ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка системи паркування з визначенням вільного місця за допомогою OpenCV»

зі спеціальності *122 комп'ютерні науки*
(код, найменування спеціальності)

освітньо-професійної програми *Штучний інтелект*
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Владислав БОВКУН
(підпис)

Виконав: здобувач вищої освіти групи ШІД-41
Владислав БОВКУН

Керівник Тетяна КИСІЛЬ
ст.викладач

Рецензент _____
Науковий ступінь, вчене звання, *ім'я, ПРІЗВИЩЕ,*

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту
_____
«____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бовкун Владислав Валерійович

1. Тема кваліфікаційної роботи: Розробка системи паркування з визначенням вільного місця за допомогою OpenCV
керівник кваліфікаційної роботи ст.викладач Кисіль Т.М ,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36
2. Строк подання кваліфікаційної роботи «31» травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи:
3.1 Науково-технічна література з питань, пов'язаних із застосуванням аналізу даних
3.2 Практичний досвід аналізу даних
3.3 Концепція побудови веб-додатків
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1 Тема дипломної роботи

4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження

4.3 Результат дослідження розробки підсистеми для аналізу даних

4.4 Результат дослідження фреймворків для веб-розробки

4.5 Результати дослідження та опис реалізації програми

4.6 Апробація результатів дослідження

4.7 Висновки

5. Перелік ілюстративного матеріалу: *презентація*

5.1 слайд 1

5.2 слайд 2

5.3 слайд 3 Постановка задачі

5.4 слайд 4 Мета розробки

5.5 слайд 5 Рішення

5.6 слайд 6 Методи

5.7 слайд 7 Висновки

6. Дата видачі завдання «27» лютого 2024 р. № 36

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів Кваліфікаційної роботи	Строк виконання Етапів роботи	Примітки
1	Підбір науково-технічної літератури	27.02.-29.02.24	
2	Аналіз та дослідження існуючих аналогів	01.03-05.03.24	
3	Дослідження програмних засобів	06.03-26.03.24	
4	Розробка системи автопарковки	27.03-29.03.24	
5	Проєктування інтерфейсів системи для адміністратора та користувача	01.04-09.05.24	
6	Тестування проєктованої системи автопарковки	10.05-12.05.24	
7	Оцінка якості роботи проєктованої системи	13.05-14.05.24	
8	Розробка демонстраційних матеріалів	14.05-15.05.24	
9	Попередній захист роботи	16.05.24	
10	Здача роботи	31.05.24	

Здобувач(ка) вищої освіти _____ Владислав БОВКУН

Керівник кваліфікаційної роботи _____ Тетяна КИСІЛЬ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 16 рис., 21 джерел.

Мета роботи - підвищення ефективності пошуку вільних місць для паркування за допомогою алгоритмів машинного навчання.

Об'єкт дослідження – процес роботи системи виявлення вільних місць на парковці.

Предмет дослідження – встановлення наявності вільних паркувальних місць за допомогою методів машинного навчання.

Короткий зміст - У першому розділі аналізується поточний стан програмного забезпечення, пов'язаного з пошуком вільних паркувальних місць. Вивчаються існуючі рішення та підходи, відзначаються їх переваги та недоліки, а також визначається напрямок для подальших досліджень.

У другому розділі ми оглядаємо та аналізуємо існуючі методи та алгоритми вирішення задачі пошуку вільних паркувальних місць. Їх переваги, обмеження та можливості розглянуто з урахуванням сучасних вимог і технічних можливостей.

У третьому розділі проводиться моделювання системи для ефективного пошуку вільних паркувальних місць на основі результатів аналізу попередніх розділів. Визначаються вимоги до системи, проектуються алгоритми та структура програмного забезпечення для оптимального вирішення поставленої проблеми.

КЛЮЧОВІ СЛОВА: OpenCV, Python, автопаркровка, розпізнавання, згорткова нейрона мережа, RCNN, штучний інтелект, нейрона мережа.

ЗМІСТ

ВСТУП……..

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІСНУЮЧИХ РІШЕНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……...11

1.1 Обґрунтування актуальності дослідження……..11

1.2 Аналіз машинного навчання та його методів……..12

1.3 Аналіз методів глибокого навчання……..14

1.3.1 Огляд методів комп'ютерного зору……15

1.3.2 Визначення та ідентифікація предметів на зображенні……17

1.4 Опис дослідницького завдання……...20

1.5 Висновки до розділу 1……21

2 ІСНУЮЧИХ МЕТОДІВ І АЛГОРИТМІВ ВИРІШЕННЯ ЗАДАЧІ ПОШУКУ ВІЛЬНИХ ПАРКУВАЛЬНИХ МІСЦЬ……22

2.1 Існуючі методи керування автопарковками……...22

2.2 Аналіз алгоритму HOG……26

2.3 Висновки до розділу 2……29

3 МОДЕЛЮВАННЯ СИСТЕМИ ПО ЗНАХОДЖЕНЮ ПАРКУВАЛЬНОГО МІСЦЯ ЯКЕ Є ВІЛЬНИМ……...31

3.1 Програмне забезпечення для виявлення вільних парковочних місць……31

3.2 Вимоги до проектованої системи……31

3.3 Функціональні можливості проектованої системи……32

3.4 Висновки до розділу 3……44

ВИСНОВКИ……

ПЕРЕЛІК ПОСИЛАНЬ……

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ……

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ……

Отформатировано: русский

ВСТУП

У сучасному світі існує проблема збору та обробки даних. Обсяги інформації швидко зростають, тому важливо вибрати оптимальні методи для її опрацювання. Це питання розглядається в галузях інтелектуального аналізу даних та машинного навчання.

Пошук вільного паркувального місця є однією з головних проблем у сучасних містах. Зростання кількості автомобілів створило дисбаланс між попитом і пропозицією паркувальних місць.

Ця проблема впливає на такі аспекти, як витрати пального, забруднення довкілля, затори на дорогах та втрачений час, і навіть спричиняє аварії через те, що водії більше уваги приділяють пошуку вільних паркувальних місць, а не спостереженню за іншими водіями та пішоходами. Тому система пошуку вільних паркувальних місць може вирішити ці проблеми, надаючи інформацію про наявність вільних місць на парковці.

Сьогодні існують різні методи вирішення цієї задачі, наприклад, використання інфрачервоних сенсорів. Проте, такі методи або недостатньо надійні, або вимагають великих витрат на встановлення та обслуговування. Завдяки розвитку машинного навчання та комп'ютерного зору цю проблему можна вирішити. Це рішення охоплює багато паркувальних місць з мінімальною кількістю камер, що підвищує надійність системи та зменшує витрати на встановлення та обслуговування. Однак, поточна проблема є складною задачею комп'ютерного зору через різні зміни освітлення, перспективи та різні точки огляду камери. Щоб подолати ці перешкоди, необхідно звернутися до алгоритмів машинного навчання, які можуть в реальному часі визначати статус кожного паркувального місця, а також підраховувати кількість вільних місць.

Різноманітність методів машинного навчання та алгоритмів створює технічну проблему для пошуку найкращої моделі для вирішення цієї проблеми за допомогою камер, які використовують алгоритми машинного навчання та комп’ютерного зору для пошуку доступних місць для паркування.

Система, яка може ідентифікувати вільні місця для паркування, дуже бажана, оскільки вона економить час на пошуки, витрати на паливо та зусилля водія. Мої дослідження зосереджені на пошуку вільних паркувальних місць за допомогою машинного навчання та комп'ютерного зору. Також така система може підраховувати загальну кількість вільних паркувальних місць, вказувати їх точне розташування та визначати статус кожного паркувального місця.

У першому розділі аналізується поточний стан програмного забезпечення, пов'язаного з пошуком вільних паркувальних місць. Вивчаються існуючі рішення та підходи, відзначаються їх переваги та недоліки, а також визначається напрямок для подальших досліджень.

У другому розділі ми оглядаємо та аналізуємо існуючі методи та алгоритми вирішення задачі пошуку вільних паркувальних місць. Їх переваги, обмеження та можливості розглянуто з урахуванням сучасних вимог і технічних можливостей.

У третьому розділі проводиться моделювання системи для ефективного пошуку вільних паркувальних місць на основі результатів аналізу попередніх розділів. Визначаються вимоги до системи, проектуються алгоритми та структура програмного забезпечення для оптимального вирішення поставленої проблеми.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІСНУЮЧИХ РІШЕНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Обґрунтування актуальності дослідження

Пошук вільних паркувальних місць є однією з ключових проблем у великих мегаполісах світу. Згідно з дослідженнями IBM, близько 40% міського трафіку складається з автомобілів, водії яких шукають місця для паркування, витрачаючи на це в середньому 8-9 хвилин свого часу. Це спричиняє перевантаження доріг у години пік.

Це питання можна розв'язати, якби водії мали доступ до інформації про зайнятість паркувальних місць заздалегідь через мобільний пристрій або інший засіб.

Сьогодні розробляються системи моніторингу паркувальних місць, які допомагають водіям зекономити час і зусилля під час пошуку вільного місця. Ці системи пропонують різні технологічні підходи до вирішення цієї проблеми. Більшість систем використовують датчики для фіксації в'їзду та виїзду автомобіля, але вони не можуть визначити точне місце вільного паркувального місця на стоянці. Крім того, ці системи не можуть виявити, коли одне паркувальне місце зайнято декількома транспортними засобами, наприклад, великогабаритними чи неправильно припаркованими автомобілями.

Існуючі підходи або є надто дорогими через витрати на встановлення та обслуговування датчиків, або не забезпечують достовірності інформації. Тому система, яка автоматично визначає вільні паркувальні місця, є дуже бажаною, оскільки вона економить час, кошти та зусилля водія.

Ця система дозволить отримувати інформацію про доступні паркувальні місця в режимі реального часу, допомагаючи водіям планувати маршрут і знаходити вільні місця для паркування.

Застосування систем пошуку вільних паркувальних місць допоможе зменшити викиди вуглекислого газу та скоротити час пошуку місця.

Існують різні системи відстеження зайнятих паркувальних місць, які можна розділити на три основні категорії.

- системи на основі датчиків руху на землі;
- системи на основі радарів;
- системи на основі обробки зображень або відеопотоків.

