МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

_____Аліна САВЧЕНКО

«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНА РОБОТА, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **«МАГІСТР»**

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

Тема: «Система автоматизованого визначення за номером транспортного засобу»

Виконавець: студент групи УС-212М Іващенко Мирон Владимирович

Керівник: _____к.т.н., доцент Холявкіна Тетяна Володимирівна

Нормоконтролер:_____ Ігор РАЙЧЕВ

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп’ютерної та програмної інженерії

Кафедра Комп’ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та техгнології”

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«_____»_____ 2022 р.

ЗАВДАННЯ
на виконання кваліфікацій роботи студента

Іващенка Мирона Володимировича
(прізвище, ім’я, по батькові)

1. Тема роботи: «Система автоматизованого визначення за номером транспортного засобу» затверджена наказом ректора № 1774/ст від 28.09.2022р.

2. Термін виконання роботи: з 26 вересня 2022 року по 27 листопада 2022 року.

3. Вихідні дані до роботи: програмний продукт розробити для всіх громадян, які проживають на території Києва.

4. Зміст пояснювальної записки: вступ, комп'ютерний зір його застосування, оптичне розпізнавання символів, структура системи автоматизованого визначення маршруту за номером транспортного засобу ,специфікація вимог до системи, прототип, висновки.

5. Перелік обов’язкового ілюстративного матеріалу: актуальність теми, комп’юерний зір і його застосування, обмеження додатку, технології які використовуються в додатку, вимоги до додатку, функціональна схема, рекомендації до використання, результати.

6. Календарний план-графік

№ п/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Проаналізувати літературу та джерела за темою дипломної роботи	26.09.22 – 01.11.22р.	
2.	Розроблення та затвердження плану дипломної роботи	02.11.22 – 04.11.22р.	
3.	Привести консультації з науковим керівником щодо створення першого розділу	05.11.22 – 06.11.22р.	
4.	Розробка розділу 1	30.09.22 – 10.10.22р.	
5.	Розробка розділу 2	10.10.22 – 20.10.22р.	
6.	Розробка розділу 3	20.10.22 – 31.10.22р.	
7.	Розробка розділу 4	31.10.22 – 04.11.22р	
8.	Висновки та оформлення пояснювальної записки дипломної роботи	4.11.22 – 06.11.22р	
9.	Підписання необхідних документів у встановленому порядку	16.11.22 –18.11.22р.	
10.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломної роботи	19.11.22 – 22.11.22р.	

7. Дата видачі завдання 26 вересня 2022р.

Керівник дипломної роботи _____ Тетяна ХОЛЯВКІНА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Мирон ІВАЩЕНКО
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Система автоматизованого визначення за номером транспортного засобу» складається зі вступу, чотирьох розділів, висновку, списку біографічних посилань і містить 76 сторінки, 1 таблицю, та 24 рисунок. Список бібліографічних посилань складається з 9 найменувань.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, НОМЕР ТРАНСПОРТНОГО ЗАСОБУ, ІНДЕТИФІКАЦІЯ, КОМП'ЮТЕРНИЙ ЗІР.

Актуальність роботи. Концепт комп'ютерного зору і його застосувань в реальному світі, комп'ютерного зору стрімко розвивається і багато де використовується в нашому житі.

Мета дипломної роботи: надання актуальної інформації про маршрут в місті Києві, розробка та тестування системи автоматизованого визначення за номером транспортного засобу.

Об'єкт дослідження: система автоматизованого визначення за номером транспортного засобу.

Предмет дослідження: розробка системи автоматизованого визначення за номером транспортного засобу.

Теоретичною основою дипломної роботи стали вітчизняні та зарубіжні

дослідження комп'ютерного зору, збірки статей наукових конференцій, тематичні публікації на сайтах.

Теоретична і практична значимість: люди не завжди точно знають, як дістатися до потрібного місця, наприклад, якщо вони іноземці і нічого не знають про маршрути місцевих транспортних засобів, тож за допомогою програми можна зрозуміти, чи буде конкретний транспортний засіб доставляти користувача до потрібного місця. На основі отриманих знань можна пободувати свою програмну реалізацію додатка системи автоматизованого визначення за номером транспортного засобу. По зібраному матеріалу можна сформувати методичний посібник для студентів

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ШІ – штучний інтелект.

КЗ – комп'ютерний зір.

РНМ – рекурентна нейронна мережа.

AWS – Amazon Web Services.

МА – Microsoft Azure.

КС – клієнт-сервер.

API – програмний інтерфейс програми.

ОРС – оптичне розпізнавання символів.

РО – розпізнавання образів.

JAVA – об'єктно-орієнтована мова програмування.

HTTP – протокол.

МТ – маршрутний транспорт.

МД – мобільний додаток.

ВСТУП

У цьому проекті була розкрита і покрита важлива тема в наш час – комп'ютерний зір. У першій частині концепт КЗ і його застосувань в реальному світі був досліджений для того, щоб розуміти актуальність, сучасні завдання комп'ютерного зору, прикладів його використання, причини з приводу питання, чому людям треба це і чому важливо розширити область комп'ютерного зору.

Технологія штучного інтелекту здатна докорінно змінити роботу компанії. Важливо зазначити, що рішення на основі штучного інтелекту повинні створюватися з урахуванням потреб компанії. Ця технологія не може бути розгорнута як універсальне рішення, незважаючи на поширену помилку. Сьогодні впроваджувати рішення штучного інтелекту на підприємстві стало простіше і доступніше, і це дає цілу низку переваг.

КЗ це область штучного інтелекту, яка дозволяє комп'ютерам і система м витягати значиму інформацію з цифрових зображень, відео і інших візуаль них даних, а також робити дії або давати рекомендації на основі цієї інформа ції. Якщо штучний інтелект дозволяє комп'ютерам думати, то комп'ютерний зір дозволяє їм бачити, спостерігати і розуміти.

Останнім часом алгоритми, які працюють з зображеннями та об'єднати кольорову інформацію про точки та їх просторове положення, стали дуже популярними, оскільки датчики, які отримують таку інформацію, стали набагато доступними та потужними, і це стало набагато простіше для роботів або комп'ютерів, щоб зрозуміти світ. Також розглядається розпізнавання символів, це є частиною комп'ютерного зору, оскільки цей концепт був використаний впродовж розвитку додатка Android з розпізнаванням номерів від камери смартфону.

У другій частині розповідь про структуру системи автоматизованого визначення за номером транспортного засобу. Описані функції ПЗ.

Користувач може навести камеру на номер транспортного засобу та відсканувати номер транспорту за допомогою камери смартфону.

Третя частина включає в собі специфікацію вимог до системи автоматизованого визначення за номером транспортного засобу. Концепція взаємодії клієнт-серверу. Технічні характеристики системи індетифікації. Технології на яких працює МД і платформа для реалізаціїї додатку. Є інформація про ОС Android, оскільки це – операційна система, на якій написаний додаток, деякі його особливості і важливі факти про Android.

Додаток має три компонента: сервіс обробки даних, сервіс збору даних, інтерфейс. Була надана інформація з API, а саме: Номер маршруту, інтервал руху, час роботи, максимальна вартість проїзду, назва маршруту, перелік вулиць.

Четверта частина включає в собі прототип і основний код класів. У наш час мобільні пристрої досить потужні, вони мають щось схоже на очі і можуть розпізнавати предмети та отримувати деяку інформацію на основі цього предмета.

РОЗДІЛ 1. КОМП'ЮТЕРНИЙ ЗІР І ЙОГО ЗАСТОСУВАННЯ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ

1.1. Концепт Комп'ютерного Зору

Значна тема – комп'ютерний зір. Була досліджена концепція комп'ютерного зору та застосування в реальному світі, щоб зрозуміти актуальність цієї галузі, прикладів його використання, причини, що стосуються питання, чому люди потребують цього, і чому важливо розширити область комп'ютерний зору.

КЗ – це дисципліна (розділ штучного інтелекту), який витягує інформацію з зображень, а зображення можуть бути різними за типом. Можуть бути фотографії, відео, набори фотографій або зображення для медичних цілей і інших галузей. Останнім часом алгоритми, які працюють з зображеннями та об'єднати кольорову інформацію про точки та їх просторове положення, стали дуже популярними, оскільки датчики, які отримують таку інформацію, стали набагато доступними та потужними, і це стало набагато простіше для роботів або комп'ютерів, щоб зрозуміти світ. Комп'ютерний зір працює приблизно так само, як і людське, за винятком того , що у людини є перевага. Людський зір має перевагу, що полягає в тому, що упродовж усього життя людина вчиться розрізняти об'єкти, визначати відста нь до них, їх рух і те, що в зображенні щось не так.

Інформація, яку витягують різні алгоритми з зображень у комп'ютерному баченні, також може мати інший тип. Деякі алгоритми просто розділяють зображення на частини, які відповідають окремим об'єктам або різним частинам об'єктів.

Наприклад, лікар може прийняти медичний образ або його частину, розбити його в інші частини і взяти той, який відповідає об'єкту інтересу, орган або пухлина, а потім вимірює об'єм або вимірюють діаметр цього об'єкта. У цьому випадку зберігається час мед. працівника.

Іншим важливим розділом КЗ є побудова тривимірних моделей з зображень, наприклад, з набору фотографій або з відео, і фотографії можуть бути прийняті одним фотографом або завантаженим з інтернету. Наприклад, тисячі фотографій, зроблених різними фотографами. Або ці дані можуть бути отримані роботом, що подорожує по незнайомому приміщенні або в деякій незнайомій області, і намагаючись побудувати тривимірну модель під час подорожі. А для того, щоб побудувати таку тривимірну модель, комп'ютер повинен відображати частини сцени, тобто, щоб зрозуміти, що в різних образах різні фрагменти відповідають тим самим об'єктах у тривимірному світі.

Це насправді дуже важко зробити. Наприклад, якщо є будівля, то нижній лівий кут вікна виглядатиме однаково для різних вікон цієї будівлі, і якщо просто подивитися на кілька пікселів, які оточують цей кут вікна, то ні людині, ні комп'ютер не зрозуміє, чи є це різні вікна. Тому для того, щоб побудувати таку модель, щоб відповідати таким частинкам зображень, комп'ютер повинен поступово побудувати модель, засновану на більш унікальних фрагментах сцени, а потім, порівняння більш унікальних фрагментів, поширюючи це знання менш унікальними . Комп'ютер використовує алгоритми оптимізації, але люди не повністю розуміють, як комп'ютери досягають цього при будівництві тривимірних "карток" у мозку.

В результаті цього комп'ютер фактично отримує тривимірну картинку сцени, і це все дуже важливо, комп'ютер розуміє позицію камери на даний момент, коли фото або відео було зроблено. І тоді ця інформація може бути використана різними способами. Наприклад, можна виміряти розміри на сцені. Або, отримана тривимірна модель та положення сцени, можуть бути

використані в комп'ютерній графічній програмі. Наприклад, такі системи використовуються у всіх сучасних фільмах, які використовують спеціальні ефекти, які поєднують реальні та віртуальні об'єкти. Крім того, тривимірна модель незабаром буде все більше, ніж у комп'ютерних іграх, реальних та збільшених додатках реальності.

Люди дуже часто змішують комп'ютерну графіку та комп'ютерний зір, хоча це є протилежним. Комп'ютерна графіка приймає опис сцени, наприклад, геометрія сцени, властивості матеріалів та створює фотографії відповідно до цього опису. Комп'ютерний зір, навпаки, робить фотографії та витягує опис сцени від них. Правда, існують системи, які після зйомки, отримавши його опис, використовує цей опис далі, щоб ідентифікувати вхідне зображення, наприклад, щоб намалювати маску над обличчям. І виявляється така система, яка поєднує в собі графіку та бачення, але вони можуть розглядатися як розширена версія системи обробки зображень.

Наприклад, при переході або проїзді вулиці за сигналами світлофора. Розпізнавання кольору лампи світлофора, що засвітилася, і знання правил дорожнього руху дозволяє прийняти правильне рішення про те, можна, чи не можна переходити вулицю в цей момент. Для оптичного РО можна застосувати метод перебору вигляду об'єкта під різними кутами, масштабами, зсувами. Для букв потрібно перебирати шрифт, властивості шрифту.

