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, професор
_____ І. М. Журавська
« __ » _____ 2024 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Система контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки OpenCV

Спеціальність 123 Комп'ютерна інженерія
123 – КМР.1 – 605.21810512

Студент _____ В. М. Кім
підпис
«__»_____202__р.

Керівник доцент, канд. фіз.-мат. наук _____ 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

Кіму Віктору Миколайовичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Система контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки 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 розпізнавання дорожніх об'єктів та виведення корисної інформації водію. Вхідними даними роботи є специфікація вимог, що описує характеристики зазначеного апаратного та програмного забезпечення.

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

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
Д-р біол. наук, професорка Григор'єва Л. І.	Кафедра екології Навчально-наукового медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці та безпеки у надзвичайних ситуаціях

Керівник роботи

_канд. фіз.-мат. наук, доцент Пузирьов Сергій Володимирович_____

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_Кім Віктор Миколайович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема: _Система контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки 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	05.01.2023	*Виконано*
9.	Аналіз результатів тестування	05.01.2023	10.01.2023	*Виконано*
10.	Розробка керівництва користувача	10.01.2023	13.01.2023	*Виконано*
11.	Відгук керівника КМР	13.01.2023	05.02.2023	*Виконано*
12.	Оформлення КМР та презентації	07.02.2023	08.02.2023	*Виконано*
13.	Попередній захист	31.01.2023	14.02.2023	*Виконано*
14.	Рецензування	12.02.2023	19.02.2023	*Виконано*
15.	Захист кваліфікаційної роботи	27.02.2023	27.02.2023	*Виконано*

Розробив здобувач ВО Кім Віктор Миколайович_____

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

«__» _____ 20__ р.

Керівник роботи канд. фіз.-мат. наук Пузирьов Сергій Володимирович__

(підпис)

«__» _____ 20__р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Система контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки OpenCV»

Студент: Кім Віктор Миколайович

Керівник: доцент, канд. фіз.-мат. наук Пузирьов С. В.

Актуальність теми кваліфікаційної роботи обумовлена тим, що в сучасному світі, де населення постійно зростає, а міські інфраструктури намагаються впоратися зі зростаючим трафіком транспорту та пішоходами. Великі та середні міста стикаються з низкою проблем, пов'язаних зі збільшенням автотранспортного руху, що призводить до заторів, дорожньо-транспортних пригод (ДТП) та загального погіршення якості життя мешканців.

Однією з основних проблем є зростання кількості ДТП, що може призвести до серйозних травм або навіть загибелі людей. Статистика ДТП свідчить про те, що велика частина аварій стається через порушення правил дорожнього руху та недбале ставлення до безпеки на дорозі.

Збільшення навантаження на моніторингові системи також є важливою проблемою. Міста намагаються впроваджувати різноманітні системи контролю за рухом, такі як відеоспостереження, системи фіксації порушень швидкісного режиму, контроль за виїздом на перехрестях та інші. Проте зростаюче навантаження може призвести до перевантаження систем та зниження їх ефективності.

Справжнім викликом є своєчасне реагування на порушення руху з мінімальним часовим інтервалом. Це означає, що системи моніторингу повинні бути досконалішими і ефективнішими, щоб оперативно реагувати на порушення та негайно приймати заходи щодо їх усунення. Наприклад, автоматизовані системи виявлення порушень можуть надсилати сповіщення поліції або автоматично застосовувати штрафи за порушення правил руху.

Отже, дослідження впровадження та оптимізація моніторингових систем для управління трафіком та безпеки на дорозі у великих та середніх містах є надзвичайно актуальними та важливими для покращення якості життя мешканців та забезпечення безпеки на дорогах.

Доповнена реальність (Augmented Reality, AR) і бібліотека OpenCV (Open Source Computer Vision Library) представляють собою два передові технологічні інструменти, які можуть бути успішно використані для досягнення цієї мети. Інтеграція AR та OpenCV в системи контролю дорожнього руху може створити потужні інструменти для аналізу, моніторингу та оптимізації дорожнього середовища, а також для підвищення безпеки всіх учасників дорожнього руху.

Мета та завдання дослідження.

Мета: розробити систему контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки OpenCV.

Об’єкт дослідження (розробки): технологія інтеграції мікропроцесорної системи з комп’ютерним зором для системи контролю дорожнього руху.

Предмет дослідження (розробки): особливості розробки системи контролю дорожнього руху в апаратно-програмних мережах на основі доповненої реальності.

Завдання:

– розглянути сучасні системи контролю дорожнього руху;

– проаналізувати недоліки існуючих систем;

– розробити апаратну частину;

– розробити програмну частину;

– створити макетну модель для демонстрації роботи.

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

Вступ містить основні обґрунтування актуальності розробки обраної теми, об’єкт, предмет дослідження, мету та завдання які необхідно виконати для досягнення поставленої мети.

В першому розділі проведено огляд засобів контролю дорожнього руху. Розроблено специфікацію вимог до апаратно-програмного забезпечення.

Другий розділ містить опис математичних методів для проєктування апаратно-програмного забезпечення.

У третьому розділі наведено опис процесу розробки апаратного забезпечення та розробки програмного забезпечення для програмно-апаратного комплексу розпізнавання та інтеграції дорожніх об’єктів.

Четвертий розділ присвячено тестування розробленого програмного та апаратного забезпечення.

У висновку описано результати виконання кваліфікаційної роботи.
Додатки містять перевірку на унікальність, код програмного забезпечення та інформацію про апробацію кваліфікаційної роботи.

В спеціальній частині з охорони праці розглянуто забезпечення вимог охорони праці на робочому місці.

Кваліфікаційна робота містить 65 сторінок (без додатків), 43 рисунка, 33 джерел посилання, 3 додатки.

Ключові слова: система контролю дорожнього руху, доповнена реальність, розпізнавання об’єктів, комп’ютерний зір, OpenCV.

ABSTRACT

of the Master’s Thesis

"Traffic control system based on augmented reality using the OpenCV library"

Student: Viktor Kim

Consultant: associate professor, cand. phys.-math. sciences Puzyrev Serhii

The relevance of the subject of the qualification work is due today's world, where the population is constantly growing, and urban infrastructures are struggling to cope with the increasing traffic of both vehicles and pedestrians. Large and medium-sized cities face a range of problems associated with the growth in motor vehicle traffic, leading to traffic jams, road accidents (RTAs), and a general deterioration in the quality of life for residents.

One of the main issues is the increase in the number of RTAs, which can result in serious injuries or even fatalities. RTA statistics indicate that a significant portion of accidents occur due to violations of traffic rules and a lack of attention to road safety.

The increasing burden on monitoring systems is also a significant problem. Cities are trying to implement various traffic control systems, such as video surveillance, speed violation detection systems, intersection control, and more. However, the growing workload can lead to system overload and a decrease in their effectiveness.

A real challenge is timely response to traffic violations with minimal time intervals. This means that monitoring systems must be more sophisticated and efficient to promptly respond to violations and take immediate action to address them. For example, automated violation detection systems can alert the police or automatically apply fines for traffic rule violations.

Therefore, research into the implementation and optimization of monitoring systems for traffic management and road safety in large and medium-sized cities is highly relevant and important for improving the quality of life for residents and ensuring road safety.

Augmented Reality (AR) and OpenCV (Open Source Computer Vision Library) are two advanced technological tools that can be successfully used to achieve this goal. The integration of AR and OpenCV into traffic control systems can create powerful tools for analyzing, monitoring, and optimizing the road environment, as well as improving the safety of all road users.

Objective: to develop a traffic control system based on augmented reality using the OpenCV library.

Object of research (development): the technology of integrating a microprocessor system with computer vision for a traffic control system.

The subject of research (development): features of the development of a traffic control system in hardware and software networks based on augmented reality.

To achieve the goal, the following **tasks** were set:

– inspect modern traffic control systems;

– analyze the shortcomings of existing systems;

– develop the hardware part;

– develop the software part;

– create a mock-up model to demonstrate the work.

The qualification work contains: a list of abbreviations, an introduction, four chapters, a conclusion, a list of reference sources and two appendices.

The introduction contains the main reasons for the relevance of the development of the chosen topic, the object, the subject of research, the goal and the tasks that must be performed to achieve the goal.

In the first section, an overview of the means of traffic control was carried out. A specification of hardware and software requirements has been developed.

The second section contains a math methods of the hardware and software design process.

The third chapter provides a description of the process of hardware development and software development for the software-hardware complex of recognition and integration of road objects.

The fourth chapter is dedicated to testing the developed software and hardware.

The conclusion describes the results of the qualification work.

Appendices contain check for uniqueness, software code and information on the approval of the qualification work.

In the special part on labor protection, the provision of labor protection requirements at the workplace is considered.

The qualification paper contains 65 pages (without appendices), 43 figures, 33 reference sources, 3 appendices.

Keywords: traffic control system, augmented reality, object recognition, computer vision, OpenCV.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МПкс	–	Мегапіксель
ПК	–	Персональний комп’ютер
API	–	Application programming interface
AWS	–	Amazon Web Services
BOT	–	Robot
BT	–	Bluetooth
FTDI	–	Future Technology Devices Interglobal
GND	–	GROUND
GPIO	–	General-purpose Input/Output
I2C	–	Inter-Integrated Circuit
IP	–	Internet Protocol
IoT	–	Internet of things
LTP	–	Line Print Terminal
MQTT	–	Message queuing telemetry transport
QoS	–	Quality of service
SDK	–	Software development kit
SPI	–	Serial Peripheral Interface
TCP	–	Transmission Control Protocol
TTL	–	Time to live
TV	–	Television
UART	–	Universal asynchronous receiver/transmitter
USB	–	Universal Serial Bus
VCC	–	Voltage Common Collector
Wi-Fi	–	Wireless Fidelity

ВСТУП

Актуальність дослідження. В сучасному світі населення постійно зростає, а міські інфраструктури намагаються впоратися зі зростаючим трафіком транспорту та пішоходами. Великі та середні міста стикаються з низкою проблем, пов'язаних зі збільшенням автотранспортного руху, що призводить до заторів, дорожньо-транспортних пригод (ДТП) та загального погіршення якості життя мешканців.

Однією з основних проблем є зростання кількості ДТП, що може призвести до серйозних травм або навіть загибелі людей. Статистика ДТП свідчить про те, що велика частина аварій стається через порушення правил дорожнього руху та недбале ставлення до безпеки на дорозі.

Збільшення навантаження на моніторингові системи також є важливою проблемою. Міста намагаються впроваджувати різноманітні системи контролю за рухом, такі як відеоспостереження, системи фіксації порушень швидкісного режиму, контроль за виїздом на перехрестях та інші. Проте зростаюче навантаження може призвести до перевантаження систем та зниження їх ефективності.

Отже, дослідження впровадження та оптимізація моніторингових систем для управління трафіком та безпеки на дорозі у великих та середніх містах є надзвичайно актуальними та важливими для покращення якості життя мешканців та забезпечення безпеки на дорогах.

Доповнена реальність (Augmented Reality, AR) і бібліотека OpenCV (Open Source Computer Vision Library) представляють собою два передові технологічні інструменти, які можуть бути успішно використані для досягнення цієї мети. Інтеграція AR та OpenCV в системи контролю дорожнього руху може створити потужні інструменти для аналізу, моніторингу та оптимізації дорожнього середовища, а також для підвищення безпеки всіх учасників дорожнього руху.

Мета та завдання дослідження.

Мета дослідження: розробити систему контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки OpenCV, що дозволить підвищити безпеку учасників дорожнього руху та покращити управління транспортними потоками.

Об’єкт дослідження: технологія інтеграції мікропроцесорної системи з комп’ютерним зором для системи контролю дорожнього руху.

Предмет дослідження: особливості розробки системи контролю дорожнього руху в апаратно-програмних мережах на основі доповненої реальності.

Завдання:

- проаналізувати існуючі системи контролю дорожнього руху та їхні обмеження;
- дослідити архітектури системи на базі доповненої реальності для контролю дорожнього руху;
- дослідити алгоритми обробки відеоданих та виявлення порушень правил дорожнього руху за допомогою OpenCV;
- розробити систему для контролю дорожнього руху;
- провести експериментальні дослідження та оцінити ефективність розробленої системи на реальних дорожніх умовах;
- дослідити вимоги та стандарти з охорони праці.

Робота пройшла **апробацію** під час XXVI Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 06–10 листопада 2023 р.). Основні положення магістерської роботи опубліковані у збірнику матеріалів XXVI Всеукраїнської науково-практичної конференції «Могилянські читання–2023» [1].

1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ПАТЕНТНОЇ ІНФОРМАЦІЇ У СФЕРІ АПАРАТНО-ПРОГРАМНИХ МОДУЛІВ

З розвитком технологій та зростанням інтенсивності дорожнього руху стає важливим завданням розробка і впровадження ефективних систем контролю дорожнього руху. Одним із перспективних напрямків є використання доповненої реальності (AR) в поєднанні з бібліотекою комп'ютерного зору OpenCV. Цей розділ присвячений аналізу існуючих підходів та технологій у галузі систем контролю дорожнього руху з акцентом на використанні AR та OpenCV.

Традиційні системи використовуються для виявлення рухомих об'єктів за допомогою камер та сенсорів. Однак ці системи мають обмежену ефективність у роботі за складних умов, таких як низьке освітлення чи погана видимість через атмосферні умови.

Доповнена реальність вже успішно впроваджується в автомобільні технології для водіїв. Вона дозволяє відображати інформацію про шлях та об'єкти в реальному часі, але її потенціал ще не повністю реалізований у сфері систем контролю дорожнього руху.

1.1 Аналіз аналогічних систем

1.1.1 Head-up displays (HUD)

HUD - це прозорий дисплей, який відображає інформацію на спеціальне скло, що не перешкоджає видимості.

Система використовується для відображення даних на борту транспортних засобів, таких як літаки, автомобілі. Це дозволяє водіям або пілотам отримувати інформацію без відволікання уваги від дороги або повітряного простору. Початково цей метод був розроблений для військових літаків для відображення важливих даних під час польоту, але тепер він широко використовується і в цивільних цілях, зокрема в автомобілях

Деякі автомобільні виробники вже впроваджують Head-Up Display (HUD) системи. Ці системи часто надають інформацію про швидкість, навігацію та інші параметри, приклад такої системи на рис.1.1.

Рисунок 1.1 – Head-up displays (HUD)

По суті, HUDs працюють, проектуючи перевернуте зображення на лобове скло, яке потім відбивається прямо в очі водія. Автомобілі з проекційним дисплеєм часто мають дивну прямокутну форму у верхній частині приладової панелі перед циферблатами; це те, звідки насправді світиться зображення.

1.1.2 Програми автопілота для автономних транспортних засобів

Самокерований автомобіль , також відомий як автономний автомобіль (AC), автомобіль без водія або робот-автомобіль (robo-car) − це автомобіль, який здатний подорожувати без участі людини. Безпілотні автомобілі відповідають за сприйняття навколишнього середовища, моніторинг важливих систем і контроль, включаючи навігацію. Система контролю приймає візуальні та аудіодані ззовні та всередині автомобіля та інтерпретує

вхідні дані для абстрактного відтворення транспортного засобу та його оточення. Потім система керування вживає заходів для переміщення транспортного засобу, враховуючи маршрут, дорожні умови, засоби керування дорожнім рухом і перешкоди.

Робота автономного транспортного засобу вимагає активації 3 основних елементів керування:

– прискорення, активація дросельної заслінки;

– напрямок, рульове керування;

– зупинка, активація гальма.

Щоб транспортний засіб відтворював традиційний людський процес розуміння свого поточного місцезнаходження, бажаного місцезнаходження та способів безпечної навігації маршрутами та небезпеками між ними, потрібні різні датчики та комп’ютерні системи. Серед наявних наразі датчиків жоден одиничний пристрій не здатний розшифрувати навколишнє середовище таким чином, щоб забезпечити достатню інформацію для автономного водіння. Натомість інженери розробляють систему, яка включає в себе комбінацію датчиків, стратегічно використовуючи конкретні можливості однієї системи, одночасно усуваючи її недоліки за рахунок використання інших, можна побачити на рис.1.2.

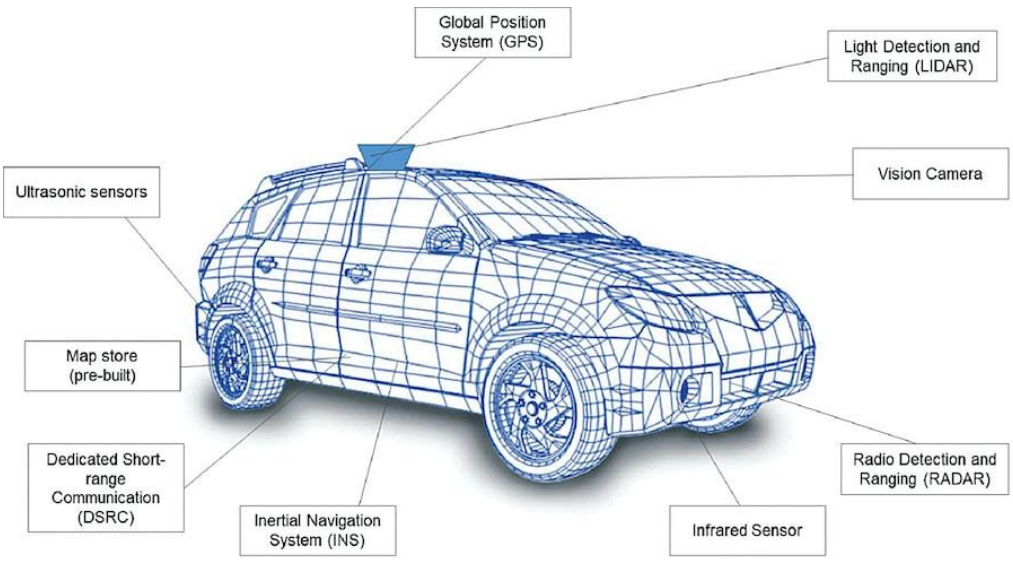


Рисунок 1.2 – Датчики та сенсори в самокерованому автомобілі

Радар використовує сплески звуку для вимірювання відстані шляхом вимірювання часу, необхідного для повернення звуку до датчика.

LIDAR – подібно до радару, але використовує лазери для вимірювання відстані замість звуку, із радіусом дії до 200 м.

Відеокамери записують декілька фотознімків (кадрів) для зйомки двовимірного відео.

Ультразвукові датчики діапазону використовують високочастотні звукові хвилі для вимірювання відстані між об'єктами.

Інерційний вимірювальний блок – комбінація акселерометрів і гіроскопів, які визначають лінійний і кутовий рух автомобіля.

Глобальна навігаційна супутникова система використовує супутники для забезпечення автономного геопросторового позиціонування. Глобальна система позиціонування (GPS) є підмножиною Глонасс.

1.1.3 Система розпізнавання дорожніх знаків (TSR)

Система контролює знаки обмеження швидкості під час руху та відображає їх водієві на дисплеї. Система використовує камеру, встановлену у верхній частині лобового скла, щоб визначити обмеження швидкості на дорожніх знаках [17]. Потім це обмеження швидкості відображається водієві на дисплеї активного водіння вашого автомобіля.

Система розпізнавання дорожніх знаків також виявляє та сповіщає водія про знаки «стоп» і «в'їзд заборонено», зображено на рис.1.3.

Рисунок 1.3 – Система розпізнавання дорожніх знаків (TSR)

Якщо під час руху транспортного засобу швидкість автомобіля перевищує знак обмеження швидкості, вказаний на дисплеї активного водіння, система сповіщає водія за допомогою індикації на дисплеї активного водіння та попереджувального звуку.

1.2 Підходи для детектування у системах контролю дорожнього руху

1.2.1 TenserFlow

Детектування об'єктів за допомогою TensorFlow – це процес використання бібліотеки машинного навчання TensorFlow для виявлення об'єктів на зображеннях або відео. Це може бути використано в різних областях, таких як системи контролю дорожнього руху, комп'ютерне зоро, медичне зображення та багато інших.