Система на основі наземних датчиків руху підраховує кількість транспортних засобів, що в'їжджають і паркуються. Такі системи оснащені датчиками на в'їзді та виїзді з паркінгу. Ви можете вказати загальну кількість паркувальних місць і кількість доступних місць, але не можете відобразити точне розташування паркувальних місць. Радарна система контролює завантаженість кожного паркувального місця за допомогою ультразвукових датчиків, встановлених на кожному паркувальному місці. Однак основною проблемою цих методів є висока вартість встановлення та обслуговування датчика. У порівнянні з іншими системами, системи, засновані на обробці відеопотоків з відеокамер, більш економічні. Ви можете контролювати та виявляти вільні та зайняті місця. Це також зменшує потребу в додатковій інфраструктурі. Крім того, такі системи можуть визначати точне розташування вільних паркувальних місць, тим самим економлячи час водіїв при пошуку вільних паркувальних місць на паркінгах.

1.2 Аналіз машинного навчання та його методів

Машинне навчання дозволяє системам автоматично навчатися та вдосконалюватись на основі досвіду без необхідності явного програмування. Основна мета — створити програми, які можуть отримувати доступ до даних і використовувати їх для самонавчання. Процес навчання починається з аналізу даних. Аналіз даних шукає закономірності та характеристики в наборі інформації, щоб приймати рішення та прогнозувати на основі нових даних. Чим ефективніший алгоритм навчання ви виберете, тим більше даних він обробляє, а отже, точніші прогнози та рішення системи. Основна мета полягає в тому, щоб дозволити комп’ютерам навчатися автоматично з мінімальним втручанням людини та адаптувати свою поведінку відповідно без будь-якого явного програмування. Процес навчання моделі автоматизований і вдосконалюється на основі досвіду системи протягом усього процесу навчання. Моделі машинного навчання

створюються на основі попередньо відібраних даних за допомогою різних алгоритмів. Вибір алгоритму залежить від типу даних і виду діяльності, яку потрібно автоматизувати.

Машинне навчання широко використовується на практиці, але воно сильно відрізняється від традиційного програмування. Основна відмінність полягає в тому, що в програмуванні вхідні дані попередньо вводяться у відповідним чином написану програму для отримання вихідних даних. У машинному навчанні вхідні та вихідні дані вже доступні на етапі навчання, що дозволяє системі самостійно розробляти програми. Загалом розробка програми машинного навчання складається з чотирьох основних кроків. Процес починається з відбору та підготовки навчальних даних. Після підготовки даних вибирається алгоритм для виконання на наборі навчальних даних.

Алгоритм — це серія статичних кроків обробки, вибір яких залежить від типу та обсягу навчальних даних і типу розв'язуваної задачі.[12] Навчання алгоритму — це ітераційний процес, у якому алгоритм порівнює змінні з отриманими результатами, коригує ваги для отримання точніших результатів і повторно запускає змінні. Після навчання точний алгоритм перетворюється на модель машинного навчання. Останнім кроком є використання моделі з новими даними. Важливо зазначити, що машинне навчання використовує алгоритми для створення систем, які навчаються на власному досвіді. Алгоритми машинного навчання часто поділяють на керовані (кероване навчання) і неконтрольовані (неконтрольоване навчання). Контрольовані алгоритми навчаються на підготовлених даних, де відомі вхідні та вихідні дані. Алгоритм навчається шляхом порівняння фактичних і правильних результатів для виявлення помилок і подальшої модифікації моделі. Таке навчання використовується в програмах, які вимагають відомих даних і ймовірностей майбутніх подій. Неконтрольоване навчання використовується для даних, де немає «правильної відповіді». Алгоритм самостійно шукає кореляції в даних і розпізнає корисні функції, не вимагаючи втручання людини в процес навчання. Їхня здатність розпізнавати схожість і

відмінності в інформації робить неконтрольовані алгоритми ідеальним вибором для дослідницького аналізу даних і розпізнавання зображень.

1.3 Аналіз методів глибокого навчання

Глибоке навчання — це розділ машинного навчання (усі методи глибокого навчання є частиною машинного навчання, але не всі методи машинного навчання є глибоким навчанням). Машинне навчання залучає експертів із предметних питань для визначення найбільш релевантних функцій. Навпаки, глибоке навчання автоматично виявляє функції й не потребує знання домену. Алгоритми глибокого навчання засновані на штучних нейронних мережах, які імітують спосіб навчання людського мозку. Це означає, що навчання моделям глибокого навчання займає значно більше часу, ніж машинне навчання, яке займає лише секунди або години. Однак для тестування ситуація зворотна. Крім того, на відміну від глибокого навчання, машинне навчання не вимагає дорогого апаратного забезпечення високого класу.

Глибоке навчання демонструє переваги під час роботи з великими обсягами даних, коли немає розуміння домену для незалежного аналізу функцій, або в таких складних завданнях, як виявлення автомобілів на зображеннях або пошук вільних місць для паркування. Моделі глибокого навчання використовують переваги великої кількості даних, пропускаючи їх через кілька рівнів обчислення та застосовуючи вагові коефіцієнти та зміщення на кожному рівні, щоб постійно коригувати та покращувати результати. Моделі глибокого навчання зазвичай класифікуються як контрольоване та неконтрольоване навчання. Типи моделей глибокого навчання, такі як згорткові нейронні мережі (CNN) [10] і рекурентні нейронні мережі (RNN) , сприяють прогресу в таких сферах, як комп’ютерне бачення та обробка природної мови.

Нейронна мережа складається з шарів вузлів, схожих на нейрони в мозку людини. Кожен шар з'єднаний із сусіднім шаром. Глибина мережі визначається кількістю її шарів. У штучній нейронній мережі сигнал проходить через вузли, і кожному вузлу присвоюється вага. Чим більша вага вузла, тим більше він впливає на наступний рівень. Останній рівень підсумовує зважені вхідні дані та створює

вихідні дані. Такі системи обробляють великі обсяги даних і тому потребують потужного обладнання. Навіть з таким обладнанням навчання може зайняти досить багато часу. Оскільки системи глибокого навчання вимагають великих обсягів даних для отримання точних результатів, інформація надається у формі великих наборів даних. Під час обробки даних штучні нейронні мережі можуть класифікувати дані на основі результатів серії запитань «правда/неправда», які вимагають дуже складних математичних розрахунків. Згодом програма самонавчається і збільшує вірогідність правильних відповідей.

1.3.1 Огляд методів комп'ютерного зору

Комп'ютерне зір — це наукова галузь, яка займається отриманням інформації з цифрових зображень (рис.1.1). Інформація на основі зображень варіюється від розпізнавання об'єктів і просторових вимірювань для навігації до додатків доповненої реальності. Інше визначення комп'ютерного зору – це сфера штучного інтелекту, яка вчить комп'ютери інтерпретувати та розуміти об'єкти на зображеннях. Використовуючи зображення та відео з камер і моделей глибокого навчання, машини можуть точно ідентифікувати та класифікувати об'єкти та реагувати на те, що вони «бачать». Комп'ютерний зір створює алгоритми, які можуть зрозуміти зміст зображення та застосувати його для різних цілей. Комп'ютерне зір є поєднанням різних дисциплін. Це можна вважати частиною інформатики, а розробка алгоритмів комп'ютерного зору вимагає знання алгоритмів і теорії машинного навчання. Комп'ютерне бачення має переваги перед іншими галузями зі спільними характеристиками, такими як обробка зображень і комп'ютерне бачення. Основні відмінності:

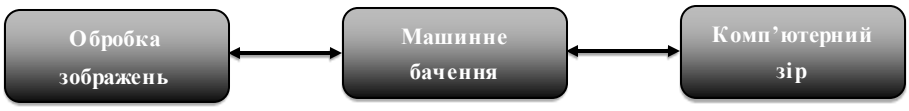


Рис. 1.1 Галузі комп'ютерного зору

Виявлення об'єктів є важливою темою в машинному навчанні та комп'ютерному зорі та передбачає пошук і обробку об'єктів на зображеннях, таких

як люди, тварини та транспортні засоби. Навколо виявленого об'єкта визначається контур, щоб ідентифікувати положення та рух об'єкта в певній сцені. Щоб повністю зрозуміти зображення, ми повинні зосередитися не тільки на класифікації зображення, але й на точному визначенні розташування та категорії об'єктів на зображенні відповідно до поставленого завдання. Це надає важливу інформацію для розуміння значення зображень і відео та актуальне для багатьох програм, таких як класифікація зображень, розпізнавання та автономне водіння. Досягнення нейронних мереж і відповідних систем навчання покращили алгоритми нейронних мереж і вплинули на методи виявлення цілей. Однак значні зміни положення, затінення та умов освітлення ускладнюють виявлення цілі, особливо в задачах локалізації цілі. Моделі виявлення цілей глибокого навчання зазвичай складаються з двох частин. Кодер приймає зображення як вхідні дані та обробляє його через кілька шарів, щоб отримати статистику для виявлення об'єктів і маркування.

Наприклад, якщо зображення містить два автомобілі, але модель призначена для виявлення лише одного об'єкта, інший об'єкт не буде обрамлено. Однак, якщо кількість об'єктів, виявлених на кожному зображенні, відома заздалегідь, модель на основі регресії може бути хорошим вибором. Це поширює регресійний підхід на регіональні мережі постачання. Ця частина моделі пропонує області зображення, де можуть бути розташовані об'єкти. Пікселі в цих регіонах обробляються підмережею класифікації для визначення міток (або викидів). Перевага цього методу полягає в тому, що він забезпечує більш точну та гнучку модель, яка може включати будь-яку кількість граничних областей. Однак додаткова точність залежить від ефективності розрахунку. Визначення завдання розпізнавання об'єктів передбачає визначення розташування об'єкта на зображенні та категорії, до якої цей об'єкт належить. Таким чином, традиційні моделі виявлення об'єктів складаються з трьох етапів: вибір діапазону інформації , виділення ознак і класифікація. Сканування всього зображення за допомогою багатомасштабного ковзного вікна є природним вибором для виявлення. Незважаючи на те, що цей підхід може виявити всі можливі розташування, він дорогий з точки зору обчислень і може призвести до надмірної кількості вікон. Вилучення ознак об'єкта

зосереджено на вилученні візуальних ознак, які ідентифікують різні об’єкти та забезпечують надійне семантичне представлення.