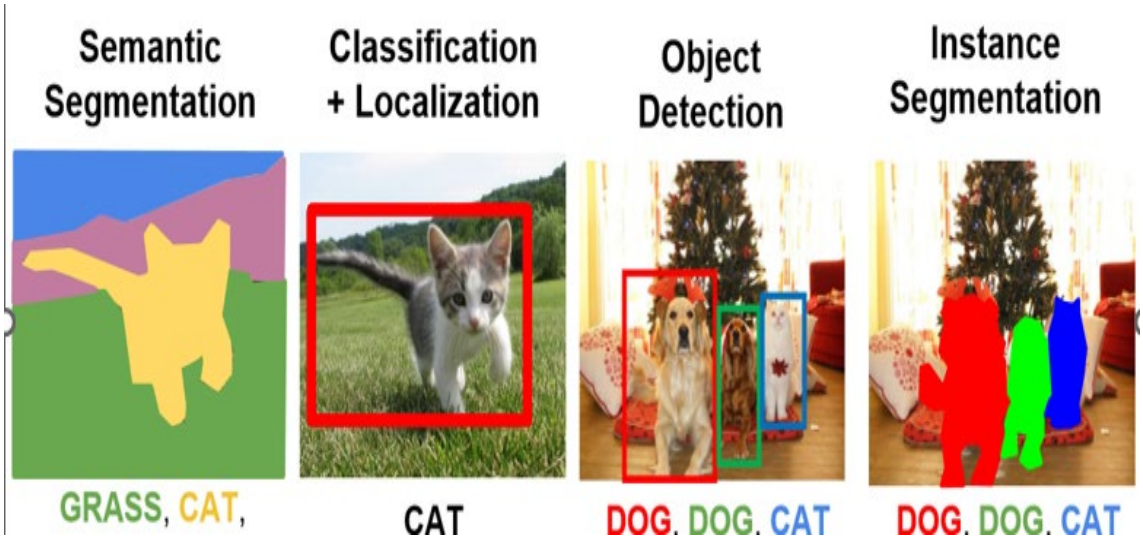

Рис. 1.1. Розпізнавання комп’ютерним зором

1.2. Області застосування комп'ютерного зору

Комп'ютерний зір потребує багато даних. Він аналізує дані знову і знову, поки не побачить відмінності і не розпізнає зображення. Наприклад, щоб навчити комп'ютер розпізнавати автомобільні шини, йому потрібно надати величезну кількість зображень шин і предметів, пов'язаних з шинами, щоб він міг побачити відмінності і розпізнати шину, особливо ту, яка не має дефектів.

Приклади комп'ютерного зору

1) Google Translate: У 2015 році технологічний лідер Google запустив сервіс миттєвого перекладу, який використовує комп'ютерний зір через камери смартфонів. Нейронний машинний переклад, ключова система, яка забезпечує миттєвий і точний переклад на основі комп'ютерного зору, була включена до веб-результатів Google Translate у 2016 році. З появою доступних хмарних рішень компанії різних вертикалей можуть легко використовувати ШІ і глибоке навчання за власним бажанням. Ці сервіси пропонують готові рішення, такі як розпізнавання зображень, розпізнавання голосу, чат-боти, предиктивні моделі тощо.

2) Amazon: До того, як Amazon впровадив штучний інтелект, він вже був одним з провідних світових маркетплейсів електронної комерції. У 2014 році компанія почала рухатися в напрямку глибокого навчання для механізмів рекомендацій – основної складової веб-сайту та його послуг сьогодні. Раніше це було нечувано в корпоративному просторі, і багато компаній не знали, що глибоке навчання можна використовувати для цієї мети. Це ознаменувало перехід компанії на ШІ, і сьогодні ця технологія є невід'ємною частиною ДНК Amazon. Перше місце, де можна побачити ШІ, це головна сторінка веб-сайту Amazon, яка унікально адаптована для кожного з 300 мільйонів користувачів. За допомогою звичок користувачів в Інтернеті Amazon може точно передбачити, що вони, найімовірніше, придбають. Сайт побудований навколо цього і розміщує кілька рекомендованих продуктів на видному місці, що допомагає користувачам помітити їх. Втім, оновлення сайту – це лише одна сторона успіху Amazon у сфері ШІ. Використовуючи алгоритми обробки природної мови і розпізнавання мови, компанія розробила і випустила Amazon Echo. Echo – це пристрій для розумного будинку, який ділиться інформацією за допомогою природної розмови. Після низки придбань у сфері штучного інтелекту компанія випустила Alexa, щоб збільшити залученість користувачів на всіх своїх платформах. Сервіс дозволяє користувачам легко отримати доступ до Amazon Music, Amazon Prime Video і навіть розміщувати замовлення на товари, доступні на сайті. Третім і, мабуть, найважливішим кроком Amazon в напрямку ШІ стало пропозицію машинного навчання як хмарного сервісу. Через свій корпоративний хмарний підрозділ, відомий як AWS, Amazon почав пропонувати доступні рішення ШІ для вирішення типових проблем підприємств. Використовуючи свою значну внутрішню інфраструктуру хмарних обчислень і поєднуючи її з розробленими рішеннями ШІ, Amazon змогла скористатися початковою частиною хвилі ШІ.

3) Anheuser-Busch InBev: найбільший у світі виробник пива. Компанія варить понад 500 відомих марок пива, таких як Budweiser, Corona,

Stella Artois, Beck's, Hoegaarden та інші. Компанія має глобальне охоплення, працюючи в більш ніж 50 країнах світу, а її бренди добре відомі серед широких верств населення. Маючи такий масштаб діяльності, компанія не одразу зрозуміла потенційні переваги впровадження штучного інтелекту. Використовуючи сервіси MA та Google Cloud Platform, AB InBev впровадила аналітику, штучний інтелект та прогнозне моделювання у свої процеси. Вони почали з використання платформи, відомої як Smart Barley, для заохочення сталого і здорового вирощування ячменю, невід'ємного інгредієнта пива. Ця платформа має на меті зменшити використання води і добрив і водночас максимізувати врожайність, використовуючи дані минулих врожаїв. Smart Barley має алгоритм штучного інтелекту, який аналізує ці дані і має на меті дозволити фермерам збирати врожай на стійкій основі. AB InBev також використовувала хмарну платформу Google для покращення процесу фільтрації пива. Останній етап фільтрації пива відомий як K-фільтр, який видаляє будь-які залишки процесу пивоваріння для створення кінцевого продукту. Збираючи дані з датчиків у фільтраційному обладнанні, щоб знайти оптимальну точку для фільтрації готового продукту. Це не тільки знизило витрати на його фільтрацію, а й дало AB InBev перевагу над конкурентами.

4) Starbacks: Starbucks розпочала свою діяльність як проста кав'ярня у Сіетлі майже 40 років тому і з того часу перебуває в авангарді культури завдяки своїй цільовій аудиторії – молодим міленіалам. Сьогодні вона має понад 30 000 закладів по всьому світу і є третьою за величиною мережею ресторанів швидкого харчування у світі. Мережа ресторанів швидкого харчування шукає нові способи використання величезного обсягу даних, які пропонують її клієнти. У 2018 році повідомлялося, що компанія здійснює понад 90 мільйонів транзакцій через свої магазини. Це величезний масив інформації щодо купівельних звичок клієнтів та продуктів, які мають найбільший попит. Starbucks почав збирати ці дані та отримувати інсайти за допомогою штучного інтелекту. Ці інсайти дозволили компанії приймати

кращі рішення, коли справа стосувалася продажів і загального напрямку бізнесу. Крім того, вони також покращили свої можливості прямого маркетингу, що дозволило використовувати таргетовану рекламу та інші подібні стратегії. Мобільний додаток Starbucks також є невід'ємною частиною їхнього руху до ШІ та машинного навчання. Маючи 17 мільйонів користувачів додатку, компанія може не тільки збирати дані, але й більш цілеспрямовано звертатися до користувачів. У додатку також є віртуальний бариста, який допомагає користувачам розміщувати замовлення, забезпечуючи таким чином дійсно безперебійний досвід на основі ШІ.

5) Twitter: Twitter вже давно є популярною соціальною мережею і з'явився в поп-культурі як альтернатива Facebook. Однак у другій половині 2010-х років, у міру свого зростання, платформа почала спостерігати збільшення кількості мови ненависті, тероризму та незаконного контенту. Оскільки відповідальність за такий контент лягла на компанію, вона почала використовувати інструменти штучного інтелекту та машинного навчання, щоб переосмислити способи роботи з таким контентом. У 2016 році Twitter придбав компанію, відому як Magic Pony Technology, для вирішення цих проблем, передавши таланти своїй власній інженерній команді. Платформа не тільки інтенсивно використовує механізми рекомендацій, але й використовує їх як інструмент для фільтрації небажаного контенту, тим самим запобігаючи його вірусному поширенню з неправильних причин. Часова шкала Твіттера також зазнала змін – вона перейшла від показу останніх твітів до складної системи рекомендацій, пристосованої до звичок кожного користувача у Твіттері. Ці нові зміни, схоже, допомогли компанії довше утримувати користувачів, оскільки наприкінці 2018 року веб-сайт повідомив про 126 мільйонів щомісячних щоденних активних користувачів. Це на 9% більше, ніж у попередньому році, і компанія отримала перший прибутковий рік у 2018 році.

Машинне навчання використовує алгоритмічні моделі, які дозволяють комп'ютеру навчитись розуміти контекст візуальних даних. Якщо через модель подається достатня кількість даних, комп'ютер "дивиться" на дані і навчається відрізняти одне зображення від іншого. Алгоритми дозволяють машині навчатися самій, а не тому, що хтось програмує її для розпізнавання зображення.

КЗ допомагає моделі машинного навчання або глибокого навчання "дивитися", розбиваючи зображення на пікселі, яким присвоюються теги або мітки. Вона використовує мітки для виконання згорток (математична операція над двома функціями для отримання третьої функції) і робить прогнози щодо того, що вона "бачить". Нейронна мережа виконує згортки і перевіряє точність своїх прогнозів в серії ітерацій, поки прогнози не почнуть збуватися. Тоді вона розпізнає або бачить зображення подібно до людини.

Подібно до того, як людина розрізняє зображення на відстані, КЗ спочатку розрізняє тверді краї і прості форми, а потім заповнює інформацію, виконуючи ітерації своїх прогнозів. РНМ використовується аналогічним чином для відео додатків, щоб допомогти комп'ютерам зрозуміти, як зображення в серії кадрів пов'язані один з одним.

Найбільш очевидна область застосування КЗ це робототехніка. Перші системи розпізнавання об'єктів на зображенні були пов'язані з промисловістю. Завданням роботів була перевірка відповідності деталі на конвеєрі заданим шаблоном.

Але з часом роботи з'явилися і в масовому виробництві, для застосування в побуті. І тут вони повинні були бачити не деталь, що лежить на конвеєрі, а складні тривимірні сцени, що складаються з безлічі предметів, вміти розпізнавати обличчя людей, їх емоції, орієнтуватися в просторі за допомогою комп'ютерного зору.

Друге перспективний напрямок – системи Big Data. Тут головні гравці – величезні корпорації, наприклад, Google і Facebook, яким щодня доводиться розпізнавати мільярди зображень.

Також КЗ є дуже корисним в системах допомоги водієві. Ми говоримо про розпізнавання дорожніх знаків і розмітки. Цікава розробка в цій області – безпілотний автомобіль Google. Для розпізнавання об'єктів він використовує цілий комплекс датчиків, радарів і відеокамер, але все ж не може рухатися на незнайомих дорогах або при поганій видимості.

Контурний аналіз – це метод розпізнавання і пошуку об'єктів по їх контурах (іншими словами, по кривій, яка окреслює межі об'єкта на зображенні). Головна перевага контурного аналізу – стабільність результатів при зміні масштабів об'єкта або його зміщення. Але є і певні обмеження:

• контур об'єкта може бути зашумлен перешкодами або мати однакову яскравість з фоном, в цьому випадку виділити його буде неможливо

• при накладенні декількох об'єктів один на інший контур буде визначатися некоректно.

Незважаючи на ці нюанси, контурний аналіз добре себе зарекомендував при розпізнаванні об'єктів з чіткими кордонами, на контрастному фоні. Приклад: розпізнавання друкованого тексту.

Template Matching застосовується для розпізнавання зображень, аналогічних якомусь заданим шаблоном. Вхідні параметри – це власне зображення, на якому потрібно знайти об'єкт, і шаблон цього самого об'єкта.

І ще один алгоритм – зіставлення по ключових точках. Алгоритм обчислює на зображенні ключові особливості, які потім використовуються для порівняння двох картинок і визначення у них загальних складових.

На відміну від контурного аналізу і пошуку шаблонів, зіставлення по точкам стійко до перешкод і трансформацій. При цьому алгоритм працює настільки швидко, що його можна застосовувати в режимі реального часу.

Технологія КЗ дозволяє розпізнати ті ділянки на обличчі, де найяскравіше відслідковуються руху мімічних м'язів. Для наочності, м'язи закріплюються точками, які рухаються під час розмови (технологія доповненої реальності). І, нарешті, для більшої достовірності в нашому додатку відстежуються рух зіниць і пульс.

Зараз всі люди користуються смартфоном. Продажі смартфонів зростають з блискавичною швидкістю, а компанії-виробники вкладають величезні суми в мобільні технології, їх розвиток і популяризацію на ринку. Мобільна розробка – дуже стрімко зростаюча область програмування, адже кількість мобільних пристроїв значно перевищує кількість персональних комп'ютерів, і ця тенденція буде тільки зростати.