У TensorFlow є кілька методів для детектування об'єктів, але одним з найбільш популярних є використання предобучених моделей, таких як модель "SSD" (Single Shot MultiBox Detector), "Faster R-CNN" (Region-based

Convolutional Neural Networks) або "YOLO" (You Only Look Once). Ці моделі мають велику кількість класів об'єктів, які вони можуть виявляти, і можуть бути використані безпосередньо або доопрацьовані для конкретних завдань.

Процес детектування об'єктів в TensorFlow зазвичай включає наступні кроки:

Підготовка даних: Підготовка набору даних для тренування моделі, включаючи зображення та відповідні мітки для об'єктів, які потрібно виявити.

Вибір моделі: Вибір підходящої моделі детектування об'єктів, яка відповідає вимогам конкретного завдання.

Тренування моделі (необов'язково): Якщо немає підходящої попередньо навченоъ моделі, може бути потрібно тренувати модель на власних даних.

Тестування моделі: Оцінка результатів роботи моделі на тестовому наборі даних для перевірки її ефективності та точності.

Виявлення об'єктів на зображеннях або відео: Застосування натренованої моделі до нових зображень або відео для виявлення об'єктів та їх класифікації, приклад можна побачити на рис.1.4.

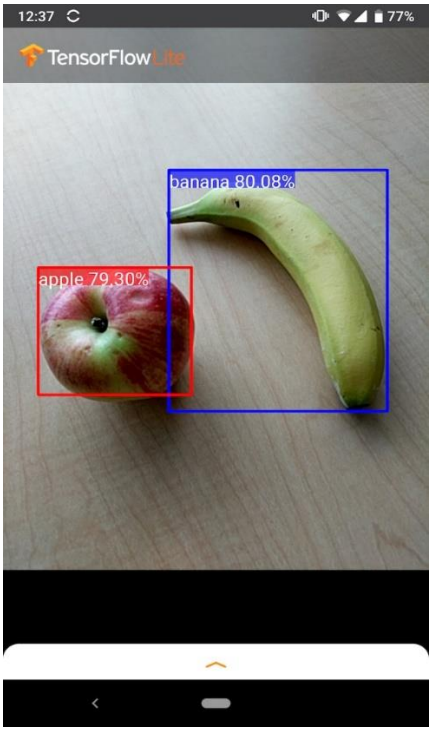


Рисунок 1.4 – Розпізнавання об'єктів за допомогою застосунку TenseFlow

Для використання TensorFlow для детектування об'єктів можна скористатися вбудованими функціями та інструментами, що надає бібліотека, а також відкритими джерелами, які розробляються спільнотою машинного навчання [10].

1.2.2 SimpleCV

SimpleCV - це бібліотека для обробки зображень та комп'ютерного зору, яка дозволяє виконувати різноманітні завдання, включаючи детектування об'єктів [8]. Для детектування об'єктів за допомогою SimpleCV можна використовувати методи та інструменти, доступні в цій бібліотеці. Ось кілька кроків, які можна виконати для детектування об'єктів з використанням SimpleCV:

Завантаження зображення: Завантажте зображення або відео, на якому потрібно виявити об'єкти, використовуючи SimpleCV.

Створення об'єкту Image: Створіть об'єкт Image для обробки зображення.

Використання методів детектування: Використовуйте методи та функції SimpleCV для детектування об'єктів на зображенні. Наприклад, ви можете використовувати функцію findBlobs() для пошуку областей, що мають певні характеристики, такі як колір або яскравість.

Обробка результатів: Обробіть результати детектування, якщо необхідно. Наприклад, ви можете відобразити області, де були виявлені об'єкти, або виконати подальші аналізи на їх основі.

Відображення результатів: Відобразіть зображення з виявленими об'єктами або іншими результатами аналізу (рис.1.5).

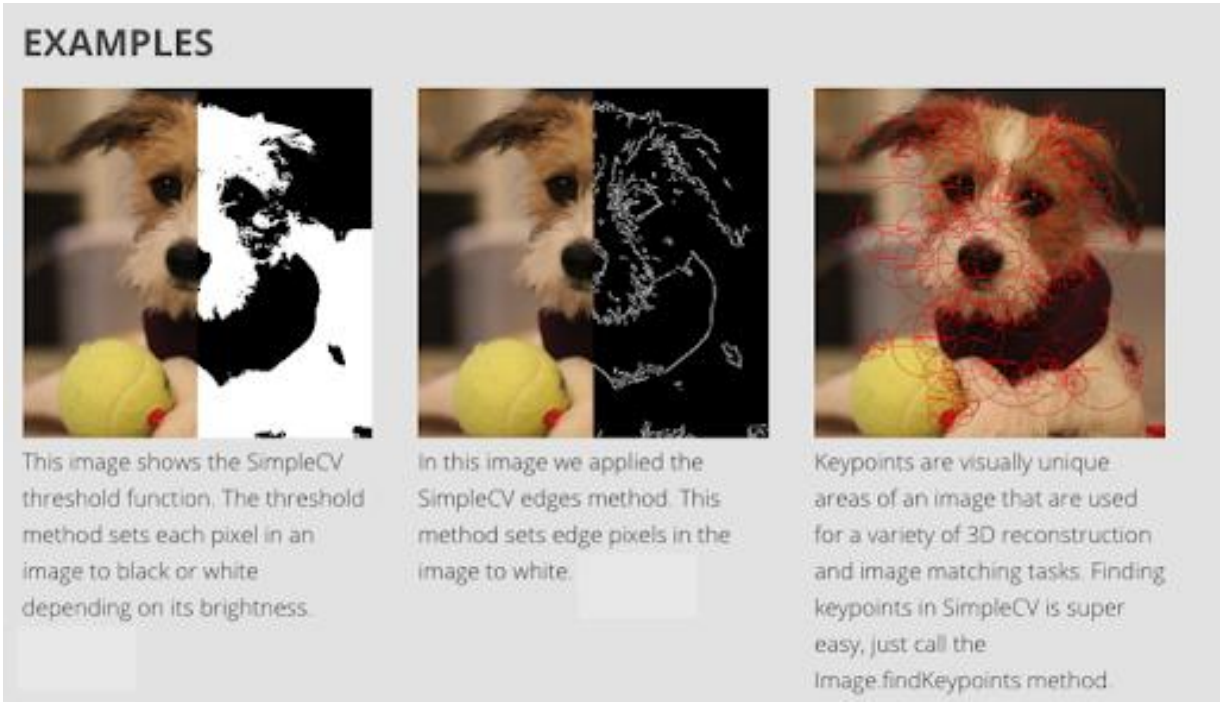


Рисунок 1.5 – Методи детектування SimpleCV

Примітно, що SimpleCV пропонує простий та зручний інтерфейс для роботи з обробкою зображень, але може бути менш потужним в порівнянні з більш розширеними бібліотеками, такими як OpenCV або TensorFlow. Однак, для простих задач детектування об'єктів SimpleCV може бути досить ефективним і зручним використанням.

1.2.3 BoofCV

BoofCV - це бібліотека комп'ютерного зору з відкритим вихідним кодом, написана на чистій Java [9]. Алгоритми, що використовуються всередині, добре оптимізовані і, як показує практика, за швидкістю у деяких випадках не поступаються реалізації на C++ в OpenCV (рис.1.6). Основні можливості бібліотеки включають:

– робота з відео та веб-камерами;

– 3D комп'ютерний зір;

– фільтри (розмиття, градієнт), шумозниження (за допомогою хвильових перетворень);

– бінаризація, морфологічні операції;

– виділення контурів (Кенні, Собель);

– пошук точок інтересу;

– пошук ліній, сегментів, прямокутників;

– стереозображення.

Рисунок 1.6 – Приклад виявлення мікро QR-коду за допомогою BoofCV

Крім цього, бібліотека містить багато інших функцій, що використовуються в комп'ютерному зорі. Підключити бібліотеку дуже просто - можна завантажити вихідний код з офіційного веб-сайту та зібрати його самостійно, або можна завантажити вже зібрані файли jar і просто підключити їх до проекту.

1.3 Формування вимог до апаратно-програмного забезпечення

1.3.1 Вимоги до апаратного забезпечення

Висока продуктивність: Апаратне забезпечення повинне мати достатню обчислювальну потужність для виконання різноманітних завдань обробки зображень та аналізу даних у реальному часі. Це включає в себе швидку

обробку великих обсягів даних та реагування на зміни у дорожньому середовищі без затримок [28].

Підтримка відповідних комунікаційних протоколів: Апаратне забезпечення повинне підтримувати необхідні комунікаційні протоколи для взаємодії з іншими пристроями та системами дорожнього контролю. Це може включати в себе підтримку бездротових технологій зв'язку, таких як Wi-Fi або Bluetooth, а також протоколи передачі даних, такі як TCP/IP.

Наявність вбудованих сенсорів: Апаратне забезпечення повинне мати вбудовані сенсори, які дозволяють зчитувати інформацію про навколишнє середовище, таку як камери, гіроскопи, акселерометри та GPS-модулі. Це дозволяє системі отримувати необхідні дані для розпізнавання дорожніх знаків та виявлення небезпечних ситуацій на дорозі.

Низьке споживання енергії: Апаратне забезпечення повинне мати оптимальне споживання енергії, щоб забезпечити тривалу автономну роботу в умовах, де доступ до електромережі обмежений. Це особливо важливо для мобільних пристроїв, таких як розумні камери на дорожніх знаках або датчики на дорожніх світлофорах.

Надійність та стабільність: Апаратне забезпечення повинне бути надійним та стабільним у роботі, навіть в екстремальних погодних умовах та умовах високого навантаження. Це забезпечить безперебійну роботу системи контролю дорожнього руху і збільшить її ефективність.

Масштабованість: Апаратне забезпечення повинне мати можливість масштабуватися з ростом потреб системи. Це дозволить легко розширювати функціональність системи та впроваджувати нові технології та можливості в майбутньому.

Врахування цих вимог під час вибору апаратного забезпечення дозволить побудувати ефективну та надійну систему контролю дорожнього руху, яка забезпечить безпеку та комфорт учасників дорожнього руху.

1.3.2 Вимоги до програмного забезпечення

Розпізнавання дорожніх об’єктів: Програмне забезпечення повинно забезпечувати надійне та швидке розпізнавання дорожніх знаків з використанням алгоритмів комп'ютерного зору та штучного інтелекту. Це включає в себе розпізнавання різноманітних типів знаків, включаючи знаки обмеження швидкості, знаки заборони, знаки попередження та інші [29].

Аналіз та інтерпретація інформації: Програмне забезпечення повинно аналізувати та інтерпретувати інформацію, отриману з дорожніх знаків, для прийняття відповідних рішень щодо керування дорожнім рухом. Це може включати в себе виявлення порушень правил дорожнього руху, виявлення небезпечних ситуацій на дорозі та вчасне оповіщення водіїв.

Інтеграція з іншими системами: Програмне забезпечення повинно бути здатним інтегруватися з іншими системами дорожнього контролю, такими як системи світлофорів, системи моніторингу транспорту та інші, для забезпечення координації та ефективного управління дорожнім рухом.

Надійність та стабільність: Програмне забезпечення повинно бути надійним та стабільним, щоб забезпечити безперебійну роботу системи контролю дорожнього руху навіть у складних умовах експлуатації та змінних погодних умовах.

Зручний інтерфейс користувача: Програмне забезпечення повинно мати зручний та інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко налаштовувати параметри системи, відслідковувати роботу та взаємодіяти з користувачами.

Ці вимоги формують основну основу для розробки програмного забезпечення для системи контролю дорожнього руху. Врахування цих вимог у процесі проектування та реалізації програмного забезпечення дозволить забезпечити ефективну та надійну роботу системи у реальних умовах експлуатації.

Висновки до розділу 1

Аналіз існуючих рішень систем контролю дорожнього руху на базі доповненої реальності з використанням комп’ютерного зору дозволив визначити ключові аспекти, тенденції та виклики у цій області. Поглиблений огляд літературних джерел та проектів показав, що застосування технологій доповненої реальності та комп'ютерного зору в системах контролю дорожнього руху є перспективним напрямом, спрямованим на покращення безпеки та ефективності управління транспортними потоками.

Отже, на основі аналітичного огляду можна визначити, що розробка та вдосконалення системи контролю дорожнього руху на базі доповненої реальності є перспективною задачею, вирішення якої вимагатиме комплексного підходу до визначення вимог та вибору оптимальних технологічних рішень.

2 МАТЕМАТИЧНІ МЕТОДИ

2.1 Система масового обслуговування

Вивчення теорії масового обслуговування спрямоване на аналіз статистичних закономірностей в операціях, які складаються з великої кількості однотипних елементарних операцій. Ці операції можуть включати складання деталей на конвеєрі, видачу інструментів, ремонт обладнання, обслуговування клієнтів у магазинах, роботу білетних кас, а також обслуговування транспорту та інше.

Теорія обслуговування, також відома як теорія черг, досліджує системи масового обслуговування, де заявки на елементарні операції надходять у випадковий час або обслуговуються протягом випадкових проміжків часу. У таких системах поява черг є неминучою наслідком. Залежно від кількості каналів обслуговування система може зазнавати збитків через можливі простої каналів або накопичення черг (рис.2.1).

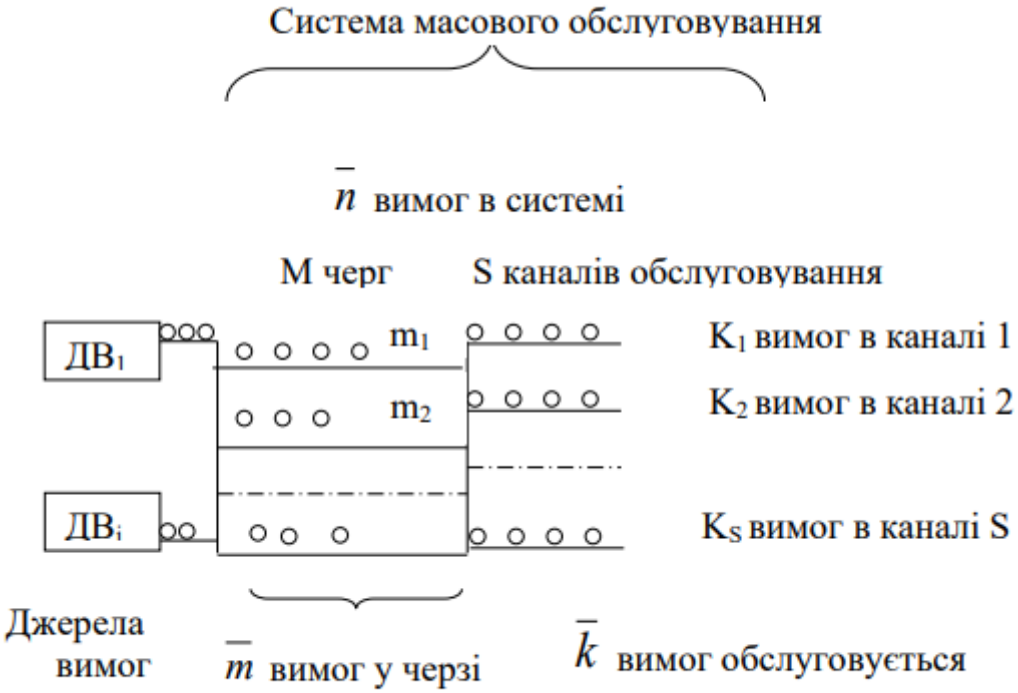


Рисунок 2.1 – Структура системи масового обслуговуваня

Основним завданням теорії масового обслуговування є вивчення статистичних закономірностей вхідного потоку заявок та тривалості обслуговування цих заявок, а також оцінка якості обслуговування за різних умов формування черг. Ці черги можуть мати різні характеристики, включаючи обмежену або необмежену довжину, обмежений час очікування тощо.

У теорії масового обслуговування розробляються математичні моделі, які пов'язують умови роботи систем обслуговування з їхньою ефективністю, що визначається їхньою здатністю впоратися з потоком заявок. Основні показники ефективності систем включають середню кількість заявок, що обслуговуються за одиницю часу, середню кількість заявок у черзі, середній час очікування на обслуговування, ймовірність відмови в обслуговуванні та інші [31].

Інтенсивність потоку () визначається як частота появи подій або середнє число подій, що надходять в систему масового обслуговування за одиницю часу.

У якості показників ефективності систем масового обслуговування можна використовувати:

– середнє число заявок, які обслуговуються за одиницю часу;

– середня кількість заявок у черзі;

– середній час очікування на обслуговування;

– ймовірність відмови в обслуговуванні без очікування;

– ймовірність того, що число заявок в черзі перевищить певне значення і таке інше.

Системи масового обслуговування можна розділити на два основних типи:

– системи з очікуванням (чергою), де заявка, що надійшла в момент зайнятості каналів, не відправляється, а стає в чергу на обслуговування;

– системи з відмовленням, де заявка, що надходить в момент, коли всі канали зайняті, отримує відмову та покидає систему, не приймаючи участі в подальшому процесі обслуговування (наприклад, заявка на телефонну розмову в момент, коли всі канали зайняті, отримує відмову і залишає систему не обслуженою).

2.2 Баєсовська модель

2.2.1 Теоретичні відомості

Баєсовська модель - це статистична модель, яка використовує теорему Баєса для оцінки ймовірностей параметрів моделі на основі даних. Вона включає в себе баєсівський підхід до статистичного виведення, в якому апріорні знання про параметри моделі комбінуються з даними, щоб отримати апостеріорні розподіли параметрів.

Основні компоненти Баєсовської моделі включають:

Апріорний розподіл: Це ймовірнісний розподіл параметрів моделі до отримання будь-яких даних. Він відображає наше початкове переконання про значення параметрів перед тим, як ми побачимо фактичні дані.

Функція правдоподібності: Це ймовірнісна функція, що визначає, як ймовірно дані при певних значеннях параметрів моделі. Вона відображає те, наскільки добре модель пояснює фактичні дані при різних значеннях параметрів.

Апостеріорний розподіл: Це оновлений ймовірнісний розподіл параметрів після врахування фактичних даних. Він визначає наші оновлені переконання про значення параметрів, враховуючи спостереження.

Байєсівське оновлення: Це процес оновлення апріорного розподілу параметрів на основі функції правдоподібності та фактичних даних, щоб отримати апостеріорний розподіл. Цей процес використовує теорему Баєса.

Баєсовські моделі є потужним інструментом для оцінки невизначеності в моделях та прийняття рішень на основі відомих даних та апріорних знань.

Вони застосовуються в різних галузях, включаючи машинне навчання, статистику, біоінформатику, фінанси та інші (рис.2.2, рис.2.3). Баєсовські методи [32] дозволяють уникнути перенавчання, роблять моделі більш стійкими до обмежених даних та надають механізм для врахування експертних знань.