Типовим прикладом є HOG.[3] Однак через різноманітність зовнішнього вигляду, освітлення та фону створення надійних функціональних описів може бути складним. Крім того, потрібен класифікатор, щоб відрізнити цільовий об’єкт від інших категорій. Впровадження регіонів підтримки CNN (R-CNN) принесло значні переваги глибоким нейронним мережам. Він має глибшу архітектуру та може вивчати більш складні функції. Ви також можете дізнаватися інформацію про об’єкти без ручної розробки функціональності. З моменту появи R-CNN було запропоновано багато вдосконалених моделей, таких як Fast R-CNN, Mask R-CNN і YOLO, які забезпечують різні покращення продуктивності виявлення об’єктів.

1.3.2 Визначення та ідентифікація предметів на зображенні

Виявлення об’єктів є важливою темою в машинному навчанні та комп’ютерному зорі та передбачає пошук Обробка зображень спрямована на перетворення необроблених зображень для виконання певних операцій. Зазвичай її метою є покращення зображення або підготовка його для конкретного завдання. У комп'ютерному зорі метою є опис і тлумачення зображень.[2] Наприклад, операції зменшення шуму, підвищення контрасту або поворот зображень, які є типовими для обробки зображень, можуть проводитися на рівні пікселів і не вимагають складного розуміння зображення, що дозволяє отримати певну інформацію про те, що в ньому відбувається.

Комп'ютерний зір використовується для виконання певних дій, зазвичай у виробничих умовах. У хімічній промисловості системи машинного зору можуть сприяти виробництву продуктів, перевіряючи контейнери на предмет їх чистоти, порожнечі та відповідності стандартам. Комп'ютерний зір також вирішує складні завдання, наприклад, розпізнавання автомобілів (для цього використовуються фільтри, такі як HOG) або детальний аналіз зображень, що дозволяє здійснювати візуальний пошук, схожий на Google Images, або застосовувати біометричні методи ідентифікації.

Для ефективної роботи комп'ютерного зору необхідно велика кількість даних. Він аналізує дані знову і знову, поки не розпізнає відмінності та не навчиться розпізнавати зображення. Наприклад, щоб навчити комп'ютер розпізнавати автомобільні шини, необхідно надати величезну кількість зображень шин та пов'язаних об'єктів, щоб навчити машину розрізняти шину.

Дві основні технології, що використовуються в цьому процесі, це тип машинного навчання, відомий як глибоке навчання, та згорткова нейронна мережа (CNN). У машинному навчанні використовуються алгоритмічні моделі, які дозволяють комп'ютеру навчатися на контексті візуальних даних. Якщо в модель подається достатньо даних, комп'ютер починає "дивитися" на дані та вчитися розрізняти одне зображення від іншого.[4] Алгоритми дозволяють машині вчитися самостійно, а не програмуватись для розпізнавання зображень.

CNN допомагає моделі глибокого навчання "бачити", розбиваючи зображення на пікселі, які позначаються тегами або мітками. Мережа використовує ці мітки для виконання згорток (математична операція між двома функціями для отримання третьої) та робить прогнози щодо побаченого. Нейронна мережа перевіряє точність своїх прогнозів через низку ітерацій, поки вони не стануть точними. Потім мережа розпізнає або "бачить" зображення.

Подібно до того, як людина спостерігає за зображеннями на відстані, CNN спочатку визначає чіткі контури та прості форми, а потім заповнює деталі через ітерації прогнозів. CNN використовується для розуміння окремих зображень. Повторювана нейронна мережа (RNN) працює подібним чином з відео, допомагаючи комп'ютерам розуміти зв'язок між кадрами.[5]

У галузі комп'ютерного зору проводиться багато досліджень, але їх мета не лише дослідити. Реальні додатки показують, наскільки важливе бачення комп'ютера для бізнесу, розваг, транспорту, медицини та повсякденного життя. Ключовий чинник зростання цих додатків — потік візуальної інформації, що надходить від смартфонів, систем безпеки, дорожніх камер та інших пристроїв з візуальними датчиками. Ці дані можуть відігравати важливу роль в різних галузях, хоча наразі залишаються не використаними в повній мірі. Ця інформація створює

середовище для навчання додатків комп'ютерного зору та сприяє їх інтеграції в різні аспекти людської діяльності і обробку об’єктів на зображеннях, таких як люди, тварини та транспортні засоби. Навколо виявленого об’єкта визначається контур, щоб ідентифікувати положення та рух об’єкта в певній сцені. Щоб повністю зрозуміти зображення, ми повинні зосередитися не тільки на класифікації зображення, але й на точному визначенні розташування та категорії об’єктів на зображенні відповідно до поставленого завдання. Це надає важливу інформацію для розуміння значення зображень і відео та актуальне для багатьох програм, таких як класифікація зображень, розпізнавання та автономне водіння. Досягнення нейронних мереж і відповідних систем навчання покращили алгоритми нейронних мереж і вплинули на методи виявлення цілей. Однак значні зміни положення, затінення та умов освітлення ускладнюють виявлення цілі, особливо в задачах локалізації цілі. Моделі виявлення цілей глибокого навчання зазвичай складаються з двох частин [6]. Кодер приймає зображення як вхідні дані та обробляє його через кілька шарів, щоб отримати статистику для виявлення об’єктів і маркування.

Наприклад, якщо зображення містить два автомобілі, але модель призначена для виявлення лише одного об’єкта, інший об’єкт не буде обрамлено. Однак, якщо кількість об’єктів, виявлених на кожному зображенні, відома заздалегідь, модель на основі регресії може бути хорошим вибором. Це поширює регресійний підхід на регіональні мережі постачання [21].

Ця частина моделі пропонує області зображення, де можуть бути розташовані об’єкти. Пікселі в цих регіонах обробляються підмережею класифікації для визначення міток (або викидів). Перевага цього методу полягає в тому, що він забезпечує більш точну та гнучку модель, яка може включати будь-яку кількість граничних областей. Однак додаткова точність залежить від ефективності розрахунку. Визначення завдання розпізнавання об’єктів передбачає визначення розташування об’єкта на зображенні та категорії, до якої цей об’єкт належить. Таким чином, традиційні моделі виявлення об’єктів складаються з трьох етапів: вибір діапазону інформації , виділення ознак і класифікація. Сканування всього зображення за допомогою багато масштабного ковзного вікна є природним

вибором для виявлення. Незважаючи на те, що цей підхід може виявити всі можливі розташування, він дорогий з точки зору обчислень і може призвести до надмірної кількості вікон. Вилучення ознак об'єкта зосереджено на вилученні візуальних ознак, які ідентифікують різні об'єкти та забезпечують надійне семантичне представлення. Типовим прикладом є HOG [7]. Однак через різноманітність зовнішнього вигляду, освітлення та фону створення надійних функціональних описів може бути складним. Крім того, потрібен класифікатор, щоб відрізнити цільовий об'єкт від інших категорій. Впровадження регіонів підтримки CNN (R-CNN) принесло значні переваги глибоким нейронним мережам. Він має глибшу архітектуру та може вивчати більш складні функції. Ви також можете дізнаватися інформацію про об'єкти без ручної розробки функціональності. З моменту появи R-CNN було запропоновано багато вдосконалених моделей, таких як Fast R-CNN, Mask R-CNN і YOLO, які забезпечують різні покращення продуктивності виявлення об'єктів[9].

Огляд системи підсумовує модель представлення та її різні властивості в багатьох областях, включаючи загальне розпізнавання об'єктів та їх видимих аспектів.

Архітектура CNN може досягти виявлення цілі за допомогою регресії в обмеженому домені та досягти виявлення цілі за допомогою локального посилення контрасту та сегментації на рівні пікселів. Виявлення транспортних засобів схоже на звичайне виявлення об'єктів і зазвичай досягається шляхом широкомасштабної адаптації лісу та об'єднання багатьох функцій. Для різних зображень потрібні різні моделі глибини. На основі архітектури YOLO виявлення об'єктів розглядається як завдання регресії просторового розміщення граничних блоків і пов'язаних з ними ймовірностей класу.

1.4 Опис дослідницького завдання

Виходячи з вищесказаного, виконане дослідження підсумовується наступним чином:

- Аналіз концепцій і завдань машинного навчання;

- Виконайте аналіз того, як визначити вільні місця для паркування;
- Оцінити алгоритми пошуку об’єктів на зображеннях;
- Розробити спосіб пошуку вільних місць для паркування на автостоянках.

1.5 Висновки до розділу 1

Пошук вільних паркувальних місць за допомогою згорткових нейронних мереж сьогодні є дуже важливим завданням. Цей розділ охоплював машинне навчання, глибоке навчання, комп’ютерне зір і виявлення об’єктів. У майбутньому буде проведено аналіз існуючих методів і алгоритмів пошуку вільних паркувальних місць на основі методів машинного навчання та комп’ютерного зору. Ми також обговорили методи попередньої обробки даних, такі як нормалізація зображення, зменшення шуму та збільшення даних, які покращують якість навчання нейронної мережі. Важливим аспектом є використання методів анотації даних для створення великого та різноманітного набору навчальних даних. Це дозволяє моделі краще узагальнювати інформацію та бути стійкою до різного освітлення та погодних умов.

На основі отриманих результатів було розроблено гібридний підхід, який поєднує в собі використання згорткових нейронних мереж для первинного розпізнавання паркувальних місць та додаткових алгоритмів пост обробки для підвищення точності та зменшення кількості хибно позитивних спрацьовувань.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ І АЛГОРИТМІВ ВИРІШЕННЯ ЗАДАЧІ ПОШУКУ ВІЛЬНИХ ПАРКУВАЛЬНИХ МІСЦЬ

2.1 Існуючі методи керування автопарковками

Зі стрімким розвитком світової економіки та постійним зростанням рівня життя збільшується кількість транспортних засобів. Для ефективного управління ними все частіше використовуються системи розпізнавання номерних знаків (LPR) та зчитувачі ультрависокої частоти (UHF) на паркувальних майданчиках і в зонах управління транспортом (рис. 2.1). Автоматична ідентифікація номерних знаків забезпечує швидкий доступ до парковок, роблячи процес паркування більш зручним.

Рис. 2.1 Структура системи (LPR) та (UHF) на паркувальних майданчиках

Система управління паркуванням ZKBioSecurity базується на використанні відеокамери для зчитування номерних знаків автомобілів (LPR-камера) та зчитувача UHF-міток великої дальності.