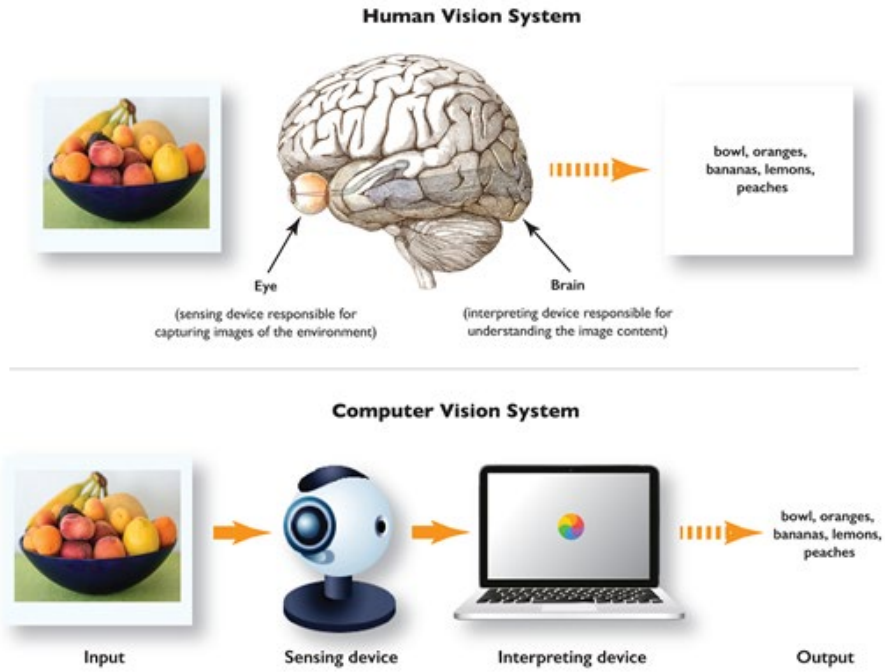

Рис. 1.2. Приклад розпізнавання обєктів КЗ

1.3. Додатки комп'ютерного зору

КЗ – фокусується на використанні автономних роботів та систем візуального контролю та вимірювання. Це означає, що технологія роботи датчиків зображення і теорії управління пов'язана з обробкою відеоданих для

управління роботом, а обробка отриманих даних в реальному часі здійснюється програмним або апаратним забезпеченням.

КЗ також фокусується на обробці тривимірних сцен, розроблених для одного або декількох зображень. Наприклад, шляхом відновлення структури або іншої інформації про 3D-сцену з одного або декількох зображень.

Візуалізація, яка спочатку була пов'язана з процесом створення зображень, але іноді мала справу з обробкою та аналізом. Наприклад, рентгенографія працює з аналізом відеоданих для медичного використання.

РО – це область, яка використовує різні методи отримання інформації з відеоданих, в основному на основі статистичного підходу. Значна частина цієї галузі присвячена практичному застосуванню цих методів [1].

КЗ, вже зараз має величезний розподіл у всіх сферах людської діяльності: від систем обробки текстових друкованих документів до систем розпізнавання обличчя людини або жестів. Простим прикладом використання КЗ в звичайному житті може служити система виявлення та зчитування штрих-коду. Штрих-код – найпоширеніший спосіб розпізнавання деяких товарів. Штрих-код – це послідовність чорно-білих смуг, що представляє деяку інформацію, зручну для читання технічними засобами. У цьому випадку певна лінія сканується у потрібній області, а потім обчислюється ширина штрихів за вказаними траєкторіями. Також КЗ використовується для перевірки зібраних комп’ютерних плат, зварювальних швів, якості кристалів, сортувальних предметів, розташованих на конвеєрній стрічці [2].

Класифікація зображень бачить зображення і може його класифікувати (собака, яблуко, обличчя людини). Точніше, вона здатна точно передбачити, що дане зображення належить до певного класу. Наприклад, компанія, що займається соціальними мережами, може використовувати його для автоматичного виявлення та відокремлення небажаних зображень, завантажених користувачами.

Виявлення об'єктів може використовувати класифікацію зображень для ідентифікації певного класу зображень, а потім виявляти і табулювати їх

появу на зображенні або відео. Приклади включають виявлення пошкоджень на складальній лінії або ідентифікацію машин, які потребують технічного обслуговування.

Відстеження об'єктів слідує за об'єктом або відстежує його після його виявлення. Це завдання часто виконується за допомогою зображень, захоплених послідовно, або відеопотоків в реальному часі. Автономні транспортні засоби, наприклад, повинні не тільки класифікувати і виявляти такі об'єкти, як пішоходи, інші автомобілі та дорожню інфраструктуру, вони повинні відстежувати їх в русі, щоб уникнути зіткнень і дотримуватися правил дорожнього руху.

Пошук зображень на основі вмісту використовує комп'ютерний зір для перегляду, пошуку і вилучення зображень з великих сховищ даних на основі вмісту зображень, а не тегів метаданих, пов'язаних з ними. Це завдання може включати автоматичну анотацію зображень, яка замінює ручне тегування зображень. Ці завдання можуть бути використані для систем управління цифровими активами і можуть підвищити точність пошуку і вилучення.

Область КЗ настільки широка, що включає навіть медицину. Комп’ютерна томографічна дозволяє отримати детальне зображення судин та оцінити характер кровотоку. Перевага цього методу: усунення ризику ускладнень від хірургічних маніпуляцій, необхідних при нормальній ангіографії, незначне променеве навантаження на пацієнта. КТ-ангіографія дозволяє отримати пошаровий ряд зображень судин; на основі даних, отриманих за допомогою комп'ютерної пост-обробки з 3D-реконструкцією, будується тривимірна модель системи кровообігу [3].

За допомогою модуля розпізнавання автомобілів номери автоматично визначаються та розпізнаються за номерами автомобілів у полі зору камери. Це дозволяє записати та зберегти розпізнаний номер у базі даних, а також зображення частини рами транспортного засобу з номерним знаком та часом реєстрації.

Найпотужнішими технічними перспективами є розробки в галузі розпізнавання жестів рукою людини та біомеханічних досліджень. Системи для біомеханічних досліджень, призначені для реєстрації та аналізу характеристик руху людини. Головною метою є швидкісне відстеження положення кінцівок та виділення характерних рис людського обличчя. Всі ці параметри можуть бути використані в системах безпеки.

1.4. Компоненти системи комп'ютерного зору

Комп'ютерний зір використовує наступні компоненти:

1. Одна або кілька цифрових або аналогових камер з відповідною оптикою для візуалізації.
2. Програмне забезпечення для створення зображень для обробки. Для камер аналога, це цифровий перетворювач зображення.
3. Процесори(сучасний ПК з багатоядерним процесором або вбудовуваним процесором).
4. Програмне забезпечення КЗ, яке забезпечує інструменти для розвитку індивідуальних додатків програмного забезпечення.
5. Додатки специфічного програмного забезпечення для обробки зображень.
6. Датчик для синхронізації деталей виявлення (часто оптичного або магнітного датчика) для обробки зображень [4].

1.5. Завдання комп'ютерного зору

Обробка зображень як складова комп’ютерного зору. Цифрова обробка зображень, або обробка зображень, є підмножиною комп'ютерного зору. Вона займається покращенням і розумінням зображень за допомогою різних алгоритмів. Це не просто підмножина, обробка зображень є попередником сучасного комп'ютерного зору, контролюючи розробку численних

алгоритмів, заснованих на правилах і оптимізації, які привели машинний зір до того, яким він є сьогодні. Обробка зображень може бути визначена як завдання виконання набору операцій над зображенням на основі даних, зібраних алгоритмами для аналізу і маніпулювання вмістом зображення або даними зображення [5].

Зображення складається з декількох пікселів, причому піксель – це найменший квант, на який можна розбити зображення. Комп'ютери обробляють зображення у вигляді масиву пікселів, де кожен піксель має набір значень, що представляють наявність та інтенсивність трьох основних кольорів: червоного, зеленого та синього. Всі пікселі збираються разом, щоб сформувати цифрове зображення. Цифрове зображення, таким чином, стає матрицею, а комп'ютерний зір – дослідженням матриць. У той час як найпростіші алгоритми комп'ютерного зору використовують лінійну алгебру для маніпулювання цими матрицями, складні програми включають такі операції, як згортки з ядрами, що навчаються, і зменшення вибірки через об'єднання.

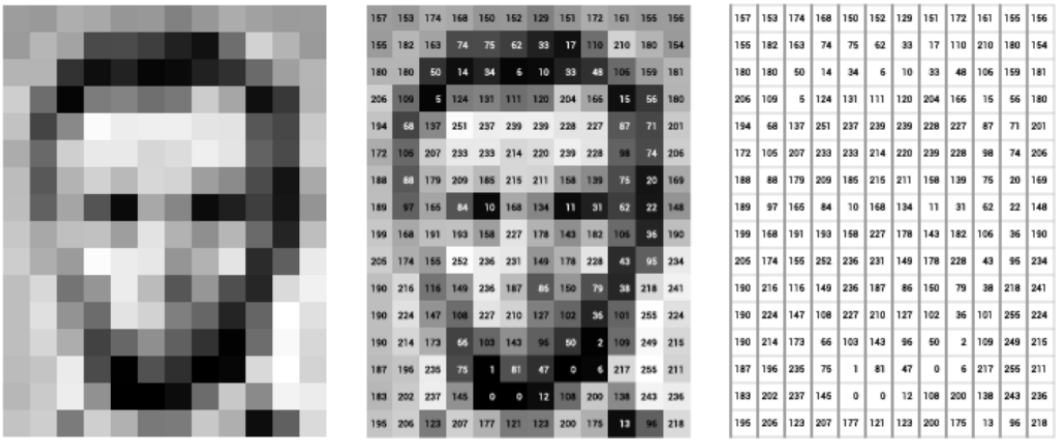

Рис. 1.3. Приклад розпізнавання обєктів КЗ

Пошук зображень в Інтернеті. Зараз існує кілька сервісів, що дозволяють шукати фотографії. Спочатку для пошуку використовувались

текстові запити. Деякий час тому в частині цих служб стало можливим пошук завантаженого зображення. Користувач повинен завантажити зображення, і служба буде шукати подібні до нього зображення в Інтернеті.

Обробка зображень як складова комп’ютерного зору

Такий пошук працює наступним чином. Спочатку зображення з Інтернету індексуються. Для них будуються цифрові уявлення, з них формується структура даних, по якій можна швидко шукати. Те ж саме відбувається із зображенням користувача – витягується якесь уявлення про нього, згідно з яким дублікати або подібні картинки шукаються в базі даних. Це завдання є складним у структурному сенсі. Мільярди зображень завантажуються в Інтернет, і використання складних методів порівняння неможливе, оскільки необхідна висока продуктивність.

Загалом, завдання комп’ютерного зору можна згрупувати за такими категоріями:

1) Визнання. Класичною проблемою в обробці зображень та комп'ютерному зорі є визначення, чи містять відеодані якийсь характерний об'єкт, ознаку чи діяльність.

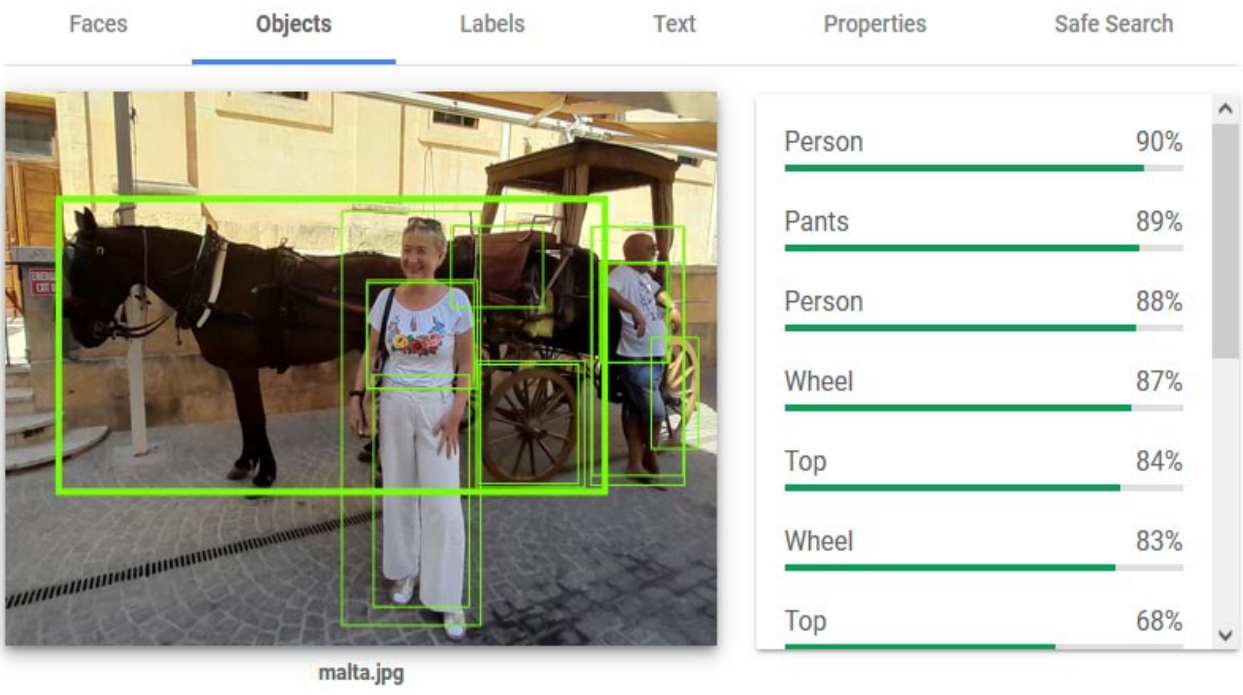

Рис. 1.4. Розпізнавання комп’ютерним зором обєктів

2) Ідентифікація. Розпізнається окремий екземпляр об’єкта. Приклади – ідентифікація конкретного людського обличчя, відбитків пальців чи автомобіля.