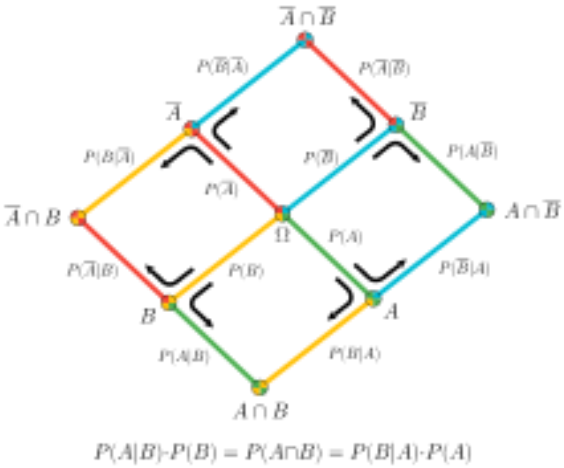


Рисунок 2.2 – Візуалізація теореми Баєса суперпозицією двох дерев ухвалення рішень

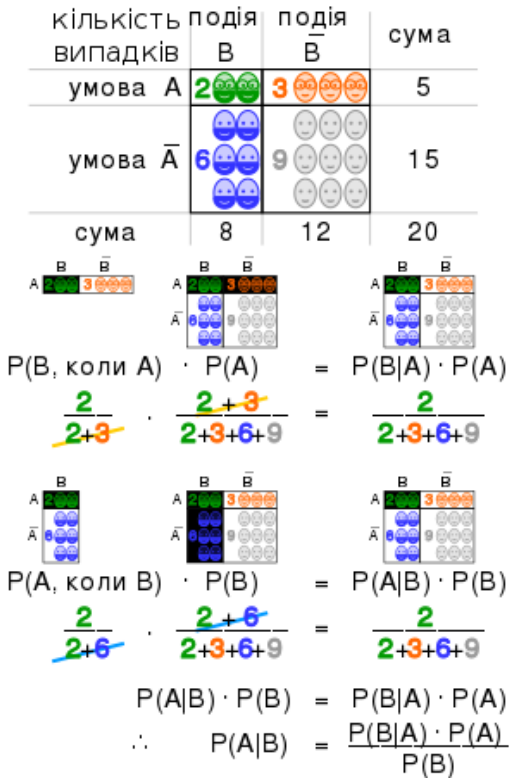


Рисунок 2.3 – Геометрична візуалізація теореми Баєса

Баєсовські методи можуть бути корисними для системи розпізнавання дорожніх об'єктів у наступних аспектах:

– обробка невизначеності: Системи розпізнавання дорожніх об'єктів часто стикаються з невизначеністю, особливо в умовах обмежених або шумних даних. Баєсовські методи дозволяють врахувати цю невизначеність, представляючи результати розпізнавання як ймовірнісні розподіли, а не одиночні точкові оцінки;

– інтеграція апріорних знань: Баєсовські моделі дозволяють інтегрувати апріорні знання або очікування про вигляд дорожніх об'єктів у процес розпізнавання. Це може покращити ефективність системи, особливо у випадках обмежених тренувальних даних;

– управління невизначеністю в середовищі:Деякі баєсівські методи, такі як фільтр Калмана та фільтр частинок, дозволяють ефективно враховувати невизначеність в середовищі та прогнозувати майбутні стани дорожніх об'єктів на основі спостережень;

– адаптивність до змінних умов: Баєсовські моделі можуть автоматично адаптуватися до змінних умов дорожнього середовища або змін у зовнішніх умовах, що може бути корисним для систем розпізнавання дорожніх об'єктів у реальному часі.

Отже, застосування баєсівських методів у системах розпізнавання дорожніх об'єктів може сприяти покращенню точності, надійності та адаптивності таких систем у різних умовах дорожнього руху.

2.3 Модель Лагранжа

Модель Лагранжа - це математична модель, яка використовується для опису системи або фізичної ситуації за допомогою функцій, відомих як лагранжіани. Ця модель базується на принципі дії Лагранжа, який формулюється у вигляді інтегрального принципу мінімальної дії.

Основні поняття моделі Лагранжа:

– лагранжіан (або функція Лагранжа): Це функція, яка описує кінетичну та потенціальну енергію системи, а також інші параметри, що характеризують її поведінку. Лагранжіан зазвичай позначається як L і залежить від координат, швидкостей та, можливо, часу;

– принцип дії Лагранжа: Це принцип, який вимагає, щоб інтеграл від різниці між кінцевим та початковим станом системи був мінімальним. Це виражається у вигляді рівнянь Ейлера-Лагранжа, які випливають з принципу екстремальної дії;

– рівняння Ейлера-Лагранжа: Це диференціальні рівняння, які описують рух системи в термінах координат, швидкостей та часу. Ці рівняння випливають з принципу дії Лагранжа і можуть бути використані для знаходження шляху, по якому дія є екстремальною;

– похідні Лагранжа: Це похідні від лагранжіану по відношенню до координат та швидкостей. Вони використовуються для похідних Ейлера-Лагранжа та для визначення траєкторії руху системи.

Модель Лагранжа широко використовується в фізиці, механіці та інженерії для моделювання та аналізу руху систем. Вона дозволяє досліджувати різноманітні фізичні процеси та вивчати їх рівняння руху з використанням математичних методів [33].

Модель Лагранжа є математичним інструментом, який зазвичай використовується для аналізу руху системи, описаного у термінах координат, швидкостей та енергії. Хоча ця модель зазвичай застосовується у фізиці та механіці, вона може бути корисною і для систем розпізнавання дорожніх об'єктів в наступних аспектах:

– аналіз руху об'єктів: Модель Лагранжа може бути використана для аналізу руху транспортних засобів та інших об'єктів на дорозі. Вона дозволяє враховувати різні фізичні параметри, такі як швидкість, прискорення та енергія, що може бути корисним для вивчення та прогнозування поведінки транспортних засобів на дорозі;

– оптимізація руху: Застосування моделі Лагранжа може допомогти в оптимізації руху на дорозі, враховуючи різні фактори, такі як швидкість руху, розмір транспортних засобів та розташування перешкод. Це може бути корисно для розробки систем управління трафіком та безпеки на дорозі;

– прогнозування руху: Модель Лагранжа може допомогти в прогнозуванні руху транспортних засобів та інших об'єктів на дорозі на основі їх поточного стану та умов дорожнього руху. Це може бути корисно для систем розпізнавання дорожніх об'єктів у визначенні їх майбутнього руху та взаємодії з іншими об'єктами на дорозі.

Отже, модель Лагранжа може бути корисною для систем розпізнавання дорожніх об'єктів у розумінні та аналізі руху транспортних засобів та інших об'єктів на дорозі з використанням математичних методів.

Висновки до розділу 2

У розділі 2 були розглянуті математичні методи, які можуть бути корисні для системи контролю дорожнього руху.

Загалом, використання цих математичних моделей в контексті системи контролю дорожнього руху на базі доповненої реальності з використанням OpenCV відкриває широкі можливості для покращення безпеки на дорозі, оптимізації транспортного потоку та покращення загального досвіду водіння. Інтеграція цих моделей з технологіями доповненої реальності та комп'ютерного зору може допомогти розробникам створити більш ефективні та інтелектуальні системи управління дорожнім рухом.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ АПАРАТНО-ПРОГРАМНОГО МОДУЛЮ

Ключовими етапами у процесі проєктування є розробка алгоритму функціонування програмного модуля та вибір відповідних технологій, які гармонійно впишуться у концепцію проєкту. Ці кроки визначають продуктивність програми та її загальну ефективність.

3.1 Обґрунтування вибору програмно-апаратних засобів для керування системою оповіщень

Розглядається вибір датчиків, камер та інших компонентів, які необхідні для забезпечення надійного функціонування системи. Аналізуються характеристики обраного обладнання та вивчаються можливості його інтеграції з технологією доповненої реальності та OpenCV [15]. Здійснюється вибір оптимальних параметрів та налаштувань для кожного апаратного компонента з метою забезпечення високої точності та швидкості обробки відеоданих.

Raspberry Pi 3 Модель B+ (RPI3-MODBP) – це повноцінний одноплатний комп'ютер розміром з банківську картку. Raspberry Pi 3 Модель B+ це покращена версія Raspberry Pi 3 Модель B. Модель базується на SoC (системі-на-чіпі) BCM2837B0, яка включає 4-ядерний ARMv8 64-бітний процесор з робочою частотою 1,4 ГГц і потужний відеопроцесор VideoCore IV.

Raspberry Pi 3 Модель B був першим Raspberry Pi, який мав функціонал бездротового та Bluetooth зв'язку. Raspberry Pi Модель B+ має такі ж можливості, але, на відміну від попередника, він має Bluetooth версії 4.2 (рис.3.1).

Рисунок 3.1 - Мікрокомп'ютер Raspberry Pi 3 Model B+

Raspberry Pi був розроблений як доступна платформа для експериментів і освіти в області комп'ютерного програмування. Raspberry Pi може використовуватися для багатьох речей, аналогічних тим, що робить звичайний настільний ПК, у тому числі обробки текстів, електронних таблиць, відео високої чіткості, ігор та програм. USB-пристрої, такі як клавіатури та миші можуть бути підключені через наявні на платі чотири порти USB.

Відмінною особливістю мікрокомп'ютерів Raspberry Pi є наявність роз'єму 40-pin GPIO для управління різними пристроями та його малий розмір, завдяки чому Raspberry Pi може також працювати як програмований контролер у найрізноманітніших додатках робототехніки та електроніки. Як варіант, він може бути об'єднаний з Pololu A-Star 32U4 Robot Controller LV із Raspberry Pi Bridge, для створення контролера для цього маленького робота.

Завдяки стандартному роз'єму 40-pin GPIO до мікрокомп'ютера можна приєднувати всілякі плати розширення для додавання необхідного функціоналу.

Raspberry Pi може працювати з широким спектром дистрибутивів ARM GNU/Linux, включаючи Ubuntu Snappy Core, Debian, Fedora та Arch Linux, а також Microsoft Windows 10 IoT Core.

Більшість моделей одноплатних комп'ютерів оснащені інтерфейсом введення-виведення загального призначення (GPIO). Цей інтерфейс дозволяє приєднувати периферійні пристрої до одноплатного комп'ютера. Контакти GPIO можуть використовуватися як входи чи виходи для передачі даних відповідно до встановлених параметрів. У більшості одноплатних комп'ютерів існують спеціалізовані виходи GPIO, які призначені для живлення і не можуть бути налаштовані для передачі даних (рис.3.2).

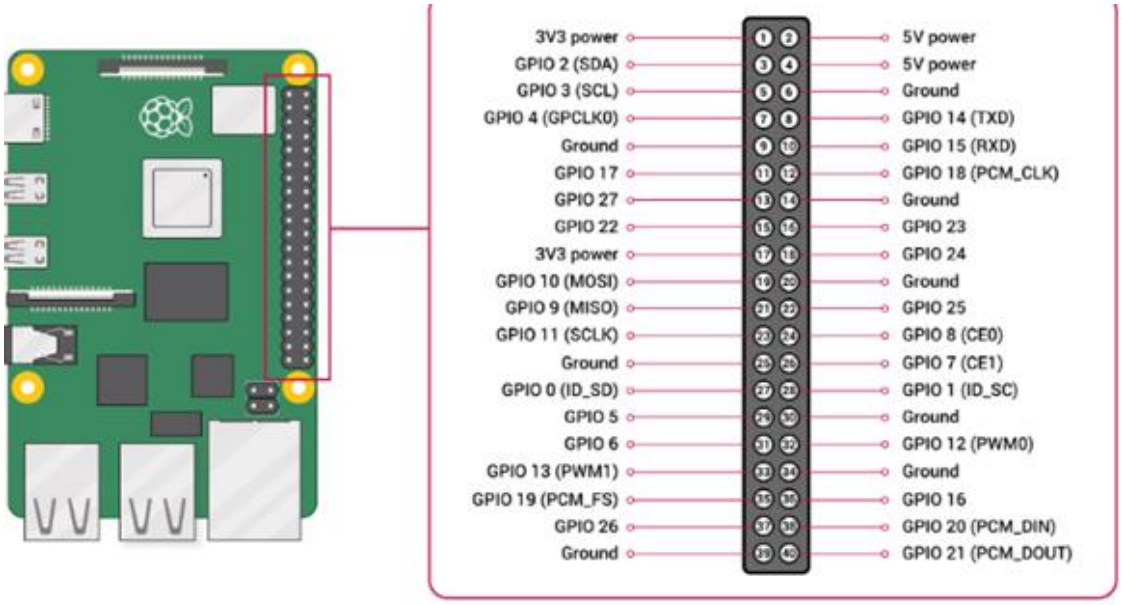


Рисунок 3.2 - Інтерфейс GPIO Raspberry Pi 3b

Виводи GPIO (введення-виведення загального призначення) Raspberry Pi 3 - це набір контактів на одноплатному комп'ютері Raspberry Pi, які можуть бути сконфігуровані як цифрові входи або виходи. Ці контакти дозволяють Raspberry Pi взаємодіяти зовнішнім світом, підключаючись до різноманітних сенсорів, пристроїв та інших електронних компонентів. Ось деяка інформація щодо GPIO на Raspberry Pi 3:

Raspberry Pi 3 має всього 40 контактів GPIO. Ці контакти розташовані на головці GPIO, яку зазвичай можна знайти вздовж краю плати Raspberry Pi [20].

Контакти GPIO можна налаштовувати як цифрові входи або виходи. У вигляді цифрових входів вони можуть зчитувати стан (високий чи низький рівень) зовнішніх пристроїв, таких як кнопки чи сенсори. У вигляді цифрових

виходів вони можуть висилати цифрові сигнали для керування світлодіодами, реле чи іншими електронними компонентами.

Контакти GPIO на Raspberry Pi 3 працюють при напрузі 3,3 вольта. Важливо враховувати цей рівень напруги при взаємодії зовнішніми компонентами, щоб уникнути пошкодження.

Деякі контакти GPIO мають конкретні функції, такі як надання апаратних сигналів PWM (широтно-імпульсна модуляція), інтерфейсу SPI (серійний периферійний інтерфейс).

Крім Raspberry Pi 3, існують інші версії цього одноплатного комп'ютера, такі як Raspberry Pi 4, який є більш потужним та має покращені можливості. Кожна нова версія часто привносить покращення в продуктивність, розширення портів та інші функції (рис.3.3).

Test	RPi 3B+	RPi 4B	Change
Boot Time	39.9	41.7	-4.5%
Idle Power (Amps)	0.505	0.684	-35.4%
Peak Power (Amps)	1.14	1.12	1.8%
CPU - sysbench primes - 1 thread	317.7	250.4	21.2%
CPU - sysbench primes - 4 threads	86.2	62.8	27.1%
RAM Bandwidth - mbw	1420	2983	110.1%
OpenGL - videogl32	30.9	35.8	15.9%
Ethernet - iperf3 (Mbps)	332	933	181.0%
WiFi 2.4Ghz - iperf3 (Mbps)	38.6	39.6	2.6%
WiFi 5Ghz (ac) - iperf3 (Mbps)	98.6	107	8.5%
External USB Drive Write - dd	35	155	342.9%
External USB Drive Read - dd	32	233	628.1%

Рисунок 3.3 – Порівняльна таблиця RPi 3B+ та RPi 4B

Крім GPIO, Raspberry Pi також має інші входи/виходи та роз'єми. Наприклад, HDMI-порт для підключення до монітора, порти USB для підключення пристроїв, Ethernet-порт для мережевого підключення, аудіороз'єми, кард-рідер для microSD-карт і т. д.

Raspberry Pi використовує операційну систему Raspbian (зазвичай засновану на Debian). Проте, користувачі можуть встановлювати інші операційні системи, такі як Ubuntu, Fedora, або навіть спеціалізовані операційні системи для проектів Інтернету речей (IoT).

Raspberry Pi користується великою спільнотою користувачів та розробників. Існують форуми, відеоуроки та різноманітні проекти, які можна використовувати для вивчення та розвитку навичок в області комп'ютерної науки та електроніки.

Raspberry Pi широко використовується для різних проектів, таких як домашні сервери, медіацентри, системи моніторингу, пристрої для навчання програмування та багато іншого. Його компактний розмір та доступність роблять його популярним серед хобістів та викладачів.

Вимагається потужне обчислювальне обладнання для виконання складних алгоритмів обробки зображень у реальному часі, забезпечуючи ефективну роботу системи.

3.2 Вибір камери

Raspberry Pi камера 8 МПкс (V2)

Raspberry Pi Camera Module v2 − це високоякісний 8-мегапіксельний датчик зображення Sony IMX219 з фіксованим фокусом, спроектований спеціально для плати Raspberry Pi. Датчик підтримує зображення з роздільною здатністю 3280×2464 пікселів, а також відео 1080p30, 720p60 та 640×480p60/90 (рис.3.4).

Рисунок 3.4 – Камера Raspberry Pi Camera Board

Він підключається до плати завдяки невеликим роз'ємам на верхній частині плати та використовує спеціальний CSi інтерфейс, спроектований

спеціально для роботи з камерами. Сам датчик невеликий, лише 25mm×23mm×9mm. Його вага складає близько 3 г, що ідеально підходить для мобільних або інших невеликих програм, де важлива вага та розмір. Для підключення до плати Raspberry Pi використовується короткий кабель.

3.3 Вибір дисплея

Система повинна бути обладнана якісним дисплеєм для відображення інформації доповненої реальності водію з належною яскравістю та чіткістю.

3.5" Дисплей сенсорний Waveshare для Raspberry Pi

Сенсорний резистивний дисплей Waveshare 3.5 дюймів. Сумісний з усіма моделями Raspberry Pi. Підтримує Raspbian. Призначений для перегляду фото, відео, буде дуже корисним у роботі з Raspberry Pi. Виконано дуже якісно від відомого виробника. У комплекті стілус і все необхідне кріплення (рис.3.5).

Рисунок 3.5 - Дисплей сенсорний Waveshare для Raspberry Pi

Повноцінна заміна стаціонарному монітору з клавіатурою та мишкою.

3.5 дюймовий резистивний сенсорний дисплей від WaveShare, який був розроблений для Raspberry PI. Дисплей підтримується на всіх платах

Raspberry, різних дистрибутивах: драйвери присутні для Raspbian, Ubuntu, Kali.

Модуль було створено як заміну стандартної периферії, а й у розробки промислових проектів, де резистивний сенсор може проявити себе повною мірою: реакція на дотик будь-якого предмета, велика точність і дешевизна. У сумі це ідеальний варіант для модуля управління промислового приладу тощо.

3.4 Програмне забезпечення

OpenCV (Open Source Computer Vision), спочатку розроблений компанією Intel, – це безкоштовна міжплатформна бібліотека комп’ютерного зору для обробки зображень у реальному часі. Програмне забезпечення OpenCV стало де-факто стандартним інструментом для всього, що стосується комп’ютерного зору [6]. Сьогодні OpenCV все ще дуже популярний, із понад 29 000 завантажень щотижня. OpenCV написаний на C і C++. Він працює під найпопулярнішими операційними системами, такими як GNU/Linux, OS X, Windows, Android, iOS тощо. Він доступний безкоштовно за ліцензією Apache Ведеться активна розробка інтерфейсів для Python, Ruby, Matlab та інших мов, що робить його легко доступним за допомогою таких команд, як «pip install opencv» для користувачів Python і «git opencv» для керування версіями. Бібліотека OpenCV містить понад 2500 алгоритмів, обширну документацію, вихідний код і зразки коду для комп’ютерного зору в реальному часі. Розробники, які використовують пакет Python разом з іншими бібліотеками Python, можуть легко інтегрувати OpenCV у свої проекти за допомогою таких команд, як «python opencv». Ця інтеграція стала ще зручнішою завдяки менеджерам пакетів, які забезпечують простий процес інсталяції та керування версіями (рис.3.6).

```cpp
#include "opencv2/objdetect.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/videoio.hpp"
#include <iostream>
using namespace std;
using namespace cv;
void detectAndDisplay( Mat frame );
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
int main( int argc, const char** argv )
{
    CommandLineParser parser(argc, argv,
                             "{help h||}"
                             "{face_cascade|data/haarcascades/haarcascade_frontalface_alt.xml|
                             "{eyes_cascade|data/haarcascades/haarcascade_eye_tree_eyeglasses.
                             "{camera|0|Camera device number.}");
    parser.about( "\nThis program demonstrates using the cv::CascadeClassifier class to detect
                  "You can use Haar or LBP features.\n\n" );
    parser.printMessage();
    String face_cascade_name = samples::findFile( parser.get<String>("face_cascade") );
    String eyes_cascade_name = samples::findFile( parser.get<String>("eyes_cascade") );
    //-- 1. Load the cascades
    if( !face_cascade.load( face_cascade_name ) )
    {
        cout << "--(!)Error loading face cascade\n";
        return -1;
    };
    if( !eyes_cascade.load( eyes_cascade_name ) )
    {
        cout << "--(!)Error loading eyes cascade\n";
        return -1;
    };
    int camera_device = parser.get<int>("camera");
    VideoCapture capture;
```

Рисунок 3.6 - OpenCV із демо-кодом детектора об’єктів C++ для виявлення облич

OpenCV було створено для максимальної ефективності та продуктивності завдань зору, що потребують інтенсивних обчислень. Тому він зосереджений на застосуванні AI бачення в реальному часі. Програмне забезпечення з відкритим вихідним кодом написано оптимізованою мовою C і може використовувати переваги багатоядерних процесорів (багатопотоковість) [7]. Мета OpenCV — створити просту у використанні інфраструктуру комп’ютерного бачення, яка допомагає людям швидко створювати складні програми бачення, надаючи понад 500 функцій, які охоплюють багато областей зору. OpenCV часто використовується для інспекції продукції заводу, медичної візуалізації, аналізу безпеки, інтерфейсу людина-машина, калібрування камери, стереобачення (3D-бачення) та роботизованого бачення. Широкі можливості обробки зображень підтримують обробку відеопотоку, об’єднання зображень (об’єднання кількох камер), калібрування камери та різноманітні завдання попередньої обробки зображень. Оскільки машинне навчання є важливим для комп’ютерного зору, OpenCV містить повну бібліотеку ML загального призначення, зосереджену на статистичному розпізнаванні образів і кластеризації. Щоб підвищити продуктивність, OpenCV підтримує прискорення NVIDIA CUDA та

графічного процесора з 2011 року, забезпечуючи функціональні можливості для обробки встановлених пакетів, пакетів сайту та двійкових файлів OpenCV, а також явний контроль над переміщенням даних між процесором і пам’яттю графічного процесора через модуль OpenCV GPU (рис.3.7).

