

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ІНТЕЛЕКТУАЛЬНА СИСТЕМА УПРАВЛІННЯ
ПЛАТНИМ ПАРКУВАННЯМ З КОМП'ЮТЕРНИМ ЗОРОМ

Виконав: студент 2 курсу, групи КНТ-512м
спеціальності _____

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні системи та мережі

ТРУБИЧИНА Д. В.

(ПРИЗВИЩЕ та ініціали)

Керівник КУДЕРМЕТОВ Р.К.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗІНА Г.Л.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти магістерський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) «Комп'ютерні системи та мережі»
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ТРУБИЧИНА Діана Василівна

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Інтелектуальна система управління платним паркуванням з комп'ютерним зором

керівник проєкту (роботи) к. т. н., доцент, КУДЕРМЕТОВ Равіль Камілович

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) аналоги систем управління паркуванням, задана економічна ефективність та функціональна гнучкість

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити

1) Аналіз предметної області;

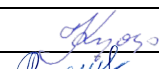
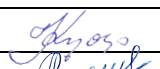
2) Технологічний огляд, планування архітектури та стратегія розробки системи;

3) Проектування системи управління паркуванням;

4) Розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КУДЕРМЕТОВ Р. К., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	01.10.2023 р.	
2	Огляд систем керування паркування	10.10.2023 р.	
3	Постановка завдання	20.10.2023 р.	
4	Технологічний огляд і вибір інструментів	25.11.2023 р.	
5	Планування архітектури системи	01.11.2023 р.	
6	Проектування користувацького інтерфейсу	05.11.2023 р.	
7	Координація та взаємодія компонентів системи	10.11.2023 р.	
8	Розробка програми класифікації паркувальних місць	15.11.2023 р.	
9	Розробка адміністративної програми	20.11.2023 р.	
10	Розробка мобільного застосунку	25.11.2023 р.	
11	Аналіз результатів роботи системи	01.12.2023 р.	
12	Оформлення пояснювальної записки	10.12.2023 р.	

Студент  **Діана ТРУБИЧИНА**
 (підпис) (ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)  **Равіль КУДЕРМЕТОВ**
 (підпис) (ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 65 с., 25 рисунків, 25 джерел, 1 додаток.

KIVY, OPENCV, PYTHON, PYTORCH, R-CNN, WXPYTHON,
АВТОМАТИЗАЦІЯ, АДМІНІСТРАТИВНА ПРОГРАММА, КЛАСИФІКАЦІЯ,
КОМП'ЮТЕРНИЙ ЗІР, ПАРКОВКА

Мета роботи – створення ефективної та гнучкої системи управління платним паркуванням, оснащеної комп'ютерним зором, здатної до інтеграції з наявними системами відеоспостереження і різними сенсорами.

У першій частині дипломної роботи проведено аналіз наявних систем управління паркуванням, з акцентом на класифікації та використаних підходах. Сформовано вимоги для проєктування та розробки системи.

У другій частині здійснено технологічний огляд, планування архітектури та розробки стратегії для інтелектуальної системи управління платним паркуванням, проведено вибір технологій комп'ютерного зору, зокрема одноступеневих і двоступеневих детекторів об'єктів, та засобів для розробки системи та інтеграції з наявною інфраструктурою.

У третій частині описано розробку користувацького інтерфейсу програмного забезпечення та алгоритмів координації взаємодії компонентів системи.

У четвертій частині приведено детальний опис процесів розробки різних компонентів системи, включаючи програму класифікації паркувальних місць, адміністративну програму, мобільний застосунок та проведено тестування системи.

ABSTRACT

Explanatory note: 65 p., 25 pictures, 25 references, 1 appendix.

KIVY, OPENCV, PYTHON, PYTORCH, R-CNN, WXPYTHON,
AUTOMATION, ADMINISTRATIVE PROGRAM, CLASSIFICATION,
COMPUTER VISION, PARKING

The goal of the work is to create an effective and flexible paid parking management system equipped with computer vision, capable of integration with existing video surveillance systems and various sensors.

In the first part of the thesis, an analysis of existing parking management systems was carried out, with an emphasis on classification and used approaches. The requirements for designing and developing the system have been formed.

In the second part, a technological review, architecture planning and strategy development for an intelligent paid parking management system were carried out, a selection of computer vision technologies, in particular single-stage and two-stage object detectors, and means for system development and integration with the existing infrastructure was carried out.

The third part describes the development of the user interface of the software and algorithms for coordinating the interaction of system components.

The fourth part provides a detailed description of the development processes of various system components, including the parking space classification program, the administrative program, the mobile application, and system testing.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної області	9
1.1 Огляд систем керування паркуванням	10
1.2 Внесок комп'ютерного зору в системи паркування.....	22
1.3 Постановка задачі.....	24
2 Технологічний огляд, планування архітектури та стратегія розробки системи.....	25
2.1 Технологічний огляд і вибір інструментів	25
2.1.1 Огляд технологій комп'ютерного зору.....	26
2.1.2 Вибір інструментів для розробки	36
2.2 Планування архітектури системи.....	38
2.3 Стратегія розробки та реалізації.....	39
3 Проєктування системи управління паркуванням	40
3.1 Архітектура системи	40
3.2 Проєктування користувацького інтерфейсу	42
3.2.1 Інтерфейс адміністративної програми	42
3.2.2 Інтерфейс мобільного застосунка.....	44
3.3 Координація та взаємодія компонентів системи	46
4 Розробка системи	50
4.1 Розробка програми класифікації паркувальних місць.....	50
4.2 Розробка адміністративної програми	53
4.3 Розробка мобільного застосунку	54
Висновки.....	56
Перелік джерел посилання	57
Додаток А Лістинги програм.....	60

ВСТУП

Розумне паркування відіграє ключову роль у поліпшенні міської інфраструктури, забезпечуючи ефективне використання обмеженого міського простору. Ця система сприяє зниженню трафіку, зменшенню часу, що витрачається на пошук паркувального місця, і скороченню рівня загазованості в містах. Шляхом інтеграції сучасних технологій, сенсорів, GPS і мобільні застосунки, розумне паркування робить значний внесок у розв'язання проблем міської мобільності, сприяючи створенню стійкішого й екологічно чистого міського середовища [1].

Основні цілі та завдання розумних паркувальних систем включають поліпшення міської інфраструктури та скорочення проблем, пов'язаних з міською мобільністю та екологією. Розумне паркування спрямоване на забезпечення ефективного використання паркувального простору, зниження трафіку і зменшення екологічного впливу. Ці системи полегшують пошук вільних місць, скорочують час, що витрачається на паркування, і зменшують паливні витрати та вуглецевий слід. Вони також допомагають знизити стрес водіїв під час пошуку паркувальних місць і зменшити завантаженість доріг, покращуючи тим самим загальну якість міського життя [2].

В умовах сучасних міст, де ефективне управління паркуванням має ключове значення для зниження транспортних проблем і підвищення комфорту міського середовища, розробка інтелектуальних систем управління платним паркуванням стає дедалі актуальнішою. Ці системи не тільки спрощують процес управління паркувальними місцями, а й сприяють оптимізації міського руху, скороченню часу, що витрачається на пошук вільного місця, а також зменшенню забруднення навколишнього середовища. Інтеграція з наявною інфраструктурою та застосування сучасних апаратних й програмних технологій, зокрема комп'ютерного зору, робить такі системи гнучкими та економічно вигідними. У рамках цієї дипломної роботи магістра

пропонується створення такої системи, яка буде особливо корисною в комерційних і приватних структурах, таких як готелі та розважальні центри, де ефективне управління паркувальними просторами відіграє важливу роль.

Таким чином дипломна робота магістра спрямована на створення системи, яка поліпшить управління паркуванням й підвищить рівень комфорту та безпеки в міському середовищі, спираючись на сучасні технології та інтеграцію з наявними системами.

Для реалізації цієї мети належить вирішити такі проблеми:

- проаналізувати наявні технології і вибрати найбільш підходящі типи для реалізації системи;
- розробити архітектуру системи й загальний алгоритм функціонування;
- розробити прототипи користувацьких інтерфейсів адміністративного та мобільного додатку;
- здійснити вибір програмних інструментів, які орієнтовані на створення гнучкої та масштабованої системи;
- здійснити програмну реалізацію користувацьких інтерфейсів;
- провести оцінку функціонування системи й інтерпретувати данні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розумна парковка (smart parking) – являє собою систему, що інтегрує сучасні технології для оптимізації процесу паркування [3]. Ця концепція заснована на використанні датчиків і програмного забезпечення для моніторингу та управління доступністю паркувальних місць, тим самим знижуючи трафік і екологічний вплив. Історичне коріння розумного паркування сягає початку XX століття, коли з появою автомобілів виникла потреба в більш ефективних паркувальних рішеннях. Розвиток цієї концепції спостерігався через впровадження автоматизованих систем, прогрес у сенсорних технологіях і впровадження GPS, що призвело до сучасних комплексних систем управління паркуванням [4].

У рамках огляду технологій, що застосовуються в розумних паркувальних системах, можна виділити такі ключові напрямки. По-перше, наземні датчики, які, використовуючи радіолокаційні технології, встановлюються в структуру паркувальних місць. Ці пристрої здатні ідентифікувати наявність автомобіля над собою, тим самим надаючи дані про зайнятість конкретного місця. Важливо відзначити, що, незважаючи на тривалий період використання, основні принципи роботи цих датчиків залишилися незмінними.

Другим значущим елементом є системи паркувальних лічильників, що часто трапляються в структурованих парковках і закритих гаражах. Ці системи зазвичай містять механізми контролю в'їзду і виїзду транспортних засобів, а також пристрої для видачі паркувальних квитків, що дає змогу ефективно керувати потоком автомобілів і процесом оплати паркування.

Третім напрямком є використання накладних датчиків паркування або камерних систем. Такі пристрої, що монтуються на ліхтарних стовпах або будівлях, дають змогу здійснювати моніторинг великих паркувальних просторів, значно розширюючи зону спостереження. Це сприяє більш

ефективному розподілу паркувальних місць і поліпшенню загальної організації міського трафіку [3].

Розумні паркувальні системи мають значний вплив на повсякденне життя міських жителів і управління міським трафіком. Вони не тільки спрощують процес пошуку вільних місць для паркування, а й сприяють зниженню загального обсягу транспортного потоку, що, своєю чергою, зменшує час, проведений у дорозі, і скорочує рівень стресу у водіїв. Завдяки точному моніторингу зайнятості паркувальних місць і динамічному управлінню цінами на паркування, ці системи сприяють більш раціональному розподілу транспортних потоків, зменшуючи тим самим завантаженість доріг у години пік [5].

Крім того, розумні паркувальні системи роблять внесок у поліпшення екологічної ситуації в містах. Зниження часу, проведеного в пошуках парковки, призводить до зменшення викидів вуглекислого газу та інших шкідливих речовин, що виділяються автомобілями. Це особливо важливо в умовах міських агломерацій, де проблема загазованості повітря стоїть особливо гостро.

На сьогодні, розумні паркувальні системи є важливим елементом сучасної міської інфраструктури, що сприяє не тільки підвищенню ефективності використання паркувального простору, а й поліпшенню якості життя міських жителів. Вони забезпечують більш зручне та екологічно чисте використання міського простору, сприяючи створенню більш стійкого і комфортного міського оточення.

1.1 Огляд систем керування паркуванням

Сучасний міський ландшафт стикається з безліччю викликів, серед яких одним з найбільш значущих є ефективне управління паркувальним простором. В умовах постійно зростаючого автомобільного парку і обмеженості міського

простору, завдання оптимізації використання паркувальних місць стає актуальним і важливим. У цьому контексті системи управління паркуванням відіграють ключову роль, надаючи рішення, спрямовані на підвищення ефективності, скорочення часу пошуку вільного місця і поліпшення загального рівня зручності для користувачів.

Розвиток технологій, зокрема, в галузі інформаційних систем, Інтернету речей (IoT) і комп'ютерного зору, відкрив нові можливості для вдосконалення систем управління паркуваннями. Ці технологічні інновації дали змогу перейти від традиційних механічних рішень до більш складних та інтелектуальних систем, здатних адаптуватися до мінливих умов і надавати дані в реальному часі [6-7].

Сучасні системи управління паркуваннями можна класифікувати за кількома ключовими параметрами, включно з типом використовуваних технологій, масштабом застосування, рівнем автоматизації та інтеграції з іншими міськими системами [8]. Ці системи варіюються від базових рішень, що ґрунтуються на використанні датчиків зайнятості місць, до складних інтегрованих платформ, що включають елементи штучного інтелекту і машинного навчання для аналізу великих даних і оптимізації процесів управління паркувальним простором.

Актуальність дослідження різних систем управління паркуванням зумовлена не тільки їхнім безпосереднім впливом на повсякденне життя міських мешканців, а й потенціалом для подальшого розвитку розумних міських інфраструктур. Нижче представлено огляд основних типів систем управління паркування, їхніх характеристик та особливостей застосування.

Системи управління паркуванням являючи собою складні та багатофункціональні технологічні рішення піддаються класифікації за різними критеріями, включно з технологіями, які використовуються, методами обробки даних і рівнем взаємодії з користувачем. Розгляд різних типів цих систем дає змогу глибше зрозуміти їхню функціональність, переваги та обмеження, а також потенціал для інтеграції в ширші міські інфраструктури. Під час

предметного аналізу дипломній роботі магістра було досліджено 4 типи системи керування:

- системи на основі сенсорів;
- системи на основі камер і комп'ютерного зору;
- мобільні додатки та геолокаційні технології;
- радіочастотно-ідентифікаційні (RFID) технології.

Системи управління паркуванням на основі датчиків являють собою різноманітні технології, спрямовані на поліпшення використання паркувальних місць і спрощення процесу паркування. Основний компонент цих систем – сенсори, що встановлюються на паркувальних місцях або поблизу них (рис. 1.1). Вони забезпечують детектування наявності автомобілів і передачу даних у центральну систему управління.

Такі системи можуть включати ультразвукові, інфрачервоні або магнітні датчики. Ультразвукові датчики використовуються для визначення наявності автомобіля шляхом вимірювання відстані між датчиком і об'єктом. Інфрачервоні датчики виявляють наявність об'єктів завдяки зміні температури навколишнього середовища, а магнітні датчики реагують на зміни в магнітному полі, викликані наявністю металевого автомобіля.



Рисунок 1.1 – Приклад розташування сенсорів у сенсорній системі управління паркуванням

Системи на основі датчиків можуть бути інтегровані з іншими технологіями, наприклад зі світлодіодними (LED) індикаторами для візуального відображення статусу паркувальних місць або системи управління паркуванням для оптимізації розподілу транспортних засобів і навігації водіїв до вільних місць. Також ці системи часто використовуються разом із мобільними застосунками, які надають користувачам інформацію про доступність місць у реальному часі [9].

Перевагами систем управління парковками на основі датчиків є їхня точність і здатність забезпечувати актуальні дані про стан паркувальних місць. Це підвищує ефективність використання паркувального простору та зменшує час, що витрачається водіями на пошук вільних місць. До того ж, такі системи сприяють зниженню рівня трафіку в зонах парковок і зменшенню викидів вуглекислого газу, оскільки водії проводять менше часу в пошуках паркування.

Однак варто відзначити і деякі обмеження. Висока вартість встановлення та обслуговування може бути стримуючим фактором для багатьох операторів парковок. Крім того, в деяких умовах навколишнього середовища (наприклад, екстремальні температури, забруднення) точність і надійність датчиків можуть знижуватися.