3) Виявлення. Відеодані перевіряються на конкретний стан. Виявлення, засноване на відносно простих і швидких розрахунках, іноді використовується для знаходження невеликих ділянок в аналізованому зображенні, які потім аналізуються з використанням методів, які вимагають більше ресурсів, щоб отримати правильну інтерпретацію [6].

4) Розпізнавання тексту. Пошук зображень за вмістом: пошук усіх зображень у великому наборі зображень, вміст яких визначений різними способами. Оцінка позиції: визначення положення або орієнтації конкретного об’єкта щодо камери. OPC: розпізнавання символів на зображеннях набраного або рукописного тексту (зазвичай для перекладу). Іноді потрібно знайти зображення тексту на малюнку і представити його у вигляді текстових даних, з якими можна буде працювати, наприклад, у текстовому редакторі. Ця технологія широко використовується в різних додатках. Зокрема, це зручний спосіб введення тексту в Інтернет перекладач. Досить сфотографувати картинку з текстом і текст на ньому буде розпізнано, і перекладач виконає переклад.

5) Відновлення сцени. Відновлення сцени має завдання відтворити тривимірну модель сцени. Більш складні методи відтворюють повну тривимірну модель.

6) Відновлення зображень. Завданням відновлення зображення є видалення шуму (шум датчика, розмиття руху). Найпростішим підходом до вирішення цієї проблеми є різні типи фільтрів. Високий рівень видалення шуму досягається під час первинного аналізу відеоданих на наявність різних

структур, таких як лінії або межі, а потім управління процесом фільтрації на основі цих даних.

7) Оптичний аналіз потоку означає пошук руху пікселів між двома зображеннями і включає кілька завдань, пов'язаних з оцінкою руху, в яких послідовність зображень (відеоданих) обробляється для знаходження оцінки швидкості кожної точки зображення або 3D сцени. Прикладами таких завдань є визначення тривимірного руху камери, відстеження, тобто відстеження руху предмета [9].

8) Керування автомобілем. Надалі будь-який автомобіль буде оснащений величезною кількістю датчиків: відеокамер, радарів, тощо. Методи комп'ютерного зору допомагають аналізувати інформацію від цих датчиків і лежать в основі систем запобігання аварій і все більш складних автопілотів.

1.6. Оптичне розпізнавання символів

ОРС або оптичний зчитувач символів – це механічне або електронне перетворення зображень, текстів, написаних від руки або надрукованих у машинно закодований текст.

Технологія оптичного розпізнавання символів – це ефективний бізнес-процес, який економить час, кошти та інші ресурси завдяки використанню можливостей автоматизованого вилучення та зберігання даних.

Серед переваг використання технології оптичного розпізнавання тексту можна виділити наступні:

- Зниження витрат
- Прискорення робочих процесів
- Автоматизація маршрутизації документів та обробки контенту
- Централізація та безпека даних (відсутність пожеж, зломів та втрати документів у підсобних сховищах)

- Покращити обслуговування, забезпечивши співробітників найактуальнішою та найточнішою інформацією.

Системи оптичного розпізнавання тексту використовують комбінацію апаратного та програмного забезпечення для перетворення фізичних друкованих документів у машинозчитуваний текст. Апаратне забезпечення – наприклад, оптичний сканер або спеціалізована плата – копіює або зчитує текст, потім програмне забезпечення, як правило, виконує подальшу обробку.

Розпізнавання за зразком використовується, коли в програму подаються приклади тексту різних шрифтів і форматів для порівняння і розпізнавання символів у відсканованому документі або файлі зображення.

Виявлення ознак відбувається, коли ОРС застосовує правила щодо ознак конкретної літери або цифри для розпізнавання символів у відсканованому документі. Особливості включають в себе кількість нахилених ліній, перехрещених ліній або кривих в символі. Наприклад, велика літера "A" зберігається у вигляді двох діагональних ліній, які перетинаються горизонтальною лінією посередині. Коли символ ідентифіковано, він перетворюється в код ASCII (American Standard Code for Information Interchange – Американський стандартний код для обміну інформацією), який комп'ютерні системи використовують для подальших маніпуляцій.

Коли хтось читає слова на екрані комп’ютера, його очі та мозок виконують роботу розпізнавання символів. Очі розпізнають слова, зображення, кольори, візерунки тощо. Вони діють як сканер, одночасно скануючи цілу групу слів. Комп’ютери можуть це робити теж, але не так швидко, як людські очі.

Комп’ютери повинні працювати більше, ніж люди, щоб сканувати будь-який документ, текст. Тож, якщо комп’ютер хоче прочитати стару книгу чи текст, потрібно представити йому зображення тієї сторінки, яку генерує оптичний сканер.

Щоб перетворити всі паперові документи в машиночитаний формат, потрібно відсканувати всі документи за допомогою хорошого сканера.

1.7. Важливість оптичного розпізнавання символів

Часи, коли всі документи організації зберігалися лише у паперовій формі – це в минулому.

В даний час майже всі організації використовують електронний документообіг. Це дозволяє їм найшвидше обробляти документи та вести детальний звіт про все.

Згідно з останніми дослідженнями, компанії зазвичай втрачають близько 10 відсотків прибутку лише через те, що під час ручної обробки документів люди роблять деякі помилки. Другий цікавий факт – 40 відсотків робочого часу витрачається на пошук оригіналу документа (у паперовій формі), якщо відсутній електронний документообіг. Дотримуючись цих фактів, можна оцінити в грошовому вираженні вигоди від впровадження систем визнання документа.

Під розпізнаванням документів розуміється, насамперед, переведення документа в оптичну копію (тобто створення малюнка), а потім використання різних програм, що витягують із цієї картинки текстову та іншу інформацію.

Найбільш відомим випадком використання ОРС є перетворення друкованих паперових документів у машинозчитувані текстові документи. Після того, як відсканований паперовий документ проходить обробку ОРС, текст документа можна редагувати за допомогою текстового процесора, такого як Microsoft Word або Google Docs.

Сучасні досягнення в розпізнаванні текстової та текстово-графічної інформації дозволяють конвертувати документи з майже нульовою ймовірністю помилок розпізнавання. Систему розпізнавання можна налаштувати для всіх основних типів паперу, що зберігаються в організації.

Крім того, можна інтегрувати електронну систему управління документами та системи розпізнавання документів.

Тобто розпізнані документи автоматично після розпізнавання та перевірки помилок потраплять в електронну систему документів.

Одне з важливих завдань – пошук архіву документів. Наприклад, у фінансовій звітності людям часто доводиться шукати інформацію про певні документи. А за відсутності електронної копії для бухгалтера знадобиться значний час, щоб знайти документ, необхідний протягом 5 хвилин. Якби існувала електронна версія документа, пошук зводився б до натискання декількох кнопок.

Завдяки розпізнаванню тексту ОРС відскановані документи можуть бути інтегровані в систему великих даних, яка тепер здатна зчитувати дані клієнтів з банківських виписок, контрактів та інших важливих друкованих документів. Замість того, щоб співробітники вивчали незліченну кількість графічних документів і вручну вводили дані в автоматизований робочий процес обробки великих даних, організації можуть використовувати ОРС для автоматизації на етапі введення даних. Програмне забезпечення ОРС може ідентифікувати текст на зображенні, витягувати текст із зображень, зберігати текстовий файл і підтримувати jpg, jpeg, png, bmp, tiff, pdf та інші формати.

Завдання розпізнавання документів актуальне не тільки для бухгалтерського обліку, але і для багатьох інших областей.

ОРС дозволяє оптимізувати моделювання великих масивів даних шляхом перетворення паперових та відсканованих графічних документів у машинозчитувані, придатні для пошуку pdf-файли. Обробка та пошук цінної інформації не може бути автоматизована без попереднього застосування розпізнавання тексту в документах, де ще немає текстових шарів.

ОРС часто використовується як прихована технологія, що забезпечує роботу багатьох відомих систем та сервісів у нашому повсякденному житті. Важливими, але менш відомими сферами застосування технології ОРС є автоматизація введення даних, допомога сліпим і людям з вадами зору та

індексування документів для пошукових систем, таких як паспорти, номерні знаки, рахунки-фактури, банківські виписки, візитні картки та автоматичне розпізнавання номерних знаків.

ОРС має багато переваг, він в основному допомагає підвищити ефективність та результативність роботи. Його здатність швидко шукати величезний вміст надзвичайно корисна, особливо в офісних налаштуваннях, що мають справу з великою кількістю документів та великою кількістю сканування.

Висновки до розділу 1

ОРС може бути використаний для різних застосувань, включаючи: сканування друкованих документів; Індексація друкованого матеріалу для пошукових систем; Автоматизація введення, вилучення та обробки даних; Розшифровка документів у текст, який можна прочитати вголос для людей із вадами зору.

Більше частин людського мозку більше присвячується обробці зорових сигналів, аніж будь-яким іншим завданням. І, незважаючи на всю славу КЗ, люди все ще далекі від можливості імітувати працю свого розуму. Однак це не означає, що прориви в комп'ютерному зорі в будь-якому випадку є незначними. Це вже спосіб навчання моделей для досягнення точності, що іноді конкурує з можливостями розпізнавання людських зображень, знаходять способи використовувати ці досягнення для зменшення операційних витрат та оптимізації бізнес-процесів.

Найголовнішим фактом є те, що комп’ютерний зір, безумовно, може полегшити життя людям у різних сферах – від медицини до навіть безпеки в процесі аналізу даних. Спектр областей застосування досить пристойний, і очевидно, що будь-які дослідження в цій галузі є справді актуальними.

РОЗДІЛ 2.
СТРУКТУРА СИСТЕМИ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ЗА НОМЕРОМ ТРАНСПОРТНОГО ЗАСОБУ

2.1 Функції ПЗ

1. Можливість відсканувати номер маршрутного транспорту за допомогою камери смартфону.

2. Зчитування номеру маршрутного транспорту (Літери і більше ніж трьох значні номери не зчитуються).

3. Коректність номеру маршрутного транспорту.

4. Витягування з API інформацію про маршрут.

5. Вивести результат.

2.1.1. Специфікації прецедентів

1. Короткий опис

Користувач заходить в мобільний додаток.

Суб’єкт – Користувач.

2. Передумова

Користувач відкриває камеру, щоб відсканувати номер маршрутного транспортного засобу.

3. Основний потік

3.1 Додаток відображає сторінку з камерою

3.2 Зчитування

3.3 Якщо МТ з міста Києва і номер відповідає трьох значному числу без літер, тоді додаток відображає сторінку з повною інформацією про маршрут.

3.4 Якщо номер МТ не відповідає вимогам і МТ не з міста Київ, повертається сторінка з текстом “Маршрут не знайдено. Додаток працює на території Києва”.

2.1.2. Діаграма прецедентів

Діаграма прецедентів показує як система буде використовуватись.