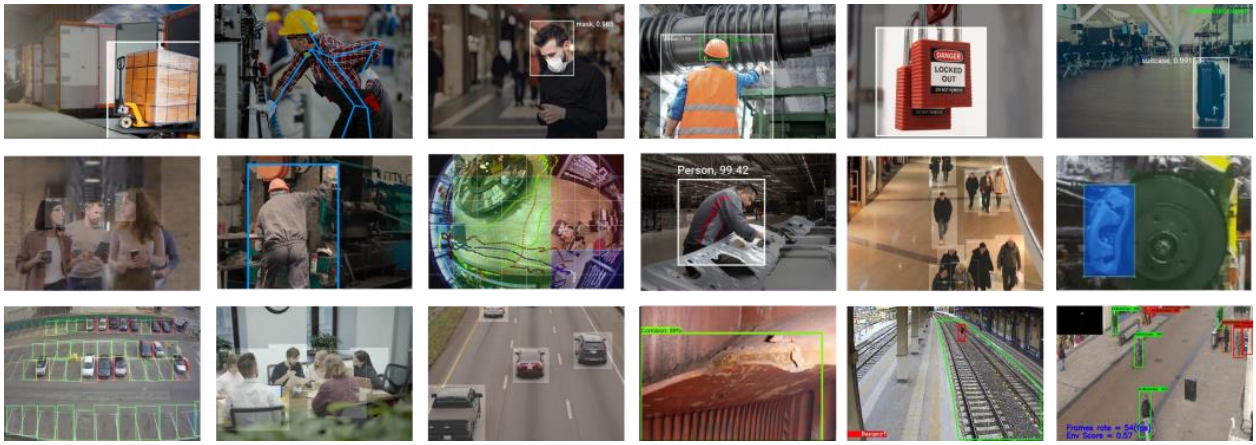


Рисунок 3.7 – Програми комп’ютерного зору, створені з використанням OpenCV і моделей глибокого навчання

Здатність змусити комп’ютери бачити за допомогою штучного інтелекту та сприймати фізичний світ за допомогою візуальних датчиків стає невід’ємною технологією для ефективної оцифровки та автоматизації операцій. В останні роки технології машинного навчання, особливо глибокого навчання, продемонстрували великий успіх у застосуванні комп’ютерного зору в різних галузях. Більшість додатків використовують штучний інтелект з IoT (AIoT), хмарні обчислення та Edge AI, щоб забезпечити та розгорнути комп’ютерне бачення будь-де та в масштабі. Використовуючи технологію зору штучного інтелекту, комп’ютери можуть розпізнавати обличчя, читати почерк, розпізнавати об’єкти, класифікувати людські рухи, виконувати автоматичний огляд або автоматично виявляти критичні ситуації за допомогою розпізнавання зображень (рис.3.8).

Рисунок 3.8 – Система комп’ютерного зору для виявлення транспортних засобів, зчитування номерних знаків і надання інформації про марку автомобіля, колір, тип і положення

OpenCV виступає як тренер, так і детектор. За допомогою OpenCV є можливість навчити класифікатор для будь-яких об’єктів, наприклад автомобілів, літаків і будівель. Існує два основні стани каскадного класифікатора зображень: перший – навчання, а інший – виявлення.

OpenCV надає дві програми для навчання каскадного класифікатора opencv_haartraining і opencv_traincascade. Ці дві програми зберігають класифікатор у різних форматах файлів.

Для навчання нам знадобиться набір зразків. Є два види зразків:

Негативний зразок: пов’язаний із необ’єктними зображеннями.

Позитивні зразки: це пов’язане зображення з об’єктами виявлення.

Набір негативних зразків необхідно підготувати вручну, тоді як колекція позитивних зразків створюється за допомогою утиліти opencv_createsamples (рис.3.9).

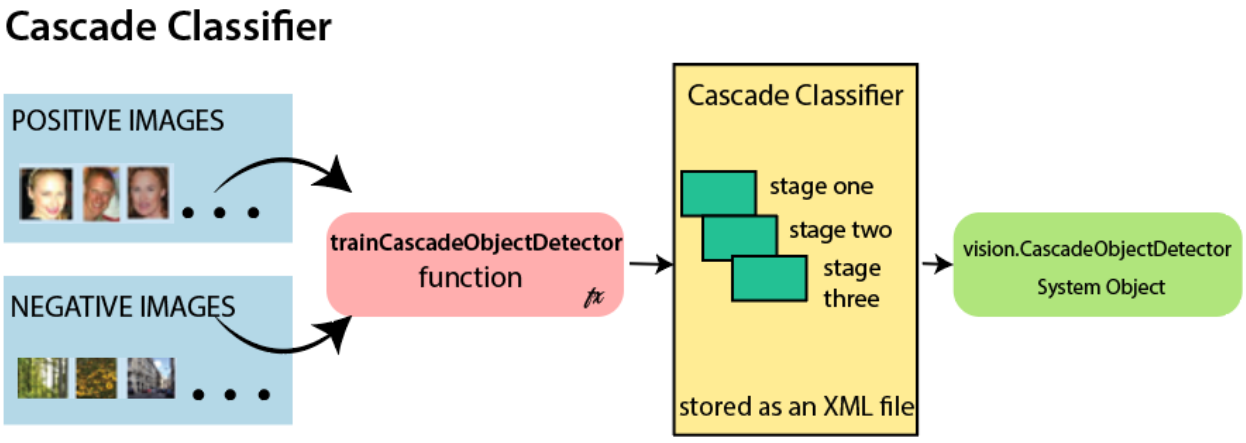


Рисунок 3.9 – Послідовність навчання розпізнавання об’єктів

Негативний зразок

Негативні зразки беруться з довільних зображень. Негативні проби додаються в текстовий файл. Кожен рядок файлу містить назву файлу зображення (відносно каталогу файлу опису) негативного зразка. Цей файл необхідно створити вручну. Визначені зображення можуть мати різні розміри.

Позитивна проба

Позитивні зразки створюються утилітою opencv_createsamples. Ці зразки можуть бути створені з одного зображення з об’єктом або з попередньої колекції. Важливо пам’ятати, що нам потрібен великий набір даних позитивних зразків, перш ніж ви надасте його згаданій утиліті, оскільки вона застосовує лише перспективне перетворення.

Доповнена реальність дозволяє системі розпізнавати різні об'єкти на дорозі, такі як транспортні засоби, дорожні знаки, пішоходи та інші елементи. Це відбувається за допомогою комп'ютерного зору та обробки відеоданих, що надає додаткові дані для аналізу та прийняття рішень системою контролю (рис.3.10).

Рисунок 3.10 - Концепт водіння за допомогою доповненої реальності

Віртуальна взаємодія та інтерфейс. Доповнена реальність створює можливість для віртуальної взаємодії між користувачем та системою. У водія або пішохода можуть відображатися важливі інформаційні елементи, такі як шляхи руху, швидкість, попередження про небезпеку тощо, що поліпшує його свідомість та безпеку [5].

Інтеграція з реальним оточенням. Доповнена реальність дозволяє системі інтегруватися з реальним оточенням. Наприклад, система може взаємодіяти з існуючими дорожніми інфраструктурними об'єктами, використовуючи їх в якості точок орієнтації та джерела інформації для коректної роботи.

Покращення сприйняття та розуміння водіїв та пішоходів. Доповнена реальність полегшує сприйняття та розуміння інформації про дорожнє середовище, роблячи його більш доступним та зрозумілим. Це сприяє покращенню реакції водіїв та пішоходів на різні ситуації на дорозі (рис.3.11).

Рисунок 3.11 – Приклад використання доповненої реальності в умовах руху на транспортному засобі

Доповнена реальність (ДР) визначається як ключовий елемент розроблюваної системи контролю дорожнього руху. Основні принципи та концепції доповненої реальності в контексті даного дослідження визначаються наступним чином:

Поєднання доповненої реальності та OpenCV може створити більш інтуїтивні та потужні системи контролю, здатні надавати додаткову інформацію для водіїв та підвищувати рівень безпеки на дорозі.

Аналіз існуючих підходів та технологій у галузі систем контролю дорожнього руху підтверджує актуальність та перспективність використання AR з інтеграцією OpenCV [3]. Подальша розробка та вдосконалення таких систем може призвести до покращення безпеки та ефективності дорожнього руху.

3.4.1 Методи проєктування програмного забезпечення

Розробка програмного забезпечення для програмно-апаратних комплексів представляє собою складний процес. Перед початком розробки

важливо детально продумати та описати структуру та алгоритм роботи цього програмного забезпечення.

Структуру програмного забезпечення можна ефективно описати за допомогою Unified Modeling Language (UML), що є графічним інструментарієм для моделювання програм. UML базується на парадигмі об'єктно-орієнтованого програмування та може використовуватися для схематичного опису бізнес-процесів, пов'язаних з програмним забезпеченням.

Мова UML включає різні типи моделей, такі як діаграми станів, діаграми класів, діаграми прецедентів та інші. Ці діаграми використовуються комплексно для забезпечення розробникам повного уявлення про архітектуру програмного забезпечення.

Діаграми UML використовуються комплексно для більш повного опису програмного забезпечення, і рекомендується використовувати ієрархію ПЗ, зокрема діаграму класів, яка є статичною структурною діаграмою. Вона відображає класи, методи, атрибути та їх взаємозв'язки. Варіації цієї моделі включають концептуальну діаграму класів (для предметної області), діаграму класів етапу специфікації ПЗ та діаграму класів етапу реалізації ПЗ, яка відображає елементи прямо в коді програмного забезпечення (рис.3.12).

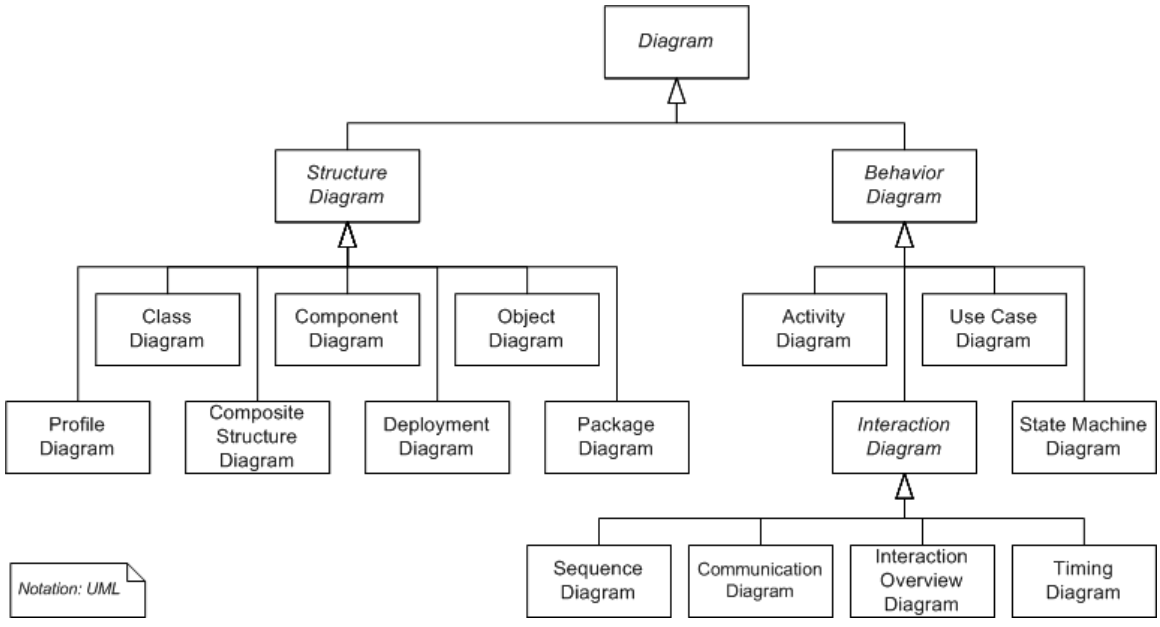


Рисунок 3.12 – Ієрархія діаграм UML

Описаний вище розділ типів діаграм класів не впливає на засоби побудови цих моделей. Програмне забезпечення для створення цих моделей входить до категорії CASE (Computer-Aided Software Engineering). Інструменти CASE не лише дозволяють будувати різноманітні моделі, але й частково генерують код на основі цих моделей. Широкий спектр програм, що входять до цього класу, включає різноманітні інструменти, але варто відзначити SoftwareIdeasModeler.

SoftwareIdeasModeler реалізує можливість побудови моделей UML, ERD-діаграм, блок-схем та інших [15]. Створені моделі можуть бути експортовані як графічні зображення для створення майбутньої нотації або у вигляді програмного коду для різних мов програмування (C++, C#, PHP, Python та ін.). Побудову та редагування моделей програмного забезпечення цей інструмент реалізує через графічний інтерфейс користувача (див. рисунок 3.13).

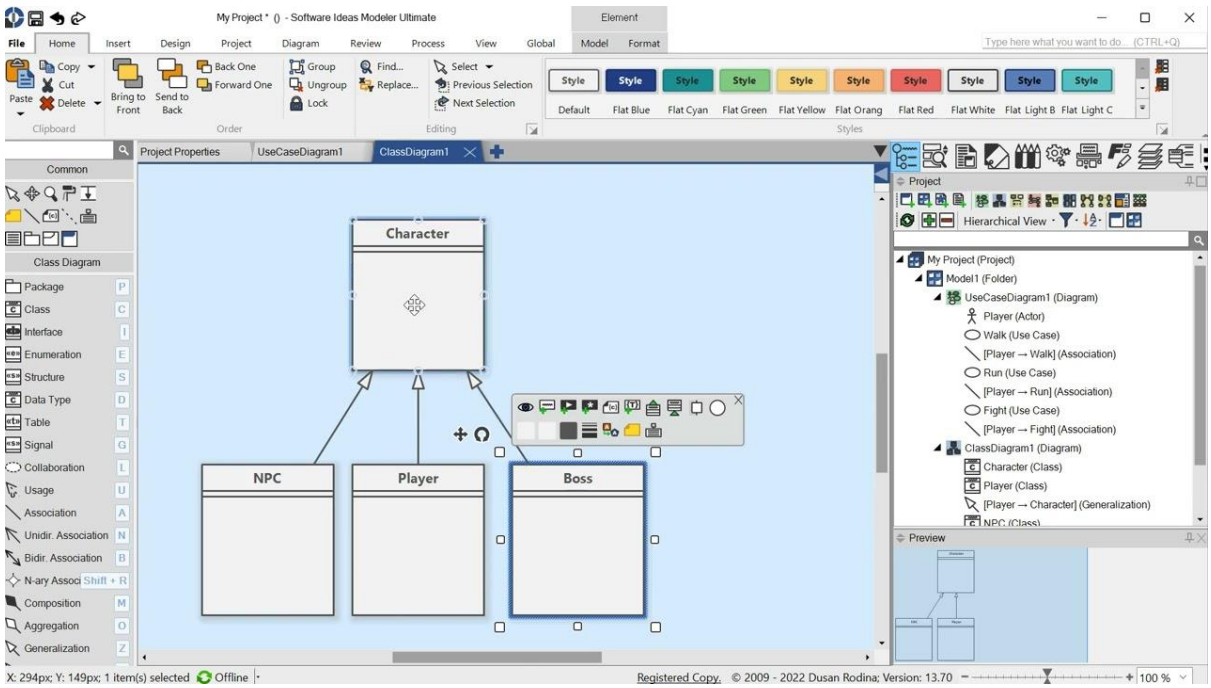


Рисунок 3.13 – Графічний інтерфейс користувача SoftwareIdeasModeler

Для забезпечення зберігання даних в програмно-апаратних комплексах використовується спеціалізоване сховище - база даних (БД). Вибір засобів проєктування БД залежить від її типу. Для проєктів з одноплатними

комп'ютерами найбільш поширеною є реляційна БД. Проєктування такої бази здійснюється на основі Entity-Relationship Diagram (ERD). Існують різні варіанти нотації для цієї моделі, але незалежно від типу, вона дозволяє відобразити взаємозв'язок між таблицями в БД (див. рисунок 3.14). Для побудови бази можна використовувати інструмент SoftwareIdeasModeler.

Також, невід'ємною частиною проєктування програмно-апаратного комплексу є опис алгоритму роботи конкретного пристрою. Це проєктування можна здійснити за допомогою блок-схеми, яка представляє собою відображення алгоритму розв'язання задачі за допомогою геометричних елементів, які вказують на операції, цикли, умови та інше.

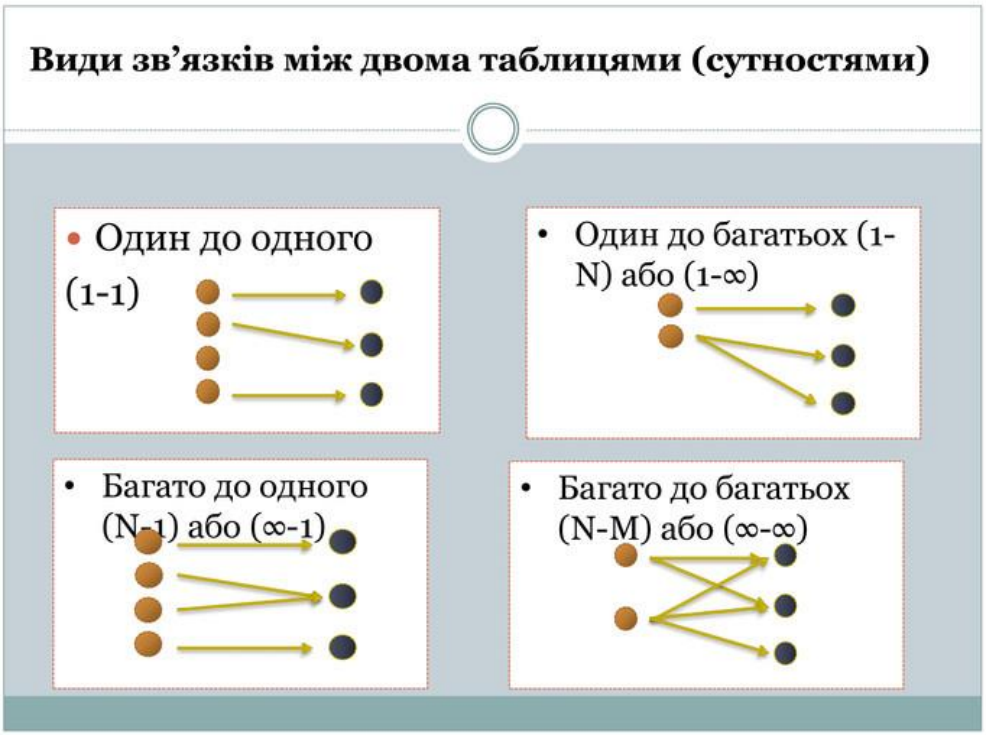


Рисунок 3.14 – Типи зв’язків діаграм и ERD

Створення графічного інтерфейсу користувача виконується за допомогою різних методів, таких як побудова мокапів, використання діаграм компонентів, створення карти сайту і т. д.

