ (Last accessed:20.01.2024).
20. Raspberry Pi 3 Model B+. URL: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-3-model-b (Last accessed:20.01.2024).
21. Raspberry Pi Remote access via SSH. URL: https://www.raspberrypi.com/documentation/computers/remote-access.html (Last accessed:20.01.2024).

22. RealVNC Documentation. URL: https://help.realvnc.com/hc/en-us/categories/360000301637-Documentation (Last accessed:20.01.2024).
23. PuTTY Documentation Page. URL: https://www.chiark.greenend.org.uk/~sgtatham/putty/docs.html (Last accessed:20.01.2024).
24. YOLOv8 Architecture. URL: https://www.researchgate.net/figure/The-architecture-of-YOLOv8-algorithm-which-is-divided-into-four-parts-including_fig2_375697582 (Last accessed:26.01.2024).
25. Raspberry Pi & YOLOv8. URL: https://docs.ultralytics.com/guides/raspberry-pi/ (Last accessed:28.01.2024).
26. YOLOv8 Object Detection Datasets. URL: https://docs.ultralytics.com/datasets/detect/ (Last accessed:29.01.2024).
27. Coral Edge TPU Raspberry Pi and YOLOv8. URL: https://docs.ultralytics.com/guides/coral-edge-tpu-on-raspberry-pi/ (Last accessed:30.01.2024).
28. Difference between YOLO and SSD. URL: https://www.geeksforgeeks.org/difference-between-yolo-and-ssd/ (Last accessed:03.02.2024).
29. Comparison between YOLO and SSD MobileNet for Object Detection. URL: https://www.researchgate.net/publication/355336797_Comparison_between_YOLO_and_SSD_MobileNet_for_Object_Detection_in_a_Surveillance_Drone (Last accessed:03.02.2024).
30. Real time embedded system for object detection using deep learning. URL: https://www.researchgate.net/publication/366316255_Real_time_embedded_system_for_object_detection_using_deep_learning (Last accessed:03.02.2024).

ДОДАТОК А

ДОВІДКА

про перевірку на унікальність пояснювальної записки кваліфікаційної магістерської роботи

на тему: «Моніторингова мережа контролю дорожнього руху на основі Raspberry Pi та OpenCV»

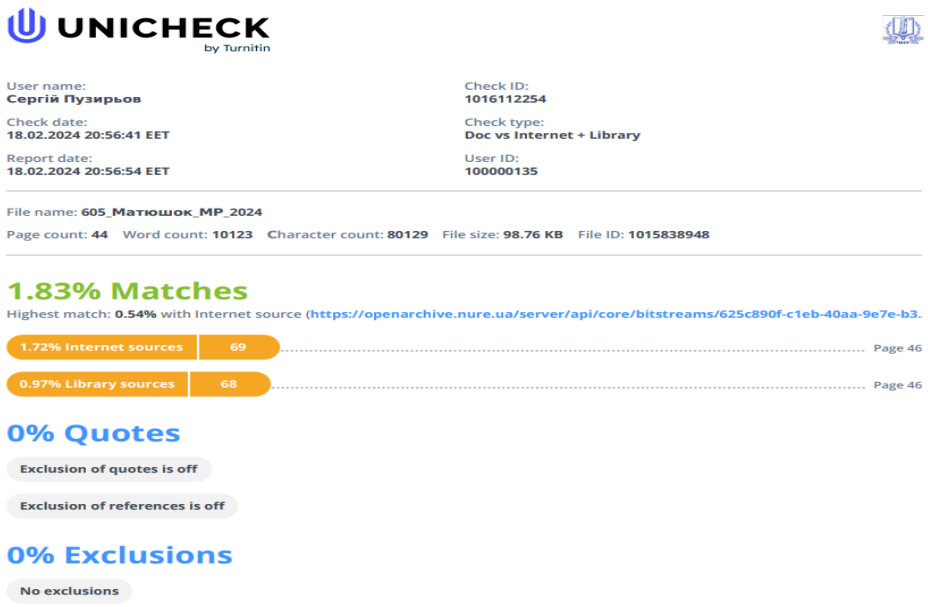
студента спеціальності 123 «Комп'ютерна інженерія», групи 605м

Матюшка Максима Миколайовича

прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту кваліфікаційної роботи: схожість складає 1,83 %.