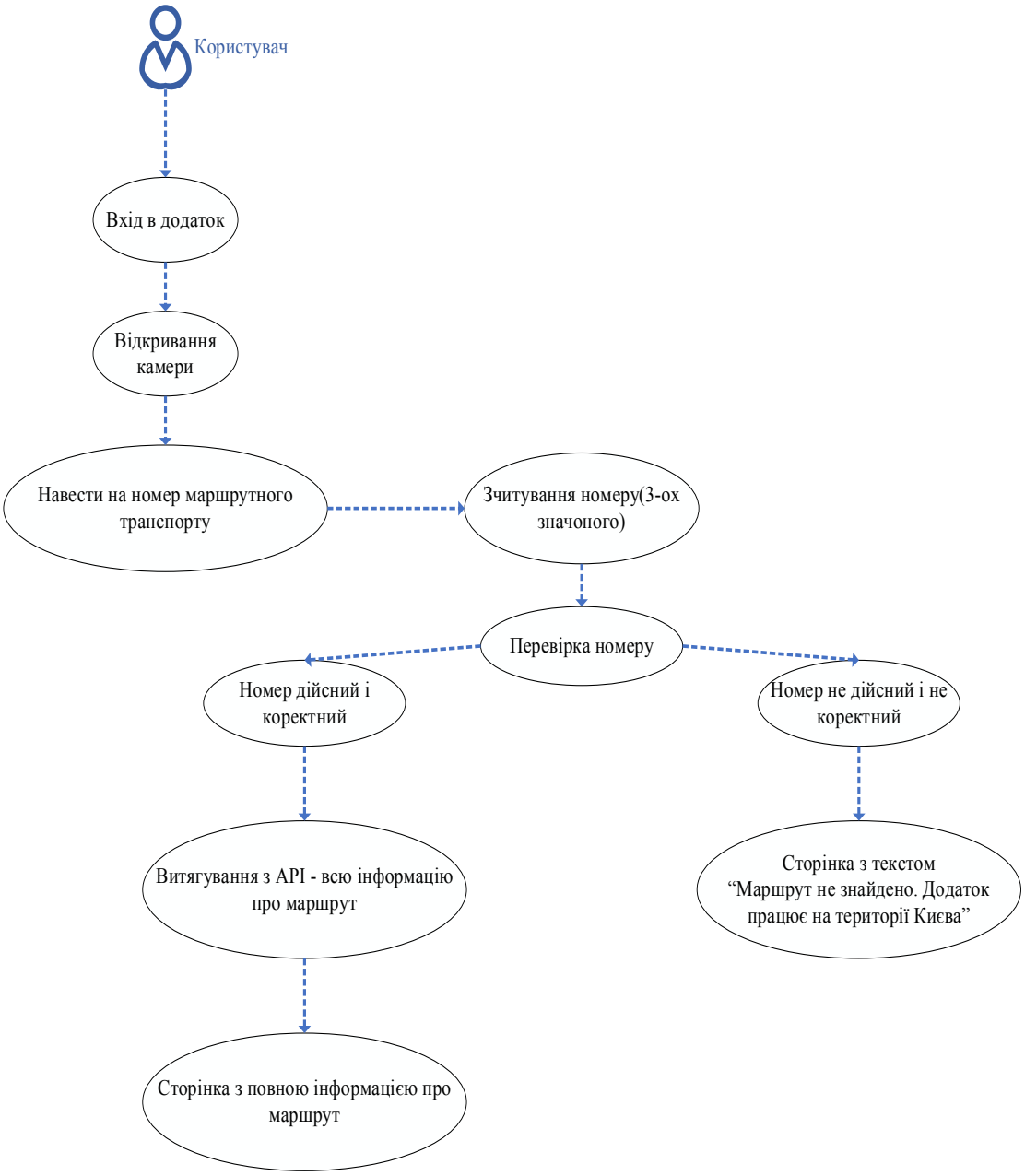

Рис 2.1. Основний сценарій подій

2.2. Опис ієрархічної будови

Користувач заходить в застосунок, йде выдкриття камери, він наводить камеру на номер маршрутного засобу, наприклад маршрутка 820, йде сканування номеру, видається вся інформація про маршрут на застосунок а саме:

1. Номер маршруту.
2. Інтервал руху
3. Час роботи.
4. Максимальна вартість проїзду.
5. Назва маршруту.
6. Перелік вулиць.

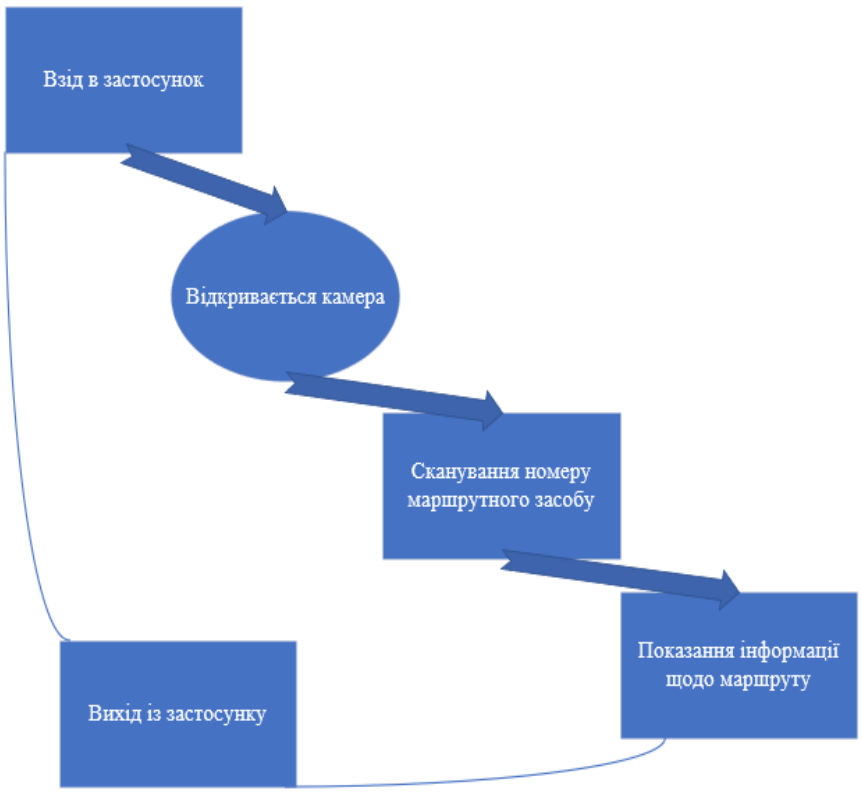

Рис 2.2. Опис ієрархічної будови

2.3. Проектування архітектури

Система автоматизованого визначення за номером транспортного засобу складається із 3 компонентів:

1. Сервіс збору даних;
2. Сервіс обробки даних;
3. Користувацький інтерфейс.
4. Кожен з них має свій перелік відповідальностей поданий в таблиці 2.1
5. Перелік відповідальностей складових елементів системи.

Таблиця 2.1

Перелік відповідальностей

Компонент	Сфера відповідальності
Сервіс збору даних	1. Отримати дані з API. 2. Згрупувати дані 3. Передати на обробку
Сервіс обробки даних	1. Згрупувати дані. 2. Обробити запит від користувача. 3. Передати інформацію на відображення в інтерфейсі.
Користувацький інтерфейс	1. Відобразити дані. (Інформація про весь маршрут)

2.4. Детальний опис функціоналу ієрархічної одиниці

Надання інформації з API EasyWay.

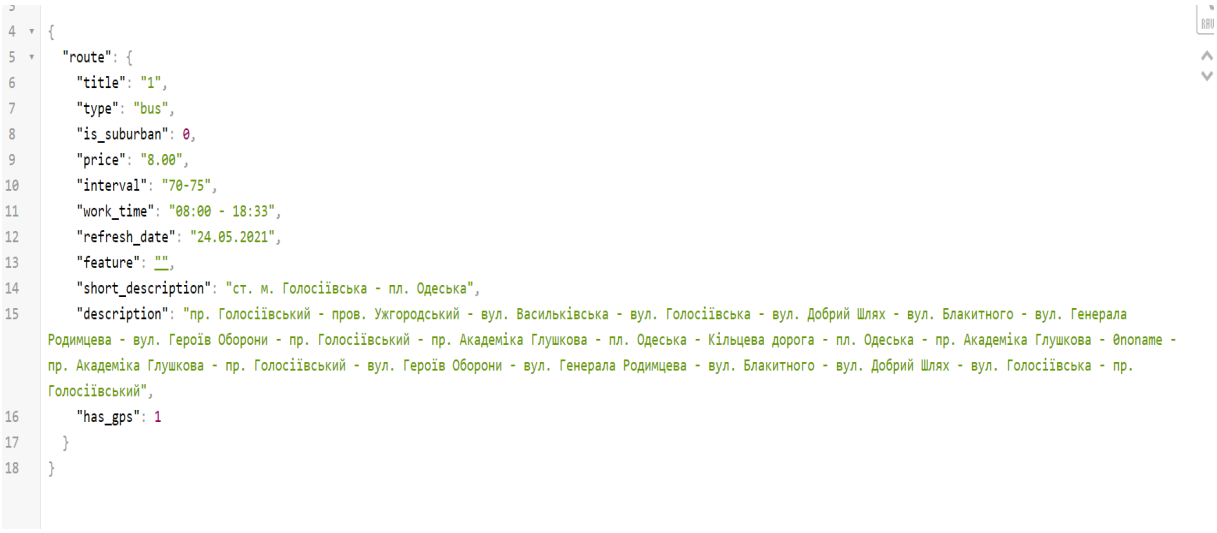

Рис 2.3. API інформація маршруту

За допомогою цього API ми бачимо маршрут.

- Отримання інформації про маршрут.
- Відображення маршруту в застосунку.
- Озвучення розпізнаного номеру.
- Сканування номеру маршрутного засобу в Києві.

Висновки до розділу 2

Описані функції ПЗ. Користувач може відсканувати номер маршрутного транспорту за допомогою камери смартфону. Номер зчитується, літери і більш ніж трьох значні номери не зчитуються. Працює тільки на території Києва.

Додаток має три компонента: сервіс обробки даних, сервіс збору даних, інтерфейс.

Була надана інформація з API, а саме: Номер маршруту, інтервал руху, час роботи, максимальна вартість проїзду, назва маршруту, перелік вулиць.

РОЗДІЛ 3. СПЕЦИФІКАЦІЯ ВИМОГ ДО СИСТЕМИ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ЗА НОМЕРОМ ТРАНСПОРТНОГО ЗАСОБУ

3.1. Вибір способу розпізнавання тексту на мобільному пристрої

Для реалізації програми для розпізнавання та ідентифікації якогось транспорту – використовувати одну з готових бібліотек, оскільки це найшвидший і найпростіший спосіб.

Мобільний зір. Бібліотека, яку було обрано. Розпізнає текст, виявляє обличчя, емоції, зчитує QR-коди. Основною причиною вибору цієї бібліотеки є той факт, що до неї досить просто прикріпити камеру і просто сфотографувати та відобразити результати. Але ще одним важливим критерієм є точність, тут вона досить висока в порівнянні з іншими бібліотеками. Розпізнавач тексту від Mobile Vision сегментує текст на блоки, рядки та слова. Google Vision API з'єднує ваш код з можливостями розпізнавання зображень Google. Ви можете думати про Google Image Search як про своєрідний API/REST-інтерфейс до images.google.com, але він робить набагато більше, ніж просто показує вам схожі зображення. Google Vision може визначити, чи є ви кішкою або людиною, а також частини вашого обличчя.

3.2. Концепція взаємодії клієнт-сервер

Концепція клієнт-сервер включає дві сторони: клієнта та сервер. Клієнт – клієнтські послуги, а сервер – постачальник послуг. Наприклад клієнтом є браузер. Клієнт і сервер взаємодіють між собою в Інтернеті або в будь-якій іншій комп’ютерній мережі за допомогою різних мережевих протоколів, таких як IP-протокол, HTTP-протокол.

Повідомлення, які надсилають клієнти, називаються HTTP-запитами. Запити мають спеціальні методи, які повідомляють серверу, як обробляти повідомлення. А повідомлення, які надсилає сервер, називаються відповідями HTTP.

Вони містять, крім корисної інформації, спеціальні коди стану, які дозволяють браузеру з'ясувати, як сервер зрозумів його запит [7]. Клієнт-серверну архітектуру можна означити, як концепцію мережі в якій основна частина її ресурсів зосереджена в серверах.

Набір серверів, які надають інформацію або інші послуги, набір клієнтів, які використовують сервіси, що надають серверами, мережа, яка забезпечує взаємодію між клієнтами та серверами, на рис 2.1 представлена схема клієнт-серверу. Правила між клієнтом і сервером називається протоколом обміну. Модель клієнт-серверної взаємодії визначається розподілом обов'язків.

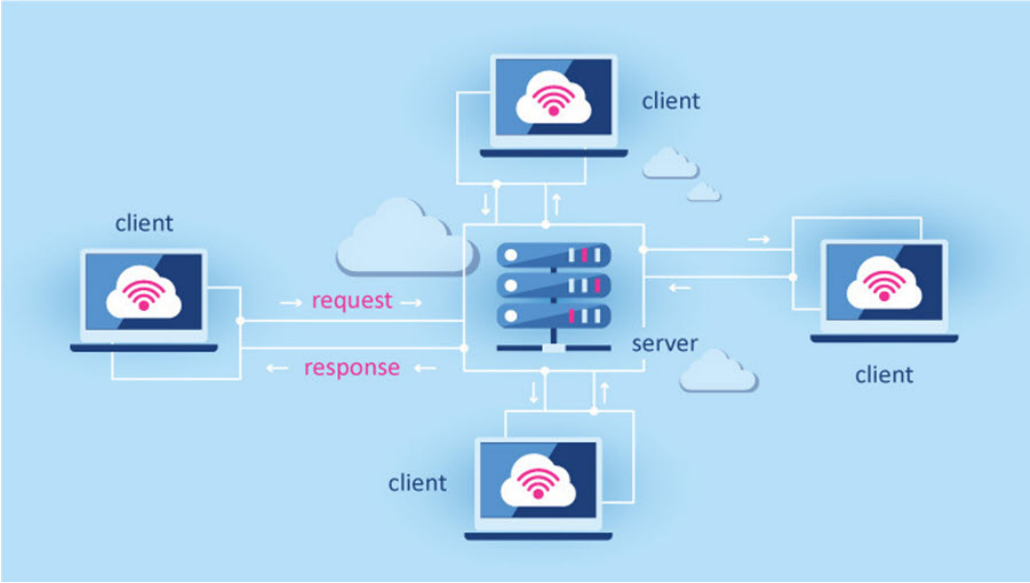

Рис. 3.1. Схема клієнт-серверу

3.3. Технічні характеристики системи індентифікації

Тема: Транспортний засіб із нанесеним номером маршруту.