Висновки до розділу 3

У третьому розділі проведено аналіз інструментів для розробки програмно-апаратного комплексу розпізнавання та інтеграції мови в побутові пристрої. Надано докладний опис інтерфейсу для взаємодії з периферійним обладнанням GPIO.

Досліджено засоби моделювання апаратного забезпечення, включаючи використання принципових схем. Проаналізовано програмне забезпечення, яке реалізує можливості цього проектування.

В рамках розділу також подано опис засобів проєктування програмного забезпечення. Розглянуто засоби опису архітектури програмного забезпечення, а також проведено дослідження основних моделей мови UML та область їх використання при створенні програмного забезпечення.

Визначено особливості проєктування баз даних та розглянуто засоби їх реалізації. Окремо розглянуті можливості розпізнавання об'єктів за допомогою машинного зору, зокрема, використання бібліотеки OpenCV.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ СИСТЕМИ КОНТРОЛЮ ДОРОЖНЬОГО РУХУ НА БАЗІ ДОПОВНЕНОЇ РЕАЛЬНОСТІ З ВИКОРИСТАННЯМ OPENCV

4.1 Налаштування операційної системи Raspberry OS

Для того, щоб використовувати Raspberry Pi легко та зручно, йому потрібна операційна система [15].

Потрібно взяти SD-карту та підключити її до свого комп'ютера.

Потім перейти на офіційний сайт та завантажити Raspberry Pi Imager відповідно до своєї операційної системи, чи це буде Mac, Linux або Windows, як показано на рис 4.1.

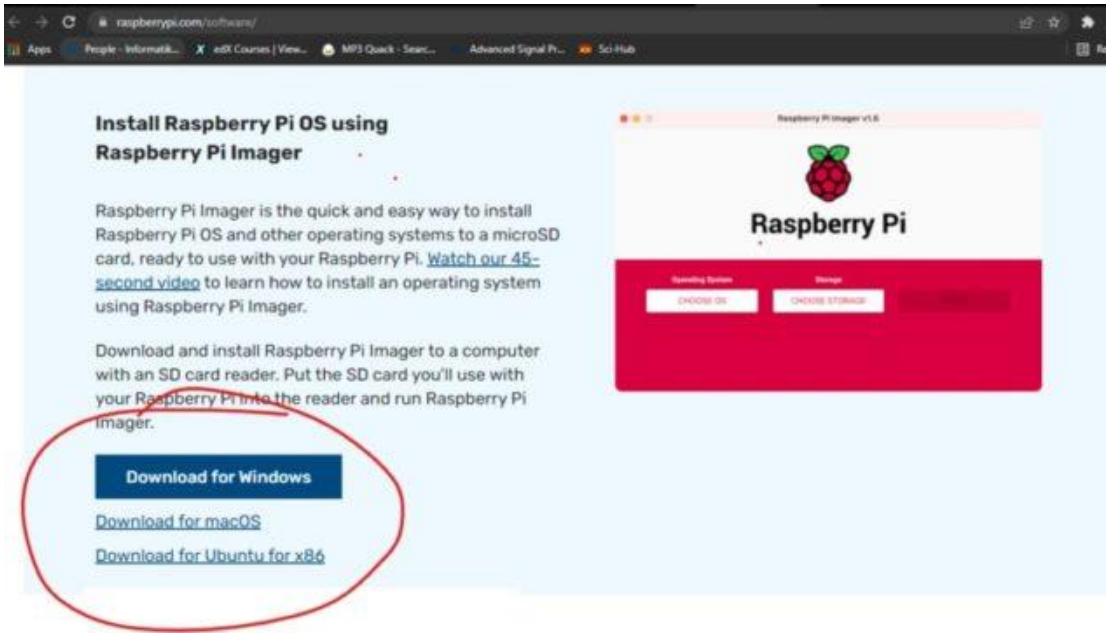


Рисунок 4.1 – Офіційна сторінка Raspberry Pi

Після завантаження необхідно встановити його повністю та відкрити (рис.4.2)

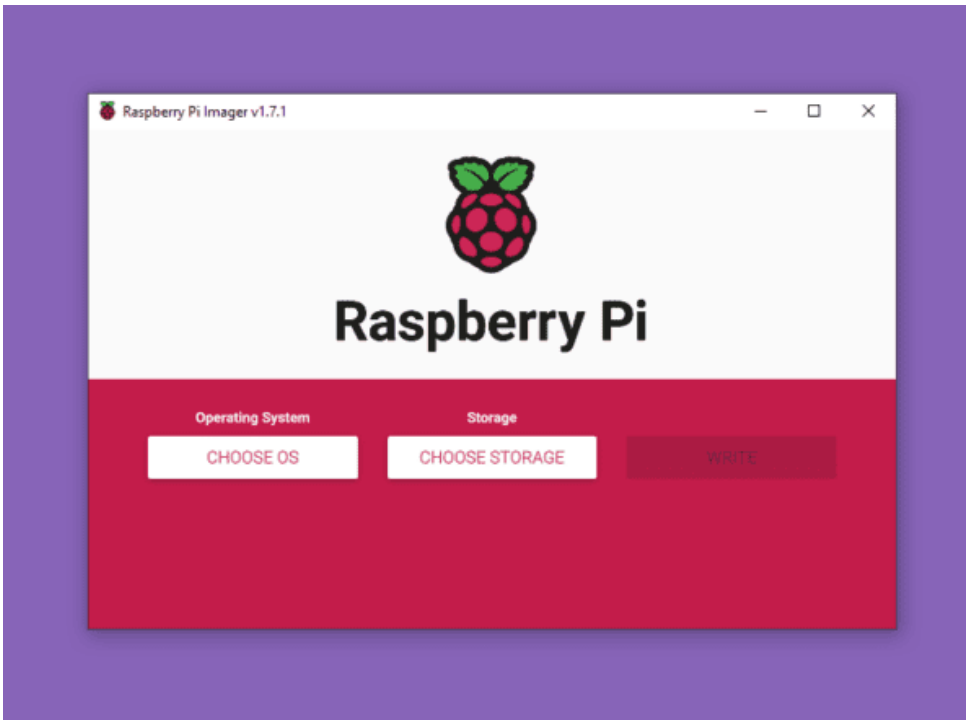


Рисунок 4.2 – Інтерфейс Raspberry Pi Imager

Потім треба натиснути «CHOOSE STORAGE» та обрати SD-карту, яка буде використовуватися для встановлення ОС, як показано на рис.4.3.

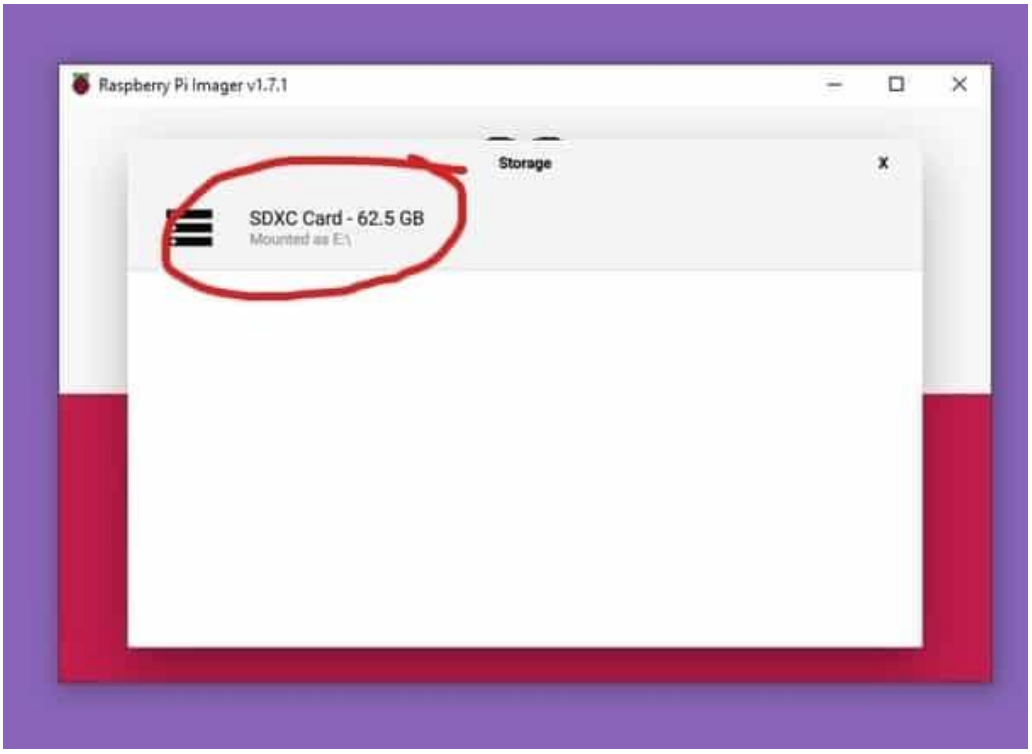


Рисунок 4.3 – Вибір SD-накопичувача

Тепер треба натиснути «CHOOSE OS » і обрати перший варіант, тобто; ОС Raspberry Pi, як на рис.4.4.

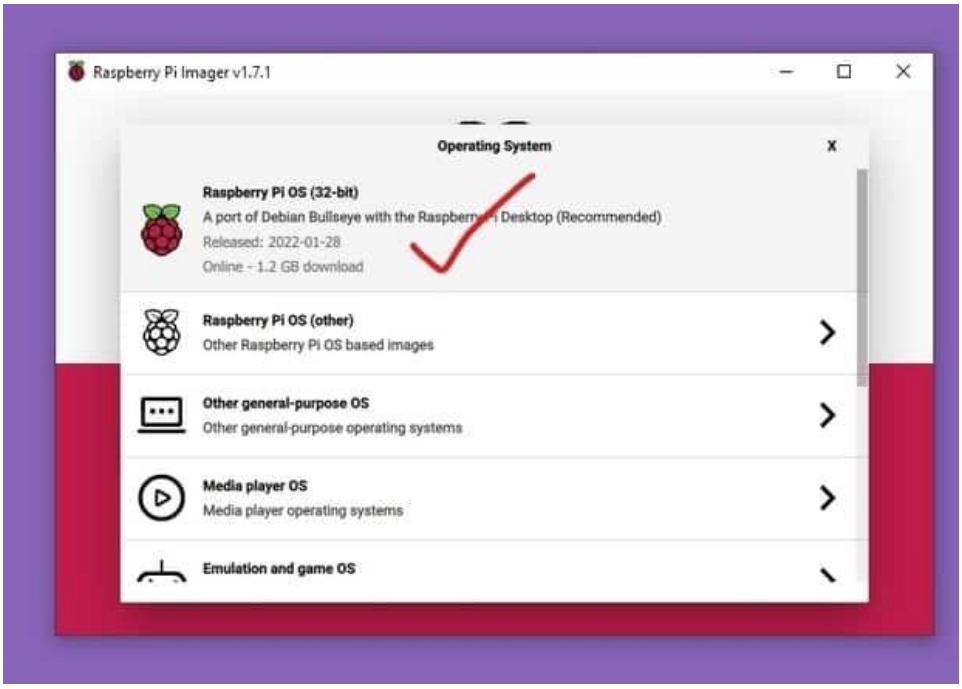


Рисунок 4.4 – Вибір ОС

Після вибору натиснути «WRITE», і розпочнеться створення образу.

Створення образу займає деякий час, після того як це буде зроблено, воно перевірить файли образу.

Після цього треба вийняти SD-карту з ПК.

4.2 Підключення Raspberry Pi

Налаштування підключення до Raspberry Pi.

Підключення клавіатури та миші, підключити обидва USB-роз’єми до відповідних USB-роз’ємів Raspberry Pi (рис.4.5).

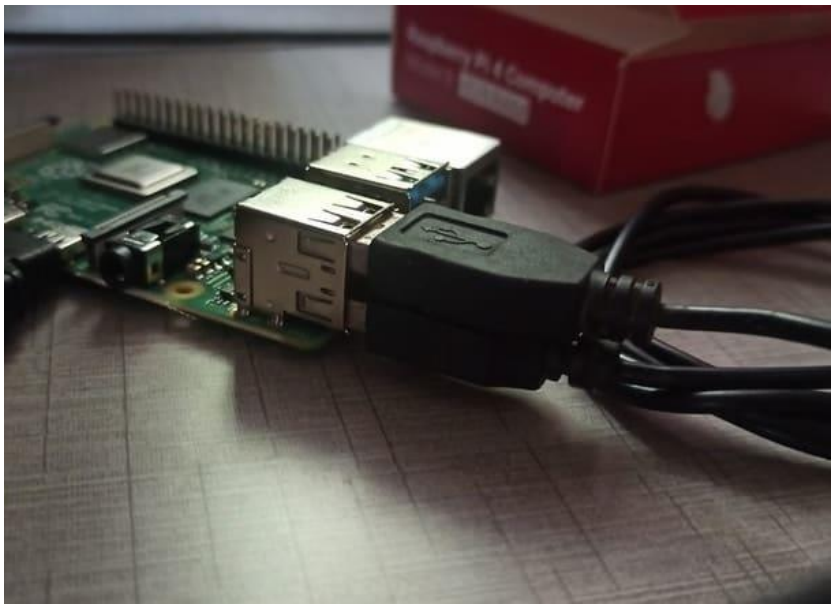


Рисунок 4.5 –Підключення клавіатури та миші

Підключення зовнішнього дисплею. На Raspberry Pi ми маємо HDMI порт. Треба підключити дріт безпечно, інакше порт може бути пошкоджений (рис.4.6).

Рисунок 4.6 – Підключення дисплею по HDMI

Тепер треба підключити SD-карту, для цього є один слот для SD-карти позаду плати Pi, потрібно вставити туди SD-карту (рис.4.7).

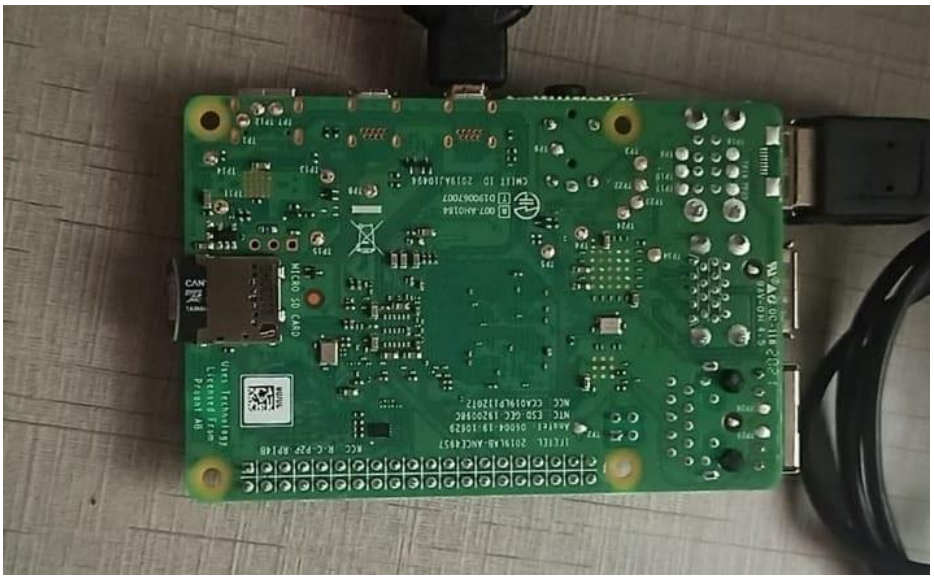


Рисунок 4.7 – Встановлення SD-карти

Потім увімкнути плату, підключивши плату до джерела живлення за допомогою роз’єму C-типу, який має вихід 5 В і 3 А. Після цього два бортових світлодіоди почнуть блимати (рис.4.8).

Рисунок 4.8 – Подача живлення на мікрокомп'ютер

Підключення камери. На Raspberry Pi ми маємо спеціальний слот для камери. Треба підключити шлейф камери безпечно, інакше слот може бути пошкоджений (рис.4.9).

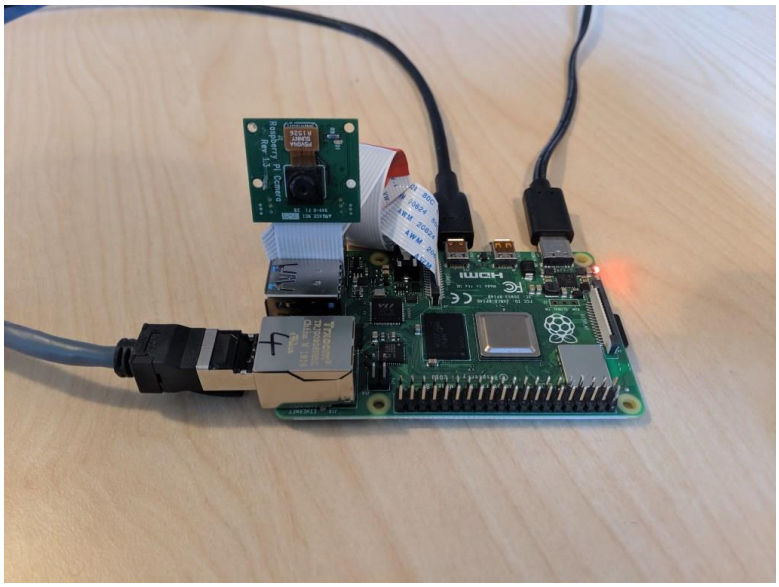


Рисунок 4.9 - Інсталяція камери

4.3 Встановлення та налаштування OpenCV на комп’ютері Raspberry Pi

Крок 1: Встановіити залежності

1. Оновлення існуючих пакетів :

Необхідно виконати таку команду, щоб оновити пакети вашої системи [16]:

```
sudo apt-get update && sudo apt-get upgrade
```

2. Встановити пакеи введення/виведення зображень :

Для підтримки різних форматів файлів зображень необхідно встанови пакети за допомогою:

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

3. Налаштування пакетів вводу/виводу відео :

Для обробки різних форматів відеофайлів і роботи з відеопотоками використоється наведені нижче команди:

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

```
sudo apt-get install libxvidcore-dev libx264-dev
```

4. Встановлення бібліотеки розробки GTK :

Щоб скомпілювати модуль highgui (використовується для відображення зображень і створення базових GUI), встановлюється бібліотека розробки GTK:

```
sudo apt-get install libgtk2.0-dev
```

5. Додаткові залежності для оптимізації OpenCV :

Для вдосконаленої оптимізації роботи OpenCV встановити ці додаткові залежності:

```
sudo apt-get install libatlas-base-dev gfortran
```

Крок 2: Встановлення pip (Інструмент керування пакетами)

Встановити pipPython 3, виконати наведену нижче команду:

```
sudo apt-get install python3-pip
```

Крок 3: Встановлення бібліотеки Numpy

Numpy надає основні математичні та числові можливості, корисні для OpenCV. Встановити його, скористатися командою:

```
pip install numpy
```

Крок 4: Доступ до OpenCV у сховищі Raspbian

Щоб знайти OpenCV у стандартному репозиторії Raspbian Buster, скористатися командою:

```
apt list python*opencv*
```

Крок 5: Встановлення OpenCV

Виконати наступну команду, щоб інсталювати OpenCV на Raspberry Pi.

```
sudo apt install python3-opencv
```

Крок 6: Перевірка встановлення OpenCV

Щоб підтвердити встановлення OpenCV, використовуйте:

```
apt show python3-opencv
```

Після виконання відобразиться на екрані, що останню версію успішно встановлено.