Розглянемо можливості роботи ***LPR-камери***[18]. LPR-камера використовує відеозображення для впізнавання автомобільних номерів. Ця техніка включає наступні кроки: Зробіть знімок номерного знака, попередньо обробіть зображення та витягніть особливості, кольори та іншу інформацію для ідентифікації номерного знака.

В даній системі доречним буде використання UHF-зчитувача [17]. Саме UHF-зчитувач — це пристрій RFID для зчитування UHF-карт і тегів, який може зчитувати кілька пасивних UHF-міток на відстані до 12 метрів одночасно. Він водонепроникний і підходить для використання в різних системах RFID, таких як керування транспортними засобами, паркування, контроль виробничих процесів і контроль доступу. (рис. 2.2).

Рис. 2.2 LPR-камера та UHF зчитувач

Існує два типи UHF-міток, сумісних з рішенням ZKTeco для управління доступом транспортних засобів: одні кріпляться на бампер автомобіля, інші на лобове скло (рис. 2.3).

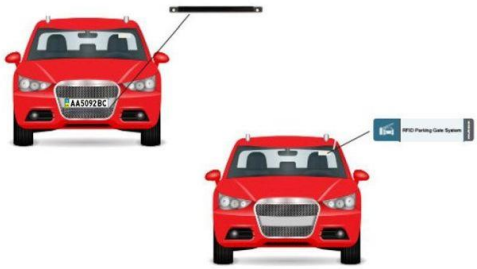


Рис. 2.3 Схематичне розміщення UHF-міток

Автоматична ідентифікація транспортних засобів за допомогою UHF-зчитувача і UHF-міток [16]. Системи доступу до паркування можуть бути реалізовані за допомогою зчитувачів UHF RFID дальнього радіусу дії та міток UHF, встановлених на автомобілях. Система активується, коли користувач з пасивною УВЧ-міткою потрапляє в зону дії УВЧ-зчитувача, розташованого на в'їзді на автостоянку. Якщо тег успішно розпізнано, доступ до захищеної зони дозволено, інакше доступ заборонено. (рис. 2.4).

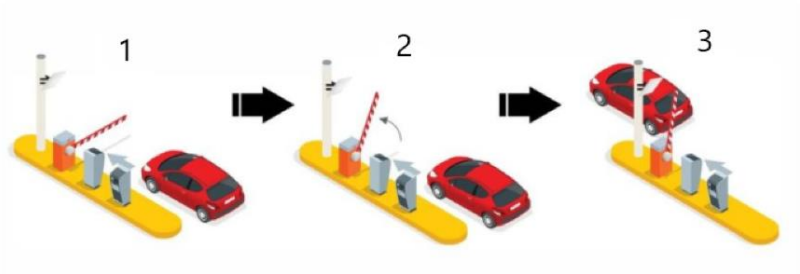


Рис. 2.4 Процес пропускної системи парковки: 1) автомобіль наближається до стоянки, UHF-зчитувач визначає UHF-мітку; 2) підняття шлагбаум доступу до стоянки після фіксації автомобіля; 3) проїзд автомобіля на стоянку парковки.

Автоматичне зчитування номерних знаків за допомогою LPR-камер. Якщо номерний знак, зчитаний камерою LPR, присутній у базі даних системи, паркувальний бар’єр автоматично піднімається, щоб дозволити доступ. Інакше у доступі буде відмовлено (рис. 2.5).

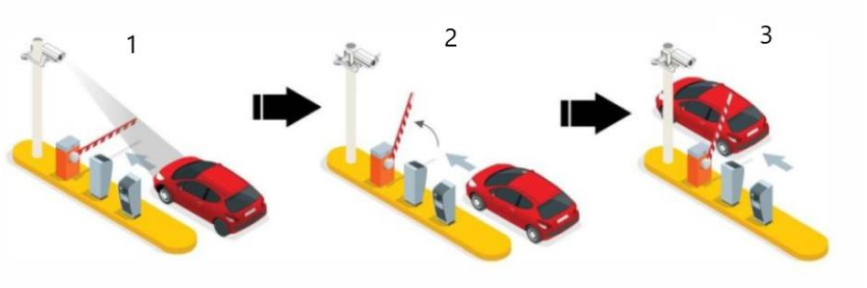


Рис. 2.5 Процес пропускної системи парковки за допомогою LPR-камер: 1) камери LPR сканують номерні знаки, коли в поле зору потрапляє автомобіль; 2) Успішна

ідентифікація збільшує бар'єри для паркування; 3) Транспортні засоби можуть заїжджати на територію, що охороняється.

Подвійна ідентифікація (UHF-зчитувач і камера LPR). Багатофакторна автентифікація підвищує безпеку та надійність систем контролю доступу (рис. 2.6). Коли автомобіль наближається до шлагбауму, зчитувач починає розпізнавати УВЧ-мітку, а камера LPR починає розпізнавати номерний знак. Після успішного збігу номерного знака та УВЧ-мітки паркувальна огорожа автоматично підніметься, щоб транспортний засіб проїхав.

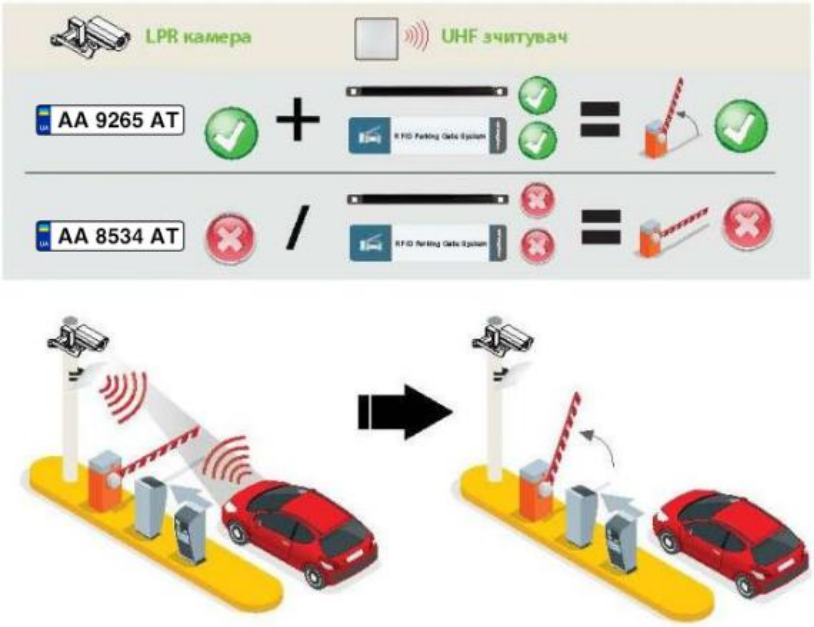


Рис. 2.6 Багатоступенева аутентифікація

White and Black Lists. Програмне забезпечення для керування паркуванням дозволяє створювати ``білі'' та ``чорні'' списки транспортних засобів (рис. 2.7). Транспортні засоби з білого списку, включаючи швидку допомогу, пожежні та поліцейські автомобілі, можуть безперешкодно заїжджати на охоронювану територію. Автомобілям з чорного списку доступ буде заблокований.

Рис. 2.7 Транспортні засоби яким дозволено проїжджати та яким заборонено проїзд

2.2 Аналіз алгоритму HOG

В даний час існує багато алгоритмів для виявлення автомобілів на зображеннях або відеопотоках. Однак для розробки методології пошуку вільних паркувальних місць були обрані такі алгоритми, як HOG, YOLO, R-CNN і Mask RCNN. HOG — це дескриптор об'єкта, який використовується для виявлення об'єктів у машинному навчанні, комп'ютерному зорі та обробці зображень. Основна ідея цього алгоритму полягає в тому, що зображення можна описати за допомогою розпізнавання образів на основі спрямованих градієнтних гістограм[13].

Створення дескриптора ознак починається з поділу зображення на клітинки та призначення кожній клітинці гістограми спрямованих градієнтів пікселів, які вона містить. Ці гістограми об'єднуються, щоб сформувати дескриптор об'єкта. Зображення зазвичай перетворюються на градації сірого для підвищення точності обробки. Контраст локальної гістограми спрямованих градієнтів нормалізовано.

Цей метод був запропонований в 2005 році і досі активно використовується для вирішення різноманітних завдань. Порівняно з іншими методами HOG має переваги[14].

Використання комірок локальної сітки зображення забезпечує інваріантність до геометричних і оптичних спотворень. Ідея полягає в тому, що зовнішній вигляд об'єкта часто добре описується та характеризується розподілом локальних градієнтів інтенсивності пікселів.[12] Градієнт тут означає наближення градієнта інтенсивності зображення, який відомий лише в одній точці. На практиці це реалізується шляхом поділу зображення на невеликі просторові області (комірки) і збору гістограми градієнтів кожного пікселя всередині комірки. Потім вони об'єднуються для створення дескриптора функції (рис.2.8). Оскільки інформація, що зберігається в кольорах зображення, мало впливає, вхідне зображення потрібно перетворити на чорно-біле, щоб зменшити вплив факторів освітлення.

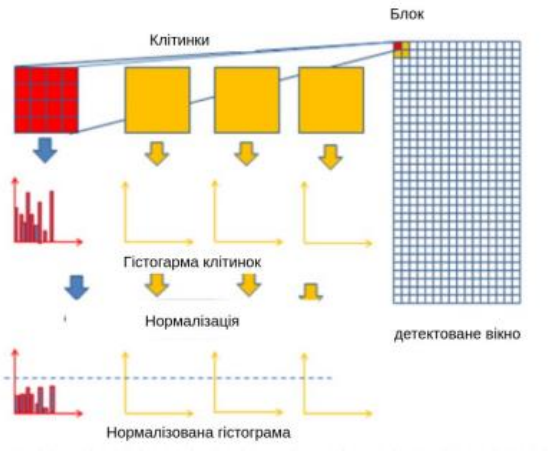


Рис. 2.8 Архітектура алгоритму (HOG)

На цій основі здійснюється обчислення градієнту вздовж горизонтальних і вертикальних напрямків зображення, а також визначається кут градієнта для кожної позиції, щоб сформувати гістограму, яка в подальшому може бути використана в алгоритмі. Це можна здійснити шляхом фільтрації зображення за допомогою спеціальних ядер зображень.