Загалом, системи управління паркуваннями на основі датчиків є значущим інструментом в арсеналі урбаністичного планування та управління транспортними потоками, надаючи цінні можливості для оптимізації міської інфраструктури та підвищення зручності для водіїв.

Наступний розглянутий тип системи, системи управління парковками, що використовують камери і комп'ютерний зір, вони є частиною інтелектуальних транспортних систем (ITS) і спрямовані, за рахунок сучасних технологій, на підвищення ефективності використання паркувальних місць.

Ці системи містять високотехнологічні оптичні сенсори та програмне забезпечення для аналізу даних, покращуючи тим самим якість обслуговування клієнтів [10]. Прикладом такої системи є система, побудована на основі пристрою Wisenet TNF-9010 від Hanwha Techwin. Цей пристрій оснащено

панорамною IP-камерою, здатною контролювати до 16 паркувальних місць, візуальне представлення інформації, що надає камера, можна побачити на рисунку 1.2. Ця камера, зображення якої наведено на рисунку 1.3, обладнана об'єктивом з оглядом на 360° і 12-мегапіксельним сенсором, а також світлодіодною індикацією для відображення стану зайнятості паркувальних місць [11].



Рисунок 1.2 – Візуальне представлення інформації з камери Wisenet TNF-9010



Рисунок 1.3 – Зовнішній вигляд модулю Wisenet TNF-9010

Згідно з інженерними рішеннями, сенсор Wisenet TNF-9010 встановлюють на парковках підземного або ангарного типу на стелі, як показано на рисунку 1.4.

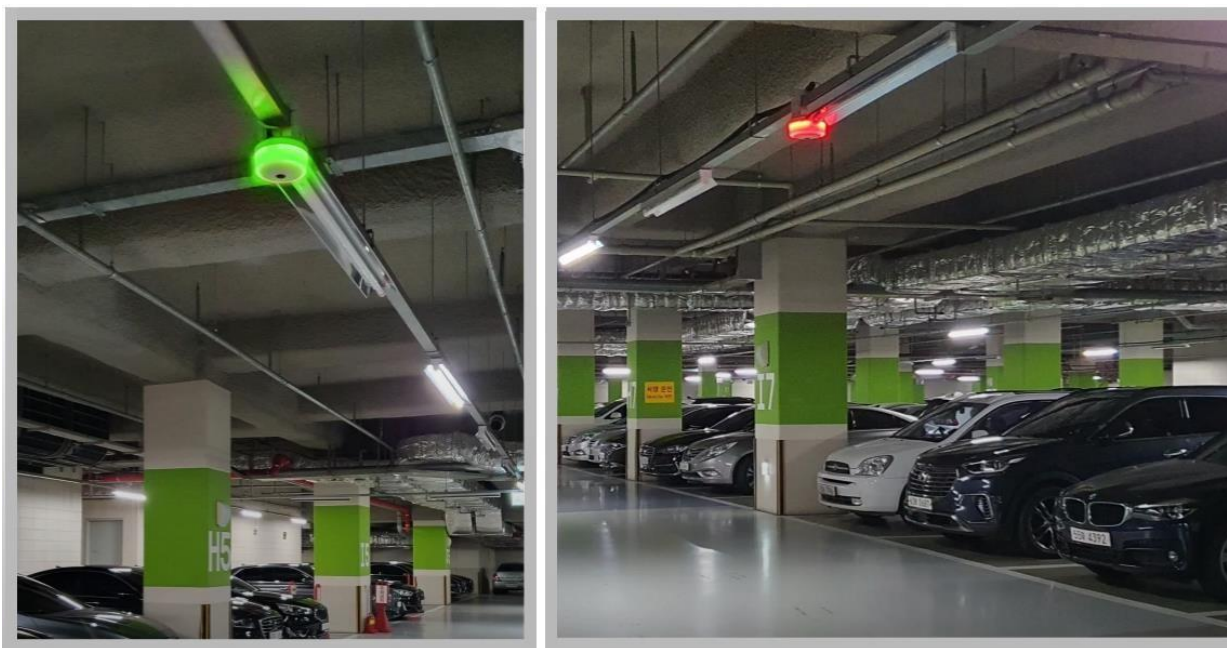


Рисунок 1.4 – Приклади розміщення сенсора у підземних парковках

Пристрої на стелі розміщуються один за одним через кожні чотири паркувальних місця (рисунок 1.5). Монтаж пристрою до конструкції стелі здійснюється за допомогою комплектної монтажної пластини. Через цю пластину пропускаються кабелі для передачі даних і живлення до пристрою.

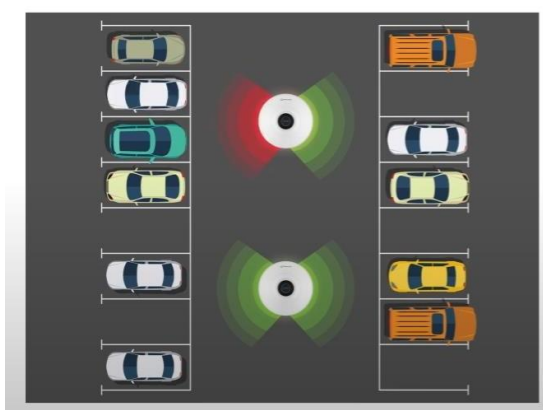


Рисунок 1.5 – Схематичне розміщення сенсора у підземних парковках

Потім користувач може під'єднати клемну колодку живлення до порту живлення DC 12 В на корпусі камери, попередньо відкривши спеціальну кришку роз'єму, а після – закрити її, забезпечивши щільність і надійність з'єднання. Також необхідно під'єднати кабель Ethernet для передавання даних

(рис. 1.6). У документації зазначено, що камера Wisenet TNF-9010, хоча й розрахована на живлення від джерела постійного струму напругою 12 В, підтримує технологію Power over Ethernet (PoE) відповідно до стандарту IEEE 802.3af. Це дає змогу здійснювати живлення стандартним кабелем Ethernet, яким передаються дані та живлення. Використання PoE спрощує установку, знижуючи необхідність у додатковій проводці, оскільки той самий кабель, який використовується для підключення до мережі, може також забезпечувати живлення камери. Згідно з документацією, якщо джерело живлення по PoE і DC 12V увімкнені одночасно, живлення пристрою буде здійснюватися як від PoE, так і від DC 12V [12].

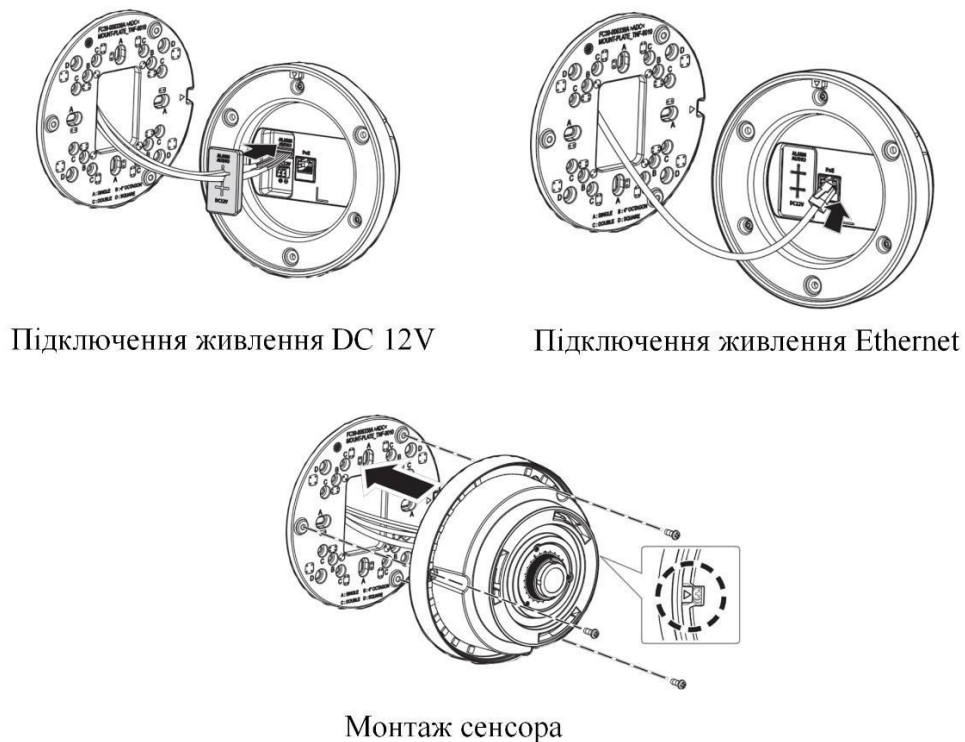


Рисунок 1.6 – Вигляд процесу під'єднання клем до пристрою й пристрою до монтажної пластини

Після монтажу кабель Ethernet від камер підключається до комутатора, а з нього до комп'ютера, на якому працює спеціальне програмне забезпечення. Воно виконує передачу візуальної інформації через інтернет на сервери компанії, де реалізовано алгоритми на базі штучного інтелекту для виявлення

об'єктів у реальному часі (рис. 1.7). За допомогою цих алгоритмів система здатна в реальному часі визначати і вказувати наявність вільних місць, використовуючи колірні сигнали. Також, камера виконує функції охоронного та ситуаційного моніторингу. Інтелектуальні паркувальні системи здатні збирати й аналізувати дані для управління дорожнім рухом, що дає змогу стратегічно планувати й оптимізувати використання паркувального простору.

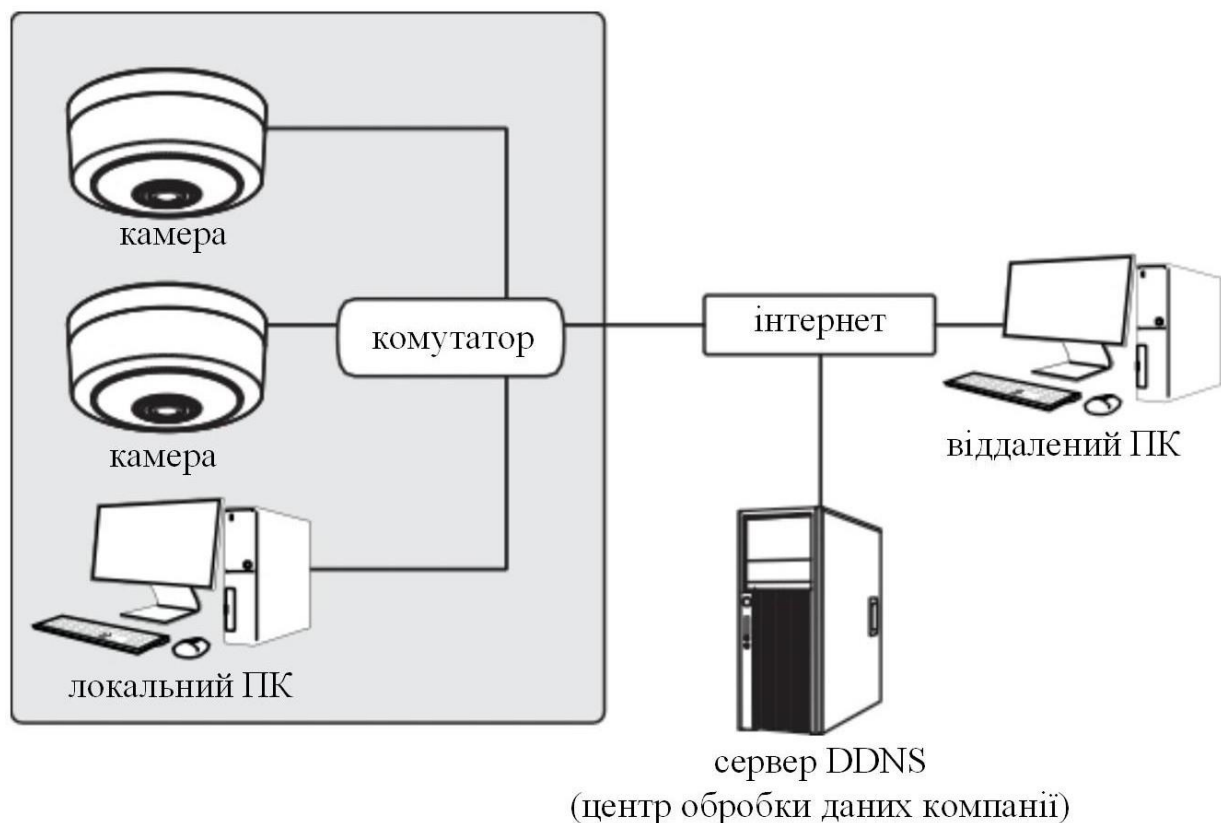


Рисунок 1.7 – Мережева схема системи

Такі системи можуть забезпечувати безпечне, скоординоване і «розумне» використання транспортних мереж.

Системи управління паркуванням на основі камер і комп'ютерного зору являють собою передове рішення, що збільшує ефективність паркувальних просторів і сприяє поліпшенню загального досвіду паркування. Вони враховують потреби сучасних міських умов, забезпечуючи оптимізацію використання паркувальних місць і покращуючи якість обслуговування

клієнтів. Використання камер і комп'ютерного зору дає змогу збирати й аналізувати дані про стан паркувальних місць у реальному часі, що сприяє в деяких рішеннях, ефективному управлінню дорожнім рухом і поліпшенню міської інфраструктури.

Серед переваг таких систем можна відзначити їхню високу здатність до виявлення вільних місць на парковках, а також моніторинг і управління паркувальними зонами за допомогою сучасних технологій. Такі системи також можуть бути інтегровані з іншими елементами інтелектуальних транспортних систем, у своїй екосистемі, забезпечуючи комплексне поліпшення міського руху та інфраструктури [13].

Важливо зазначити, що для успішного функціонування таких систем необхідно враховувати різні фактори, такі як стабільний й високошвидкісний інтернет, якість зображень, умови освітлення, погодні умови та можливі перешкоди для камер. Незважаючи на ці виклики, системи управління паркуванням на основі камер і комп'ютерного зору являють собою перспективний напрямок у розвитку інтелектуальних транспортних систем і «розумних» міст.

Систем управління паркуваннями, у яких в основі лежать мобільні застосунки та геолокаційні технології. Технологія зосереджена на підвищенні зручності та ефективності управління парковочними просторами за рахунок можливостей смартфонів. Ці технології являють собою синтез мобільного зв'язку, геолокаційного позиціонування і часто взаємодіють з датчиками та іншими IoT-пристроями, що дозволяють відстежувати наявність місць для паркування в реальному часі.

Центральна серверна система аналізує зібрані дані, визначаючи наявність вільних місць, і надсилає цю інформацію користувачам через мобільні застосунки. Окрім інформування про вільні місця, деякі системи також пропонують функцію навігації, керуючи водіїв до найближчих доступних місць для паркування. Інтеграція з платіжними системами в мобільних додатках

дозволяє здійснювати оплату за паркування безпосередньо зі смартфона, що підвищує зручність використання парковок [14].

Переваги систем управління паркуванням, що базуються на використанні мобільних застосунків та геолокаційних технологій, є значними. По-перше, вони сприяють покращенню ефективності використання парковочних місць: швидке інформування водіїв про вільні місця зменшує час, витрачений на пошук парковки. По-друге, оптимізація процесу пошуку парковки також сприяє зменшенню трафіку та викидів вуглекислого газу, що є важливим аспектом у боротьбі з міськими заторами та забрудненням навколишнього середовища. По-третє, інтеграція з платіжними системами та реалізація навігаційних функцій підвищують зручність та інноваційність використання парковок. Додатково, збір та аналіз даних про використання парковок допомагає в аналізі ефективності парковочних місць та сприяє більш обґрунтованому плануванню інфраструктури.