4.4 Налаштування системи розпізнавання дорожніх знаків з використанням OpenCV

4.4.1 Методологія

Вибрана база даних правильно сегрегована, передоброблена, а потім перейменована у відповідні теки.

Модель належним чином навчена за допомогою згорткової нейронної мережі (CNN), після чого відбувається класифікація [17].

Вхідне тестове зображення отримане, передоброблене, а потім перетворене у форму масиву для порівняння.

Відбувається порівняння тестового зображення та навченої моделі, за яким відображається результат (рис.4.10).

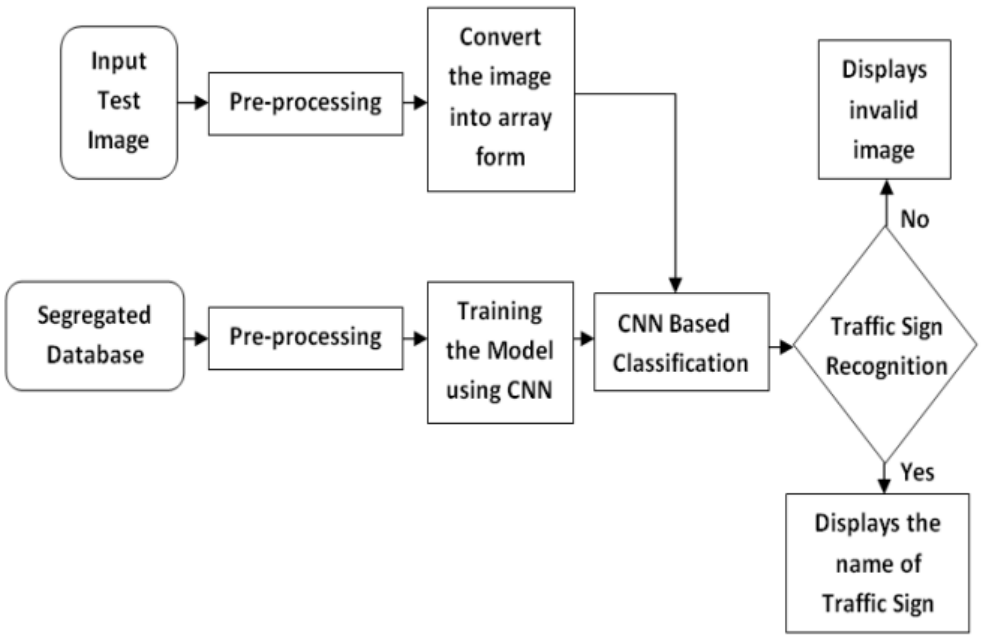


Рисунок 4.10 – Етапи методології

Набір даних зібраний з Kaggle під назвою Німецький Бенчмарк Розпізнавання Дорожніх Знаків (GTSRB) та Класифікація Дорожніх Знаків. З 43 класів набору даних, 7 класів, а саме Зупинка, Поворот ліворуч наперед, Поворот праворуч наперед, Заборонено поворот ліворуч, Заборонено поворот праворуч, Поворот ліворуч та Поворот праворуч. Загалом навчено 1 012

зображень, з яких 672 зображення належать до вищезгаданих 7 класів, а решта 340 зображень навчено випадковим чином (рис.4.11).

Рисунок 4.11 – Приклад набору даних

Цей набір даних розділений на різні теки, передоброблений та перейменований, так що кожна тека представляє певний клас дорожнього знаку. Усі зображення з цих тек мають мітки та надаються моделі CNN для навчання. Пізніше відбувається класифікація цих зображень.

Розроблено графічний інтерфейс користувача, за допомогою якого користувач може завантажити зображення з тестових даних. На це зображення застосовується передобробка, а потім подається до навченої моделі, де відбувається порівняння між тестовим зображенням та навченою моделлю. Після цього мітка завантаженого дорожнього знака відображається на графічному інтерфейсі користувача, а також надсилається як повідомлення-сповіщення на зареєстрований номер мобільного телефону за допомогою облачної служби зв'язку Twilio [18].

4.4.2 Системна архітектура

До зображень, взятих з набору даних, застосовується передобробка даних. Після цього до передоброблених зображень застосовуються відбір ознак та вилучення ознак. Потім класифікація виконується на основі результатів попередніх зображень (рис.4.12).

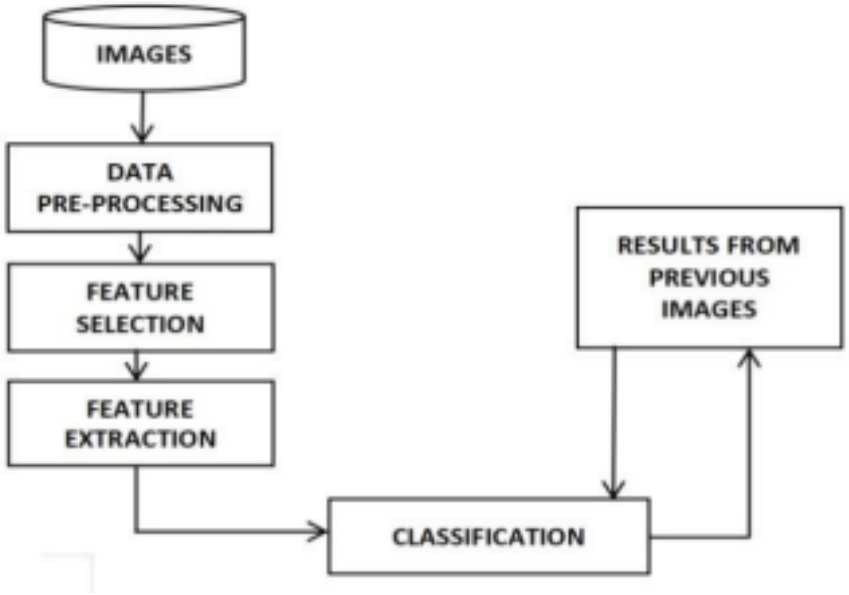


Рисунок 4.12 – Системна архітектура

4.4.3 Flow Diagram

Спочатку набір даних, який складається з дорожніх знаків, піддається передобробці. Ці попередньо оброблені дані використовуються для підготовки даних, де набір даних розбивається на навчальний і тестовий набори, і таким чином отримується навчальні дані. З використанням цих даних модель навчається, і підготовлена модель зберігається.

Коли користувач завантажує зображення, це зображення піддається передобробці, а потім подається до збереженої моделі. Після класифікації модель повертає мітку або клас, до якого належить зображення (рис.4.13).

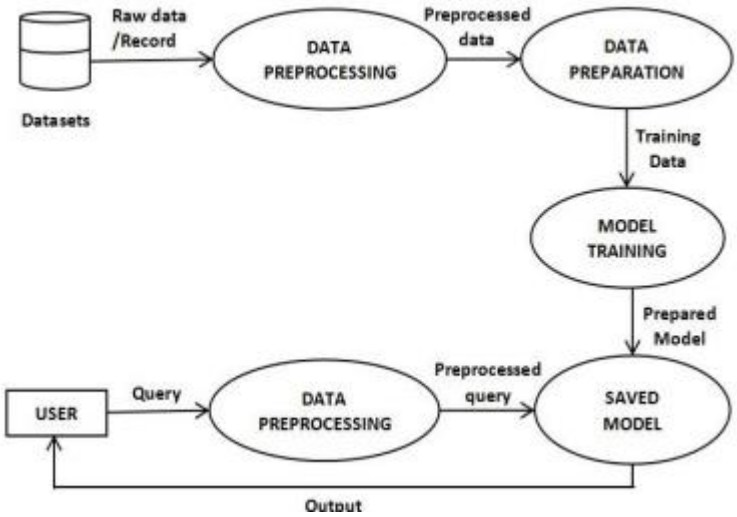


Рисунок 4.13 – Блок-схема

4.4.4 Фаза навчання та фаза класифікації

Фаза навчання:

а) отримання набору даних: Завантажується набір даних, що містить зображення дорожніх знаків;

б) передобробка даних:

1) зображення розділяються, перейменовуються на основі їхнього значення та зберігаються в каталозі для навчання (TRAIN_DIR);

2) кожне зображення з TRAIN_DIR призначається мітка класу на основі назви, що їй дана;

3) попередня обробка даних є обов'язковим завданням для очищення даних та підготовки їх для моделі машинного навчання, що також підвищує точність та ефективність. Основний обмежувач у CNN: для подачі набору даних зображень в згорткову нейронну мережу вони всі повинні бути одного розміру. Тому кожне зображення з TRAIN_DIR читається за допомогою функції cv2.imread() та змінюється до стандартного розміру 50x50 за допомогою функції cv2.resize();

4) кожне змінене зображення та його мітка класу перетворюються в масив і додаються до масиву training_data;

5) Цей масив перемішується і потім зберігається у формі файлу масиву NumPy.

в) Вибір ознак:

1) це процес зменшення кількості вхідних змінних при розробці прогностичної моделі. Він використовується для виявлення та видалення зайвих ознак, які не сприяють точності прогностичної моделі, тим самим прискорюючи процес навчання;

2) вибрані ознаки у запропонованій роботі - це колір, розмір, форма, текстура тощо.

г) класифікація: Це процес аналізу зображення та визначення класу або мітки, до якого воно належить. Класифікація зображень виконується за допомогою згорткової нейронної мережі (CNN), де класифікатор складається з повністю зв'язаних шарів (рис.4.14).

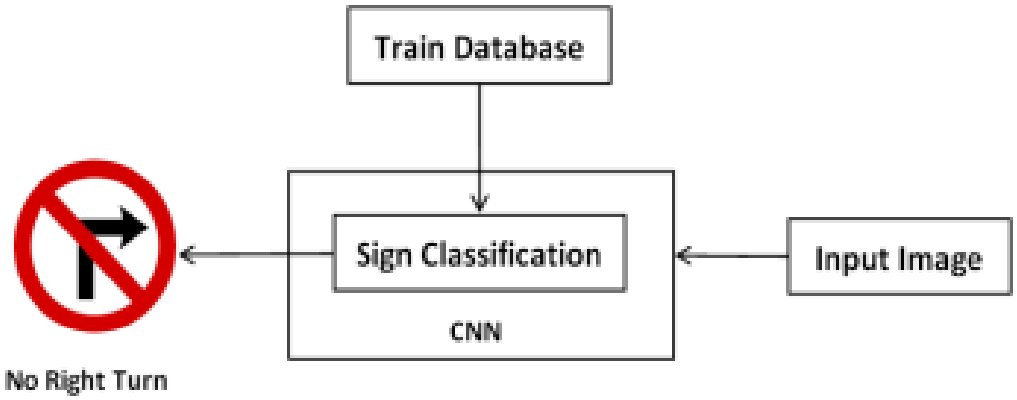


Рисунок 4.14 – Процес класифікації

д) Модель згорткової нейронної мережі (CNN): У запропонованій роботі використовуються три шари CNN: згортковий шар, шар пулінгу та повністю зв'язаний шар.

1) згортковий шар: Цей шар ефективно витягує різні ознаки з вхідних зображень. Математична операція згортки виконується між вхідним зображенням та фільтром/ядром розміром 3x3. Фільтр переміщується по вхідному зображенню, а потім отримується добуток точок між частинами

вхідного зображення та фільтром. Це призводить до отримання карти ознак, яка надає інформацію про зображення;

2) шар пулінгу: Цей шар зменшує розмір отриманої після згортки карти ознак, зменшуючи зв'язки між шарами. Максимальне значення з карти ознак вибирається за допомогою операції максимального пулінгу. Шар пулінгу діє як міст між згортковим і повністю зв'язаним шарам. Він допомагає зменшити обчислювальну вартість, кількість навчальних параметрів і тим самим контролює перенавчання.

3) fully-connected шар: Вихід з останнього шару максимального пулінгу передається на вхід цього шару. Тут отримані ознаки порівнюються з ознаками тестового зображення, і подібні ознаки асоціюються з певною міткою. Зазвичай мітки кодуються числами для зручності обчислень, і пізніше вони будуть перетворені в відповідні рядки;

4) шар Dropout: Коли нові дані надаються певній моделі, яка дуже добре працює на навчальних даних, відбувається перенавчання, що має негативний вплив на продуктивність цієї моделі. Цю проблему вирішується за допомогою шару випадкового відключення (Dropout). Під час процесу навчання кілька нейронів видаляються з моделі CNN;

5) функція активації: Вона обчислює зважену суму входів та зсувів, з якої можна прийняти рішення про те, чи можна активувати нейрон. У запропонованій роботі використовуються дві функції активації: ReLU (Rectified Linear Unit) і Softmax;

6) ReLU: Вона дозволяє лише позитивні значення і відкидає від'ємні значення. Вона перетворює всі від'ємні входи в нуль, тим самим запобігаючи активації нейронів. В результаті цього лише кілька нейронів буде активовано в певний момент часу;

7) Функція Softmax: Вона використовується при роботі з багатокласовими виходами. Для заданого входу вона відображає вихід на число від 0 до 1 таким чином, що їхня загальна сума дорівнює 1 (рис.4.15).

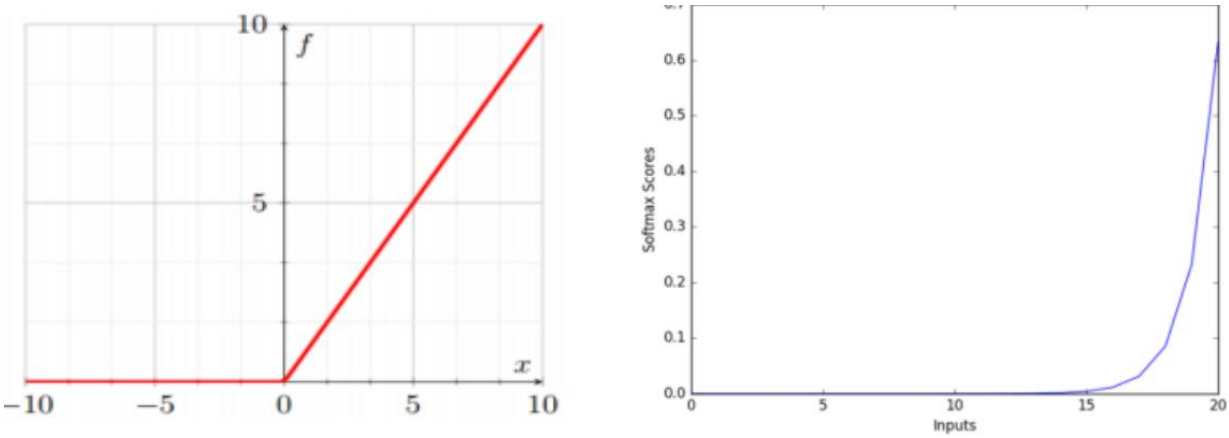


Рисунок 4.15 – Графік відображає функції активації ReLU і Softmax

У запропонованій роботі зображення розміром 50x50 з трьома каналами (RGB) та розміром партії None подаються на вхід функції input_data(), що призводить до 4D тензору [розмір партії, 50, 50, 3], який подається на вхід у CNN. Використовується шар згорткової обробки, який приймає на вхід вхідний 4D тензор та використовує тридцять два згорткові фільтри розміром 3x3, а також активацію ReLU. Він видає 4D тензор, який подається на вхід у шар максимального пулінгу, що використовує ядро згортки розміром 3x3. Наступні чотири шари є комбінацією шару згорткової обробки і шару максимального пулінгу, де шар згорткової обробки використовує шістдесят чотири, сто двадцять вісім, тридцять два і шістдесят чотири фільтри розміром 3x3 відповідно, а всі шари максимального пулінгу використовують ядро згортки розміром 3x3. Наступний шар - це повністю з'єднаний шар, який приймає вхідний 4D тензор, використовує активацію ReLU та видає 2D тензор, який подається на вхід у шар випадкового відключення (Dropout). У шарі випадкового відключення зберігаються 80% вузлів, а решта 20% вузлів випадково відключаються з CNN. 2D тензор подається на вхід у наступний повністю з'єднаний шар, який використовує активацію Softmax для класифікації зображень на 8 різних класів. Вихід останнього шару, який є 2D тензором, подається на вхід у шар регресії, який використовує функцію втрати категоріальної крос-ентропії для обчислення помилок та оптимізатор моменту адаптивного зміни (Adam) разом із своїм коефіцієнтом навчання для

мінімізації функції втрат. Ця модель CNN називається, завантажується, а потім зберігається [19].

Зі зібраного набору даних взято 793 зображення як навчальний набір та 199 зображень як тестовий набір. Модель CNN навчається з використанням навчального та тестового набору з епохою 90, а потім навчена модель зберігається.

4.4.5 Тестування програмно-апаратного комплексу

Мета навчальної програми полягає в тому, щоб створити послідовну модель, навчити модель, зберегти модель і виконати остаточний тест, використовуючи випадкове зображення з тестового набору. Програма також виводить графік, але це не впливає на процес навчання, а лише для уточнення процесу згодом для полегшення розуміння результату.

Навчання програми за допомогою терміналу в duster (ОС Linux) показує дані під час навчання, дані показані на рисунку 4.16 нижче.

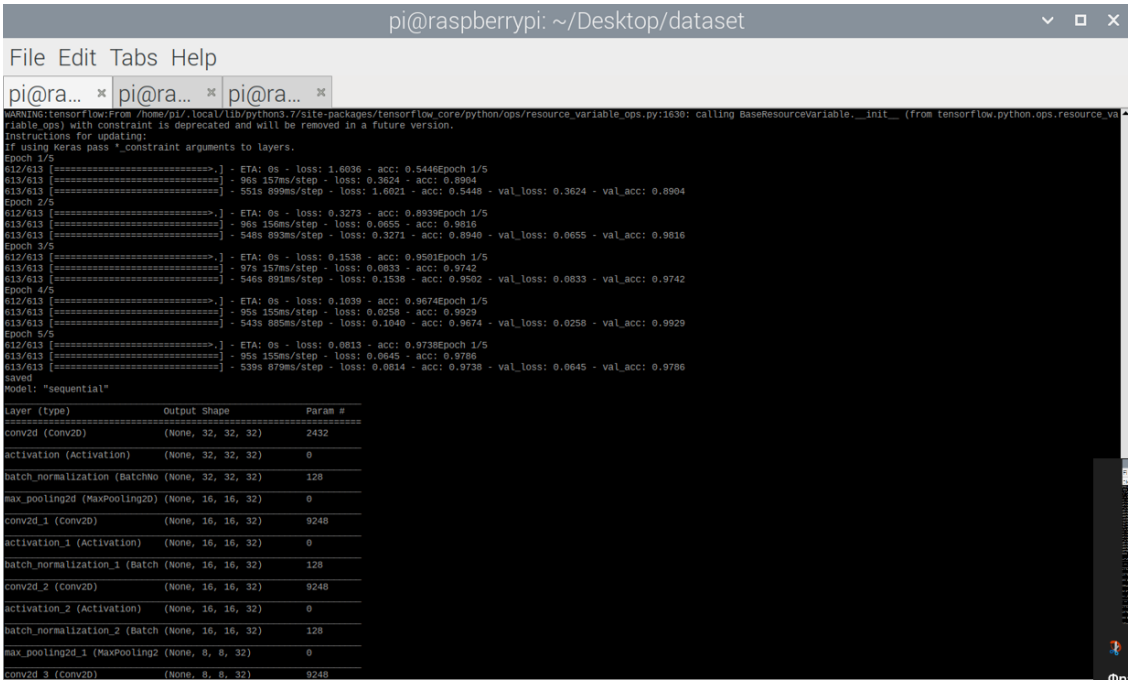


Рисунок 4.16 - Процес навчання

Це показує номер кроку навчання для цієї епохи, кожна епоха створює новий рядок даних. Наприкінці кроків відбувається перевірка, яка потім завершує епоху та переходить до нової.