Об’єкт: мешканці які проживають на території Києва.

Апаратне забезпечення: Смартфон.

Операційна система: ОС Android версії 4.0.2 і пізніше.

Мова програмування: Java/Kotlin.

Мова інтерфейсу: Українська.

Додаткові умови:

- Особливості для людей з поганим зором.
- Система повинна працювати в різних погодних умовах.
- Використання API: Mobile Vision, EasyWay.
- Доступ до Інтернету.
- При наведенні на номер маршрутного транспорту, відразу показує маршрут.
- Дані обробляються в режимі реального часу.

3.4 Вибір способу розпізнавання тексту на мобільному пристрої

Для реалізації програми для розпізнавання та ідентифікації якогось транспорту – використовувати одну з готових бібліотек, оскільки це найшвидший і найпростіший спосіб.

Мобільний зір. Бібліотека, яку було обрано. Розпізнає текст, виявляє обличчя, емоції, зчитує QR-коди. Основною причиною вибору цієї бібліотеки є той факт, що до неї досить просто прикріпити камеру і просто сфотографувати та відобразити результати. Але ще одним важливим критерієм є точність, тут вона досить висока в порівнянні з іншими бібліотеками. Розпізнавач тексту від Mobile Vision сегментує текст на блоки, рядки та слова.

3.5.Технології на яких працює додаток

Описання всіх технологій, на яких працює МД, для системи автоматизованого визначення за номером транспортного засобу.

Використовується: ОС: Android, Android SDK, API, HTTP протокол, JSON, Мобільна бібліотека зору від Google.

3.5.1. ОС Android

Операційна система Android – це мобільна операційна система, розроблена Google, яка в основному використовується для сенсорних пристроїв, мобільних телефонів та планшетів. Його конструкція дозволяє користувачам інтуїтивно маніпулювати мобільними пристроями за допомогою рухів пальців, що відображають загальні рухи, такі як пощипування, проведення та натискання. Хоча Android пропонує користувачам життєздатну альтернативу іншим мобільним операційним системам, деякі обмеження все ще залишаються. З боку розробника, кодування складного користувацького досвіду та інтерфейсів часто є складним завданням, яке вимагає більшої довіри до Java, ніж до Objective-C.

Рис. 3.2. ОС Android

3.5.2. Android SDK

SDK (Software Development Kit) – це набір засобів розробки, що дозволяє спеціалістам з програмного забезпечення створювати програми для

певного програмного пакету, основних засобів розробки програмного забезпечення, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем та інших платформ.

Переваги використання SDK:

1) Проста інтеграція з системою для нових клієнтів.

2) Повторне використання коду.

3) Якість коду.

Компоненти:

1) Емулятор Android.

2) інструменти платформи.

3) інструменти побудови.

4) SDK-інструменти.

Рис. 3.3. ОС Android

3.5.3. API

Абревіатура розшифровується як Application Programming Interface (Інтерфейс програмування програм) або інтерфейс для програмування програм.

Інтерфейс, який дозволяє розробникам використовувати кілька готових блоків для створення додатків.

У випадку веб-додатків API може надавати дані у форматі, відмінному від стандартного HTML. Сторонні відкриті API найчастіше надають дані в

одному з двох форматів: XML або JSON. Максимальне розділення інтерфейсу та бекенда.

Всі необхідні способи отримання та надсилання інформації за допомогою API, у цьому проекті використовуються два API – Mobile Vision та EasyWay API.

Дані Вашого телефону ніколи не будуть повністю доступні серверу, так само як і сервер ніколи не буде повністю доступний Вашому телефону. Замість цього вони обмінюються невеликими пакетами даних, передаючи лише те, що необхідно – як при замовленні їжі на винос. Ви говорите ресторану, що б ви хотіли з'їсти, вони кажуть вам, що їм потрібно натомість, і, врешті-решт, ви отримуєте свою їжу.

API стали настільки цінними, що складають значну частину доходів багатьох компаній. Такі великі компанії, як Google, eBay, Salesforce.com, Amazon та Expedia – це лише деякі з компаній, які заробляють гроші на своїх API.

Сучасні API дотримуються стандартів (як правило, HTTP і REST), які є зручними для розробників, легкодоступними і широко зрозумілими

Вони розглядаються більше як продукти, ніж як код. Вони призначені для споживання конкретною аудиторією (наприклад, розробниками мобільних пристроїв), вони задокументовані та мають версії таким чином, що користувачі можуть мати певні очікування щодо їх обслуговування та життєвого циклу.

Оскільки вони набагато більш стандартизовані, вони мають набагато сильнішу дисципліну безпеки і управління, а також контролюються і управляються для забезпечення продуктивності і масштабування.

Як і будь-яке інше програмне забезпечення, сучасний API має власний життєвий цикл розробки програмного забезпечення (SDLC), що включає проектування, тестування, збірку, управління та версіювання. Крім того, сучасні API добре задокументовані для споживання та версійності.

Рис. 3.4. EasyWay

3.5.4. HTTP протокол

Протокол веб-сайту, також відомий як протокол передачі даних, являє собою своєрідний набір правил, що описують порядок, особливості взаємодії двох або більше пристроїв, підключених до однієї мережі та взаємодіючих. Протокол сайту допомагає врегулювати все це і дозволяє всім користувачам (клієнтам) без проблем взаємодіяти з серверами (хостами).

Найбільш базовими протоколами є HTTP та HTTPS. Ці 2 протоколи використовуються для всього, що цікавить користувача.

HTTP – це протокол програми передачі даних. Існує сервер, який у пасивному режимі постійно чекає з'єднання з ним. Цей зв'язок з ним рано чи пізно буде встановлений клієнтом, тобто інтерфейсом користувача машини через Інтернет. Сервер здатний обробляти запити користувачів завдяки інструкціям, які надав йому протокол HTTP. Якщо запит не може бути оброблений, сервер знає, яку помилку він повинен дати.

HTTPS – це вдосконалена версія HTTP. Основна відмінність полягає в тому, що тепер запити від клієнта надсилаються не в голому вигляді, а зашифровані з використанням криптографічних механізмів SSL і TLS. Використання цього протоколу дозволяє досягти такого результату, що запит

від клієнта дійсно може бути прочитаний лише на стороні сервера, і жодним чином не може бути перехоплений третьою стороною десь посередині.

Протокол HTTPS – це найбезпечніший спосіб спілкування між пристроями сьогодні. Його не можна зламати, цей протокол передачі даних невразливий.

Рис. 3.5. HTTPS протокол

3.5.5. JSON

JSON – це скорочення від "JavaScript Object Notation" і це спосіб зберігання інформації в організованому та легкодоступному вигляді. Це дає нам зручну для читання колекцію даних, до якої ми можемо отримати доступ.

JSON побудований на двох структурах:

Колекція пар ім'я/значення. У різних мовах це реалізується як об'єкт, запис, структура, словник, хеш-таблиця, список з ключами або асоціативний масив.

Впорядкований список значень. У більшості мов реалізується як масив, вектор, список або послідовність.

Це універсальні структури даних. Практично всі сучасні мови програмування підтримують їх у тому чи іншому вигляді. Логічно, що

формат даних, взаємозамінний з мовами програмування, також має базуватися на цих структурах.

Рис. 3.6. JSON

3.5.6. Мобільна бібліотека зору від Google

Google КЗ дозволяє розробникам зрозуміти зміст зображення, інкапсулюючи потужні моделі машинного навчання в простий у використанні REST API: бібліотека КЗ з відкритим кодом. Був розроблений для обчислювальної ефективності та з сильним акцентом на додатки реального часу. Написана на оптимізованому C / C ++, бібліотека може скористатися перевагами багатоядерної обробки. Увімкнений з OpenCL, він може скористатися апаратним прискоренням базової різнорідної обчислювальної платформи [8].

Рис. 3.7. Бібліотека зору Google

3.6. Платформа для реалізації

Платформою для реалізації було обрано середовище Android Studio – найбільш популярним і зручним середовищем для розробки мобільного додатку, має потужні можливості.

3.7. Функціональні вимоги

1. Збір, систематизація і накопичення і віддавання даних.
2. Накопичення, збереження даних.
3. Візуалізація.
4. Надійність системи .
5. Відновлення.

3.8. Системні вимоги

Мінімальна апаратна конфігурація мобільного пристрою:

- Частота оновлення екрану 90 Гц;
- ОС Android 4.2, Android 4.4.2 і більше;

- Процесор Intel Atom Processor 1.2 ГГц;
- Навігація GPS;
- Оперативная пам’ять об’ємом не менше 1 Гб;
- Вільний простір 1ГБ;
- ОЗУ 1ГБ;
- Доступ до інтернету;
- Роздільна здатність екрану 1920x1080;
- Сенсорний екран;
- Стандарт зв’язку 3G
- Камера 20 Мп;

3.9. Функціональна схема

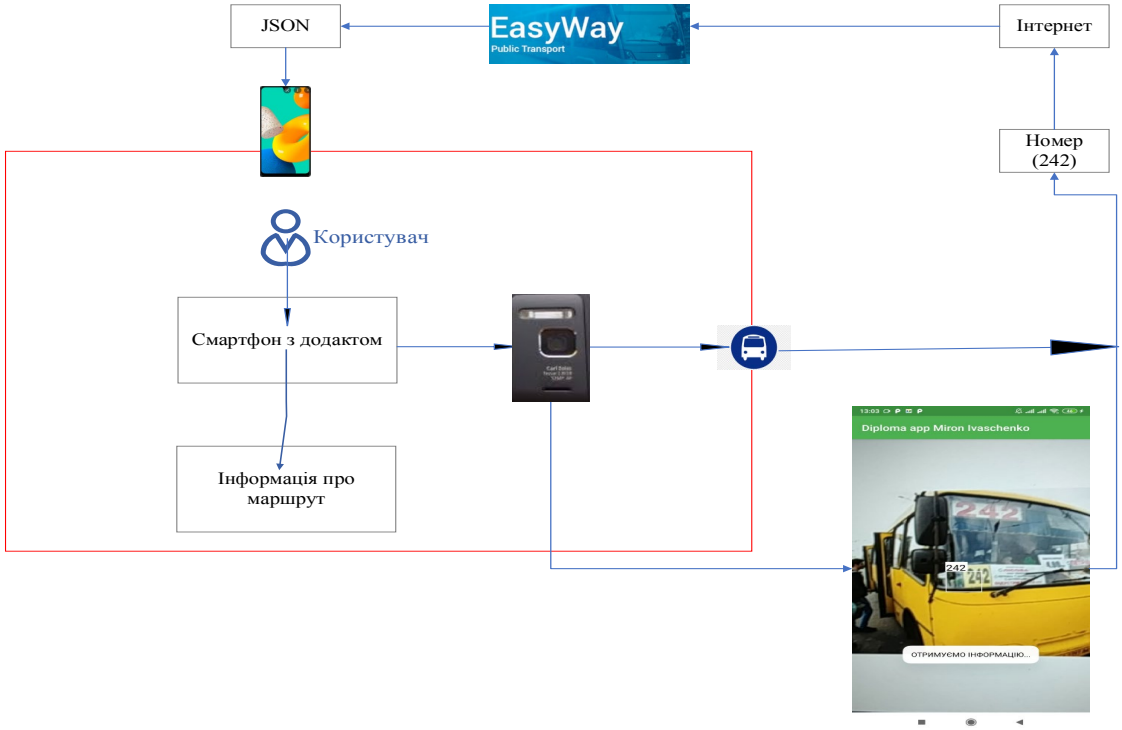

Рис. 3.8. Функціональна схема додатку

Висновки до розділу 3

API є важливою складною програмою програмного забезпечення, і вони існують на всіх рівнях програмного забезпечення.

Про те, що API полегшує життя як розробників, так і користувачів, у цьому проекті використовуються два API – Mobile Vision та EasyWay API.

EasyWay – це сервіс, який надає інформацію про всі маршрути та зупинки транспортних засобів України, включаючи Київ, що є достатнім для розробки додаткових матеріалів.

Мобільну бібліотеку зору від Google було обрано для використання під час реалізації програми для розпізнавання номера маршрутного транспорту в режимі реального часу за допомогою камери на смартфоні.

Описані системні та функціональні вимоги додатку. Платформою для реалізації обрано середовище Android Studio. Також була спроектована функціональна схема додатку.

РОЗДІЛ 4. ПРОТОТИП СИСТЕМИ АВТОМАТИЗОВАНОГО ВИЗНАЧЕННЯ ЗА НОМЕРОМ ТРАНСПОРТНОГО ЗАСОБУ

4.1. Впровадження

Ідея програми полягає в тому, що люди не завжди точно знають, як дістатися до потрібного місця, наприклад, якщо вони іноземці і нічого не знають про маршрути місцевих транспортних засобів, тож за допомогою програми можна зрозуміти, чи буде конкретний транспортний засіб доставити користувача до потрібного місця.