Окрім зазначених переваг, системи управління паркуванням на базі мобільних додатків та геолокаційних технологій мають ряд обмежень та потенційних проблем. Передусім слід звернути увагу на необхідність постійного підтримання та оновлення технологій, що є ключовим для забезпечення стабільної роботи системи. Це часто вимагає значних додаткових витрат на технічне обслуговування та модернізацію інфраструктури. Водночас, існує проблема забезпечення приватності даних, адже збір та аналіз інформації про використання парковок, включаючи геолокаційні дані та історію переміщень користувачів, може порушувати приватність. Також ефективність цих систем значно залежить від стабільності та наявності інтернет-з'єднання, що може бути обмежувальним фактором у регіонах з обмеженим доступом до інтернету. Додатково, системи можуть бути менш доступні для окремих груп користувачів, зокрема для тих, хто не використовує смартфони або не ознайомлений з сучасними технологіями, що обмежує їх загальну доступність та ефективність.

Технологія радіочастотної ідентифікації (RFID) знаходить своє застосування в управлінні паркувальними просторами, пропонуючи своє вихідне рішення для автоматизації та підвищення ефективності паркувальних систем. Вона включає в себе використання радіохвиль для ідентифікації та відстеження об'єктів, транспортних засобів, якщо стає центром реалізації паркових систем.

Основний принцип роботи RFID-базованих систем полягає в наступному: кожен транспортний засіб, що в'їжджає на парковку, обладнаний RFID-міткою (рис. 1.8), яка містить унікальний ідентифікаційний код. При наближенні автомобіля до паркувального майданчика, RFID-читачі, розміщені на в'їздах, розпізнають цю мітку.



Рисунок 1.8 – Зовнішній вигляд RFID-мітки

Після цього вони передають інформацію про мітку до центральної системи управління паркуванням, це може бути персональний комп'ютер або віддалений сервер, просту схему системи приведено на рисунку 1.9. Така система, у свою чергу, перевіряє отриману інформацію з наявною базою даних транспортних засобів. У разі успішної верифікації, система надає доступ автомобілю та призначає місце для паркування [15].

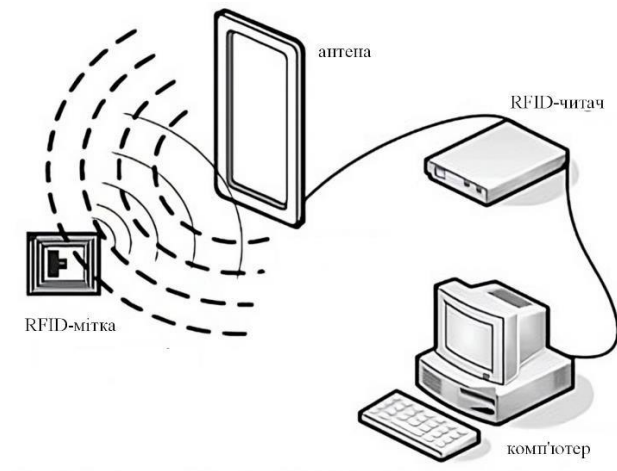


Рисунок 1.9 – Проста схема системи зчитування інформації на базі RFID

Протягом часу перебування на парковці, система постійно моніторить присутність автомобіля, забезпечуючи підвищення безпеки та контролю над несанкціонованим паркуванням та перевищенням відведеного часу на парковці. Система на базі RFID також дозволяє здійснювати безготівкові розрахунки, пов'язані з обліковим записом водія, що сприяє безперешкодному та швидкому процесу виїзду з парковки.

Головними перевагами систем управління паркуванням на базі RFID є їх висока ефективність, оскільки вони зменшують потребу в ручному втручанні. Ці системи також підвищують безпеку через постійний моніторинг та обмежений доступ. Вони забезпечують покращення задоволеності клієнтів, знижуючи час очікування та уникання заторів у паркових зонах.

Враховуючи специфічні особливості та обмеження систем на базі RFID, ці системи можуть бути не найкращим вибором для деяких контекстів, наприклад, для паркування клієнтів готелів. Основна проблема полягає в тому, що наявність RFID-міток на транспортних засобах не завжди є стандартом, особливо для тимчасових або випадкових відвідувачів, як це часто трапляється в готелях. Тому такі системи можуть бути більш підходящими для парковок в компаніях, де використовуються корпоративні автомобілі або де робітники регулярно використовують особистий транспорт.

У таких випадках RFID-базовані системи можуть бути вигідними завдяки їх ефективності та можливості забезпечувати підвищений контроль за доступом до парковки. Для робітників, що регулярно відвідують компанію, можна встановити постійні RFID-мітки, що значно спрощує процес в'їзду та виїзду з парковки. Це також сприяє безпеці, оскільки система може відстежувати тільки авторизовані транспортні засоби та запобігати несанкціонованому доступу.

Підсумовуючи, RFID-базовані системи управління паркуванням пропонують переваги в ефективності, безпеці та управлінні даними, але також стикаються з викликами, присутністю RFID-мітки, високими витратами, технічною надійністю.

1.2 Внесок комп'ютерного зору в системи паркування

На сьогоднішній день завдяки стрімкому розвитку нейромережевих алгоритмів і рішень в області комп'ютерного зору значно розширилися можливості управління паркувальними системами. Комп'ютерний зір, будучи ключовим компонентом інтелектуальних систем, стає центральним елементом в автоматизації та оптимізації процесів паркування. Цей напрямок технології пропонує нові можливості та рішення для сучасних міських інфраструктур, особливо в контексті управління паркувальними просторами.

Розвиток технологій комп'ютерного зору дав змогу реалізувати нові підходи до обробки зображень і відео в реальному часі, що є важливою складовою для сучасних паркувальних систем. Впровадження цих технологій сприяє поліпшенню управління та моніторингу паркувальних місць, а також значно підвищує рівень безпеки на парковках. Такі можливості, як ідентифікація вільних місць, розпізнавання номерних знаків і аналіз поведінки водіїв, стали можливими завдяки прогресу в галузі комп'ютерного зору. Це

підкреслює необхідність інтеграції комп'ютерного зору в паркувальні системи як засобу підвищення їхньої ефективності та надійності.

Ці переваги роблять комп'ютерний зір невід'ємною частиною сучасних систем управління паркуванням, сприяючи поліпшенню міської інфраструктури та підвищенню якості життя міських жителів. Ефективне управління паркувальними місцями з використанням комп'ютерного зору спрощує процес паркування і сприяє розвитку більш розумних і стійких міських середовищ.

Технології комп'ютерного зору, що використовуються в паркувальних системах, містять низку методів, таких як класифікація, кластеризація, виявлення об'єктів і трекінг об'єктів. Ці методи дають змогу автоматизувати й оптимізувати процеси управління паркуваннями. У рамках розвитку технологій комп'ютерного зору, що застосовуються в системах управління парковками, особливу увагу приділяють алгоритмам машинного навчання, що базуються на принципах роботи згорткових нейронних мереж (CNN). Ці мережі, виходячи з результатів досліджень [16], є ефективним інструментом для опрацювання та аналізу візуальних даних, забезпечуючи високу точність у задачах класифікації, кластеризації, виявлення і трекінгу об'єктів.

Згорткові нейронні мережі, завдяки своїй унікальній архітектурі, здатні моделювати просторові кореляції між сусідніми пікселями, що робить їх особливо придатними для вилучення складних ознак із зображень. Ця властивість CNN дає їм змогу ефективно обробляти й інтерпретувати візуальну інформацію, що критично важливо для систем управління паркуванням, де необхідно точно визначати стан паркувальних місць у реальному часі.

Однак, процес навчання CNN вимагає значних обчислювальних ресурсів і часу [17]. Незважаючи на це, після завершення навчання, етап прогнозування за допомогою навченої мережі відбувається швидко і з високою точністю. Це робить CNN кращим вибором для завдань, де потрібне швидке і точне опрацювання великих обсягів даних, як у випадку з системами управління парковками.

1.3 Постановка задачі

На основі проведеного аналізу а також детального розгляду актуальних досліджень у сфері систем управління паркуванням, представлених у статтях [14 - 17], було виявлено ключові обмеження систем, що переважають на ринку. Основний фокус поточних рішень припадає на міську інфраструктуру, при цьому вони демонструють значну залежність від спеціалізованого обладнання, зокрема, від камер, які не передбачають можливість інтеграції з уже наявними камерами в рамках комерційних об'єктів. Такий підхід неминуче призводить до збільшення капітальних витрат і обмежує гнучкість адаптації системи до конкретних умов експлуатації. Як розв'язання цих проблем пропонується розробка системи, що розв'язує цю проблему. Метою даної роботи є розробка інтелектуальної системи управління платним паркуванням з комп'ютерним зором, яка призначена для використання в різних комерційних і приватних структурах, зокрема в готелях, розважальних центрах, а також в інших установах, де керування паркувальними місцями відіграє економічне та операційне значення для таких структур. Особлива увага приділяється можливості інтеграції системи з уже наявними відеоспостережними системами та різноманітними сенсорами, що сприяє її економічній ефективності та функціональній гнучкості.

Для досягнення цієї мети було визначено такі завдання:

- аналіз і вибір технологій (дослідити наявні програмні та апаратні рішення для релізації системи, вибрати найбільш підходящі технології комп'ютерного зору, обґрунтувати їх застосування в контексті паркувальної системи);
- розроблення архітектури системи (створити концептуальну модель системи, включно з архітектурою обробки даних і відеоаналізу, визначити структуру баз даних і способи інтеграції системи із зовнішніми системами);

- планування інтерфейсів (розробити користувацькі інтерфейси для взаємодії як з кінцевими користувачами, так і з адміністраторами системи);
- провести розробку та тестування системи (відповідно до планування архітектури системи та інтерфейсів розробити систему до фази тестування, використовуючи набір даних для моделювання різних сценаріїв роботи системи).

2 ТЕХНОЛОГІЧНИЙ ОГЛЯД, ПЛАНУВАННЯ АРХІТЕКТУРИ ТА СТРАТЕГІЯ РОЗРОБКИ СИСТЕМИ

Наведено огляд у якому розглядаються технології комп'ютерного зору та алгоритми виявлення об'єктів, включаючи одноступеневі та двоступеневі детектори. Значна увага приділяється розробці інтелектуальної системи управління платним паркуванням, з акцентом на вибір інструментів і технологій. Представлено планування архітектури системи, орієнтованої на ефективну інтеграцію в наявні інфраструктури. Остання частина розділу зосереджується на стратегії розробки та реалізації системи, включаючи етапи попереднього проєктування, розробки та тестування.

2.1 Технологічний огляд і вибір інструментів

Технології комп'ютерного зору, алгоритми виявлення об'єктів на зображення, зазвичай поділяють на дві основні категорії (рисунок 2.1), залежно від кількості разів, коли одне й те саме зображення передають у мережу для

обробки. Цей поділ охоплює одноступеневі детектори і двоступеневі детектори [18].



Рисунок 2.1 – Два види алгоритмів детектування об'єктів на зображеннях

2.1.1 Огляд технологій комп'ютерного зору

Двоступеневі детектори ґрунтуються на принципах алгоритмів штучного інтелекту, є фундаментальним елементом у розробці сучасних систем, де можна використовувати данні з камер. У центрі цих технологій перебувають алгоритми штучних нейронних мереж, що забезпечують високу точність і ефективність у задачах обробки та аналізу зображень. Серед найбільш значущих досягнень у цій царині виділяються конволюційні нейронні мережі (CNN).

Основна структура CNN містить кілька ключових елементів (рис. 2.2), кожен з яких відіграє важливу роль у процесі навчання і функціонування мережі [19].

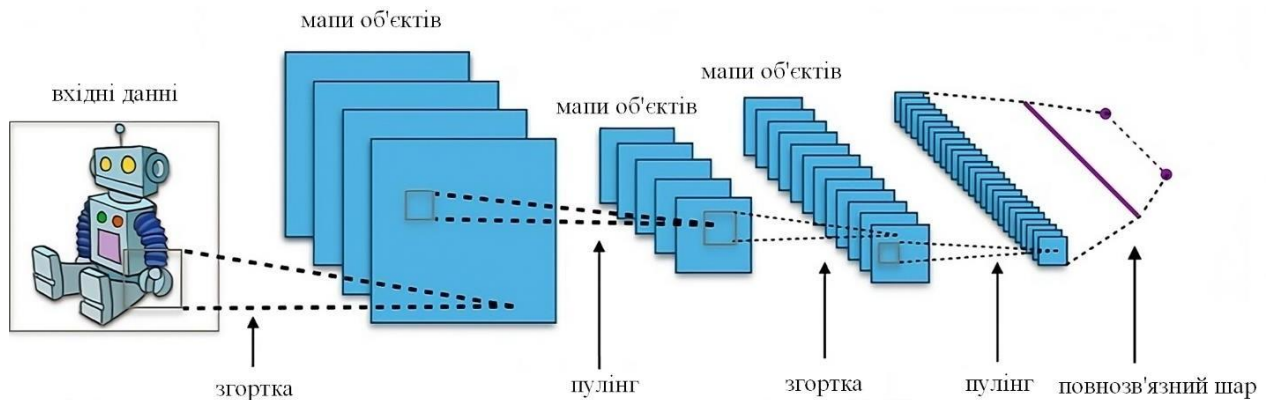


Рисунок 2.2 – Типова архітектура згорткової нейронної мережі

Шар згортки є основним блоком у CNN. У цьому шарі кожен канал зображення обробляється своїм фільтром, або ядром згортки. Ці фільтри аналізують попередній шар за фрагментами, виділяючи ключові характеристики зображення. Важливою особливістю згорткового шару є його здатність ефективно обробляти візуальну інформацію з відносно невеликою кількістю параметрів, що значно збільшує продуктивність обробки даних.

Шар активації слідує за кожним згортковим шаром і застосовує нелінійну функцію активації до результатів згортання. Найпоширенішою функцією активації є ReLU (Rectified Linear Unit) та її модифікації. Ці функції допомагають у збільшенні нелінійності в моделі, що навчається, що дає змогу мережі навчатися на складніших і різноманітніших даних [20].

Шар пулінгу або субдискретизації використовується для зменшення розмірності сформованих карт ознак. Це прискорює обчислення і робить мережу більш інваріантною до масштабу зображення. Існує два основних типи пулінгу: максимальний і середній, кожен з яких має свої переваги залежно від завдання.

Повнозв'язний шар зазвичай розташовується на виході згорткових шарів. У цих шарах кінцеві карти ознак подаються на вхід, і шари навчаються на виявленні нелінійних комбінацій високорівневих ознак. Це дає змогу мережі робити точніші та складніші передбачення на основі обробленої інформації.

Ключовим аспектом навчання CNN є процес зворотного поширення помилки (backpropagation).

Під час цього процесу мережа навчається на безлічі прикладів, наприклад, зображень, кожне з яких супроводжується відповідною міткою або відповіддю. Коли мережа робить помилку в передбаченні, інформація про цю помилку поширюється назад мережею. Це дає змогу коригувати ваги нейронів таким чином, щоб поліпшити продуктивність мережі на майбутніх даних. Такий підхід забезпечує поступове й ефективне навчання мережі, даючи їй змогу адаптуватися до складних і різноманітних завдань розпізнавання образів [21].

CNN є важливим інструментом у сфері комп'ютерного зору, який використовується в різних додатках або інших алгоритмах комп'ютерного зору завдяки своїм унікальним характеристикам. Однією з ключових переваг CNN є їхня здатність ефективно обробляти великі обсяги візуальних даних. Ці мережі здатні виділяти ключові ознаки зі складних візуальних візерунків, що робить їх особливо цінними для завдань, пов'язаних із розпізнаванням образів. Крім того, CNN демонструють високу точність у таких завданнях, завдяки глибокому навчанню та складній архітектурі. Гнучкість застосування також є значною перевагою CNN, оскільки вони можуть бути адаптовані до широкого спектру завдань у галузі комп'ютерного зору.