Потім програма друкує короткий опис моделі (рис.4.17).

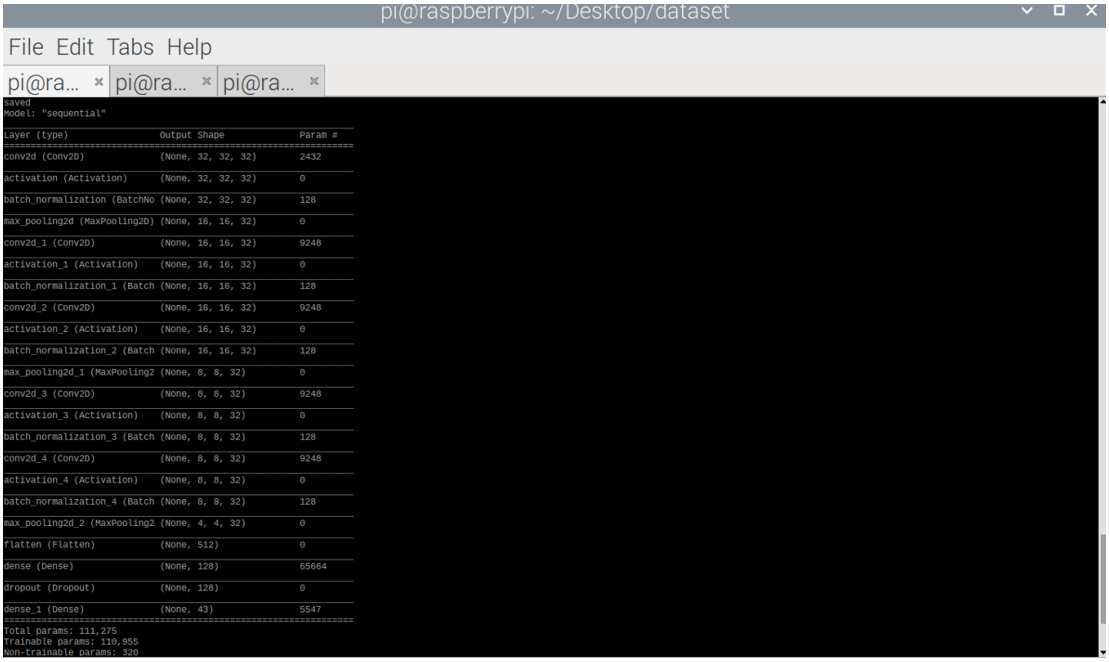


Рисунок 4.17 - Опис отриманої моделі

Рисунок 4.18 відображає вивід матриці з тестового зображення після того, як воно пройшло прогнозування моделі.

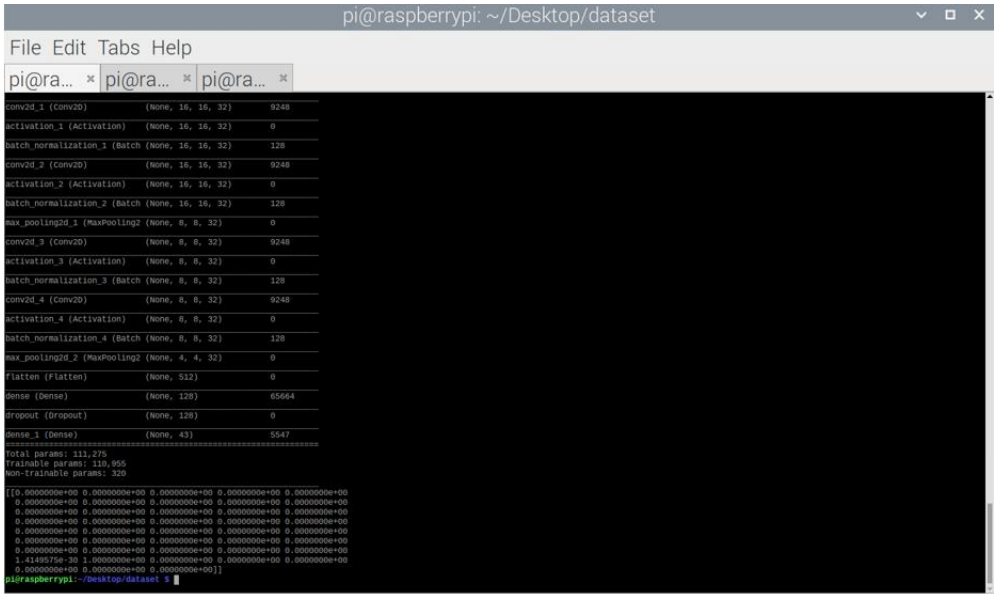


Рисунок 4.18 - Вивід матриці з тестового зображення

На зображенні 4.19 показано графік навчання моделі. Це показує, що протягом 5 епох точність зростає, а втрати зменшуються.

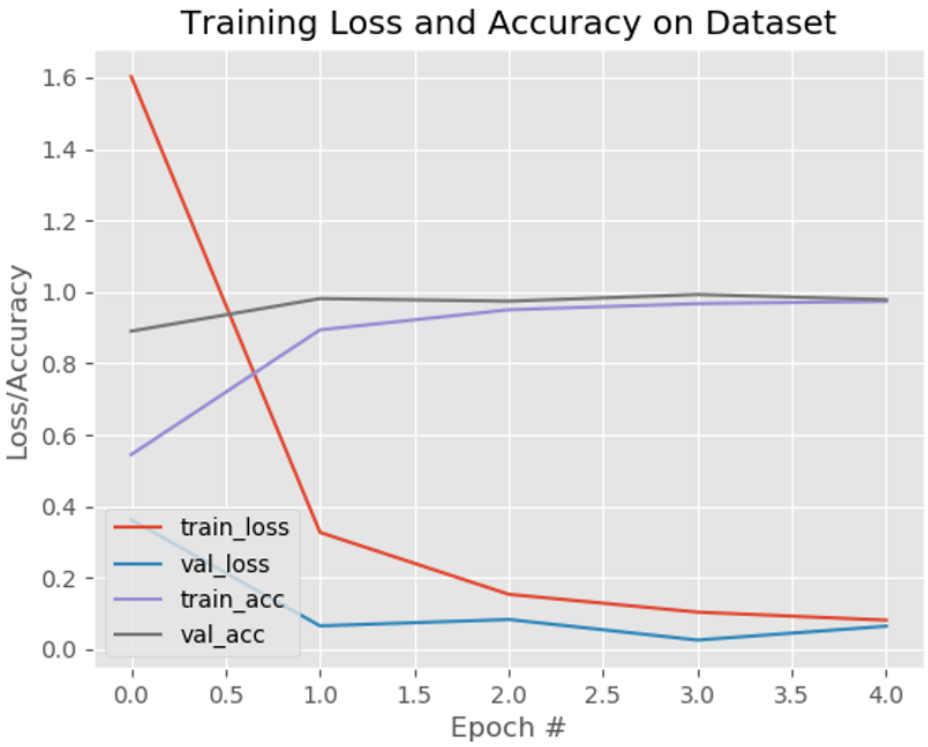


Рисунок 4.19 - Графік навчання моделі

Наступний експеримент показує, що розпізнавання працює, на рисунку 4.20 нижче, бібліотека opencv показує відеоканал у вікні та друкує прогнозований вивід у терміналі.

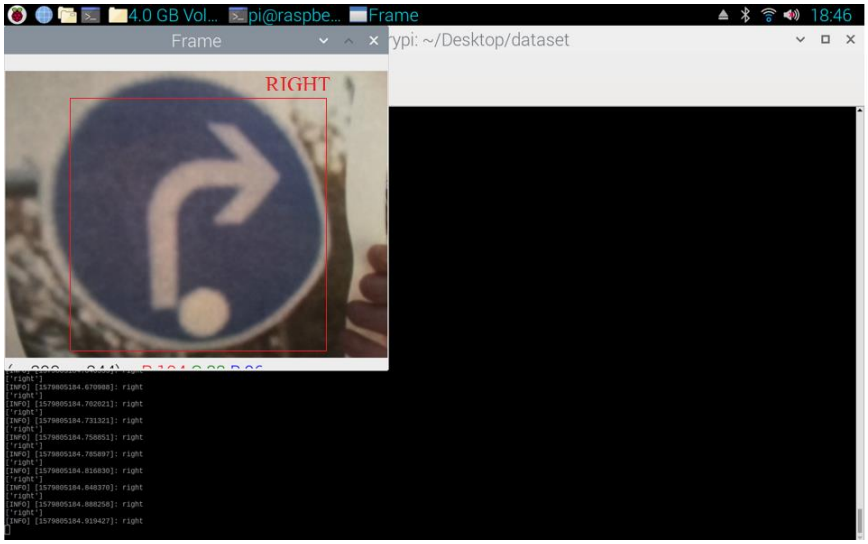


Рисунок 4.20 - Результат роботи комплексу

На рис.4.20 видно пряму трансляцію з Pi-камери, під час виконання скрипту Pi- камери. Для кожного кадру робиться прогнознозування, яке виводиться в термінал. Frame показує те, що «бачить» камера.

Висновки до розділу 4

У цьому розділі було проведено практичну реалізацію програмно-апаратного комплексу для системи контролю дорожнього руху на базі технології доповненої реальності з використанням OpenCV.

Цей комплекс охоплює налаштування операційної системи Raspberry Pi, підключення пристрою, встановлення та налаштування бібліотеки OpenCV, а також налаштування системи розпізнавання дорожніх знаків.

Результатом практичної реалізації став створений програмно-апаратний комплекс, який може ефективно використовуватися для контролю дорожнього руху. Його можна використовувати для розпізнавання дорожніх знаків у реальному часі та забезпечення відповідних заходів управління дорожнім рухом на їх основі.

Також цей комплекс відкриває нові можливості для покращення безпеки дорожнього руху та ефективного управління транспортним потоком. Його можна використовувати в різних областях, включаючи міську інфраструктуру, автодороги та інші місця з великим потоком транспорту.

Отже, практична реалізація цього програмно-апаратного комплексу відкриває нові перспективи для впровадження технологій доповненої реальності в системи контролю дорожнього руху та сприяє покращенню безпеки на дорогах.

ВИСНОВКИ

У даній роботі проведено аналіз та дослідження сучасних технологій та методів у сфері апаратно-програмних модулів для систем контролю дорожнього руху. Розглянуті аналогічні системи, такі як Head-up displays (HUD), програми автопілота для автономних транспортних засобів, системи розпізнавання дорожніх знаків (TSR).

Аналіз підходів для детектування у системах контролю дорожнього руху, таких як TensorFlow та SimpleCV, надає глибше розуміння можливостей та обмежень таких технологій у вирішенні завдань систем контролю дорожнього руху.

Формулювання вимог до апаратного та програмного забезпечення є кроком у напрямку створення оптимального та ефективного програмно-апаратного комплексу для системи контролю дорожнього руху. Вибір апаратних засобів та програмного забезпечення ґрунтується на ретельному аналізі потреб та можливостей системи.

Практична реалізація програмно-апаратного комплексу на базі технології доповненої реальності з використанням OpenCV на пристрої Raspberry Pi підкреслює практичну застосовність отриманих у роботі знань та методів. Це відкриває перспективи для реалізації новаторських технологій у сфері дорожньої безпеки та транспортного управління.

Отже, результатом цієї роботи є створення програмно-апаратного комплексу, який може відігравати ключову роль у покращенні безпеки дорожнього руху та ефективному управлінні транспортним потоком.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кім В. М., Пузирьов С. В. Система контролю дорожнього руху на базі доповненої реальності з використанням OpenCV. // Могиляньскі читання – 2023 / Матеріали XXVI Всеукраїнської науково-практичної конференції. Миколаїв: Видавництво ЧНУ ім. Петра Могили. 2023. 423 с.

2. Augmented Reality with OpenCV. URL: https://www.opencvhelp.org/tutorials/applications/augmented-reality/ (дата звернення: 15.12.2023).

3. Augmented Reality with OpenCV and OpenGL: the tricky projection matrix. URL: https://fruty.medium.com/augmented-reality-with-opencv-and-opengl-the-tricky-projection-matrix-394cc6050b00 (дата звернення: 16.12.2023).

4. Lane Detection and Traffic Sign Recognition using OpenCV and Deep Learning for Autonomous Vehicles. URL: https://www.slideshare.net/irjetjournal/lane-detection-and-traffic-sign-recognition-using-opencv-and-deep-learning-for-autonomous-vehicles (дата звернення: 17.12.2023).

5. Advanced Lane Detection for Autonomous Vehicles using Computer Vision techniques. URL: https://towardsdatascience.com/advanced-lane-detection-for-autonomous-vehicles-using-computer-vision-techniques-f229e4245e41 (дата звернення: 17.12.2023).

6. Moving Object Detection with OpenCV using Contour Detection and Background Subtraction. URL: https://learnopencv.com/moving-object-detection-with-opencv/ (дата звернення: 19.12.2023).

7. OpenCV: Open Source Computer Vision Library. URL: https://github.com/opencv/opencv (дата звернення: 19.12.2023).

8. SimpleCV. URL: https://github.com/sightmachine/SimpleCV (дата звернення: 22.12.2023).

9. BoofCV. URL: https://boofcv.org/index.php?title=Main_Page (дата звернення: 22.12.2023).

10. Image Recognition using TensorFlow. URL: https://www.geeksforgeeks.org/image-recognition-using-tensorflow/
(дата звернення: 24.12.2023).

11. Приклади кода для роботи с GPIO на C/C++. URL: https://github.com/cryptotronix/cryptotronix-cpp-gpio
(дата звернення: 24.12.2023).

12. Exploring Computer Vision with OpenCV: Building Image Recognition Systems. URL: https://medium.com/@lotfi-habbiche/exploring-computer-vision-with-opencv-building-image-recognition-systems-71fb39161d65
(дата звернення: 25.12.2023).

13. Machine Learning Overview. URL: https://docs.opencv.org/3.4/dc/dd6/ml_intro.html (дата звернення: 26.12.2023).

14. Face Recognition With Raspberry Pi and OpenCV URL: https://core-electronics.com.au/guides/face-identify-raspberry-pi/
(дата звернення: 26.12.2023).

15. Raspberry Pi OS. URL: https://www.raspberrypi.com/software/ (дата звернення: 26.11.2023).

16. Accessing the Raspberry Pi Camera with OpenCV and Python. URL: https://pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/ (дата звернення: 25.11.2023).

17. Traffic Signs Recognition using CNN and Keras in Python. URL: https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/ (дата звернення: 24.11.2023).

18. Understanding the Technology behind Traffic Sign Recognition (TSR) Systems. URL: https://www.design-reuse.com/articles/41154/traffic-sign-recognition-tsr-system.html (дата звернення: 26.11.2023).

19. Detecting and recognizing traffic signs : patent US20080137908A1. Unites States: US11/951. № US20080137908A1 ; applied on 06.12.2007; published on 13.06.2008.

20. GPIO (general purpose input/output) verification system and method : patent CN105844056A. China: G06F30/398. № CN105844056A ; applied on 15.04.2016 ; published on 10.08.2016.

21. Variable frame length virtual GPIO with a modified UART interface : patent US9880965B2. United States : G06F13/4221. № US9880965B2 ; applied on 10.09.2015 ; published on 17.03.2016.

22. Variable frame length virtual GPIO with a modified UART interface : patent US9880965B2. United States : G06F13/4221. № US9880965B2 ; applied on 10.09.2015 ; published on 17.03.2016.

23. Multi-port multi-sideband-GPIO consolidation technique over a multi-drop serial bus : patent US10467154B2. United States : G06F13/126. № US10467154B2 ; applied on 08.01.2018 ; published on 16.08.2018.

24. GPIO to GPIO communication over multi-node link networks : patent CN107025200B. China : G06F13/364. № CN107025200B ; applied on 26.01.2017 ; published on 08.08.2017.

25. Virtual GPIO : patent US9582456B2. United States : G06F13/4282. № US9582456B2 ; applied on 13.08.2015 ; published on 10.12.2015.

26. ДСТУ ISO/IEC 12207:2016. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення (ISO/IEC 12207:2008, IDT).

27. Городецька О. С., Гикавий В. А., Онищук О. В. Комп’ютерні мережі : навч. посіб. Вінниця : ВНТУ, 2017. 129 с.

28. Задерейко О. В., Логінова Н. І., Толокнов А. А. Комп’ютерні мережі : навч. посібник. Одеса : Фенікс, 2023. 249 с.

29. Лемешко А. В., Кирпач Л. А., Сорокін Д. В., Бученко І. А., Шрам М. М. Проектування безпроводових комп’ютерних мереж. Київ : ДУТ, 2021. 147 с.

30. Ткаченко О. М., Торошанко Я. І., Лемешко А. В. та ін. Комп’ютерні мережі: контроль та прогнозування перевантажень : посібник. Київ : ДУТ, 2021. 77 с.

31. Основні поняття систем масового обслуговування. URL: https://stud.com.ua/163945/informatika/osnovni_ponyattya_sistem_masovogo_obslugovuvannya (дата звернення: 25.11.2023).

32. Теорема гіпотез (формула Байєса). URL: http://ebooks.git-elt.hneu.edu.ua/tvms/p-2-5.html (дата звернення: 30.11.2023).

33. Множники Лагранжа. URL: https://ukrayinska.libretexts.org/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0/%D1%80%D0%BE%D0%B7%D1%80%D0%B0%D1%85%D1%83%D0%BD%D0%BA%D1%83/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0%3A_%D0%9E%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_(OpenStax)/14%3A_%D0%94%D0%B8%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D0%B0%D1%86%D1%96%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%B9_%D0%B4%D0%B5%D0%BA%D1%96%D0%BB%D1%8C%D0%BA%D0%BE%D1%85_%D0%B7%D0%BC%D1%96%D0%BD%D0%BD%D0%B8%D1%85/14.08%3A_%D0%9C%D0%BD%D0%BE%D0%B6%D0%BD%D0%B8%D0%BA%D0%B8_%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6%D0%B0
(дата звернення: 30.11.2023).

ДОДАТОК А

ДОВІДКА

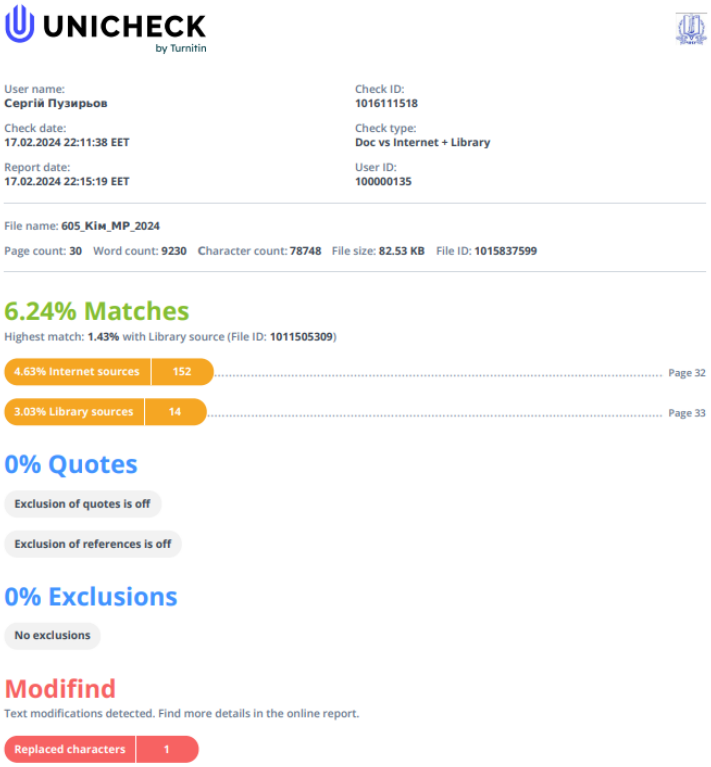
про перевірку на унікальність пояснювальної записки
кваліфікаційної магістерської роботи на тему:
«Система контролю дорожнього руху на основі доповненої реальності з використанням бібліотеки OpenCV»
студента спеціальності 123 «Комп'ютерна інженерія», 605м групи
Кіма Віктора Миколайовича

прізвище, ім'я, по-батькові

Шифр роботи: 123 – КМР.1 – 605.21810512

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту кваліфікаційної роботи: схожість складає 6,24 %.