UNICHECK
by Turnitin

User name: Сергій Пузирьов
Check ID: 1016112254
Check date: 18.02.2024 20:56:41 EET
Check type: Doc vs Internet + Library
Report date: 18.02.2024 20:56:54 EET
User ID: 100000135

File name: 605_Матюшок_МР_2024
Page count: 44 Word count: 10123 Character count: 80129 File size: 98.76 KB File ID: 1015838948

1.83% Matches
Highest match: 0.54% with Internet source (https://openarchive.nure.ua/server/api/core/bitstreams/625c890f-c1eb-40aa-9e7e-b3.

1.72% Internet sources 69 Page 46
0.97% Library sources 68 Page 46

0% Quotes
Exclusion of quotes is off
Exclusion of references is off

0% Exclusions
No exclusions

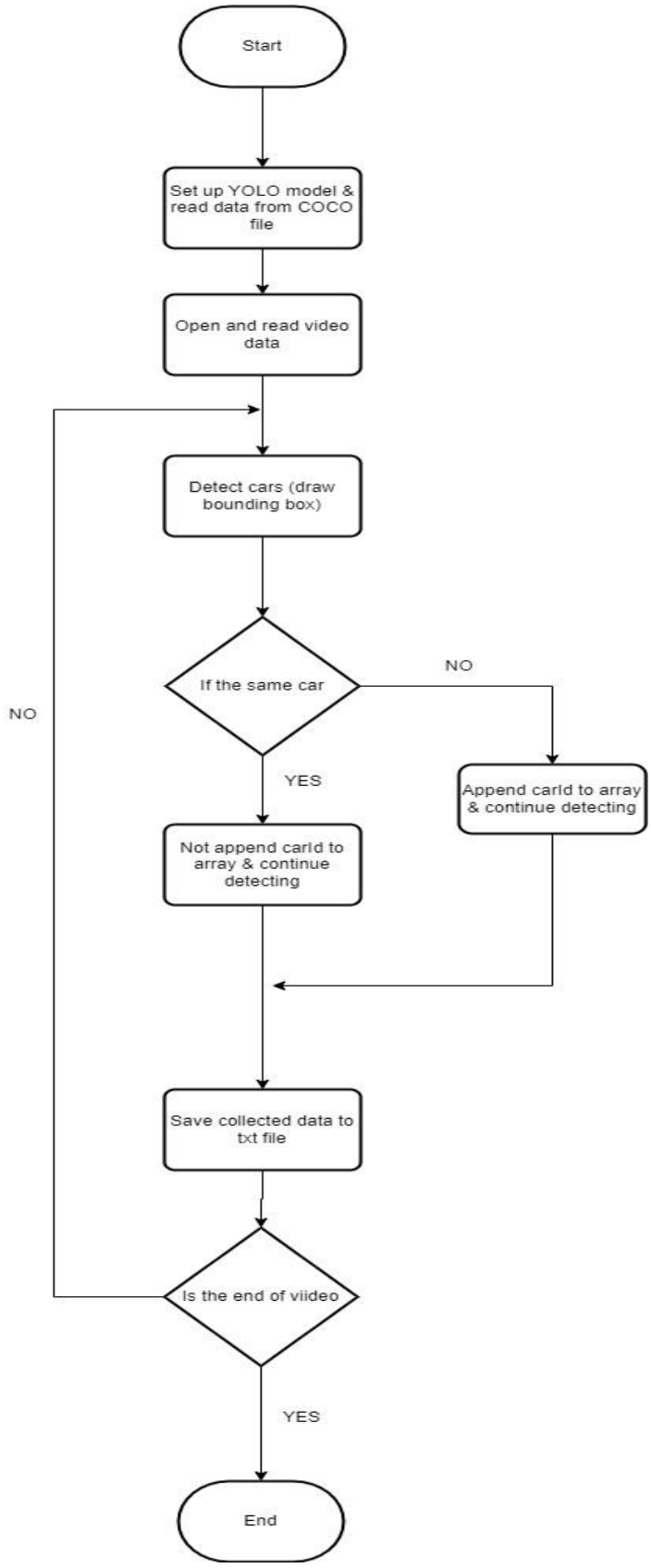
Студент:

М. М. Матюшок
підпис ініціали, прізвище

Керівник:
канд. фіз.-мат. наук, доц
С. В. Пузирьов
підпис ініціали, прізвище

Дата: «___» _____ 2023 р.