Проте існують і певні недоліки, пов'язані з використанням CNN. Вони вимагають значних обчислювальних ресурсів для навчання і функціонування, особливо під час роботи з великими наборами даних.

Розроблення та налаштування CNN можуть виявитися складними завданнями через велику кількість параметрів і шарів, що входять до їхньої структури. Ця обставина вимагає від розробників поглиблених знань у галузі машинного навчання і може вимагати додаткового часу на вивчення та експериментування для оптимізації моделі. Крім того, складність CNN може призвести до підвищення вартості проєкту, особливо у випадках, коли потрібен

запуск мережі на спеціалізованих платформах без використання готових інструментів.

Ще одним значущим аспектом, який слід враховувати під час роботи з CNN, є ризик перенавчання, особливо під час використання обмежених або зміщених наборів даних провокуючи модель занадто точно адаптується до навчальних даних, втрачаючи при цьому здатність до узагальнення на нових даних. Це може призвести до зниження ефективності моделі під час її застосування в реальних умовах, оскільки вона може некоректно інтерпретувати дані, відмінні від тих, на яких вона була навчена. Розв'язання цієї проблеми часто вимагає ретельного добору даних для навчання і застосування технік, що запобігають перенавчанню, таких як регуляризація або використання методів аугментації даних [22].

Для того щоб алгоритм CNN був більш здатний до локалізації та класифікації об'єктів, застосовується методика вилучення пропозицій за регресією, що є ключовим аспектом в архітектурі R-CNN (Region-based Convolutional Neural Networks). Ця архітектура відрізняється від традиційних підходів CNN, орієнтованих на аналіз зображення загалом, і пропонує більш точний механізм для виявлення та розпізнавання об'єктів всередині зображень.

В основі R-CNN лежить ідея, що ефективна локалізація об'єктів вимагає попереднього виділення потенційно значущих областей на зображенні. Це досягається завдяки використанню алгоритмів, що генерують пропозиції регіонів (рисунк 2.3). Такі алгоритми, як вибіркового пошук, створюють безліч кандидатних областей, в яких можуть перебувати об'єкти, що цікавлять. Кожну з цих областей потім аналізують окремо, що дає змогу більш детально вивчити і класифікувати об'єкти [22].

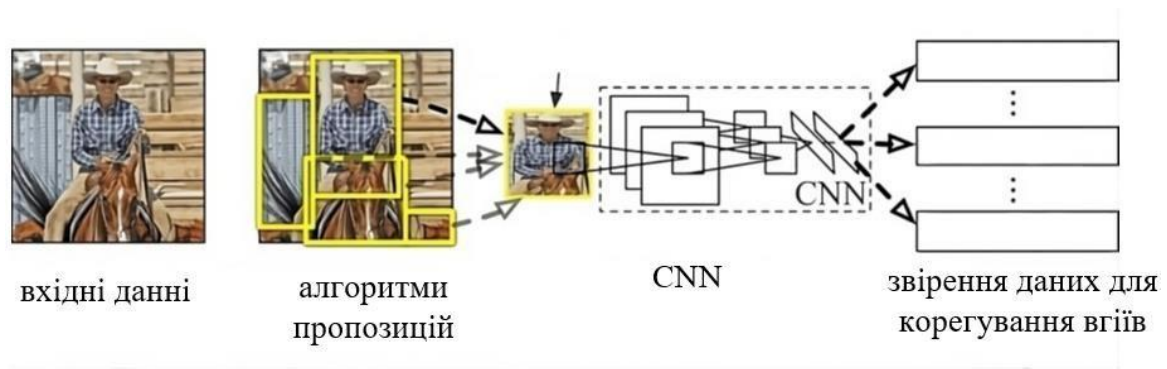


Рисунок 2.3 – Архітектура R-CNN

Після визначення областей, що цікавлять, R-CNN використовує CNN мережі для вилучення ознак з кожного запропонованого регіону. Ці ознаки стануть основою для подальшої класифікації об'єктів. У більшості випадків для класифікації використовують такі методи, як машини опорних векторів (SVM), які дають змогу точно визначити, до якого класу належить об'єкт у запропонованому регіоні. На додаток до цього, здійснюється процес регресії обмежувальних рамок, який сприяє точній локалізації об'єктів усередині зображення.

Необхідно зазначити, що обробка в R-CNN відбувається послідовно для кожного запропонованого регіону, що, з одного боку, підвищує точність обробки, але з іншого боку, збільшує час і ресурси, необхідні для обробки зображення. Крім того, R-CNN вимагає окремого навчання для різних частин моделі, що робить навчання складнішим і часозатратнішим порівняно з традиційними CNN.

Таким чином, R-CNN є значним кроком уперед у сфері комп'ютерного зору, пропонуючи поліпшену здатність до локалізації та класифікації об'єктів, особливо в складних візуальних сценаріях.

Ця модель знаходить широке застосування в різних галузях, де потрібна висока точність розпізнавання об'єктів. Однак, незважаючи на свої переваги, R-CNN, як і раніше, стикається з певними викликами, включно з високою ресурсомісткістю і складністю навчання, що стимулює пошук і розробку більш ефективних архітектур нейронних мереж, таких як Fast R-CNN і Faster R-CNN.

Ці поліпшення спрямовані на оптимізацію процесу оброблення зображень, зберігаючи при цьому високу точність розпізнавання і класифікації об'єктів.

Для усунення недоліків розглянутих мереж CNN і R-CNN, особливо щодо перенавчання і складності опрацювання великих обсягів даних, архітектура Faster R-CNN являє собою значний прогрес. Основна ідея цієї моделі полягає в об'єднанні мережі регіональних пропозицій (Region Proposal Network, RPN) з Fast R-CNN, тим самим створюючи більш ефективну і цілісну систему виявлення об'єктів.

Основні інновації Faster R-CNN містять у собі:

- мережа регіональних пропозицій (RPN): RPN автоматично генерує регіональні пропозиції високої якості, які потім використовуються Fast R-CNN для подальшого виявлення об'єктів. Це усуває необхідність у повільних і менш ефективних зовнішніх методах генерації регіонів, таких як вибірковий пошук;
- якорі: використання анкерів (або якорів) дає змогу RPN врахувати різні масштаби та співвідношення сторін об'єктів. Це сприяє точному виявленню об'єктів різних розмірів і форм, що особливо важливо в умовах керування паркуваннями, де автомобілі можуть сильно відрізнятися за розміром і розташуванням;
- оцінка об'єктності: RPN оцінює ймовірність того, що запропонований регіон містить об'єкт, що допомагає відфільтрувати менш значущі регіони та зменшити кількість помилкових спрацьовувань;
- спільне використання функцій між RPN і Fast R-CNN: знижує час навчання і підвищує ефективність, оскільки ті самі згортувальні обчислення використовуються як в RPN, так і в Fast R-CNN.

З точки зору реалізації, Faster R-CNN забезпечує більш швидку і точну обробку зображень порівняно з CNN.

Однак важливо зазначити, що, незважаючи на ці поліпшення, Faster R-CNN все ще може зіткнутися з проблемою тривалого часу навчання, особливо під час роботи з великими і різноманітними наборами даних. Це може бути пов'язано з кореляцією даних, оскільки всі приклади в міні-пакеті витягуються з

одного зображення. Таким чином, хоча Faster R-CNN значно покращує процес виявлення об'єктів, потрібен ретельний підбір даних і методів навчання для досягнення оптимальних результатів.

На додаток до згаданих вище інновацій, Faster R-CNN також долає обмеження своїх попередників завдяки використанню шару об'єднання ROI (Region of Interest Pooling), який уніфікує розміри векторів ознак, витягнутих із різних пропозицій регіонів. Це забезпечує більш ефективне і точне вилучення ознак, що критично важливо для систем паркування, де необхідно точно ідентифікувати і класифікувати різні об'єкти, такі як автомобілі різних типів і розмірів [23].

Метод навчання Faster R-CNN демонструє гнучкість: існує можливість почергового, наближеного спільного, і непримірного спільного навчання. Ці методи дають змогу гнучкіше підходити до навчання моделі, оптимізуючи її під конкретні вимоги та умови використання.

Проте, важливо усвідомлювати, що, незважаючи на ці переваги, Faster R-CNN може зіткнутися з проблемами під час роботи з дуже великими або складними наборами даних, що вимагає додаткових ресурсів і часу на навчання. Також, як і у випадку з будь-якою складною моделлю машинного навчання, існує ризик перенавчання, що може вплинути на здатність моделі генералізувати свої знання на нові, невідомі дані. Це особливо актуально в динамічному середовищі вуличних парковок, де умови та сценарії можуть швидко змінюватися.

Одноступеневий детектор You Only Look Once (YOLO) – це значний прогрес в області виявлення об'єктів, що використовує наскрізний нейромережовий підхід. На відміну від розглянутих алгоритмів виявлення об'єктів, які зазвичай перепрофілюють класифікатори або використовують мережі пропозицій регіонів, YOLO виконує виявлення шляхом одночасного прогнозування обмежувальних рамок і ймовірностей класів за один прохід через мережу.

такі як зміни в архітектурі CNN і стратегіях навчання, постійно підвищують ефективність виявлення, роблячи їх високоефективними для застосування в реальному часі [24].

Висока швидкість обробки даних і можливості виявлення в реальному часі роблять YOLO ідеальним рішенням для таких застосувань, як відеоспостереження, автономні транспортні засоби та інших сфер, де потрібне швидке і точне виявлення об'єктів. Незважаючи на свої сильні сторони, YOLO стикається з проблемами при виявленні невеликих об'єктів і в умовах великого скупчення людей, що є сферами постійних досліджень і розробок.

З розвитком комп'ютерного зору еволюція YOLO та подібних алгоритмів виявлення об'єктів продовжує відігравати ключову роль у розширенні меж можливого в виявленні та аналізі об'єктів у реальному часі.

Одноступінчатий детектор об'єктів (Single Shot Detector, SSD). SSD об'єднує переваги швидкості та ефективності, завдяки чому він знаходить широке застосування в різних галузях, включно із системами відеоспостереження та автономним водінням.

Архітектура мережі SSD базується на CNN, яка модифікована шляхом додавання додаткових шарів (рисунок 2.5). Ці шари, звані шарами передбачень (prediction layers), одночасно виконують завдання виявлення і класифікації об'єктів. Вони здатні обробляти зображення різних розмірів, забезпечуючи високу точність виявлення об'єктів на різних масштабах [25].

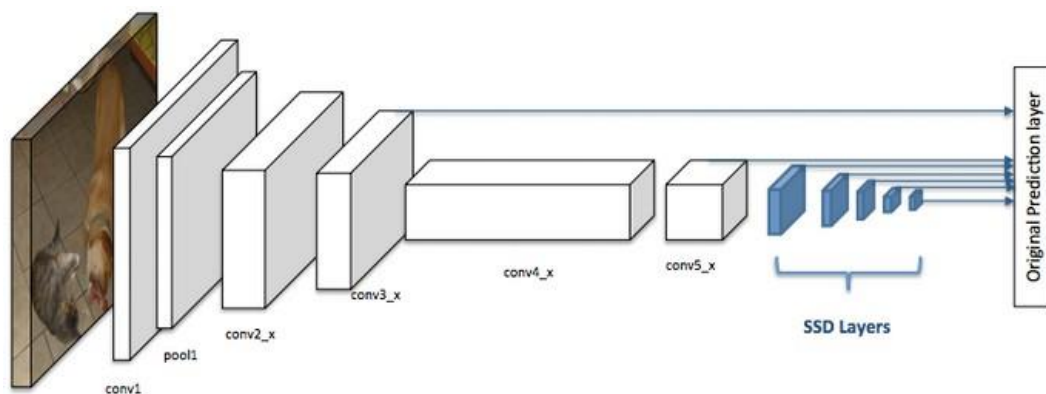


Рисунок 2.5 – Архітектура нейромережі SSD

Основна особливість SSD полягає у використанні декількох згорткових шарів для передбачення межових рамок і класифікації об'єктів. На відміну від двоетапних детекторів, таких як R-CNN, які спочатку генерують пропозиції областей і потім класифікують їх, SSD виконує виявлення об'єктів за один прохід, що значно прискорює процес.

SSD застосовує концепцію якірних рамок - фіксованих рамок різних масштабів і пропорцій, рівномірно розподілених по зображенню. На кожному рівні згорткової мережі SSD оцінює ймовірність наявності об'єкта в кожній якірній рамці та коригує її розміри для точного охоплення об'єкта.

У процесі навчання SSD мінімізує функцію втрат, що враховує як помилку локалізації (точність визначення меж об'єкта), так і помилку класифікації (правильність визначення класу об'єкта). Це подвійне цільове завдання сприяє точному визначенню як положення, так і класу об'єктів.

Однією з переваг SSD є його швидкість і ефективність, особливо порівняно з двоетапними методами. Проте, він може бути менш точним у визначенні дрібних об'єктів порівняно з деякими іншими алгоритмами, що застосовують методи збільшення зображення.

У науковій статті, присвяченій порівняльному аналізу нейромережевих алгоритмів: Faster R-CNN, YOLO v3 і SSD, представлено результати, що виявляють ключові характеристики та ефективність кожного методу. Дослідження детально оцінює алгоритми з точки зору їхнього застосування для виявлення сільськогосподарських теплиць на супутникових знімках високої роздільної здатності. Ключові критерії, за якими проводилося порівняння, включають точність виявлення (mAP), швидкість оброблення (FPS) і візуальну оцінку результатів.

YOLO v3 продемонстрував найвищу точність виявлення серед усіх розглянутих моделей (mAP), а також мав найвищу швидкість обробки (FPS). Це робить його кращим вибором для оперативного моніторингу та ефективного розпізнавання об'єктів на супутникових знімках різної роздільної здатності.

Модель виявилася найбільш придатною для завдань, де потрібне швидке і точне виявлення об'єктів.

SSD, своєю чергою, має значну перевагу у швидкості обробки, хоча трохи поступається YOLO v3 у точності (mAP). Більш висока швидкість обробки робить SSD придатним для завдань, де час обробки є критичним фактором. Незважаючи на це, SSD може бути менш точним у визначенні деяких об'єктів у порівнянні з більш повільними методами.

Faster R-CNN показав конкурентоспроможні результати по точності (mAP), близькі до результатів SSD, але мав нижчу швидкість обробки (FPS). Це обмежує його застосування в завданнях, що вимагають швидкої обробки даних. Однак, він чудово підходить для сценаріїв, де потрібна висока точність виявлення, особливо в умовах складного середовища за наявності дрібномасштабних об'єктів. Faster R-CNN рекомендується для ситуацій, де важлива не тільки швидкість, а й високий ступінь точності виявлення.

На основі даних, представлених у статті, можна зробити висновки про те, що кожен з алгоритмів найбільше підходить для певних типів завдань з обробки зображень. YOLO v3 є найкращим вибором для завдань, що вимагають швидкого і точного виявлення об'єктів, завдяки його високій швидкості та точності. SSD підходить для додатків, де першорядне значення має швидкість обробки, навіть за рахунок деякого зниження точності. Нарешті, Faster R-CNN рекомендується для сценаріїв, де найбільш важлива точність виявлення, особливо в складних умовах і за наявності дрібних об'єктів.