Ядра зображення - це невеликі матриці, які застосовуються для різних операцій обробки зображень, таких як розмиття, підвищення різкості та визначення контурів. У машинному навчанні ці ядра також використовуються для виділення важливих або характерних частин зображення. Зазвичай вони використовуються у згорткових нейронних мережах для класифікації зображень, де вони допомагають знаходити контури та визначати ключові особливості на зображенні.[11]

Для кожного пікселя можна визначити показники розміру та напрямку градієнта, який описує зміну кольору пікселя вздовж осей x та y. Це визначення відповідає нахилу неперервної функції, яка є вектором частинних похідних за всіма змінними. Якщо f(x, y) є кольором пікселя в (x, y), то вектор градієнта в пікселі (x, y) визначається як:

$$f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1,y) - f(x-1),y \\ f(x,y+1) - f(x,y-1) \end{bmatrix} \quad (2.1)$$

Часткова похідна ∂f/∂x вимірюється як різниця кольорів між пікселями, розташованими ліворуч і праворуч від певної точки (f(x + 1, y) - f(x - 1, y)). Змінна ∂y – різниця кольорів між сусідніми пікселями вище та нижче цієї точки (f(x, y + 1) – f(x, y – 1)).

Основна ідея полягає в тому, що градієнт збільшується там, де є велика зміна інтенсивності, що вказує на наявність країв на зображенні. Це дозволяє відфільтрувати непотрібну інформацію та зберегти лише важливі частини зображення. Градієнти можна використовувати для створення зображень, які усувають непотрібні елементи та підкреслюють значиму інформацію.

Зображення розбивається на клітинки для обчислення гістограми градієнтів для кожної з них. Наприклад, якщо розмір зображення складає 64 на 128 пікселів, його можна поділити на клітинки розміром 8 на 8. Це дозволяє створювати компактні дескриптори функцій для представлення певних областей зображення, що забезпечує більшу стійкість до шуму та зменшує розмір зображення. Гістограма розбивається на дев'ять окремих сегментів, кожен з яких відповідає конкретним кутам від 0 до 180 градусів з кроком 20 градусів. Ключове завдання — визначити

всі пікселі на гістограмі. Залежно від вибраного напрямку вибираються біни, а значення пікселів розподіляються між двома бінами пропорційно їх відстані від кожного біна.

Освітлення може впливати на обчислення градієнтів. Нормалізація здійснюється шляхом розрахунок довжини вектора та поділ усіх його елементів на цю довжину. Цю операцію нормалізації можна виконати на різних розмірах блоків (8 на 8 або 16 на 16). Кінцевий крок - об'єднати всі блоки.

Використання методу HOG для виявлення об'єктів та розпізнавання зображень базується на дескрипторах функцій, які виділяють корисну інформацію та відкидають надлишкові. HOG розраховує горизонтальні та вертикальні компоненти градієнтів кожного окремого пікселя, а потім організовує цю інформацію в гістограму для визначення зміщень у даних. Нормалізація блоків може покращити модель, зменшивши її упередження. Метод HOG може бути застосований у різних областях, від автономних транспортних засобів до AR та VR, для розв'язання завдань, пов'язаних з виявленням об'єктів на зображеннях.

Проте для пошуку вільного паркувального місця метод HOG може бути не найкращим рішенням через складнощі з розміщенням автомобілів під різними кутами відносно камери, що призводить до ускладнення розпізнавання автомобілів на зображенні чи відеопотоці.

2.3 Висновки до розділу 2

У цьому розділі було проведено аналіз основних методик пошуку вільного паркувального місця. Було проведено дослідження сучасних систем управління парковками та принципів їх організації, в результаті чого визначено основні структурні компоненти на рівні апаратного та програмного забезпечення. Було досліджено корпоративні бізнес-орієнтовані рішення, що застосовуються для управління мережами парковок, що дозволило виявити їх недоліки та обґрунтувати необхідність розробки методів і засобів на основі прогнозування доступності паркомісць з урахуванням параметрів оптимальної вартості та маршруту від місця розташування водія до найближчої парковки. Були визначені основні

характеристики кожного алгоритму. Кожен із розглянутих алгоритмів має переваги та недоліки.

Однак за допомогою проаналізованого підходу можна вирішити проблему і розробити метод пошуку вільного паркувального місця на автостоянці. Для моделювання системи пошуку вільних паркувальних місць необхідно вибрати один із запропонованих алгоритмів і розробити програмний код.

3 МОДЕЛЮВАННЯ СИСТЕМИ ПО ЗНАХОДЖЕНЮ ПАРКУВАЛЬНОГО МІСЦЯ ЯКЕ Є ВІЛЬНИМ

3.1 Програмне забезпечення для виявлення вільних парковочних місць

Під час розробки методики пошуку вільного паркувального місця та впровадження практичної частини були використані мова програмування Python та такі бібліотеки: TensorFlow, Keras, NumPy, OpenCV, os, Matterport .

Модуль os надає можливість використання функціоналу, залежного від операційної системи.

TensorFlow є відкритою платформою для машинного навчання, яка включає гнучку екосистему інструментів та бібліотек, дозволяючи дослідникам впроваджувати найсучасніші технології машинного навчання, а розробникам — легко створювати та розгортати додатки .

OpenCV (Open Source Computer Vision Library) — це бібліотека з відкритим кодом, що містить багато алгоритмів комп'ютерного зору .

NumPy є основним пакетом для наукових обчислень на Python, який забезпечує багатовимірний об’єкт масиву та різноманітні підпрограми для швидких операцій з масивами, включаючи математичні, логічні операції, маніпуляції з фігурами, сортування, вибір, введення/виведення, дискретні перетворення Фур'є, лінійну алгебру, статистичні операції, випадкове моделювання та багато іншого [22].

Keras — це API, розроблене для зручності користувачів, яке пропонує прості та послідовні API, мінімізує кількість дій, необхідних для загальних випадків використання, та надає чіткі повідомлення про помилки. Крім того, Keras має обширну документацію та посібники для розробників [24].

Matterport — це реалізація алгоритму Mask R-CNN на Python 3 з використанням Keras та TensorFlow, яка базується на Feature Pyramid Network та ResNet 101.

3.2 Вимоги до проектованої системи

Щоб система працювала належним чином, вона повинна відповідати таким вимогам:

- чітко визначати об'єкт, зокрема, розпізнавати, чи зайняте паркувальне місце або вільне;
- відстежувати наявність транспортного засобу;
- передавати інформацію на сервер для доступу користувачів через інтерфейс (мобільний додаток);
- повідомляти систему про зміну стану паркувального місця після його звільнення.

Дана система буде працювати за принципом, який наведено на (Рис. 3.1)

Рис. 3.1 Схема роботи системи паркувальника

3.3 Функціональні можливості проектованої системи

Ключові функціональні вимоги, визначені під час розробки програмного забезпечення для управління автоматичним шлагбаумом, показані на (Рис. 3.2).

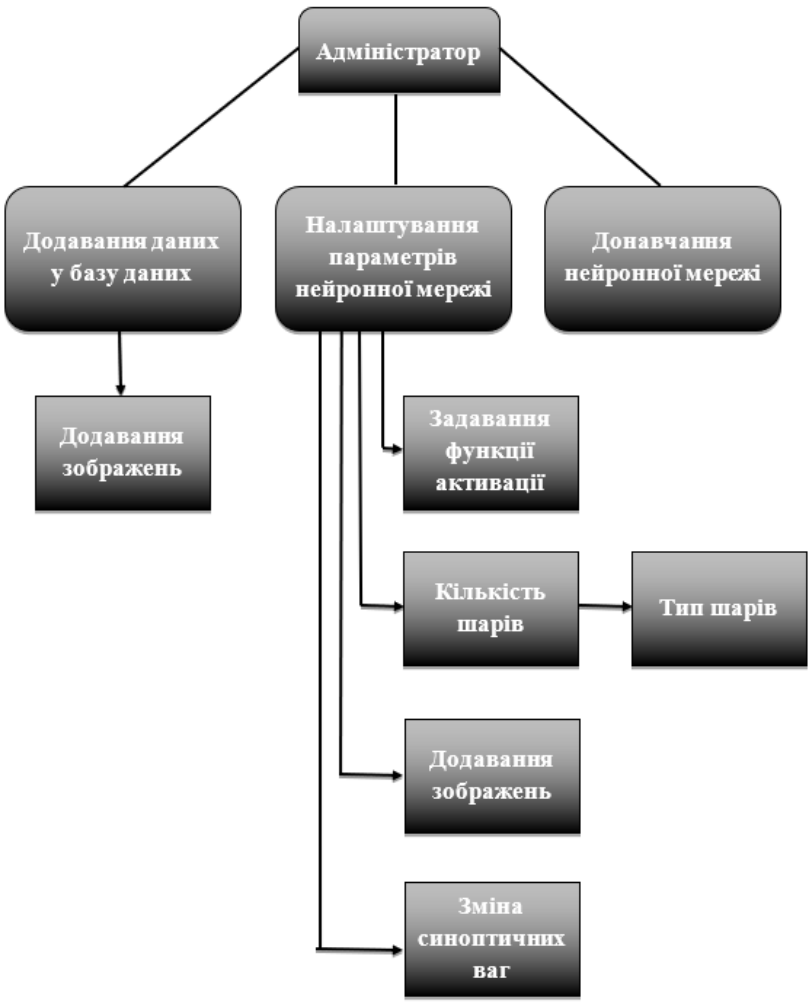


Рис. 3.2 Діаграма варіантів використання при розробці моделі інтелектуально комп'ютеризованої системи управління парковками

Вхідним даним є відеопотік з камери, спрямованої на парковку (Рис. 3.3):

Рис 3.3 Відеопоток з камери яка направлена на парковку

Будуть передаватись кожен кадр відео через конвеєр по одному за раз. Паркувальні місця знаходяться там, де машини стоять протягом тривалого часу (Рис. 3.4):

Рис. 3.4 - Заповненні паркомісця

Таким чином, якщо ми зможемо розпізнати автомобілі і визначити, які з них залишаються нерухомими між кадрами, ми зможемо визначити розташування паркувальних місць.

Розпізнавання автомобілів на відеокадрі є типовим завданням розпізнавання об'єктів. Для цього можна застосувати різні підходи на основі машинного навчання.