Студент: Керівник:

_____ В. М. Кім _____ канд. фіз.-мат. наук, доцент С. В. Пузирьов

підпис ініціали, прізвище підпис ініціали, прізвище

Дата: « ___ » _____ 2024 р.

ДОДАТОК Б

Код програми

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt
from skimage.feature import blob_dog, blob_log, blob_doh
import imutils
import argparse
import os
import math

from classification import training, getLabel

SIGNS = ["ERROR",
         "STOP",
         "TURN LEFT",
         "TURN RIGHT",
         "DO NOT TURN LEFT",
         "DO NOT TURN RIGHT",
         "ONE WAY",
         "SPEED LIMIT",
         "OTHER"]

# Clean all previous file
def clean_images():
file_list = os.listdir('./')
for file_name in file_list:
       if '.png' in file_name:
              os.remove(file_name)


### Preprocess image
def constrastLimit(image):
    img_hist_equalized = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
    channels = cv2.split(img_hist_equalized)
    channels[0] = cv2.equalizeHist(channels[0])
    img_hist_equalized = cv2.merge(channels)
    img_hist_equalized = cv2.cvtColor(img_hist_equalized, cv2.COLOR_YCrCb2BGR)
    return img_hist_equalized

def LaplacianOfGaussian(image):
    LoG_image = cv2.GaussianBlur(image, (3,3), 0)           # paramter
    gray = cv2.cvtColor( LoG_image, cv2.COLOR_BGR2GRAY)
    LoG_image = cv2.Laplacian( gray, cv2.CV_8U,3,3,2)       # parameter
    LoG_image = cv2.convertScaleAbs(LoG_image)
    return LoG_image

def binarization(image):
    thresh = cv2.threshold(image,32,255,cv2.THRESH_BINARY)[1]
    #thresh  =  cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY,11,2)
    return thresh

def preprocess_image(image):
    image = constrastLimit(image)
    image = LaplacianOfGaussian(image)
```

```
            image = binarization(image)
            return image

        # Find Signs
        def removeSmallComponents(image, threshold):
            #find all your connected components (white blobs in your image)
            nb_components,         output,         stats,         centroids         =
cv2.connectedComponentsWithStats(image, connectivity=8)
            sizes = stats[1:, -1]; nb_components = nb_components - 1

            img2 = np.zeros((output.shape),dtype = np.uint8)
            #for every component in the image, you keep it only if it's above threshold
            for i in range(0, nb_components):
                if sizes[i] >= threshold:
                    img2[output == i + 1] = 255
            return img2

        def findContour(image):
            #find contours in the thresholded image
            cnts = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE    )
            cnts = cnts[0] if imutils.is_cv2() else cnts[1]
            return cnts

        def contourIsSign(perimeter, centroid, threshold):
            #  perimeter, centroid, threshold
            # # Compute signature of contour
            result=[]
            for p in perimeter:
                p = p[0]
                distance = sqrt((p[0] - centroid[0])**2 + (p[1] - centroid[1])**2)
                result.append(distance)
            max_value = max(result)
            signature = [float(dist) / max_value for dist in result ]
            # Check signature of contour.
            temp = sum((1 - s) for s in signature)
            temp = temp / len(signature)
            if temp < threshold: # is  the sign
                return True, max_value + 2
            else:                  # is not the sign
                return False, max_value + 2

        #crop sign
        def cropContour(image, center, max_distance):
            width = image.shape[1]
            height = image.shape[0]
            top = max([int(center[0] - max_distance), 0])
            bottom = min([int(center[0] + max_distance + 1), height-1])
            left = max([int(center[1] - max_distance), 0])
            right = min([int(center[1] + max_distance+1), width-1])
            print(left, right, top, bottom)
            return image[left:right, top:bottom]

        def cropSign(image, coordinate):
            width = image.shape[1]
            height = image.shape[0]
            top = max([int(coordinate[0][1]), 0])
            bottom = min([int(coordinate[1][1]), height-1])
            left = max([int(coordinate[0][0]), 0])
            right = min([int(coordinate[1][0]), width-1])
            #print(top,left,bottom,right)
            return image[top:bottom,left:right]
```

```
    def findLargestSign(image, contours, threshold, distance_theshold):
        max_distance = 0
        coordinate = None
        sign = None
        for c in contours:
            M = cv2.moments(c)
            if M["m00"] == 0:
                continue
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])
            is_sign, distance = contourIsSign(c, [cX, cY], 1-threshold)
            if is_sign and distance > max_distance and distance > distance_theshold:
                max_distance = distance
                coordinate = np.reshape(c, [-1,2])
                left, top = np.amin(coordinate, axis=0)
                right, bottom = np.amax(coordinate, axis = 0)
                coordinate = [(left-2,top-2),(right+3,bottom+1)]
                sign = cropSign(image,coordinate)
        return sign, coordinate


    def findSigns(image, contours, threshold, distance_theshold):
        signs = []
        coordinates = []
        for c in contours:
            # compute the center of the contour
            M = cv2.moments(c)
            if M["m00"] == 0:
                continue
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])
            is_sign, max_distance = contourIsSign(c, [cX, cY], 1-threshold)
            if is_sign and max_distance > distance_theshold:
                sign = cropContour(image, [cX, cY], max_distance)
                signs.append(sign)
                coordinate = np.reshape(c, [-1,2])
                top, left = np.amin(coordinate, axis=0)
                right, bottom = np.amax(coordinate, axis = 0)
                coordinates.append([(top-2,left-2),(right+1,bottom+1)])
        return signs, coordinates

    def  localization(image,  min_size_components,  similitary_contour_with_circle,
model, count, current_sign_type):
        original_image = image.copy()
        binary_image = preprocess_image(image)

        binary_image = removeSmallComponents(binary_image, min_size_components)

        binary_image          =          cv2.bitwise_and(binary_image,binary_image,
mask=remove_other_color(image))

        #binary_image = remove_line(binary_image)

        cv2.imshow('BINARY IMAGE', binary_image)
        contours = findContour(binary_image)
        #signs,       coordinates       =       findSigns(image,       contours,
similitary_contour_with_circle, 15)
        sign,     coordinate     =     findLargestSign(original_image,     contours,
similitary_contour_with_circle, 15)
```

```
            text = ""
            sign_type = -1
            i = 0

            if sign is not None:
                sign_type = getLabel(model, sign)
                sign_type = sign_type if sign_type <= 8 else 8
                text = SIGNS[sign_type]
                cv2.imwrite(str(count)+'_'+text+'.png', sign)

            if sign_type > 0 and sign_type != current_sign_type:
                cv2.rectangle(original_image, coordinate[0],coordinate[1], (0, 255, 0),
1)
                font = cv2.FONT_HERSHEY_PLAIN
                cv2.putText(original_image,text,(coordinate[0][0],  coordinate[0][1]  -
15), font, 1,(0,0,255),2,cv2.LINE_4)
            return coordinate, original_image, sign_type, text

        def remove_line(img):
            gray = img.copy()
            edges = cv2.Canny(gray,50,150,apertureSize = 3)
            minLineLength = 5
            maxLineGap = 3
            lines = cv2.HoughLinesP(edges,1,np.pi/180,15,minLineLength,maxLineGap)
            mask = np.ones(img.shape[:2], dtype="uint8") * 255
            if lines is not None:
                for line in lines:
                    for x1,y1,x2,y2 in line:
                        cv2.line(mask,(x1,y1),(x2,y2),(0,0,0),2)
            return cv2.bitwise_and(img, img, mask=mask)

        def remove_other_color(img):
            frame = cv2.GaussianBlur(img, (3,3), 0)
            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
            # define range of blue color in HSV
            lower_blue = np.array([100,128,0])
            upper_blue = np.array([215,255,255])
            # Threshold the HSV image to get only blue colors
            mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

            lower_white = np.array([0,0,128], dtype=np.uint8)
            upper_white = np.array([255,255,255], dtype=np.uint8)
            # Threshold the HSV image to get only blue colors
            mask_white = cv2.inRange(hsv, lower_white, upper_white)

            lower_black = np.array([0,0,0], dtype=np.uint8)
            upper_black = np.array([170,150,50], dtype=np.uint8)

            mask_black = cv2.inRange(hsv, lower_black, upper_black)

            mask_1 = cv2.bitwise_or(mask_blue, mask_white)
            mask = cv2.bitwise_or(mask_1, mask_black)
            # Bitwise-AND mask and original image
            #res = cv2.bitwise_and(frame,frame, mask= mask)
            return mask

        def main(args):
        #Clean previous image
            clean_images()
            #Training phase
```

```
            model = training()

            vidcap = cv2.VideoCapture(args.file_name)

            fps = vidcap.get(cv2.CAP_PROP_FPS)
            width = vidcap.get(3)  # float
            height = vidcap.get(4) # float

            # Define the codec and create VideoWriter object
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
            out = cv2.VideoWriter('output.avi',fourcc, fps , (640,480))

            # initialize the termination criteria for cam shift, indicating
            # a maximum of ten iterations or movement by a least one pixel
            # along with the bounding box of the ROI
            termination = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)
            roiBox = None
            roiHist = None

            success = True
            similitary_contour_with_circle = 0.65   # parameter
            count = 0
            current_sign = None
            current_text = ""
            current_size = 0
            sign_count = 0
            coordinates = []
            position = []
            file = open("Output.txt", "w")
            while True:
                success,frame = vidcap.read()
                if not success:
                    print("FINISHED")
                    break
                width = frame.shape[1]
                height = frame.shape[0]
                #frame = cv2.resize(frame, (640,int(height/(width/640))))
                frame = cv2.resize(frame, (640,480))

                print("Frame:{}".format(count))
                #image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
                coordinate,    image,    sign_type,    text    =    localization(frame,
args.min_size_components,    args.similitary_contour_with_circle,    model,    count,
current_sign)
                if coordinate is not None:
                    cv2.rectangle(image, coordinate[0],coordinate[1], (255, 255, 255),
1)
                print("Sign:{}".format(sign_type))
                if sign_type > 0 and (not current_sign or sign_type != current_sign):
                    current_sign = sign_type
                    current_text = text
                    top = int(coordinate[0][1]*1.05)
                    left = int(coordinate[0][0]*1.05)
                    bottom = int(coordinate[1][1]*0.95)
                    right = int(coordinate[1][0]*0.95)

                    position   =   [count,   sign_type   if   sign_type   <=   8   else   8,
coordinate[0][0], coordinate[0][1], coordinate[1][0], coordinate[1][1]]
                    cv2.rectangle(image, coordinate[0],coordinate[1], (0, 255, 0), 1)
                    font = cv2.FONT_HERSHEY_PLAIN
```

```
                    cv2.putText(image,text,(coordinate[0][0],  coordinate[0][1]  -15),
font, 1,(0,0,255),2,cv2.LINE_4)

                    tl = [left, top]
                    br = [right,bottom]
                    print(tl, br)
                    current_size      =      math.sqrt(math.pow((tl[0]-br[0]),2)      +
math.pow((tl[1]-br[1]),2))
                    # grab the ROI for the bounding box and convert it
                    # to the HSV color space
                    roi = frame[tl[1]:br[1], tl[0]:br[0]]
                    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
                    #roi = cv2.cvtColor(roi, cv2.COLOR_BGR2LAB)

                    # compute a HSV histogram for the ROI and store the
                    # bounding box
                    roiHist = cv2.calcHist([roi], [0], None, [16], [0, 180])
                    roiHist = cv2.normalize(roiHist, roiHist, 0, 255, cv2.NORM_MINMAX)
                    roiBox = (tl[0], tl[1], br[0], br[1])

                elif current_sign:
                    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
                    backProj = cv2.calcBackProject([hsv], [0], roiHist, [0, 180], 1)

                    # apply cam shift to the back projection, convert the
                    # points to a bounding box, and then draw them
                    (r, roiBox) = cv2.CamShift(backProj, roiBox, termination)
                    pts = np.int0(cv2.boxPoints(r))
                    s = pts.sum(axis = 1)
                    tl = pts[np.argmin(s)]
                    br = pts[np.argmax(s)]
                    size = math.sqrt(pow((tl[0]-br[0]),2) +pow((tl[1]-br[1]),2))
                    print(size)

                    if  current_size < 1 or size < 1 or size / current_size > 30 or
math.fabs((tl[0]-br[0])/(tl[1]-br[1])) > 2 or math.fabs((tl[0]-br[0])/(tl[1]-br[1]))
< 0.5:
                        current_sign = None
                        print("Stop tracking")
                    else:
                        current_size = size

                    if sign_type > 0:
                        top = int(coordinate[0][1])
                        left = int(coordinate[0][0])
                        bottom = int(coordinate[1][1])
                        right = int(coordinate[1][0])

                        position = [count, sign_type if sign_type <= 8 else 8, left,
top, right, bottom]
                        cv2.rectangle(image, coordinate[0],coordinate[1], (0, 255, 0),
1)
                        font = cv2.FONT_HERSHEY_PLAIN
                        cv2.putText(image,text,(coordinate[0][0],  coordinate[0][1]  -
15), font, 1,(0,0,255),2,cv2.LINE_4)
                    elif current_sign:
                        position = [count, sign_type if sign_type <= 8 else 8, tl[0],
tl[1], br[0], br[1]]
                        cv2.rectangle(image, (tl[0], tl[1]),(br[0], br[1]), (0, 255, 0),
1)
                        font = cv2.FONT_HERSHEY_PLAIN
```

```
                    cv2.putText(image,current_text,(tl[0],   tl[1]   -15),   font,
1,(0,0,255),2,cv2.LINE_4)

                if current_sign:
                    sign_count += 1
                    coordinates.append(position)

                cv2.imshow('Result', image)
                count = count + 1
                #Write to video
                out.write(image)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            file.write("{}".format(sign_count))
            for pos in coordinates:
                file.write("\n{}            {}            {}            {}            {}
{}".format(pos[0],pos[1],pos[2],pos[3],pos[4], pos[5]))
            print("Finish {} frames".format(count))
            file.close()
            return


        if __name__ == '__main__':
            parser = argparse.ArgumentParser(description="NLP Assignment Command Line")

            parser.add_argument(
              '--file_name',
              default= "./MVI_1049.avi",
              help= "Video to be analyzed"
              )

            parser.add_argument(
              '--min_size_components',
              type = int,
              default= 300,
              help= "Min size component to be reserved"
              )


            parser.add_argument(
              '--similitary_contour_with_circle',
              type = float,
              default= 0.65,
              help= "Similitary to a circle"
              )

            args = parser.parse_args()
            main(args)
```

ДОДАТОК В

Публікації за темою роботи

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

ЧНУ

«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

УДК 004.932:53.087.4

Кім В. М.,
студент 6-го курсу,
Пузирьов С. В.,
канд. фіз.-мат. наук, доцент,
доцент кафедри КІ,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

СИСТЕМА КОНТРОЛЮ ДОРОЖНЬОГО РУХУ НА БАЗІ ДОПОВНЕНОЇ РЕАЛЬНОСТІ З ВИКОРИСТАННЯМ OPENCV

Сучасний світ на кожному кроці свідчить про безперервний розвиток технологій та інновацій, які впливають на різні сфери життя суспільства. Однією з галузей, яка невпинно трансформується та покращується завдяки передовим технологіям, є дорожній рух і безпека на дорогах. На фоні зростаючої кількості транспортних засобів і складності дорожнього руху, необхідність вдосконалення систем контролю дорожнього руху та підвищення рівня безпеки водіїв і пішоходів стає надзвичайно актуальною.

Доповнена реальність (Augmented Reality, AR) і бібліотека OpenCV (Open Source Computer Vision Library) представляють собою два передові технологічні інструменти, які можуть бути успішно використані для досягнення цієї мети. Інтеграція AR та OpenCV в системи контролю дорожнього руху може створити потужні інструменти для аналізу, моніторингу та оптимізації дорожнього середовища, а також для підвищення безпеки всіх учасників дорожнього руху (рис. 1).

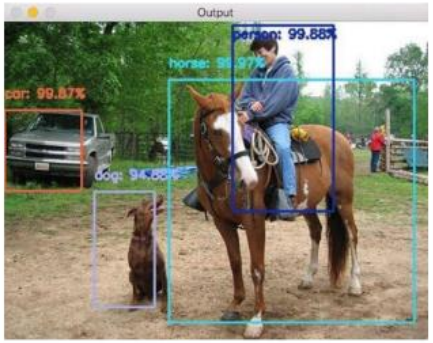


Рисунок 1 – Приклад зображення, на якому розпізнано кілька об'єктів, у тому числі автомобіль, собаку, коня та людину, за допомогою OpenCV, глибокого навчання та виявлення об'єктів

Використання OpenCV та AR для контролю дорожнього руху має свої переваги і недоліки, які варто враховувати при розробці та впровадженні таких систем.

Переваги використання OpenCV та AR:

– підвищена безпека: Системи на базі OpenCV та AR можуть виявляти небезпечні ситуації на дорозі та надавати водіям та пішоходам важливу інформацію для запобігання аваріям та нещасним випадкам;

– покращена навігація: AR може надавати водіям та пішоходам інтерактивні вказівки та віртуальні шляхи, що допомагає їм точно навігувати;

– розпізнавання дорожніх знаків: OpenCV може автоматично розпізнавати дорожні знаки та інші об'єкти на дорозі, забезпечуючи додаткову інформацію водіям;

– ефективність системи контролю дорожнього руху: Об'єднання OpenCV та AR може покращити ефективність систем контролю дорожнього руху, дозволяючи використовувати реальний час інформації для прийняття рішень.

Недоліки використання OpenCV та AR:

– високі вимоги до обладнання: Використання OpenCV та AR може вимагати потужних обчислювальних ресурсів та дорогих обладнання, що може бути недоступним для всіх дорожніх систем;

– споживання енергії: Завдяки постійному аналізу відеопотоку, системи на базі OpenCV можуть споживати багато електроенергії, що може бути проблемою для портативних пристроїв;

– спроби проникнення та приватність: Використання AR може створювати потенційні проблеми з приватністю, оскільки він може відображати особисту інформацію на публічних місцях;

– системи не завжди надійні: Залежно від умов освітлення, погодних умов і інших факторів, системи на базі OpenCV та AR можуть бути менш надійними в певних ситуаціях;

– витрати на розробку та підтримку: Розробка та підтримка систем на базі OpenCV та AR може вимагати значних витрат на час та кошти.

Впровадження AR та OpenCV в сферу дорожнього руху може відкрити нові перспективи для зменшення кількості дорожніх аварій, підвищення ефективності дорожньої інфраструктури та створення безпечніших умов для всіх учасників руху. Завдяки цим технологіям, ми маємо можливість революціонізувати спосіб, яким ми бачимо, розуміємо і контролюємо дорожній рух, і ця дипломна робота є першим кроком у цьому напрямку.

Список використаних джерел

1. OpenCV: Object Detection. URL: https://docs.opencv.org/3.4/d5/d54/group__objdetect.html (Last accessed: 01.10.2023).

2. Object detection with deep learning and OpenCV. URL: https://pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/ (Last accessed: 01.10.2023).

3 Traffic Signs Detection by YOLO v3, OpenCV, Keras. URL: https://www.kaggle.com/code/valentynsichkar/traffic-signs-detection-by-yolo-v3-opencv-keras (Last accessed: 01.10.2023).

4. OpenCV Augmented Reality (AR). URL: https://pyimagesearch.com/2021/01/04/opencv-augmented-reality-ar/ (Last accessed: 01.10.2023).