2.1.2 Вибір інструментів для розробки

Для розроблення інтелектуальної системи управління платним паркуванням з комп'ютерним зором, на основі всебічного аналізу типів систем і з огляду на поставлені вимоги у кінці першої частини дипломної роботи, було сформовано набір програмних інструментів і технологій. Метою цього вибору є створення системи, здатної інтегруватися з наявною інфраструктурою комерційних і приватних об'єктів, що має економічну ефективність і функціональну гнучкість.

Інструменти для розробки користувацького та адміністративного інтерфейсу:

- wxPython;
- PyTorch;
- OpenCV.

Використання wxPython для створення адміністративного інтерфейсу системи обґрунтовується його здатністю надавати нативний, високо функціональний графічний інтерфейс для широкого спектра операційних систем.

PyTorch і OpenCV ці бібліотеки обрані для реалізації функцій комп'ютерного зору і глибокого навчання PyTorch підтримує просунуті алгоритми глибокого навчання, включно з моделями виявлення об'єктів, як-от Faster R-CNN, YOLO і SSD, що дає змогу обробляти й аналізувати відеодані з паркування, включно з розпізнаванням автомобілів і визначенням їхнього місця розташування.

База даних для адміністративної програми - SQLite. Використання SQLite в адміністративній програмі підкреслює потребу в легковажній, але ефективній системі управління базами даних, здатній забезпечити швидкий доступ і обробку даних без необхідності складного налаштування.

Розробка серверної частини - Django. Фреймворк Django обрано для серверної розробки з огляду на його потужні можливості в галузі створення безпечних і масштабованих веб-додатків. Django надає інструменти для ефективної роботи з базами даних, а також інтегровану підтримку систем адміністрування.

Розробка мобільного додатка - Kivy. Для розроблення мобільного застосунку, призначеного для навігації водіїв, вибір Kivy зумовлений його здатністю створювати кросплатформні застосунки, що дає змогу забезпечити широку доступність застосунку на різних мобільних пристроях.

Вибір цих технологій та інструментів дає змогу реалізувати систему, що відповідає сучасним вимогам до управління паркувальними місцями,

забезпечуючи водночас гнучкість інтеграції та економічну ефективність. Використання Python як основної мови програмування підтримує широкий спектр технологій і забезпечує гладку інтеграцію різних компонентів системи, полегшуючи розробку і подальшу підтримку системи.

2.2 Планування архітектури системи

Для забезпечення основи в розробці інтелектуальної системи управління платним паркуванням із застосуванням комп'ютерного зору, використовується стратегічне планування загальної архітектури системи. Цей процес охоплює визначення ключових компонентів і механізмів їхньої взаємодії, що є значущим для створення концептуального плану, який відповідає як функціональним вимогам, так і загальним цілям проєкту. Розроблення архітектури на початку розробки передбачає чітке розмежування етапів розроблення, що сприяє більш цілеспрямованому та систематизованому процесу проєктування.

Камери, встановлені на парковці, є первинними джерелами даних у системі. Можливе застосування додаткових датчиків для збору інформації про стан паркувальних місць. Важливим завданням є обробка отриманих відеоданих та інформації про зайнятість паркувальних місць, що потребує подальшого аналізу та обробки.

Локальний комп'ютер або сервер, що обробляє дані з камер, використовує алгоритми комп'ютерного зору на базі OpenCV і PyTorch. Це дає змогу аналізувати стан паркувальних місць і ухвалювати рішення про їхню зайнятість.

Сервер, що працює на Django, виконує роль центру обробки та зберігання даних. Після первинного опрацювання даних локальним комп'ютером, цю інформацію передають на сервер для подальшого зберігання та управління в

базі даних SQLite. Таким чином, сервер стає ключовим елементом системи, централізуючи управління інформацією.

Інтеграція з інтернет-технологіями та хмарними сервісами дає змогу передавати дані від центру обробки до кінцевих користувачів. Це забезпечує додаткові можливості для зберігання даних і управління навантаженнями.

Мобільний застосунок для кінцевих користувачів надає інтерфейс для доступу до інформації про вільні паркувальні місця, їх бронювання та оплати. Додаток зв'язується із сервером через API, даючи змогу користувачам отримувати актуальну інформацію та надсилати запити.

Система також передбачає двосторонній обмін інформацією між локальним комп'ютером/сервером обробки та центром обробки і зберігання даних. Це забезпечує адаптивність системи до мінливих умов і дає змогу в реальному часі оновлювати параметри обробки даних.

Архітектура системи орієнтована на створення інтегрованої та масштабованої платформи, здатної забезпечити високу продуктивність і надійність. Ця структура полегшує інтеграцію системи в наявні інфраструктури та дає змогу легко масштабувати її відповідно до мінливих вимог і розширення функціоналу.

2.3 Стратегія розробки та реалізації

Для розробки проєкту дипломної роботи магістра розроблення інтелектуальної системи управління платним паркуванням з використанням комп'ютерного зору ключове значення має чітке планування стратегії розроблення та реалізації системи. Цей етап охоплює планування всіх аспектів проєкту, починаючи від попереднього проєктування і закінчуючи впровадженням системи.

Визначення етапів розроблення:

- попереднє проєктування (на основні визначених вимог до системи, проводиться розробка первинної специфікації і створюється початковий проєкт архітектури);
- розробка (здійснюється безпосереднє програмування і створення всіх компонентів системи, включно з розробкою адміністративного застосунку, серверної частини, мобільного застосунку, а також інтеграцією з базами даних);
- тестування (проводиться тестування ключових аспектів системи, включно з функціональними тестами та тестами продуктивності).

3 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ПАРКУВАННЯМ

3.1 Архітектура системи

Наведена на рисунку 3.1 архітектура інтелектуальної системи управління платним паркуванням демонструє багаторівневу структуру, у центрі якої перебуває оброблення та аналіз і зберігання даних, що надходять від камер і датчиків. Камери, встановлені на парковці або поблизу від неї, є основним джерелом інформації. Вони можуть відрізнятися за виробником, типом та інтерфейсом під'єднання, але зрештою всі вони розраховані на передачу даних на локальний комп'ютер, використовуючи різні способи під'єднання, включно з Ethernet, USB або Wi-Fi. Це забезпечує гнучкість системи щодо використовуваного обладнання і робить її сумісною з широким спектром пристроїв.

Крім камер, система також підтримує інтеграцію різних датчиків, які можуть бути під'єднані безпосередньо до системи через USB або через спеціальні перетворювачі сигналів SPI, I²C, UART в USB. Це розширює можливості системи зі збору даних, даючи змогу отримувати більш точну інформацію про стан паркувальних місць і умови навколишнього середовища.

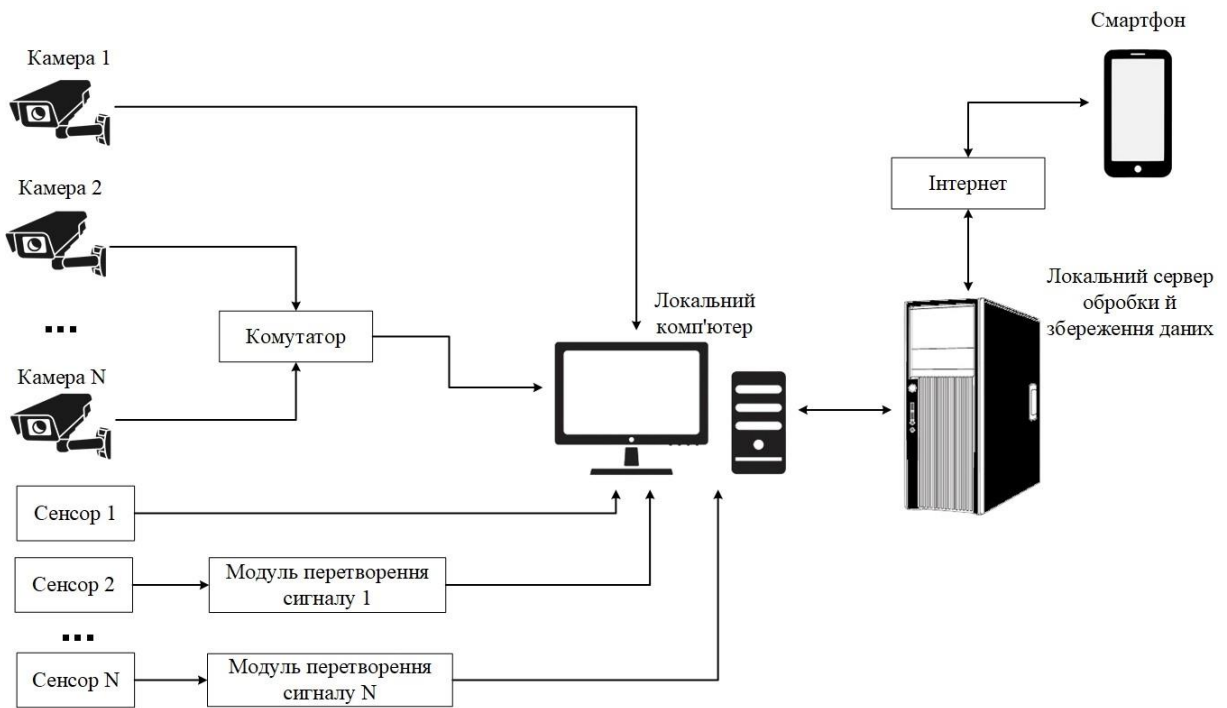


Рисунок 3.1 – Архітектура системи

Локальний комп'ютер обробляє надходженні дані за допомогою алгоритмів комп'ютерного зору і, залежно від режиму роботи, передає команди або дані на сервер. Сервер виконує функції як зберігання даних, так і їх подальшого опрацювання команд або даних. Він розташований у локальній мережі компанії, що забезпечує швидкий і безпечний обмін інформацією між комп'ютером і сервером.

Ключовим елементом системи є передача даних від сервера на смартфони користувачів через інтернет. Це дає змогу користувачам у реальному часі отримувати актуальну інформацію про доступні паркувальні місця, здійснювати бронювання та оплату паркування. Інтеграція з мобільними пристроями робить систему зручною і доступною для широкого кола користувачів.

Таким чином, архітектура системи являє собою комплексне рішення, яке забезпечує високий ступінь інтеграції, гнучкості та масштабованості системи, що є ключовим для її ефективного функціонування в різних умовах і середовищах.

3.2 Проєктування користувацького інтерфейсу

У розробці інтелектуальної системи управління платним паркуванням початковим етапом стало проєктування користувацького інтерфейсу. Для візуалізації концепції інтерфейсу використовувався інструмент дизайну Figma, який забезпечив можливість швидкого створення детальних макетів графічного інтерфейсу і прототипування. Цей підхід дав змогу точно визначити розташування ключових функціональних елементів кнопок та елементи керування і встановити візуальну ієрархію компонентів. Основною метою було створення базового інтерфейсу, який має логічну і зрозумілу структуру, що забезпечує зручність навігації і полегшує доступ до основних функцій системи.

3.2.1 Інтерфейс адміністративної програми

Специфікація інтерфейсу адміністративного додатка орієнтована на забезпечення ефективного моніторингу та управління системою управління платним паркуванням. Цей інтерфейс надає адміністратору комплексні інструменти для спостереження за станом паркувальних місць і регулювання параметрів системи. На рисунку 3.2 представлено макет основного вікна адміністративної програми, яке є центральним елементом управління.

Основне вікно програми, що слугує стартовим інтерфейсом для адміністратора, забезпечує можливість перегляду відеозображення з камер паркування. Адміністратор має можливість перемикатися між різними камерами зі списку, які доступні у системі. У верхній частині інтерфейсу розташована навігаційна панель, яка дає змогу переходити в різні режими адміністрування системи. Крім стандартного режиму моніторингу, передбачено додаткові режими, – це режим навчання і режим виведення статусу паркувальних місць.

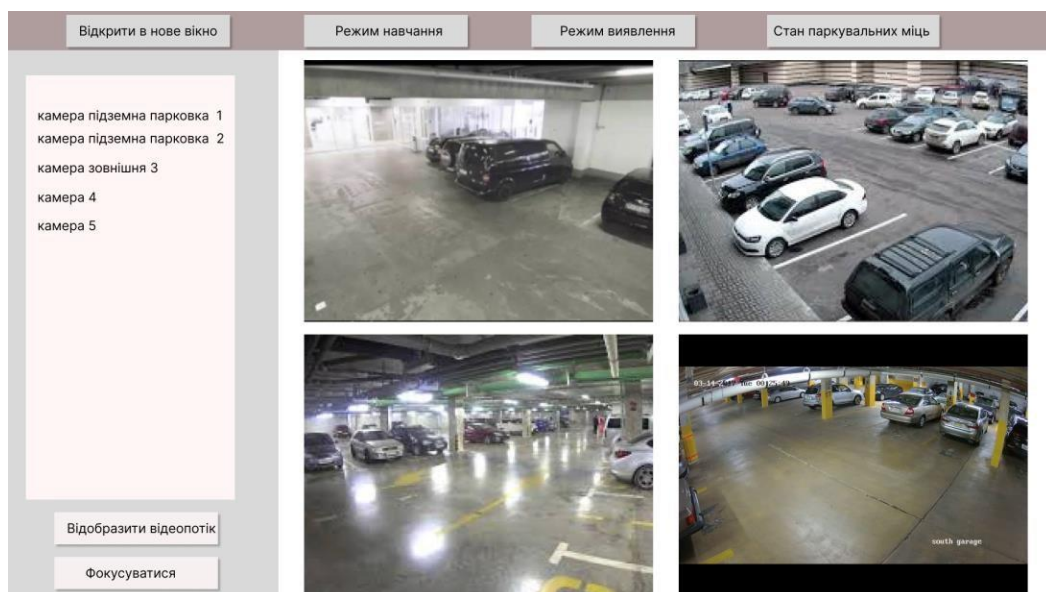


Рисунок 3.2 – Макет інтерфейсу режиму виявлення, головне вікно адміністративної програми

Макет інтерфейсу для режиму навчання, представлений на рисунку 3.3, дає адміністратору можливості керування системою навчання, збирання даних і перемикання між камерами. У цьому режимі адміністратор може налаштовувати та навчати модель для детектування вільних паркувальних місць для кожної окремої камери.