4.2. Прототп та опис інтерфейсу додатка

Прототип це чорновий варіант програми, створений здебільш для того, щоб перевірити влучність концепції, архітектури, функціональних та технологічних рішень, а також для того, щоб продемонструвати робочу програму.

Для відображення користувацького інтерфейсу були створені прототипи. Прототипи є зручним способом відображення очікуваних елементів, які будуть показані користувачу.

Можна побачити, як додаток зчитує текст, який він бачить, і видає одразу на екран мобільного пристрою, як показано на рис 4.1.

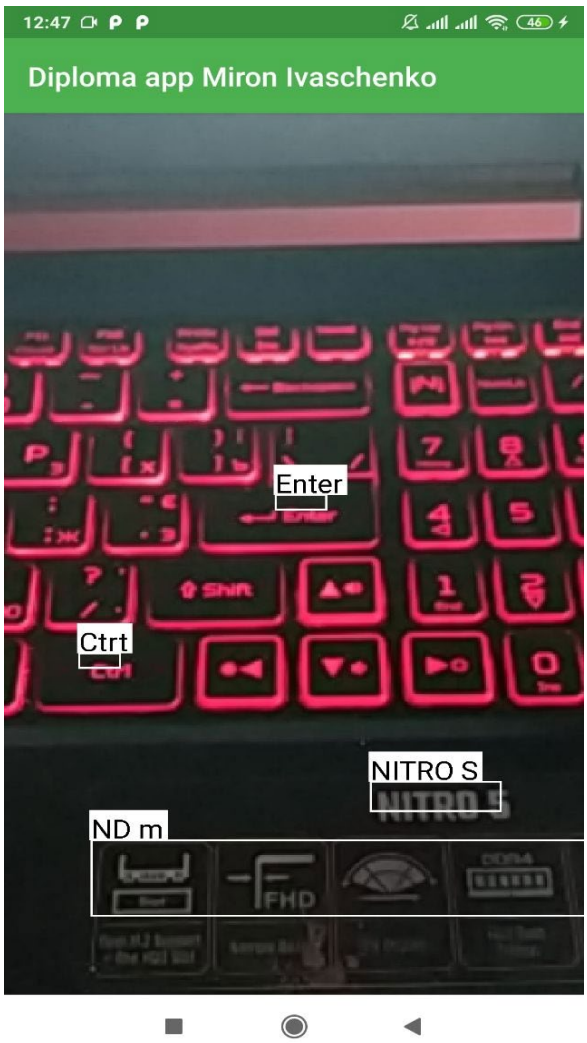

Рис 4.1. Зчитування тексту

Коли користувач запускає програму, з'являється екран із камерою смартфона та тимчасовим повідомленням тексту, номер. Але, щоб дадаток зчитав успішно номер, це повинен бути трьох значний номер, без літер, тоді додаток зчитає трьохзначний номер вірно.

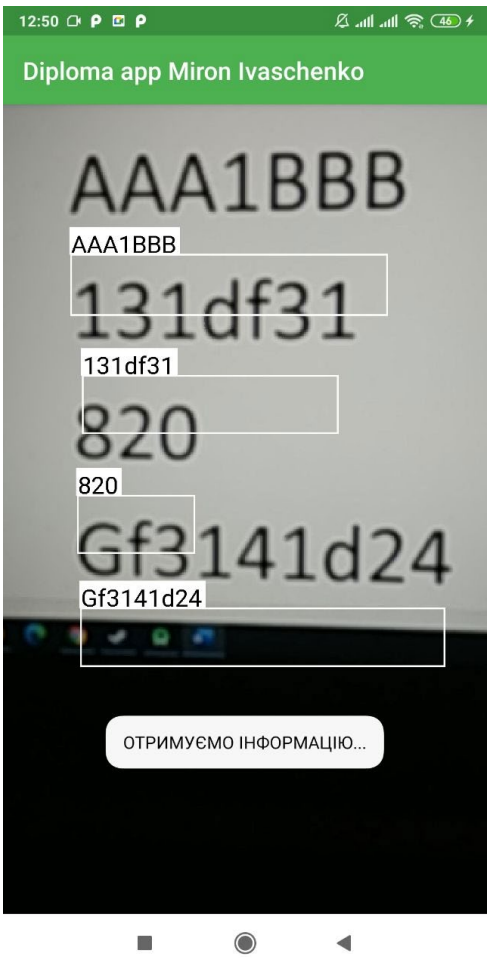

Рис 4.2. Зчитування номеру

Він зчитує всі символи і номери, які представленні, але щоб видалась інформація, додаток пропустить всі неправильні номери, текст, тощо і зчитає трьох значний номер.

Користувач повинен направити свою камеру на транспортний засіб із його номером. МТ засіб повинен знаходитись в Києві, тобто номер транспортного засобу є київським. Після цього програма розпізнає номер. Після зчитування номера програма надсилає запит на сервер EasyWay і отримує звідти інформацію про маршрут. Якщо сервер не містить жодної інформації про номер, на екрані відображатиметься сторінка з інформацією, що маршрут не знайдено.

Рис 4.3. Зчитування номеру львівського транспортного засобу

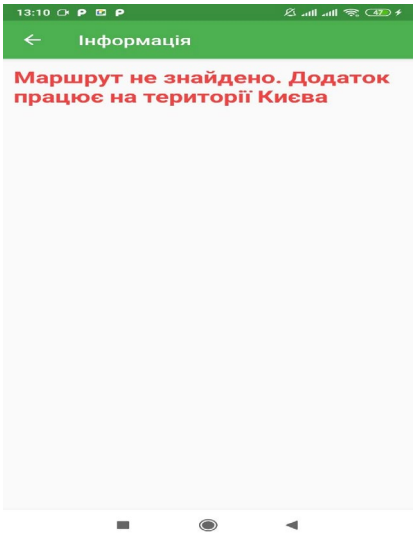

Рис 4.4. Сторінка з текстом “Маршрут не знайдено”

Якщо номер існує, і номер транспортного засобу відповідає вимогам додатку, тоді є вся інформація, яку можна отримати із серверів EasyWay. Приклад отриманого результату на основі наданого номера.

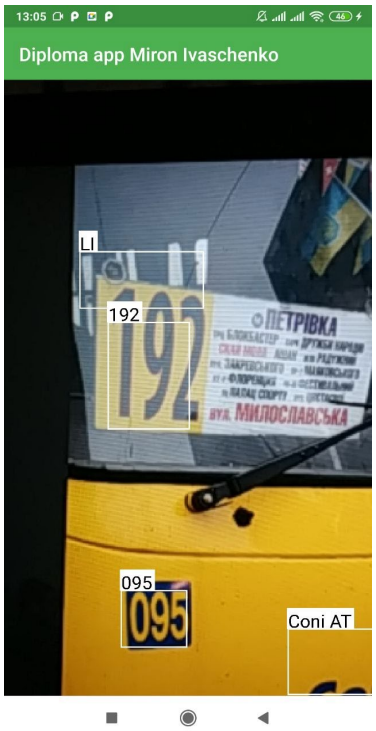

Рис 4.5. Транспортний засіб з номером 192 м.Київ

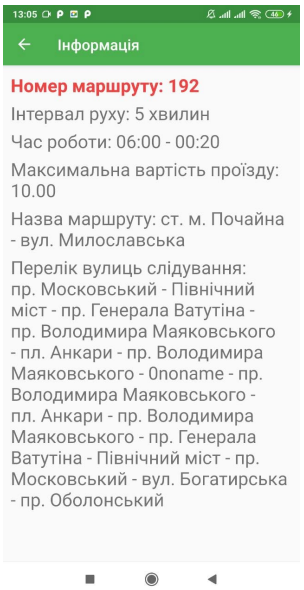

Рис 4.6. Вся інформація про маршрут з номером 192

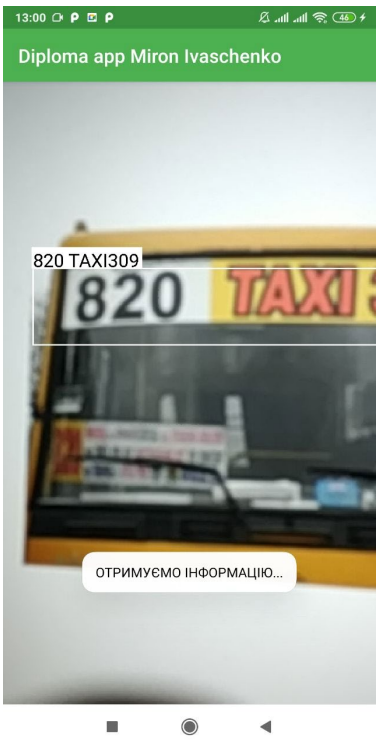

Рис 4.7. Транспортний засіб з номером 820 м.Київ

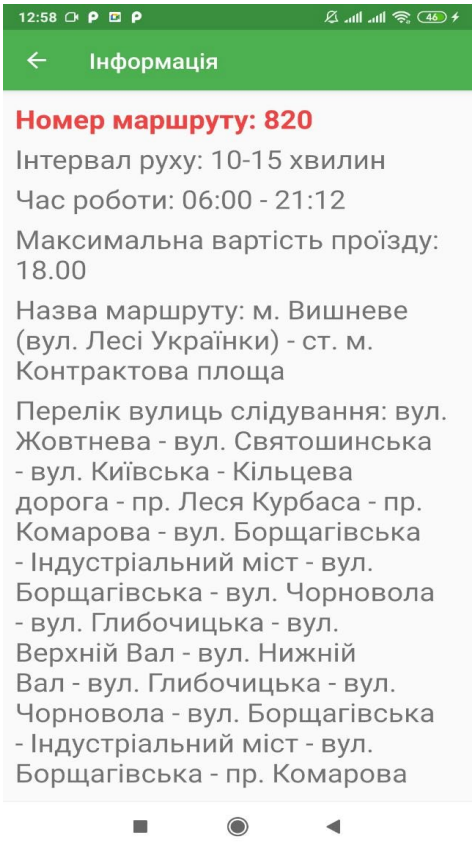

Рис 4.8. Вся інформація про маршрут з номером 820

Існує два запити на розпізнавання одного номера завдяки тому, як сервери EasyWay зберігають інформацію. Послуга надавала доступ до маршрутів.GetRouteInfo та GetRoutesList. Перший повертає інформацію про конкретний маршрут у формі JSON.

Розпізнавання номерів було реалізовано за допомогою API Google mobile vision. Він здатний розпізнати багато речей, наприклад обличчя людини, тощо.

Рис 4.8. Розпізнавання Google vision

4.3. Програмування системи автоматизованого визначення за номером транспортного засобу

Клас для отримання JSON

```
public class GetJSONTask extends AsyncTask<String, Void, JSONObject> {

    private static final String ROUTES_URL =
"https://api.easyway.info/?login=miron.ivasch" +

"&password=2Jeh4uie48Cfa3&city=kyiv&function=cities.GetRoutesList";

    private static final String ROUTE_URL =
```

```
"https://api.easyway.info/" +
            "?login=miron.ivasch&password=2Jeh4uie48Cfa3" +
            "&city=kyiv&function=routes.GetRouteInfo&id=";

    @Override
    protected JSONObject doInBackground(String... strings)  {

        JSONParser jsonParser = new JSONParser();
        Map<String,String> map;

        try {
            JSONObject jsonObject = getJSONByUrl(ROUTES_URL);

            map = jsonParser.parseRoutesJSON(

jsonObject.getJSONObject("routesList").getJSONArray("route"));

            jsonObject = getJSONByUrl(ROUTE_URL +
getIdByTitle(strings[0], map));

            return jsonObject;

        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }

    private JSONObject getJSONByUrl(String request) throws
IOException, JSONException {

        HttpURLConnection urlConnection;

        URL url = new URL(request);
        urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setRequestMethod("GET");
        urlConnection.setReadTimeout(10000 /* milliseconds */ );
        urlConnection.setConnectTimeout(15000 /* milliseconds */ );
        urlConnection.setDoOutput(true);
        urlConnection.connect();

        BufferedReader br = new BufferedReader(new
InputStreamReader(url.openStream()));
        StringBuilder sb = new StringBuilder();

        String line;
        while ((line = br.readLine()) != null) {
            sb.append(line).append("\n");
        }
```

```
        br.close();

        String jsonString = sb.toString();

        return new JSONObject(jsonString);
    }

    private String getIdByTitle(String title, Map<String,String> map)
{

        for (Map.Entry<String, String> entry : map.entrySet()) {
            if(entry.getKey().equals(title)) {
                return entry.getValue();
            }
        }
        return null;
    }
}
```

Клас для відображення даних

```
public Queue<String> parseJSON(JSONObject jsonObject) {

    Queue<String> queue = new LinkedList<>();

    try {
        queue.add("Номер маршруту: " + jsonObject.getString("title"));
        queue.add("Вартість проїзду: " +
jsonObject.getString("price"));
        queue.add("Інтервал руху " + jsonObject.getString("interval")
+ " хвилин");
        queue.add("Час роботи: " + jsonObject.getString("work_time"));
        queue.add("Назва маршруту: " +
jsonObject.getString("short_description"));
        queue.add("Перелік вулиць слідування: " +
jsonObject.getString("description"));
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return queue;
}
```