Можна підготувати детектор, використовуючи метод HOG (Histogram of Oriented Gradients, гістограми спрямованих градієнтів), та просканувати ним всю картинку для виявлення всіх автомобілів. Цей застарілий підхід, який не базується на глибокому навчанні, дозволяє досягти відносно швидкої роботи, проте не надто ефективний у випадку автомобілів, що розташовані в різних позиціях.

Можна навчити детектор, використовуючи згорткову нейронну мережу (CNN, Convolutional Neural Network) та просканувати ним всю картинку, поки не будуть виявлені всі автомобілі. Цей метод працює дуже точно, але менш ефективно, оскільки потрібно просканувати зображення кілька разів за допомогою CNN, щоб знайти всі автомобілі. Хоча таким чином ми можемо виявити автомобілі, що розташовані в різних позиціях, нам необхідно мати значно більше навчальних даних, ніж для HOG-детектора.

Можна скористатися новими методиками глибокого навчання, такими як Mask R-CNN, Faster R-CNN або YOLO, які об'єднують в собі точність CNN та набір технічних рішень, що дозволяють значно прискорити процес розпізнавання. Такі моделі здатні працювати досить швидко (з використанням GPU), якщо у нас є достатньо даних для їх навчання.

В цілому, ми шукаємо найбільш просте рішення, яке працюватиме ефективно і вимагатиме мінімальну кількість тренувальних даних. Проте в нашому конкретному випадку, Mask R-CNN є розумним вибором, навіть якщо він вважається новим і швидким.

Архітектура Mask R-CNN була спроектована таким чином, що вона здатна ефективно розпізнавати об'єкти на всьому зображенні без застосування методу ковзного вікна. Іншими словами, вона працює досить швидко. З сучасним GPU можливо розпізнавати об'єкти на відео високої якості зі швидкістю кількох кадрів на секунду. Цього повинно вистачити для наших потреб у цьому проекті.

Більшість алгоритмів розпізнавання надають лише прямокутну рамку, що обмежує кожен об'єкт. Однак Mask R-CNN видає більше інформації: крім місця, де знаходиться кожен об'єкт, він також надає його контур, або маску (Рис. 3.5):

Рис. 3.5 Визначення маска R-CNN

Для тренування Mask R-CNN потрібна велика кількість зображень об'єктів, які планується розпізнавати. Одним з таких датасетів є COCO (Common Objects In Context), що містить зображення з анотаціями масок об'єктів. У цьому датасеті є понад 12 000 фотографій автомобілів з уже нанесеними позначками.

Ці дані ідеально підходять для тренування моделі з використанням Mask R-CNN. Кожного разу, коли модель розпізнає об'єкт, вона повертає 4 характеристики:

- Цілочисельна змінна, що визначає тип об'єкта. Модель COCO має можливість розпізнавати 80 різних типів об'єктів, які часто зустрічаються, таких як автомобілі та вантажівки. Повний список цих об'єктів доступний за посиланням.
- Рівень впевненості в результаті розпізнавання. Чим вище значення, тим більш впевнена модель у правильності розпізнавання об'єкта.
- Прямокутна рамка, яка обмежує об'єкт, виражена у вигляді координат X та Y пікселів на зображенні.

Маска вказує, які пікселі всередині прямокутної рамки є частиною об'єкта. За допомогою цієї маски можна визначити контур об'єкта.

Внизу наведено Python-код для виявлення прямокутних рамок, які обмежують машини, за допомогою попередньо навченої моделі Mask R-CNN та OpenCV. При програмуванні будь-якої моделі в області комп'ютерного зору,

першим кроком є підключення потрібних бібліотек, що містять основні алгоритми для обробки зображень:

Лістинг 3.1

```
import numpy as np
import cv2
import mrcnn.config
import mrcnn.utils
from mrcnn.model import MaskRCNN
from pathlib import Path

# Конфігурація, яку використовуватиме бібліотека Mask-RCNN.
class MaskRCNNConfig(mrcnn.config.Config):
  NAME = "coco_pretrained_model_config"
  IMAGES_PER_GPU = 1
  GPU_COUNT = 1
  NUM_CLASSES = 1 + 80  # У датасеті COCO знаходиться 80 класів + 1 фоновий клас.
  DETECTION_MIN_CONFIDENCE = 0.6

# Фільтруємо список результатів розпізнавання, щоб залишилися лише автомобілі.
def get_car_boxes(boxes, class_ids):
  car_boxes = []

  for i, box in enumerate(boxes):
    # Якщо знайдений об'єкт не автомобіль, пропускаємо його.
    if class_ids[i] in [3, 8, 6]:
      car_boxes.append(box)

  return np.array(car_boxes)
```

```
# Коренева директорія проекту.
ROOT_DIR = Path(".")

# Директорія для збереження логів та навченої моделі.
MODEL_DIR = ROOT_DIR / "logs"

# Локальний шлях до файлу з навченими вагами.
COCO_MODEL_PATH = ROOT_DIR / "mask_rcnn_coco.h5"

# Завантажуємо датасет COCO за потреби.
if not COCO_MODEL_PATH.exists():
    mrcnn.utils.download_trained_weights(COCO_MODEL_PATH)

# Директорія із зображеннями для обробки.
IMAGE_DIR = ROOT_DIR / "images"

# Відеофайл або камера обробки — Вставте значення 0, якщо потрібно використовувати камеру, а не відеофайл.
VIDEO_SOURCE = "test_images/parking.mp4"

# Створюємо модель Mask-RCNN як виведення.
model = MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=MaskRCNNConfig())

# Завантажуємо передбачувану модель.
model.load_weights(COCO_MODEL_PATH, by_name=True)

# Розташування паркувальних місць.
parked_car_boxes = None
```

```
# Завантажуємо відеофайл, для якого хочемо запустити розпізнавання.
video_capture = cv2.VideoCapture(VIDEO_SOURCE)

# Проходимося в циклі з кожного кадру.
while video_capture.isOpened():
    success, frame = video_capture.read()
    if not success:
        break

    # Конвертуємо зображення з колірної моделі BGR (використовується OpenCV) у RGB.
    rgb_image = frame[:, :, ::-1]

    # Подаємо зображення моделі Mask R-CNN для отримання результату.
    results = model.detect([rgb_image], verbose=0)

    # Mask R-CNN припускає, що ми розпізнаємо об'єкти на численних зображеннях.
    # Ми передали лише одне зображення, тому отримуємо лише перший результат.
    r = results[0]

    # Змінна r тепер містить результати розпізнавання:
    # - r['rois'] — рамка, що обмежує, для кожного розпізнаного об'єкта;
    # - r['class_ids'] - ідентифікатор (тип) об'єкта;
    # - r['scores'] - ступінь впевненості;
    # - r['masks'] - маски об'єктів (що дає вам їхній контур).

    # Фільтруємо результат для отримання рамок автомобілів.
    car_boxes = get_car_boxes(r['rois'], r['class_ids'])
    print("Cars found in frame of video:")
```

```
    # Показуємо кожну рамку на кадрі.
    for box in car_boxes:
        print("Car:", box)
        y1, x1, y2, x2 = box

        # Малюємо рамку.
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 1)

    # Показуємо кадр на екрані.
    cv2.imshow('Video', frame)

    # Натисніть 'q', щоб вийти.
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Очищаємо після завершення.
video_capture.release()
cv2.destroyAllWindows()
```

Після виконання цього скрипта на екрані відобразиться зображення з прямокутною рамкою навколо кожної виявленої машини.

У консоль також будуть виведені координати кожної з машин:

Лістинг 3.2

```
Cars found in frame of video:
Car: [492 871 551 961]
Car: [450 819 509 913]
Car: [411 774 470 856]
```

Надається доступ до піксельних координат кожної машини. Після перегляду декількох послідовних кадрів, ми можемо легко виявити, які з машин залишилися

нерухомими, і припустити, що ці місця можуть бути вільними для паркування. Але, в даному випадку, можна визначити, коли машина покинула паркувальне місце. Машини частково перекривають одна одну в рамках (Рис. 3.6):

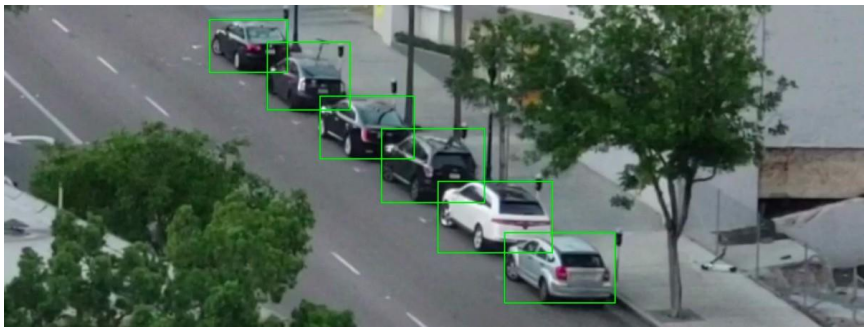


Рис. 3.6. Перекриття машин при розпізнаванні

Якщо уявити, що кожна рамка відповідає парковочному місцю, може виникнути ситуація, коли воно, здавалося б, частково зайняте автомобілем, хоча насправді вільне. Нам необхідно розробити метод оцінки ступеня перетину між цими двома об'єктами.

Ми використовуватимемо метрику, відому як Intersection Over Union (відношення площі перетину до суми площ) (Рис. 3.7), або IoU. Щоб знайти значення IoU, необхідно порахувати кількість пікселів, де перетинаються два об'єкти, і поділити її на загальну кількість пікселів, які займають ці об'єкти:

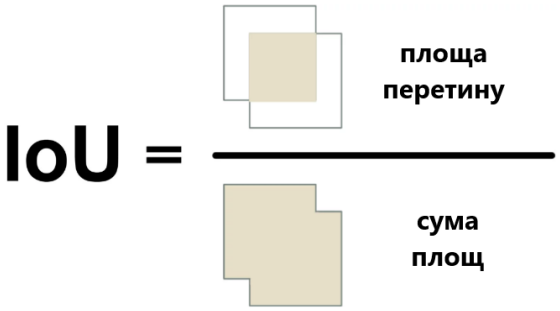


Рис. 3.7 Відношення площі перетину до суми площ

Це допоможе нам розібратися, як прямокутна рамка, що обмежує машину, перетинається з рамкою паркувального місця. Це спростить визначення доступності парковки. Низьке значення IoU, наприклад, 0.15, вказує на те, що машина займає лише невелику частину паркувального місця. У випадку високого значення, наприклад, 0.6, машина займає більшу частину місця і паркуватися там неможливо.