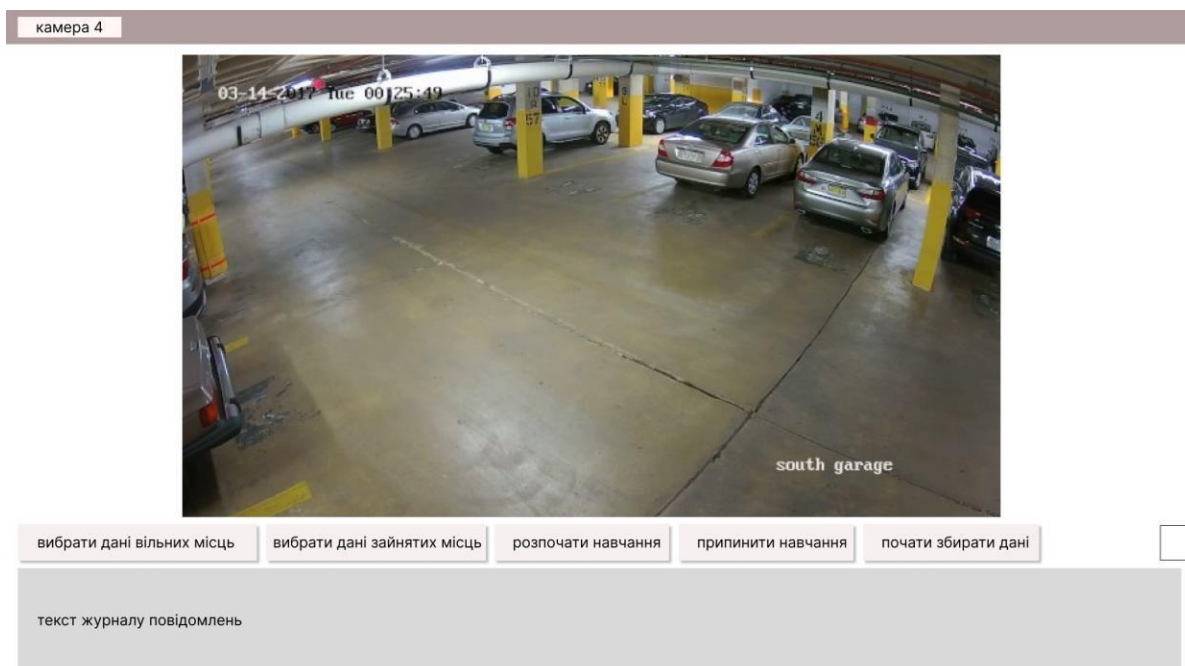


Рисунок 3.3 – Макет інтерфейсу режиму навчання

Режим стану паркувальних місць, макет якого представлений на рисунку 3.4, дає змогу адміністратору отримувати актуальну інформацію про заповненість паркувальних місць. Додатково цей режим містить у собі функціонал допоміжних інструментів для формування даних, необхідних для навчання моделей. У цьому контексті адміністратор може вручну виділяти зони акцентування на зображеннях з камер і зберігати ці дані, цей функціонал є важливим для процесу навчання алгоритмів комп'ютерного зору.

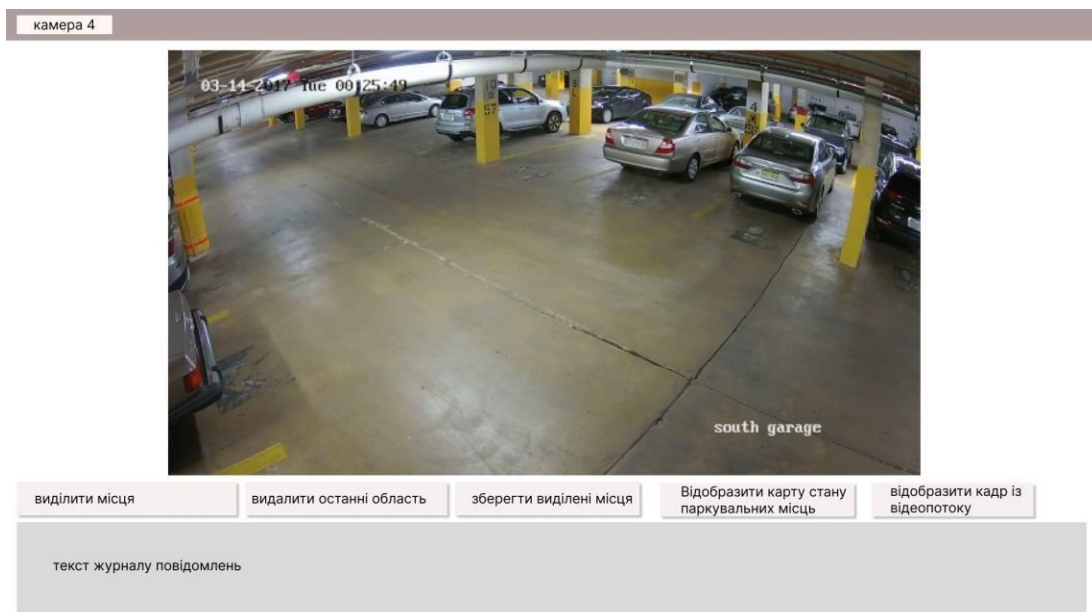


Рисунок 3.4 – Макет інтерфейсу стану паркувальних місць

3.2.2 Інтерфейс мобільного застосунка

Інтерфейс мобільного застосунку системи, що розробляється, орієнтований на забезпечення зручності та ефективності взаємодії кінцевих користувачів із функціоналом системи. Важливим аспектом є розробка інтуїтивно зрозумілого інтерфейсу, який здатний забезпечити швидкий доступ до ключових функцій додатка. На рисунку 3.5 представлено макет інтерфейсу мобільного застосунку, що демонструє базовий функціонал.

Основний екран застосунку являє собою меню, що містить список базових функцій: перегляд паркувальних місць, бронювання місця, оплата паркування, історія парковок і зворотний зв'язок.

Перший пункт зі списку – «Перегляд паркувальних місць» – слугує для інформування про доступні місця та їхній статус (зайнято/вільно), даючи змогу користувачам швидко орієнтуватися в умовах паркування та обирати відповідне місце відповідно до їхніх потреб.

Функція «Бронювання паркувальних місць» включає інтерактивний процес вибору місця на карті, а також зазначення часу і тривалості бронювання.

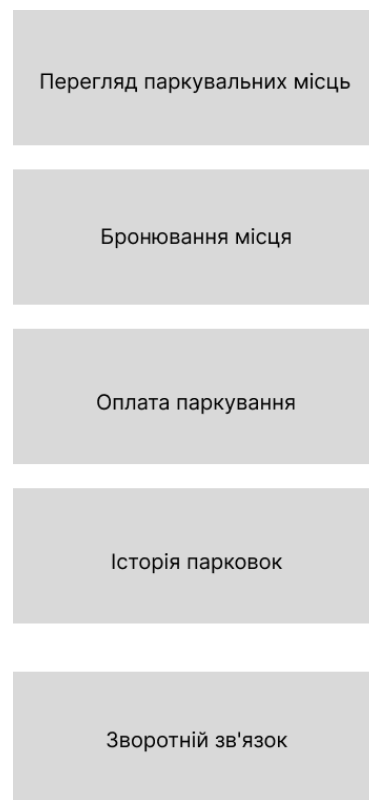


Рисунок 3.5 – Макет інтерфейсу головної сторінки мобільного застосунку

Для зручності оплати паркування в застосунку передбачена оплата безпосередньо через мобільний пристрій.

«Історія паркувань» надається у вигляді легко доступного запису про минулі паркувальні сесії користувача, включаючи інформацію про час, тривалість і вартість кожної сесії.

Розділ «Зворотній зв'язок і підтримка» доступний користувачам для повідомлень про проблеми або пропозицій щодо поліпшення сервісу.

3.3 Координація та взаємодія компонентів системи

При розробці інтелектуальної системи управління платним паркуванням ключового значення набуває ефективна координація та взаємодія між основними компонентами системи: сервером, адміністративним локальним комп'ютером і мобільним додатком. Основне завдання системи полягає у збиранні, обробці та розподілі інформації про стан паркувальних місць, а також у наданні користувацького інтерфейсу для управління та моніторингу цих процесів.

На рисунку 3.6 представлено загальний алгоритм функціонування серверної програми, що є невід'ємною частиною інтелектуальної системи управління платним паркуванням. Цей алгоритм підкреслює ключові етапи роботи сервера, забезпечуючи ефективну взаємодію з іншими компонентами системи.

Сервер починає свою роботу з ініціалізації та запуску, під час яких здійснюється перевірка на наявність можливих системних винятків, таких як відмова жорстких дисків тощо. У разі виявлення подібних винятків сервер записує відповідні звіти і припиняє свою роботу. У разі успішного проходження цієї стадії сервер переходить у режим очікування підключення з боку адміністративної програми.

Основний стан сервера – це очікування підключення. Щойно сервер виявляє підключення адміністративної програми, він переходить до наступного етапу – очікування даних або команд. Отримання команд від адміністративної програми ініціює процес їх обробки. Після обробки сервер надсилає відповідь назад на локальний комп'ютер, на якому запущено адміністративну програму. Якщо команди надходять не від адміністративної програми, а від мобільного застосунку, сервер виконує опрацювання даних, призначених для цього застосунку, і надсилає їх назад у мобільний застосунок.

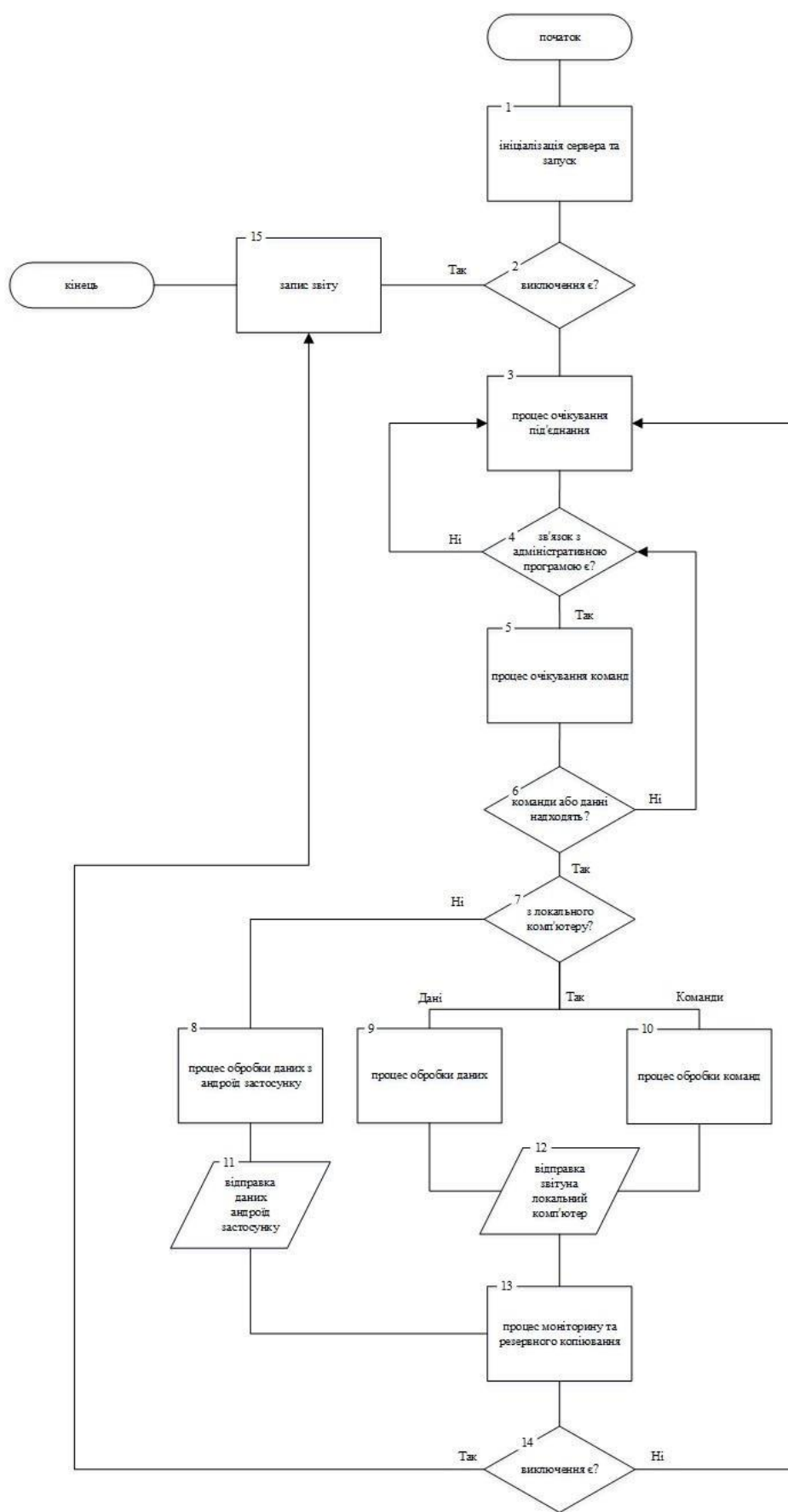


Рисунок 3.6 – Загальний алгоритм роботи серверної програми

Завершальним етапом роботи сервера є процес моніторингу та резервного копіювання. У разі виявлення винятків у роботі системи сервер фіксує відповідні звіти. За відсутності винятків сервер продовжує роботу, знову переходячи в режим очікування підключення. Цей циклічний процес забезпечує безперервне та стабільне функціонування системи.

На рисунку 3.7 представлено загальний алгоритм функціонування адміністративної програми. Цей елемент так само є важливим елементом системи, що розробляється, забезпечуючи взаємодію із сервером і управління даними паркувальних місць.

Адміністративна програма починає свою роботу з етапу ініціалізації та запуску. Проводячи перевірку на наявність системних винятків. У разі їх виявлення програма реєструє відповідні звіти про неполадки і приймає рішення про завершення своєї роботи. У разі якщо винятків немає, програма переходить до наступного етапу – завантаження навчених моделей.

Завантаження навчених моделей – важливий момент у роботі програми. У разі виникнення помилок на цьому етапі, програма також фіксує необхідну інформацію у звітах і завершує свою роботу. Успішне завантаження моделей дає змогу програмі перейти до спроби встановлення зв'язку з локальним сервером. Наявність стабільного зв'язку із сервером необхідна для функціонування деяких функцій програми.

У разі, якщо зв'язок із сервером не вдається встановити, адміністративна програма переходить до виконання процесів із локальним обліком даних. Це передбачає зчитування даних безпосередньо з камер і датчиків, а також подальше опрацювання та аналіз цієї інформації. Після локального збору та аналізу даних програма знову намагається встановити зв'язок із сервером.

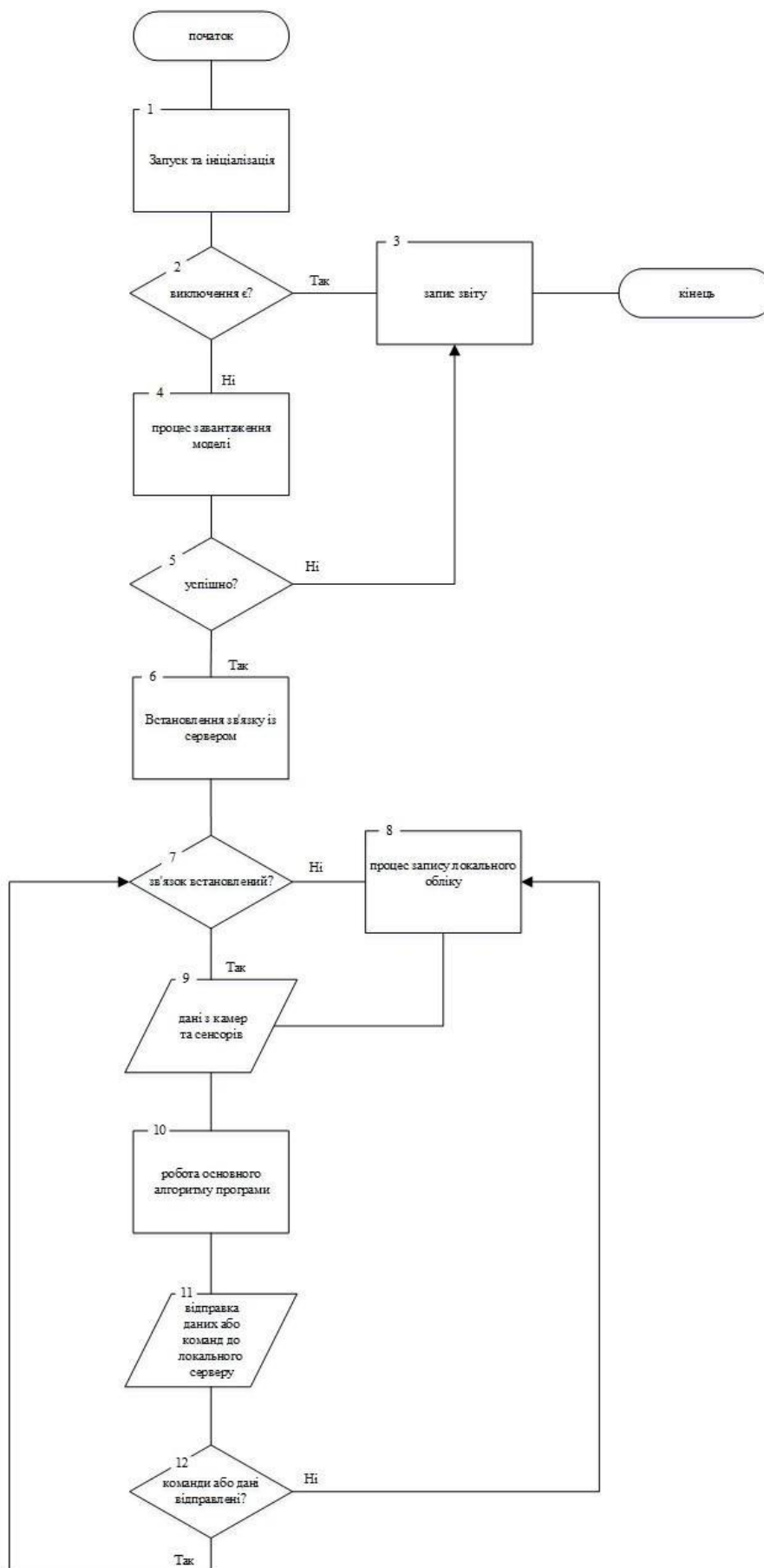


Рисунок 3.7 – Загальний алгоритм роботи адміністративної програми

Основна робота адміністративної програми полягає в надсиланні даних і команд на локальний сервер залежно від режиму її роботи. У разі успішного надсилання даних, програма повторно перевіряє наявність зв'язку із сервером. За відсутності зв'язку програма знову переходить до локального обліку даних та їх обробки. Це забезпечує безперервність роботи системи навіть за тимчасових збоїв у зв'язку із сервером.