ДОДАТОК Б


Start
Set up YOLO model & read data from COCO file
Open and read video data
Detect cars (draw bounding box)
If the same car
NO
YES
NO
Append carId to array & continue detecting
Not append carId to array & continue detecting
Save collected data to txt file
Is the end of viideo
YES
End

ДОДАТОК В

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України

«МОГИЛЯНСЬКІ ЧИТАННЯ – 2023:
досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти»

XXVI Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

Миколаїв, 6–10 листопада 2023 року

Миколаїв – 2023

Тези доповідей

Підсекція:
ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ СИСТЕМИ

УДК 004.01

Матюшок М. М.,
магістрант,
Пузирьов С. В.,
канд. фіз.-мат. наук, доцент,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

ПОТОКОВА ПЕРЕДАЧА ВІДЕО ЗАСОБАМИ WEBRTC

Найбільші проблеми при організації потокової передачі відео – це втрата якості зображення, затримка відео трансляції, складність організації такої системи та її вартість. Однак WebRTC технологія допомагає вирішувати вище перелічені проблеми.

WebRTC – це технологія для передачі відео, аудіо та інших даних в режимі реального часу. За допомогою WebRTC можна створити надійний засіб комунікації без будь-яких додаткових програм чи плагінів.

Більшість систем відео- та аудіозв'язку зазвичай розробляються на C/C++, і встановлення стабільного робочого з'єднання може займати кілька місяців розробки. У відміну від цього, WebRTC використовує рівень API на основі Javascript, який можна використовувати прямо у веб-браузері. Хоча технічно WebRTC все ще написано мовами C/C++, проте, використання API на основі JavaScript значно спрощує процес розробки та зменшує витрати часу. WebRTC працює у більшості сучасних веб-браузерах і надає можливість отримувати доступ до пристроїв та демонструвати екран у реальному часі (рис. 1).

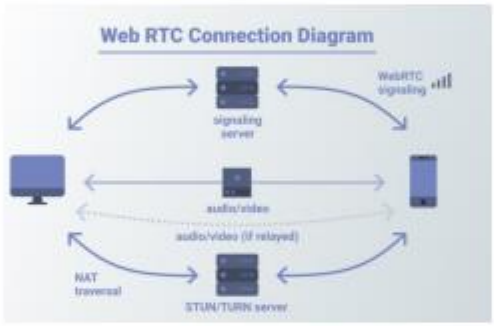


Рисунок 1 – Діаграма WebRTC з'єднання

Як і будь-яка інша технологія, WebRTC має свої переваги та недоліки, про які слід знати, перш ніж починати розробку продукту на основі WebRTC.

Переваги WebRTC:

- висока якість зв'язку завдяки сучасним відеокодекам (VP8, H264) і аудіокодекам (Opus);
- якість відео автоматично підлаштовується під будь-який тип підключення;
- вбудована підтримка шумозаглушення та придушення відлуння;
- автоматичне керування чутливістю мікрофона;
- високий рівень безпеки: всі з'єднання захищені (HTTPS) і зашифровані (SRTP);
- багатоплатформність.

Проте існують і деякі вагомі недоліки:

- оскільки браузери не можуть синхронізувати кілька вхідних потоків, потрібен сервер для мікшування аудіо та відео для проведення групових аудіо- чи відеоконференцій;
- перекодування групових конференцій вимагає великих обчислювальних ресурсів. Підтримка групових відеоконференцій для WebRTC зазвичай вимагає платної підписки (у випадку хмарних рішень).

Завдяки бібліотекам JavaScript на основі API WebRTC можна додати функціонал цієї технології будь-який проект за короткий термін і з мінімальними зусиллями.

У WebRTC існують три основні категорії API:

1) багатоплатформність;