Оскільки IoU використовується досить часто у комп'ютерному зорі, більшість відповідних бібліотек мають вбудовану реалізацію цього показника. У нашій бібліотеці Mask R-CNN ця реалізація доступна як mrcnn.utils.compute_overlaps().

Якщо у нашому розпорядженні є перелік прямокутних рамок для парковочних місць, ми можемо легко додати перевірку наявності автомобілів у цих рамках, просто додавши ще один рядок коду:

Лістинг 3.3

```
# Фільтруємо результат для отримання рамок автомобілів.
  car_boxes = get_car_boxes(r['rois'], r['class_ids'])


  # Дивимося, як сильно машини перетинаються з відомими місцями для паркування.
  overlaps = mrcnn.utils.compute_overlaps(car_boxes, parking_areas)


  print(overlaps)
```

Отриманий результат має бути приблизно наступного вигляду:

Лістинг 3.4

```
[
[1.         0.07040032 0.         0.        ]
[0.07040032 1.         0.07673165 0.        ]
[0.         0.         0.02332112 0.        ]
]
```

У цьому двовимірному масиві кожен рядок відповідає одній рамці парковочного місця, а кожен стовпець вказує на те, наскільки це місце перетинається з однією з виявлених машин. Результат 1.0 свідчить про повне зайняття місця автомобілем, тоді як низьке значення, наприклад, 0.02, вказує на те, що автомобіль частково зайняв місце, але його ще можна використати для паркування. Щоб визначити невикористані місця, достатньо перевірити кожен рядок у цьому масиві. Якщо всі значення наближаються до нуля, це означає, що місце, швидше за все, вільне.

Проте слід мати на увазі, що розпізнавання об'єктів не завжди працює ідеально у реальному часі на відео. Навіть якщо модель на базі Mask R-CNN досить точна, іноді вона може пропустити інший автомобіль в одному з кадрів відео. Тому перед тим як вирішити, що місце для паркування вільне, важливо переконатися, що воно залишається вільним ще протягом 5–10 наступних кадрів відео. Таким чином, ми можемо уникнути ситуацій, коли система помилково вважає місце вільним через глюк у одному з кадрів відео. Як тільки ми переконаємося, що місце залишається вільним протягом кількох кадрів, ми можемо надсилати повідомлення!

Останній етап нашого процесу - надсилання SMS-повідомлення при звільненні парковочного місця.

Відправлення повідомлень за допомогою Python стає надзвичайно простим з використанням Twilio [19]. Twilio - це відомий API, який дозволяє надсилати SMS з майже будь-якої мови програмування за допомогою лише кількох рядків коду.

Для користування Twilio потрібно спочатку зареєструватися та отримати пробний обліковий запис, створити номер телефону Twilio та отримати автентифікаційні дані облікового запису. Після цього встановіть клієнтську бібліотеку:

Лістинг 3.5

$ pip3 install twilio

Після цього скористайтеся наступним кодом для надсилання повідомлення:

Лістинг 3.6

```
from twilio.rest import Client

# Дані облікового запису Twilio.
twilio_account_sid = 'Ваш Twilio SID'
twilio_auth_token = 'Ваш токен аутентифікації Twilio'
twilio_source_phone_number = 'Ваш номер телефона Twilio'

# Створюємо об'єкт клієнта Twilio.
client = Client(twilio_account_sid, twilio_auth_token)

# Відправляємо SMS.
message = client.messages.create(
  body="Тіло повідомлення",
  from_=twilio_source_phone_number,
  to="Ваш номер, куди прийде повідомлення" )
```

Для включення можливості надсилання повідомлень у нашому скрипті просто додайте цей код. Однак важливо забезпечити, щоб повідомлення не надсилалося кожного разу, коли виявляється вільне місце. Тому ми використовуємо прапорець, який встановлюється у певний стан і не дозволяє надсилання повідомлень протягом певного часу або поки не звільниться інше місце.

3.4 Висновки до розділу 3

В третьому розділі було розроблено метод для пошуку вільних паркувальних місць, на основі якого було створено програмний код для вирішення проблеми пошуку вільних паркувальних місць на автостоянці і було реалізовано відповідні алгоритми для роботи інтелектуальної комп'ютеризованої системи управління парковками, що дозволило надалі провести аналіз ефективності запропонованих рішень шляхом порівняння з наявними системами.

Результати аналізу показали, що запропонована система значно перевершує традиційні методи як за швидкістю виявлення вільних місць, так і за точністю їх визначення. Використання сучасних технологій, таких як машинне навчання та обробка великих даних, забезпечило більш високу адаптивність і надійність роботи системи. Крім того, було враховано фактори економії ресурсів та зниження впливу на навколишнє середовище завдяки оптимізації маршруту водіїв до вільних місць. Система також включає в себе інтуїтивно зрозумілий інтерфейс для користувачів, що сприяє зручності її використання.

Таким чином, проведене дослідження підтверджує, що розроблена інтелектуальна комп'ютеризована система управління парковками є ефективним інструментом для покращення організації паркування на автостоянках, що, в свою чергу, сприяє підвищенню комфорту водіїв та оптимізації використання міських ресурсів.

ВИСНОВКИ

Під час виконання бакалаврської роботи було досліджено сферу машинного навчання, зокрема глибокого навчання та комп'ютерного зору, з метою вирішення завдання пошуку паркувальних місць які є вільними.

Було досліджено поняття машинного навчання, глибокого навчання, комп'ютерного зору та детекції об’єктів на зображеннях і у відеопотоці для вирішення поставленої задачі. Проаналізовано різноманітні алгоритми для виявлення об'єктів та вивчено їх структуру і можливості.

У ході роботи було розглянуто та проаналізовано такі алгоритми машинного навчання і комп'ютерного зору, як HOG, YOLO, R-CNN та Mask R-CNN. Описано принципи їх роботи та виявлено проблеми, з якими вони стикаються.

Було створено мережу на основі алгоритму Mask R-CNN для пошуку вільних паркувальних місць за допомогою машинного навчання та комп'ютерного зору.

Була розроблена інтелектуальна комп'ютеризована система управління парковками, з метою поліпшення ефективності та безпеки автоматичних стоянок. Вона може бути використана для модернізації існуючих автостоянок або встановлення на нових об'єктах. Основними функціями цієї системи будуть: Автоматичне виявлення та реєстрацію в'їзду та виїзду автомобілів; Керування доступом; Моніторинг зайнятості парковки: Завдяки використанню датчикам та камер, система може визначати, які місця на парковці зайняті, а які вільні. Ця інформація може бути відображена на інформаційних табло або мобільних додатках.

Інтелектуальна комп'ютеризована система управління парковками сприяє покращенню управління автостоянками, зменшенню часу пошуку місць для паркування, покращенню безпеки та зручності для водіїв. Вона також може забезпечити збір даних про використання парковки для подальшого аналізу та оптимізації її роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1~~4~~. Adrian Rosebrock/pyimagesearch/OpenCV: Automatic License/Number Plate Recognition (ANPR) with Python/September 21, 2020/ https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/

2~~10~~. Gary Bradski, Adrian Kaehler/Learning OpenCV: Computer Vision with the OpenCV Library/2011-07-22/ https://books.google.com.ua/books?id=seAgiOfu2EIC&pg=PA267&lpg=PA267&dq=openCV+parking&source=bl&ots=hWE69leEQc&sig=ACfU3U1OoFtSopHqBx1SEUzDZqZDf44wKQ&hl=uk&sa=X&ved=2ahUKEwjqiY_Y1KOGAxUESPEDHWMaAgs4RhDoAXoECAQQAw#v=onepage&q=openCV%20parking&f=false

3~~7~~. Jashwanth Dasari, Shivani Vodnala, Swapna Enugala/https://www.ijraset.com/2023-08-25/https://www.ijraset.com/research-paper/smart-parking-system-using-python-and-opencv

4~~3~~. Licenseplatesrecognition/ HowLPRworks/ https://www.licenseplatesrecognition.com/how-lpr-works.html

5. Mask R-CNN Kaiming He Georgia Gkioxari Piotr Dollar Ross Girshick ´ Facebook AI Research (FAIR)

6~~5~~. Olgarose/Parking Space Detection in OpenCV/ https://olgarose.github.io/ParkingLot/

7~~6~~. Priya Dwivedi/https://medium.com/Oct 16, 2018/ https://towardsdatascience.com/find-where-to-park-in-real-time-using-opencv-and-tensorflow-4307a4c3da03

8. Yuvraj Raulji/Car Parking Space Management System/March 12, 2024/ https://prakashinfotech.com/car-parking-space-management-system

9~~2~~. Zkteco/ Керування парковкою/ https://zkteco.technology/solution/parking-management/

10~~12~~. База даних для системи COCO https://github.com/matterport/Mask_RCNN

11. Бовкун В.В., Кисіль Т.М. Застосування штучного інтелекту в системах паркування автотранспорту. IV Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез.– К.: ДУІКТ, 2024. С.290-292.

12. Бовкун В.В., Кисіль Т.М., Етичні аспекти застосування штучного інтелекту в сучасному суспільстві / Науково-практична конференція «Проблеми комп’ютерної інженерії». К.: ДУІКТ, 2023 р. с. 56-58.

13~~1~~. Брайант Р., О'Халларон Д. Комп'ютерні системи. Архітектура та програмування, 3-є видання. Print2print. 358 с.

14. Доля Г. Комп'ютерні системи штучного інтелекту. Університет «Україна». 2011.296 с.

15. Кисіль Т.М., Бовкун В.В., 3D-ПРИНТЕР HEWLETT-PACKARD З ІНТЕГРОВАНИМ ШТУЧНИМ ІНТЕЛЕКТОМ ВІДЕО / Науково-практична конференція «Сучасні досягнення компанії HEWLETT PACKARD ENTERPRISE в

галузі ІТ та нові можливості їх вивчення і застосування». Збірник тез. – К.: ДУІКТ, 2023р., с. 33-35.

16~~8~~. Офіційна документація бібліотеки numpy https://numpy.org/install/

17~~9~~. Офіційна документація бібліотеки opencv https://opencv.org/

18~~11~~. Офіційна документація бібліотеки tensorflow https://www.tensorflow.org/

19. Платформа для відсилання sms повідомлень twilio https://www.twilio.com/

20. Ясній О.П., Галас М.М. Архітектура інтелектуальної комп'ютерної системи управління доступністю паркомісць. Матеріали XII міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (6-7 грудня 2023 року). Тернопіль: ТНТУ. 2022. С. 463.