Таким чином, адміністративна програма відіграє важливу роль у забезпеченні безперебійної роботи системи управління платним паркуванням, адаптуючись до різних умов експлуатації та забезпечуючи ефективну взаємодію із сервером та іншими компонентами системи.

4

РОЗРОБКА СИСТЕМИ

4.1 Розробка програми класифікації паркувальних місць

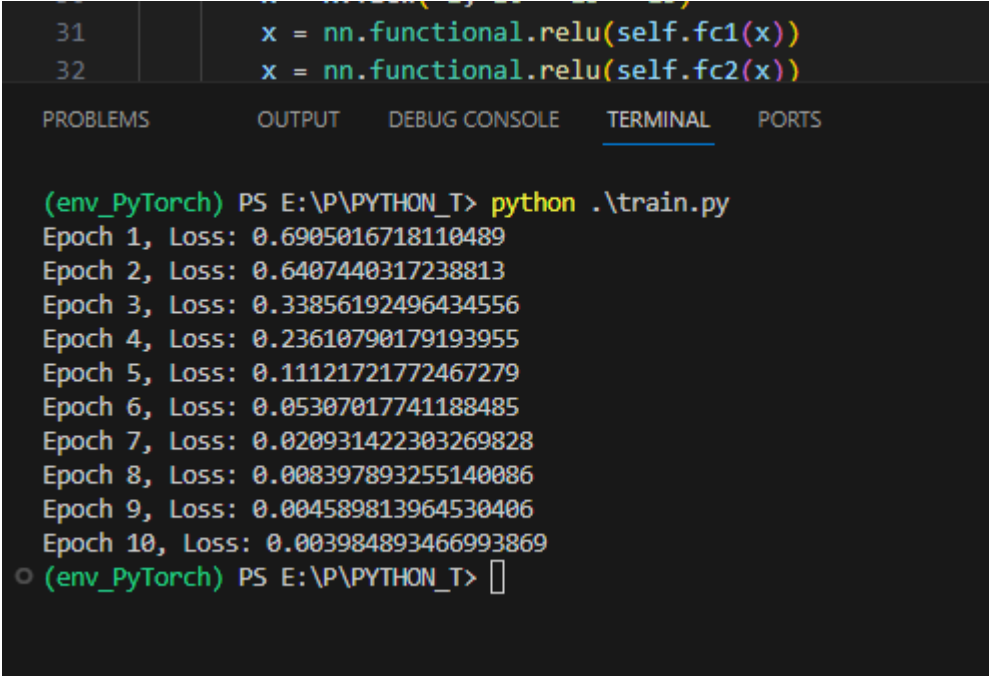
У процесі розробки програми для класифікації паркувальних місць була використана бібліотека PyTorch. Ця бібліотека надає великі інструменти для реалізації різних алгоритмів нейронних мереж у сфері комп'ютерного зору. Вона містить популярні DataNetworks, моделі архітектур і методи перетворення зображень.

Для навчання нейромережі використовували набір даних із проєкту «parking-space-counter» на GitHub, автором якого є користувач під ніком computervisioneng. Цей набір даних складається із серії зображень паркувальних місць, зайнятих і вільних, що стало основою для розроблення Python-скрипта, спрямованого на класифікацію паркувальних місць.

Скрипт базується на простій архітектурі згорткової нейронної мережі (CNN), що складається з декількох згорткових і повнозв'язних шарів. Оскільки дані являють собою зображення, немає необхідності у виділенні регіональних зон або опрацюванні даних у формі пропозицій про регіони (region proposals).

Робота скрипта відбувається так: зображення з навчального набору даних перетворюються і нормалізуються для підготовки до обробки нейромережею. Це передбачає зміну розміру зображень до 64x64 пікселів і їх конвертацію в тензори.

Модель CNN задається з використанням декількох шарів. Навчання моделі здійснюється із застосуванням оптимізатора SGD і функції втрат CrossEntropyLoss. Процес навчання триває 10 епох, протягом яких дані обробляються партіями (батчами), після чого відбувається зворотне поширення помилки для корекції ваг нейромережі (рис. 4.1).



```
31 x = nn.functional.relu(self.fc1(x))
32 x = nn.functional.relu(self.fc2(x))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(env_PyTorch) PS E:\P\PYTHON_T> python .\train.py
Epoch 1, Loss: 0.6905016718110489
Epoch 2, Loss: 0.6407440317238813
Epoch 3, Loss: 0.33856192496434556
Epoch 4, Loss: 0.23610790179193955
Epoch 5, Loss: 0.11121721772467279
Epoch 6, Loss: 0.05307017741188485
Epoch 7, Loss: 0.020931422303269828
Epoch 8, Loss: 0.008397893255140086
Epoch 9, Loss: 0.004589813964530406
Epoch 10, Loss: 0.003984893466993869
(env_PyTorch) PS E:\P\PYTHON_T> 
```

Рисунок 4.1 – Процес навчання нейромережі

Після завершення навчання стан моделі зберігається у файл для подальшого використання в адміністративній програмі. Скрипт функціонує на сервері та передає навчену модель через локальну мережу для роботи адміністративної програми, надаючи функціонал для аналізу та визначення стану паркувальних місць.

На додаток, було розроблено програму для тестування алгоритму класифікації. Ця програма використовує Python і бібліотеки, як-от OpenCV (cv2), PyTorch, torchvision, і PIL, для обробки й аналізу відео в реальному часі.

Програма завантажує попередньо навчену модель згорткової нейронної мережі з файлу, використовуючи PyTorch. Модель, названа SimpleCNN, проходить через процес оцінювання, щоб бути готовою до класифікації зображень.

Потім, використовуючи OpenCV, програма відкриває підготовлений відеопотік. Кожен кадр відео перетворюється у формат PIL для подальшої обробки. Застосовується серія перетворень, включно зі зміною розміру і конвертацією в тензор, що необхідно для сумісності з моделлю нейронної мережі.

Після обробки зображення модель передбачає клас кожного кадру (рис. 4.2). У цій програмі використовуються два класи: «Class A» місце парковки вільне і «Class B» місце парковки зайнято. Передбачений клас потім відображається на кадрі з використанням функцій OpenCV.

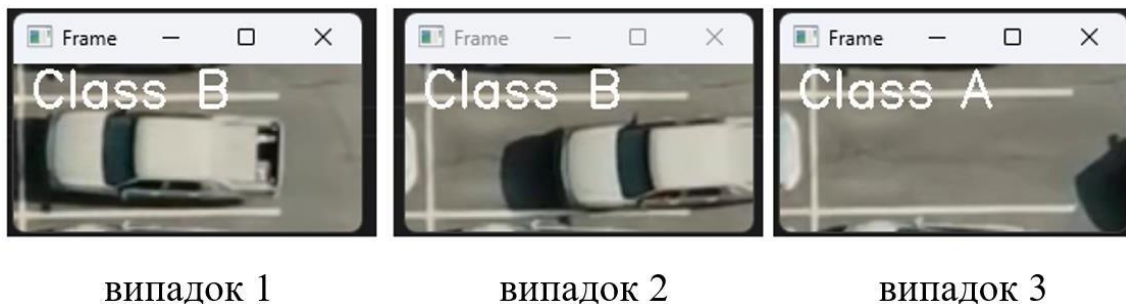


Рисунок 4.2 – Результат на класифікації стану паркувального місця

Цей підхід дав змогу перевірити працездатність і точність моделі нейронної мережі в умовах, наближених до реальних, і демонструє практичне застосування розробленої системи для класифікації паркувальних місць.

4.2 Розробка адміністративної програми

У додатку А представлено код адміністративної програми (лістинг А.1). Ця програма, виконана на основі бібліотеки wxPython, забезпечує гнучке і кросплатформенне середовище для візуалізації та управління паркувальними місцями.

Клас CameraPanel є основою для візуалізації відеопотоку з камер. У ньому використовується бібліотека OpenCV для захоплення та обробки відео, що дає змогу працювати із зображеннями з різних джерел. Кожна камера в програмі представлена індивідуально, що дає можливість вибору конкретного відеопотоку для моніторингу (рис. 4.3).

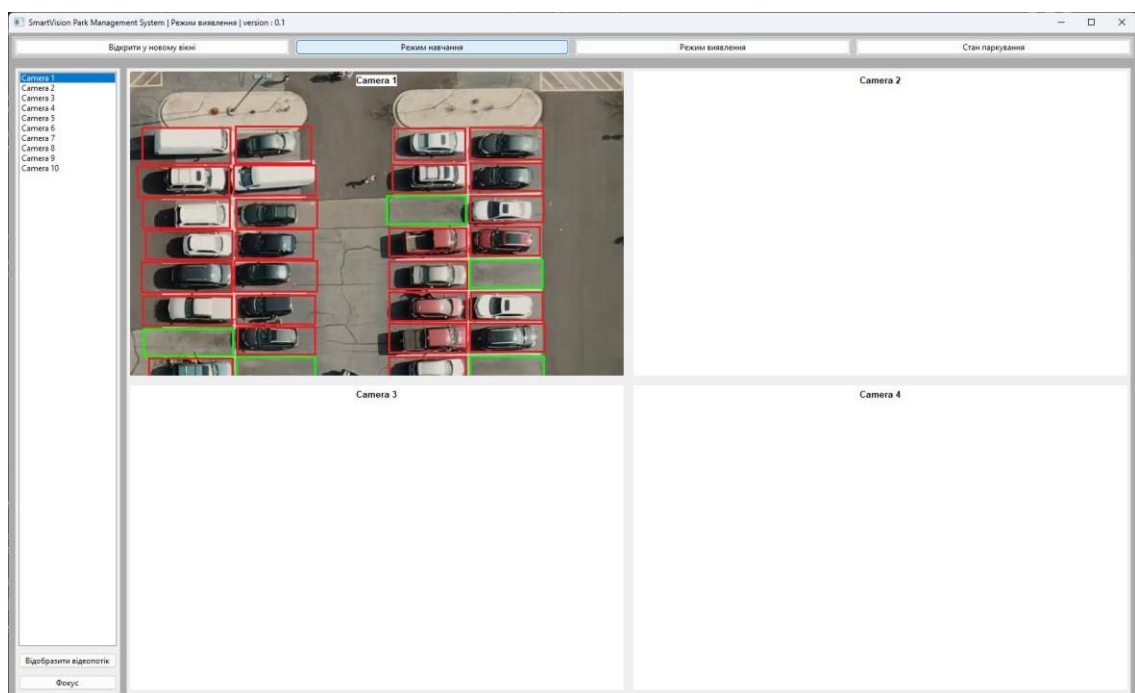


Рисунок 4.3 – Програмна реалізація головного вікна програми

Клас ModeButtonPanel відповідає за реалізацію користувацького інтерфейсу програми. У цьому класі реалізовано кнопки для перемикання між різними режимами роботи програми, такими як режим навчання, режим

виявлення та режим відображення стану паркування. Використання wxPython дає змогу створити інтуїтивно зрозумілий і функціональний інтерфейс.

Клас AdminApp являє собою основне вікно програми. Цей клас інтегрує всі компоненти інтерфейсу, включно з панеллю камер і панеллю керування режимами. Важливою особливістю є можливість вибору камер зі списку і відображення відеопотоку в реальному часі. Програма також має функціональність фокусування на певній камері, що спрощує моніторинг специфічних зон паркування.

Інтеграція із Зовнішніми Скриптами здійснюється через модуль subprocess. Це дає змогу адміністративній програмі взаємодіяти з іншими компонентами системи, як-от модулі навчання, виявлення та відображення статусу паркування. Така архітектура забезпечує гнучкість у розширенні функціоналу програми та спрощує процес інтеграції нових компонентів.

4.3 Розробка мобільного застосунку

У додатку А наведено код мобільного додатка (лістинг А.2), що є важливою складовою інтелектуальної системи управління платним паркуванням. Розроблення застосунку проводили з використанням фреймворка Kivu, що дає змогу створювати інтерактивні, мультимедійні та кросплатформні мобільні застосунки.

Програмний код розділений на кілька класів, кожен з яких являє собою окремий екран у застосунку. Основними екранами є: екран перегляду паркувальних місць, екран бронювання, екран оплати паркування, екран історії парковок і екран зворотного зв'язку згідно розробці макету була реалізована організація функціоналу та зручність навігації для користувача.

ВИСНОВКИ

У дипломній роботі магістра було здійснено комплексну розробку інтелектуальної системи управління платним паркуванням, яка інтегрує комп'ютерний зір для оптимізації управління паркувальними місцями. Проєкт охоплює аналіз, проєктування, розробку та тестування системи, з акцентом на гнучкість та інтеграцію з наявними інфраструктурними рішеннями.

У програмній частині роботи було розроблено ключові компоненти системи, включно з адміністративним програмним забезпеченням, мобільним додатком і алгоритмами оброблення даних, що демонструють можливості автоматизації та ефективності управління паркувальними місцями.

Наукова новизна дипломної роботи магістра полягає в розробці комплексного рішення, що дає змогу підвищити ефективність управління паркуванням за допомогою сучасних технологій комп'ютерного зору, а також у її гнучкій інтеграції з різними типами відеоспостережних систем і датчиків, забезпечуючи широкі можливості для реалізації в різних комерційних і приватних структурах.

Практична цінність розробленої системи полягає в її здатності інтегрувати вже наявні системи відеоспостереження і сенсори, що значно знижує капітальні витрати і підвищує ефективність управління паркувальними місцями. Система забезпечує точне виявлення зайнятості паркувальних місць і надає зручний інтерфейс для кінцевих користувачів через мобільний додаток. Її гнучка архітектура дає змогу легко масштабувати систему й адаптувати її до різних умов експлуатації.

Таким чином, розроблена система є важливим внеском у сферу інтелектуального управління парковками, демонструючи значні переваги застосування сучасних технологій комп'ютерного зору та мобільних рішень у розв'язанні актуальних завдань оптимізації міської інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smart parking solutions- top 5 benefits for the drivers. Conure. URL: <https://www.conurets.com/smart-parking-solutions-top-5-benefits-for-the-drivers/> (date of access: 03.01.2023).
2. Annansingh F. Smart parking: the cornerstone of a smart city. Examining the socio-technical impact of smart cities. 2021. P. 17. URL: <https://doi.org/10.4018/978-1-7998-5326-8.ch007> (date of access: 05.01.2023).
3. Smart Parking | Types of smart parking systems. Cleverciti. URL: <https://www.cleverciti.com/en/smart-parking> (date of access: 01.12.2023).
4. The history of smart parking. Cleverciti. URL: <https://www.cleverciti.com/en/resources/blog/history-of-smart-parking> (date of access: 02.01.2023).
5. Krishnamurthy C. K. B., Ngo N. S. The effects of smart-parking on transit and traffic: Evidence from SFpark. Journal of environmental economics and management. 2020. Vol. 99. P. 102273. URL: <https://doi.org/10.1016/j.jeem.2019.102273> (date of access: 06.01.2023).
6. Intelligent parking management systems using iot and machine learning techniques for real-time space availability estimation / R. Raman et al. 2023 international conference on sustainable communication networks and application (ICSCNA), Theni, India, 15–17 November 2023. 2023. URL: <https://doi.org/10.1109/icscna58489.2023.10370636> (date of access: 08.01.2023).
7. Lyu L., Fan N. Research and design of intelligent parking management system based on UML technology. 2022 IEEE conference on telecommunications, optics and computer science (TOCS), Dalian, China, 11–12 December 2022. 2022. URL: <https://doi.org/10.1109/tocs56154.2022.10016081> (date of access: 09.01.2023).
8. Fahim A., Hasan M., Chowdhury M. A. Smart parking systems: comprehensive review based on various aspects. Heliyon. 2021. Vol. 7, no. 5. P.

e07050. URL: <https://doi.org/10.1016/j.heliyon.2021.e07050> (date of access: 10.01.2023).

9. Gautam S. Parking sensors' advantages and drawbacks. Get My Parking Blog. URL: <https://blog.getmyparking.com/2018/12/17/parking-sensors-advantages-and-drawbacks/> (date of access: 11.01.2023).

10. Lee H., Chatterjee I., Cho G. Enhancing parking facility of container drayage in seaports: A study on integrating computer vision and AI. 2023 IEEE 6th international conference on knowledge innovation and invention (ICKII), Sapporo, Japan, 11–13 August 2023. 2023. URL: <https://doi.org/10.1109/ickii58656.2023.10332699> (date of access: 12.01.2023).

11. Wisenet TNF-9010 kamera. SecuriFocus. URL: <https://www.securifocus.com/hir/2021/10/26/wisenet-tnf-9010-kamera> (date of access: 13.01.2023).

12. NETWORK CAMERA user manual. 28 p. URL: <https://www.use-ip.co.uk/datasheets/tnf9010usermanual.pdf>.

13. Research on parking guidance system based on computer vision / Y. Liu et al. Journal of physics: conference series. 2023. Vol. 2425, no. 1. P. 012054. URL: <https://doi.org/10.1088/1742-6596/2425/1/012054> (date of access: 14.01.2023).

14. Розумна система паркування з використанням iot" (smart parking system using iot). Mokosmart. URL: <https://www.mokosmart.com/uk/smart-parking-system-using-iot/> (дата звернення: 15.01.2023).

15. RFID based automatic car parking system. International journal of advanced engineering, management and science (IJAEMS). 2016. Vol. 2, no. 4. P. 135.

16. Muhammad Z. A systematic review of machine-vision-based smart parking systems. Scientific journal of informatics. 2020. Vol. 7, no. 2. P. 2013.

17. Lee H., Chatterjee I., Cho G. A systematic review of computer vision and AI in parking space allocation in a seaport. Applied sciences. 2023. Vol. 13, no. 18. P. 10254. URL: <https://doi.org/10.3390/app131810254> (date of access: 19.01.2023).

18. Kundu R. OLO algorithm for object detection explained. V7 Labs. URL: <https://www.v7labs.com/blog/yolo-object-detection> (date of access: 18.01.2023).
19. Craig L. What is a convolutional neural network (CNN)?. TechTarget - SearchEnterpriseAI. URL: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network> (date of access: 19.01.2023).
20. Convolutional neural network: benefits, types, and applications. Datagen. URL: <https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/> (date of access: 20.01.2023).
21. Umer A. Understanding convolutional neural networks : a beginner's journey into the architecture. Medium, published in CodeX. URL: <https://medium.com/codex/understanding-convolutional-neural-networks-a-beginners-journey-into-the-architecture-aab30dface10> (date of access: 21.01.2023).
22. Everything about mask R-CNN: a beginner's guide. Viso.ai. URL: <https://viso.ai/deep-learning/mask-r-cnn/> (date of access: 22.01.2023).
23. Chen C., Wang S., Huang S. An improved faster RCNN-based weld ultrasonic atlas defect detection method. Measurement and control. 2022. P. 002029402210920. URL: <https://doi.org/10.1177/00202940221092030> (date of access: 23.01.2023).
24. Redmon J. You only look once: unified, real-time object detection. 10 p. URL: <https://arxiv.org/pdf/1506.02640.pdf> (date of access: 23.01.2023).
25. How single-shot detector (SSD) works?. ArcGIS API for Python - ArcGIS Developers. URL: <https://developers.arcgis.com/python/guide/how-ssd-works/> (date of access: 24.01.2023).

ДОДАТОК А

Лістинги програм

Лістинг А.1 – Management Smart Vision Park.py (Адміністративна програма)

```
import wx
import cv2
import threading
import subprocess

version = 0.1 # curent status version of program product

def detect_cameras():
    index = 0
    arr = []
    while True:
        cap = cv2.VideoCapture(index)
        if not cap.read()[0]:
            break
        else:
            arr.append(f"Камера {index}")
        cap.release()
        index += 1
    return arr

class CameraPanel(wx.Panel):
    def __init__(self, parent, id):
        super(CameraPanel, self).__init__(parent, id=id)
        self.SetBackgroundColour(wx.WHITE)
        self.camera_label = f"Camera {id}"
        self.label = wx.StaticText(self, label=self.camera_label)
        self.label.SetFont(wx.Font(10, wx.DEFAULT, wx.NORMAL,
wx.BOLD))
        self.capture = None
        self.bmp = None
        self.timer = wx.Timer(self)
        self.Bind(wx.EVT_PAINT, self.OnPaint)
        self.Bind(wx.EVT_TIMER, self.OnTimer, self.timer)
        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(self.label, 0, wx.ALL | wx.CENTER, 5)
        self.SetSizer(sizer)

    def LoadVideo(self, filepath):
        self.capture = cv2.VideoCapture(filepath)
        ret, frame = self.capture.read()
        if ret:
```

```

        self.bmp = wx.Bitmap.FromBuffer(frame.shape[1],
frame.shape[0], frame)
        self.timer.Start(int(1000./30)) # Frame rate

    def OnPaint(self, evt):
        if self.bmp:
            dc = wx.ClientDC(self)
            dc.DrawBitmap(self.bmp, 0, 0)

    def OnTimer(self, evt):
        ret, frame = self.capture.read()
        if ret:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            self.bmp.CopyFromBuffer(frame)
            self.Refresh(False)

class ModeButtonPanel(wx.Panel):
    def __init__(self, parent):
        super(ModeButtonPanel, self).__init__(parent)
        self.SetBackgroundColour(wx.Colour(200, 200, 200))

        self.new_window_button = wx.Button(self, label="Відкрити у
новому вікні")
        self.training_mode_button = wx.Button(self, label="Режим
навчання")
        self.detection_mode_button = wx.Button(self, label="Режим
виявлення")
        self.parking_status_button = wx.Button(self, label="Стан
паркування")

        sizer = wx.BoxSizer(wx.HORIZONTAL)
        sizer.Add(self.new_window_button, 1, wx.EXPAND | wx.ALL,
5)
        sizer.Add(self.training_mode_button, 1, wx.EXPAND |
wx.ALL, 5)
        sizer.Add(self.detection_mode_button, 1, wx.EXPAND |
wx.ALL, 5)
        sizer.Add(self.parking_status_button, 1, wx.EXPAND |
wx.ALL, 5)

        self.SetSizer(sizer)
        # Прив'язка подій до кнопок
        self.new_window_button.Bind(wx.EVT_BUTTON,
self.OnNewWindow)
        self.training_mode_button.Bind(wx.EVT_BUTTON,
self.OnTrainingMode)
        self.detection_mode_button.Bind(wx.EVT_BUTTON,
self.OnDetectionMode)
        self.parking_status_button.Bind(wx.EVT_BUTTON,
self.OnParkingStatus)

    def OnNewWindow(self, event):
        subprocess.Popen(["python", "main.py"])

```

```

def OnTrainingMode(self, event):
    subprocess.Popen(["python", "traningMod.py.py"])

def OnDetectionMode(self, event):
    subprocess.Popen(["python", "dtProcess.py"])

def OnParkingStatus(self, event):
    subprocess.Popen(["python", "ParkingStat.py.py"])

class AdminApp(wx.Frame):
    def __init__(self, parent, title):
        super(AdminApp, self).__init__(parent, title=title,
size=(1000, 600))
        self.selected_camera_index = None
        self.camera_panels = []
        self.is_focused = False
        self.InitUI()
        self.Centre()
        self.Show()

    def InitUI(self):
        main_sizer = wx.BoxSizer(wx.VERTICAL)

        mode_button_panel = ModeButtonPanel(self)
        main_sizer.Add(mode_button_panel, 0, wx.EXPAND | wx.ALL,
5)

        camera_control_sizer = wx.BoxSizer(wx.HORIZONTAL)

        control_panel = wx.Panel(self)
        control_sizer = wx.BoxSizer(wx.VERTICAL)
        self.camera_listbox = wx.ListBox(control_panel,
choices=[f"Camera {i+1}" for i in range(10)], style=wx.LB_SINGLE)
        show_camera_button = wx.Button(control_panel,
label="Відобразити відеопотік")
        control_sizer.Add(self.camera_listbox, 1, wx.EXPAND |
wx.ALL, 5)
        control_sizer.Add(show_camera_button, 0, wx.EXPAND |
wx.ALL, 5)
        control_panel.SetSizer(control_sizer)
        camera_control_sizer.Add(control_panel, 0, wx.EXPAND |
wx.ALL, 5)

        video_panel = wx.Panel(self)
        video_sizer = wx.GridSizer(2, 2, 5, 5)
        for i in range(4):
            camera_panel = CameraPanel(video_panel, id=i+1)
            self.camera_panels.append(camera_panel)
            video_sizer.Add(camera_panel, 1, wx.EXPAND | wx.ALL,
5)

        video_panel.SetSizer(video_sizer)

```

```

        camera_control_sizer.Add(video_panel, 1, wx.EXPAND |
wx.ALL, 5)

        main_sizer.Add(camera_control_sizer, 1, wx.EXPAND |
wx.ALL, 5)
        self.SetSizer(main_sizer)

        self.camera_listbox.Bind(wx.EVT_LISTBOX,
self.OnCameraSelect)
        show_camera_button.Bind(wx.EVT_BUTTON, self.OnShowCamera)

        focus_button = wx.Button(control_panel, label="Фокус")
        control_sizer.Add(focus_button, 0, wx.EXPAND | wx.ALL, 5)
        focus_button.Bind(wx.EVT_BUTTON, self.OnFocusCamera)

    def OnFocusCamera(self, event):
        if not self.is_focused:

            for i in range(1, 4):
                self.camera_panels[i].Hide()

self.camera_panels[0].SetSize(self.GetClientSize())
        self.is_focused = True
    else:

        for i in range(4):
            self.camera_panels[i].Show()
            self.camera_panels[i].SetSize(self.GetClientSize())
// (2, 2))
        self.is_focused = False
        self.Layout()

    def OnCameraSelect(self, event):
        self.selected_camera_index = event.GetSelection()

    def OnShowCamera(self, event):
        if self.selected_camera_index is not None:
            selected_camera_label =
self.camera_listbox.GetString(self.selected_camera_index)
            target_camera_panel =
self.camera_panels[self.selected_camera_index % 4]

target_camera_panel.label.SetLabel(selected_camera_label)
            target_camera_panel.Refresh()

if __name__ == '__main__':
    app = wx.App()
    frame = AdminApp(None, title=f'SmartVision Park Management
System | Режим виявлення | version : {version}')

frame.camera_panels[0].LoadVideo("E:\\P\\PYTHON_T\\input_data\\car
Park.mp4")

```

```
app.MainLoop()
```

Лістинг А.2 – MovieApp.py (програма мобільного застосунку)

```
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.uix.image import Image

class ParkingScreen(Screen):
    def switch_to(self, screen_name):
        self.manager.current = screen_name

class ParkingInfoScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(Label(text='Доступні паркувальні
місця'))

        # Загрузка и отображение изображения
        parking_image = Image(source='data/startus.png')
        layout.add_widget(parking_image)

        back_button = Button(text='Назад')
        back_button.bind(on_press=lambda x:
self.switch_to('main'))
        layout.add_widget(back_button)
        self.add_widget(layout)

class BookingScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(Label(text='Бронювання місця:'))
        back_button = Button(text='Назад')
        back_button.bind(on_press=lambda x:
self.switch_to('main'))
        layout.add_widget(back_button)
        self.add_widget(layout)

class PaymentScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(Label(text='Оплата парковки:'))
        back_button = Button(text='Назад')
        back_button.bind(on_press=lambda x:
self.switch_to('main'))
        layout.add_widget(back_button)
        self.add_widget(layout)
```

```

class HistoryScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(Label(text='Історія парковок'))
        back_button = Button(text='Назад')
        back_button.bind(on_press=lambda x:
self.switch_to('main'))
        layout.add_widget(back_button)
        self.add_widget(layout)

class FeedbackScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical')
        layout.add_widget(Label(text='Зворотній зв'язок:'))
        back_button = Button(text='Назад')
        back_button.bind(on_press=lambda x:
self.switch_to('main'))
        layout.add_widget(back_button)
        self.add_widget(layout)

class MainScreen(ParkingScreen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation='vertical', padding=10,
spacing=10)
        layout.add_widget(Button(text='Перегляд паркувальних
місць', on_press=lambda x: self.switch_to('parking_info')))
        layout.add_widget(Button(text='Бронювання місця',
on_press=lambda x: self.switch_to('booking')))
        layout.add_widget(Button(text='Оплата паркування',
on_press=lambda x: self.switch_to('payment')))
        layout.add_widget(Button(text='Історія парковок',
on_press=lambda x: self.switch_to('history')))
        layout.add_widget(Button(text='Обратная связь',
on_press=lambda x: self.switch_to('feedback')))
        self.add_widget(layout)

class ParkingApp(App):
    def build(self):
        sm = ScreenManager()
        sm.add_widget(MainScreen(name='main'))
        sm.add_widget(ParkingInfoScreen(name='parking_info'))
        sm.add_widget(BookingScreen(name='booking'))
        sm.add_widget(PaymentScreen(name='payment'))
        sm.add_widget(HistoryScreen(name='history'))
        sm.add_widget(FeedbackScreen(name='feedback'))
        return sm

if __name__ == '__main__':
    ParkingApp().run()

```