21. Ясній О.П., Галас М.М. Нейронна мережа розпізнавання номерних знаків при організації системи керування парковкою. Матеріали XI науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (13-14 грудня 2023 року). Тернопіль: ТНТУ. 2022. С. 141.

ДОДАТОК А ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ

Отформатировано: По центру

```
import numpy as np
import cv2
import mrcnn.config
import mrcnn.utils
from mrcnn.model import MaskRCNN
from pathlib import Path
from twilio.rest import Client

# Конфігурація, яку використовуватиме бібліотека Mask-RCNN.
class MaskRCNNConfig(mrcnn.config.Config):
    NAME = "coco_pretrained_model_config"
    IMAGES_PER_GPU = 1
    GPU_COUNT = 1
    NUM_CLASSES = 1 + 80  # у датасеті COCO знаходиться 80 класів
+ 1 фоновий клас.
    DETECTION_MIN_CONFIDENCE = 0.6

# Фільтруємо список результатів розпізнавання, щоб залишилися лише
автомобілі
def get_car_boxes(boxes, class_ids):
    car_boxes = []

    for i, box in enumerate(boxes):
        # Якщо знайдений об'єкт не автомобіль, то пропускаємо
його.
        if class_ids[i] in [3, 8, 6]:
            car_boxes.append(box)
```

```
    return np.array(car_boxes)

# Конфигурация Twilio.
twilio_account_sid = 'Ваш Twilio SID'
twilio_auth_token = 'Ваш токен аутентифікації Twilio'
twilio_phone_number = 'Ваш номер телефону Twilio'
destination_phone_number = 'Номер, куди прийде повідомлення'
client = Client(twilio_account_sid, twilio_auth_token)

# Коренева директорія проекту.
ROOT_DIR = Path(".")

# Директорія для збереження логів та навченої моделі.
MODEL_DIR = ROOT_DIR / "logs"

# Локальний шлях до файлу з навченими вагами.
COCO_MODEL_PATH = ROOT_DIR / "mask_rcnn_coco.h5"

# Завантажуємо датасет COCO за необхідності.
if not COCO_MODEL_PATH.exists():
    mrcnn.utils.download_trained_weights(COCO_MODEL_PATH)

# Директорія із зображеннями для обробки.
IMAGE_DIR = ROOT_DIR / "images"

# Відеофайл або камера обробки — вставте значення 0, якщо
використовувати камеру, а не відеофайл.
VIDEO_SOURCE = "test_images/parking.mp4"
```

```
# Створюємо модель Mask-RCNN у режимі виводу.
model = MaskRCNN(mode="inference", model_dir=MODEL_DIR,
config=MaskRCNNConfig())

# Завантажуємо просунуту модель.
model.load_weights(COCO_MODEL_PATH, by_name=True)

# Розташування паркувальних місць.
parked_car_boxes = None

# Завантажуємо відеофайл, для якого хочемо запустити
розпізнавання.
video_capture = cv2.VideoCapture(VIDEO_SOURCE)

# Скільки кадрів поспіль із порожнім місцем ми вже бачили.
free_space_frames = 0

# Ми вже відправляли SMS?
sms_sent = False

# Проходимося в циклі з кожного кадру.
while video_capture.isOpened():
    success, frame = video_capture.read()
    if not success:
        break

    # Конвертуємо зображення з колірної моделі BGR в RGB.
    rgb_image = frame[:, :, ::-1]
```

```
    # Подаємо зображення моделі Mask R-CNN для отримання результату.
    results = model.detect([rgb_image], verbose=0)

    # Mask R-CNN припускає, що ми розпізнаємо об'єкти на численних зображеннях.
    # Ми передали лише одне зображення, тому отримуємо лише перший результат
    r = results[0]

    # Змінна r тепер містить результати розпізнавання:
    # - r['rois'] — рамка, що обмежує, для кожного розпізнаного об'єкта;
    # - r['class_ids'] — ідентифікатор (тип) об'єкта;
    # - r['scores'] — ступінь упевненості;
    # - r['masks'] — маски об'єктів (що дає вам їхній контур).

    if parked_car_boxes is None:
        # Це перший кадр відео - припустимо, що всі виявлені машини стоять на парковці.
        # Зберігаємо місце розташування кожної машини як місце для паркування і переходимо до наступного кадру.
        parked_car_boxes = get_car_boxes(r['rois'], r['class_ids'])
    else:
        # Ми вже знаємо, де місця. Перевіряємо, чи є вільні.

        # Шукаємо машини на поточному кадрі.
        car_boxes = get_car_boxes(r['rois'], r['class_ids'])
```

```
        # Дивимося, як сильно ці машини перетинаються з відомими
місцями для паркування.
        overlaps                                                =
mrcnn.utils.compute_overlaps(parked_car_boxes, car_boxes)

        # Припускаємо, що вільних місць немає, доки знайдемо хоча
б одне.
        free_space = False

        # Проходимося в циклі по кожному відомому паркувальному
місцю.
        for parking_area, overlap_areas in zip(parked_car_boxes,
overlaps):

            # Шукаємо максимальне значення перетину з будь-якою
виявленою
            # На кадрі машиною (неважливо, який).
            max_IoU_overlap = np.max(overlap_areas)

            # Отримуємо верхню ліву та нижню праву координати
місця для паркування.
            y1, x1, y2, x2 = parking_area

            # Перевіряємо, чи вільне місце, перевіривши значення
IoU.
            if max_IoU_overlap < 0.15:
                # Місце вільне! Малюємо зелену рамку навколо
нього.
```

```
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255,
0), 3)
                # Зазначаємо, що ми знайшли щонайменше воно вільне
місце.
                free_space = True
            else:
                # Місце все ще зайняте – малюємо червону рамку.
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0,
255), 1)

            # Записуємо значення IoU усередині рамки.
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, f"{max_IoU_overlap:0.2}", (x1 + 6,
y2 - 6), font, 0.3, (255, 255, 255))

        # Якщо хоча б одне місце було вільним, починаємо вважати
кадри.
        # Це для того, щоб переконатися, що місце дійсно вільне
        # і не надіслати зайвий раз повідомлення.
        if free_space:
            free_space_frames += 1
        else:
            # Якщо все зайнято, обнулюємо лічильник.
            free_space_frames = 0

        # Якщо місце вільне протягом кількох кадрів, можна
сказати, що воно вільне.
        if free_space_frames > 10:
            # Відображаємо напис SPACE AVAILABLE!! вгорі екрану.
```

```
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, f"SPACE AVAILABLE!", (10, 150),
font, 3.0, (0, 255, 0), 2, cv2.FILLED)

            # Надсилаємо повідомлення, якщо ще не зробили це.
            if not sms_sent:
                print("SENDING SMS!!!")
                message = client.messages.create(
                    body="Parking space open - go go go!",
                    from_=twilio_phone_number,
                    to=destination_phone_number
                )
                sms_sent = True

        # Показуємо кадр на екрані.
        cv2.imshow('Video', frame)

    # Натисніть 'q', щоб вийти.
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Натисніть 'q', щоб вийти.
video_capture.release()
cv2.destroyAllWindows()
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Розробка системи паркування з визначенням вільного місця за допомогою OpenCV»

Виконав: здобувач вищої освіти гр. ШІД-41
Владислав БОВКУН

Керівник: *ст.викладач* Тетяна КИСІЛЬ

2024 рік

2

Мета роботи: підвищення ефективності роботи паркування за допомогою алгоритмів машинного навчання**.**

Об'єкт дослідження: процес пошуку та виявлення вільних місць на автопарковці.

Предмет дослідження: розпізнавання наявності вільних паркувальних місць за методами машинного навчання та глибокого навчання.

Основні завдання кваліфікаційної роботи:

1. Дослідження і аналіз існуючих методів визначення вільних паркувальних місць.
2. Розробка алгоритму визначення вільних місць на парковці на основі OpenCV.
3. Аналіз результатів тестування та виправлення виявлених недоліків або неточностей.

Постановка задачі

3

- Проаналізувати існуючі системи для пошуку вільного паркувального місця.

- Розробити систему яка використовує алгоритм машинного навчання для розпізнавання автомобілів та паркувальних місць на зображеннях та у відеопотоці.
- Розробити систему яка може допомогти водіям знайти вільне паркувальне місце, що може заощадити час, паливо та зменшити рівень стресу водія.

Мета розробки

4

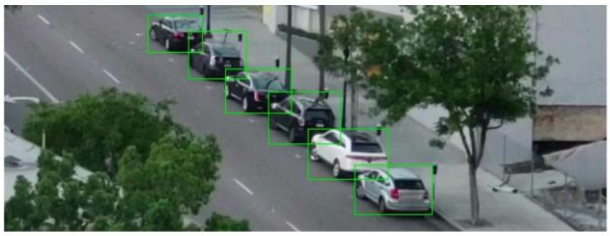
- Розробити систему була для підвищення ефективності керування парковкою в ТРЦ “RETROVILE”

- Також цю систему можна використовувати і для інших відкритих парковок

Рішення

5

- Розроблено систему для пошуку вільного паркувального місця.
- Система використовує алгоритм Mask R-CNN для розпізнавання автомобілів та паркувальних місць.
- Система може працювати з зображеннями та відеопотоком.



Методи

6

Система складається з трьох основних компонентів:

- Модуль розпізнавання об'єктів: використовує алгоритм Mask R-CNN для розпізнавання автомобілів та паркувальних місць.
- Модуль визначення вільних місць: використовує метрику Intersection Over Union (IOU) для визначення, чи є паркувальне місце вільним.
- Модуль інтерфейсу користувача: надає користувачеві інформацію про вільні паркувальні місця.

- Контрольоване навчання (навчання з учителем)
- Ітераційний процес
- Штучні нейронні мережі
- Згорткові нейронні мережі (CNN)
- Рекурентні нейронні мережі (RNN)

Висновки

- Розроблено систему для пошуку вільного паркувального місця.
- Система може допомогти водіям знайти вільне паркувальне місце, що може заощадити час та зменшити стрес.
- Система може бути використана для створення мобільного додатку або веб-сайту, який допомагатиме водіям знаходити паркувальні місця.