Клас для камери додатку

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate");

    setContentView(R.layout.activity_vision_live_preview);

    preview = findViewById(R.id.preview_view);
    if (preview == null) {
```

```
      Log.d(TAG, "Preview is null");
    }
    graphicOverlay = findViewById(R.id.graphic_overlay);
    if (graphicOverlay == null) {
      Log.d(TAG, "graphicOverlay is null");
    }

    Spinner spinner = findViewById(R.id.spinner);
    List<String> options = new ArrayList<>();
    options.add(TEXT_RECOGNITION);
    options.add(OBJECT_DETECTION);
    options.add(OBJECT_DETECTION_CUSTOM);
    options.add(CUSTOM_AUTOML_OBJECT_DETECTION);
    options.add(FACE_DETECTION);
    options.add(BARCODE_SCANNING);
    options.add(IMAGE_LABELING);
    options.add(IMAGE_LABELING_CUSTOM);
    options.add(CUSTOM_AUTOML_LABELING);
    options.add(POSE_DETECTION);
    options.add(SELFIE_SEGMENTATION);

    // Creating adapter for spinner
    ArrayAdapter<String> dataAdapter = new ArrayAdapter<>(this,
R.layout.spinner_style, options);
    // Drop down layout style - list view with radio button

dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dr
opdown_item);
    // attaching data adapter to spinner
    spinner.setAdapter(dataAdapter);
    spinner.setOnItemSelectedListener(this);

    ToggleButton facingSwitch = findViewById(R.id.facing_switch);
    facingSwitch.setOnCheckedChangeListener(this);

    ImageView settingsButton = findViewById(R.id.settings_button);
    settingsButton.setOnClickListener(
        v -> {
          Intent intent = new Intent(getApplicationContext(),
SettingsActivity.class);
          intent.putExtra(
              SettingsActivity.EXTRA_LAUNCH_SOURCE,
SettingsActivity.LaunchSource.LIVE_PREVIEW);
          startActivity(intent);
        });

    if (allPermissionsGranted()) {
      createCameraSource(selectedModel);
    } else {
      getRuntimePermissions();
    }
```

```
  }

  @Override
  public synchronized void onItemSelected(AdapterView<?> parent, View
view, int pos, long id) {
    // An item was selected. You can retrieve the selected item using
    // parent.getItemAtPosition(pos)
    selectedModel = parent.getItemAtPosition(pos).toString();
    Log.d(TAG, "Selected model: " + selectedModel);
    preview.stop();
    if (allPermissionsGranted()) {
      createCameraSource(selectedModel);
      startCameraSource();
    } else {
      getRuntimePermissions();
    }
  }

  @Override
  public void onNothingSelected(AdapterView<?> parent) {
    // Do nothing.
  }

  @Override
  public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
    Log.d(TAG, "Set facing");
    if (cameraSource != null) {
      if (isChecked) {
        cameraSource.setFacing(CameraSource.CAMERA_FACING_FRONT);
      } else {
        cameraSource.setFacing(CameraSource.CAMERA_FACING_BACK);
      }
    }
    preview.stop();
    startCameraSource();
  }

  private void createCameraSource(String model) {
    // If there's no existing cameraSource, create one.
    if (cameraSource == null) {
      cameraSource = new CameraSource(this, graphicOverlay);
    }

    try {
      switch (model) {
        case OBJECT_DETECTION:
          Log.i(TAG, "Using Object Detector Processor");
          ObjectDetectorOptions objectDetectorOptions =

PreferenceUtils.getObjectDetectorOptionsForLivePreview(this);
```

```
          cameraSource.setMachineLearningFrameProcessor(
              new ObjectDetectorProcessor(this,
objectDetectorOptions));
          break;
        case OBJECT_DETECTION_CUSTOM:
          Log.i(TAG, "Using Custom Object Detector Processor");
          LocalModel localModel =
              new LocalModel.Builder()

.setAssetFilePath("custom_models/object_labeler.tflite")
                  .build();
          CustomObjectDetectorOptions customObjectDetectorOptions =

PreferenceUtils.getCustomObjectDetectorOptionsForLivePreview(this,
localModel);
          cameraSource.setMachineLearningFrameProcessor(
              new ObjectDetectorProcessor(this,
customObjectDetectorOptions));
          break;
        case CUSTOM_AUTOML_OBJECT_DETECTION:
          Log.i(TAG, "Using Custom AutoML Object Detector Processor");
          LocalModel customAutoMLODTLocalModel =
              new
LocalModel.Builder().setAssetManifestFilePath("automl/manifest.json").
build();
          CustomObjectDetectorOptions customAutoMLODTOptions =

PreferenceUtils.getCustomObjectDetectorOptionsForLivePreview(
                  this, customAutoMLODTLocalModel);
          cameraSource.setMachineLearningFrameProcessor(
              new ObjectDetectorProcessor(this,
customAutoMLODTOptions));
          break;
        case TEXT_RECOGNITION:
          Log.i(TAG, "Using on-device Text recognition Processor");
          cameraSource.setMachineLearningFrameProcessor(new
TextRecognitionProcessor(this));
          break;
        case FACE_DETECTION:
          Log.i(TAG, "Using Face Detector Processor");
          FaceDetectorOptions faceDetectorOptions =

PreferenceUtils.getFaceDetectorOptionsForLivePreview(this);
          cameraSource.setMachineLearningFrameProcessor(
              new FaceDetectorProcessor(this, faceDetectorOptions));
          break;
        case BARCODE_SCANNING:
          Log.i(TAG, "Using Barcode Detector Processor");
          cameraSource.setMachineLearningFrameProcessor(new
BarcodeScannerProcessor(this));
          break;
```

```
        case IMAGE_LABELING:
          Log.i(TAG, "Using Image Label Detector Processor");
          cameraSource.setMachineLearningFrameProcessor(
              new LabelDetectorProcessor(this,
ImageLabelerOptions.DEFAULT_OPTIONS));
          break;
        case IMAGE_LABELING_CUSTOM:
          Log.i(TAG, "Using Custom Image Label Detector Processor");
          LocalModel localClassifier =
              new LocalModel.Builder()

.setAssetFilePath("custom_models/bird_classifier.tflite")
                  .build();
          CustomImageLabelerOptions customImageLabelerOptions =
              new
CustomImageLabelerOptions.Builder(localClassifier).build();
          cameraSource.setMachineLearningFrameProcessor(
              new LabelDetectorProcessor(this,
customImageLabelerOptions));
          break;
        case CUSTOM_AUTOML_LABELING:
          Log.i(TAG, "Using Custom AutoML Image Label Detector
Processor");
          LocalModel customAutoMLLabelLocalModel =
              new
LocalModel.Builder().setAssetManifestFilePath("automl/manifest.json").
build();
          CustomImageLabelerOptions customAutoMLLabelOptions =
              new
CustomImageLabelerOptions.Builder(customAutoMLLabelLocalModel)
                  .setConfidenceThreshold(0)
                  .build();
          cameraSource.setMachineLearningFrameProcessor(
              new LabelDetectorProcessor(this,
customAutoMLLabelOptions));
          break;
        case POSE_DETECTION:
          PoseDetectorOptionsBase poseDetectorOptions =

PreferenceUtils.getPoseDetectorOptionsForLivePreview(this);
          Log.i(TAG, "Using Pose Detector with options " +
poseDetectorOptions);
          boolean shouldShowInFrameLikelihood =

PreferenceUtils.shouldShowPoseDetectionInFrameLikelihoodLivePreview(th
is);
          boolean visualizeZ =
PreferenceUtils.shouldPoseDetectionVisualizeZ(this);
          boolean rescaleZ =
PreferenceUtils.shouldPoseDetectionRescaleZForVisualization(this);
          boolean runClassification =
```

```
PreferenceUtils.shouldPoseDetectionRunClassification(this);
          cameraSource.setMachineLearningFrameProcessor(new
PoseDetectorProcessor(
              this, poseDetectorOptions, shouldShowInFrameLikelihood,
visualizeZ, rescaleZ,
              runClassification, /* isStreamMode = */true));
          break;
        case SELFIE_SEGMENTATION:
          cameraSource.setMachineLearningFrameProcessor(new
SegmenterProcessor(this));
          break;
        default:
          Log.e(TAG, "Unknown model: " + model);
      }
    } catch (RuntimeException e) {
      Log.e(TAG, "Can not create image processor: " + model, e);
      Toast.makeText(
              getApplicationContext(),
              "Can not create image processor: " + e.getMessage(),
              Toast.LENGTH_LONG)
          .show();
    }
  }

  /**
   * Starts or restarts the camera source, if it exists. If the camera
source doesn't exist yet
   * (e.g., because onResume was called before the camera source was
created), this will be called
   * again when the camera source is created.
   */
  private void startCameraSource() {
    if (cameraSource != null) {
      try {
        if (preview == null) {
          Log.d(TAG, "resume: Preview is null");
        }
        if (graphicOverlay == null) {
          Log.d(TAG, "resume: graphOverlay is null");
        }
        preview.start(cameraSource, graphicOverlay);
      } catch (IOException e) {
        Log.e(TAG, "Unable to start camera source.", e);
        cameraSource.release();
        cameraSource = null;
      }
    }
  }

  @Override
  public void onResume() {
```

```
    super.onResume();
    Log.d(TAG, "onResume");
    createCameraSource(selectedModel);
    startCameraSource();
  }

  /** Stops the camera. */
  @Override
  protected void onPause() {
    super.onPause();
    preview.stop();
  }

  @Override
  public void onDestroy() {
    super.onDestroy();
    if (cameraSource != null) {
      cameraSource.release();
    }
  }

  private String[] getRequiredPermissions() {
    try {
      PackageInfo info =
          this.getPackageManager()
              .getPackageInfo(this.getPackageName(),
PackageManager.GET_PERMISSIONS);
      String[] ps = info.requestedPermissions;
      if (ps != null && ps.length > 0) {
        return ps;
      } else {
        return new String[0];
      }
    } catch (Exception e) {
      return new String[0];
    }
  }

  private boolean allPermissionsGranted() {
    for (String permission : getRequiredPermissions()) {
      if (!isPermissionGranted(this, permission)) {
        return false;
      }
    }
    return true;
  }

  private void getRuntimePermissions() {
    List<String> allNeededPermissions = new ArrayList<>();
    for (String permission : getRequiredPermissions()) {
      if (!isPermissionGranted(this, permission)) {
```

```
        allNeededPermissions.add(permission);
      }
    }

    if (!allNeededPermissions.isEmpty()) {
      ActivityCompat.requestPermissions(
          this, allNeededPermissions.toArray(new String[0]),
PERMISSION_REQUESTS);
    }
  }

  @Override
  public void onRequestPermissionsResult(
      int requestCode, String[] permissions, int[] grantResults) {
    Log.i(TAG, "Permission granted!");
    if (allPermissionsGranted()) {
      createCameraSource(selectedModel);
    }
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
  }

  private static boolean isPermissionGranted(Context context, String
permission) {
    if (ContextCompat.checkSelfPermission(context, permission)
        == PackageManager.PERMISSION_GRANTED) {
      Log.i(TAG, "Permission granted: " + permission);
      return true;
    }
    Log.i(TAG, "Permission NOT granted: " + permission);
    return false;
  }
}
```

Клас для активності вибору

```
public final class ChooserActivity extends AppCompatActivity
    implements OnRequestPermissionsResultCallback,
AdapterView.OnItemClickListener {
  private static final String TAG = "ChooserActivity";
  private static final int PERMISSION_REQUESTS = 1;

  private static final Class<?>[] CLASSES =
      VERSION.SDK_INT < VERSION_CODES.LOLLIPOP
          ? new Class<?>[] {
            LivePreviewActivity.class, StillImageActivity.class,
          }
          : new Class<?>[] {
            LivePreviewActivity.class,
            StillImageActivity.class,
            CameraXLivePreviewActivity.class,
```

```
            CameraXSourceDemoActivity.class,
          };

  private static final int[] DESCRIPTION_IDS =
      VERSION.SDK_INT < VERSION_CODES.LOLLIPOP
          ? new int[] {
            R.string.desc_camera_source_activity,
R.string.desc_still_image_activity,
          }
          : new int[] {
            R.string.desc_camera_source_activity,
            R.string.desc_still_image_activity,
            R.string.desc_camerax_live_preview_activity,
            R.string.desc_cameraxsource_demo_activity,
          };

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    if (BuildConfig.DEBUG) {
      StrictMode.setThreadPolicy(
          new
StrictMode.ThreadPolicy.Builder().detectAll().penaltyLog().build());
      StrictMode.setVmPolicy(
          new StrictMode.VmPolicy.Builder()
              .detectLeakedSqlLiteObjects()
              .detectLeakedClosableObjects()
              .penaltyLog()
              .build());
    }
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate");

    setContentView(R.layout.activity_chooser);

    // Set up ListView and Adapter
    ListView listView = findViewById(R.id.test_activity_list_view);

    MyArrayAdapter adapter = new MyArrayAdapter(this,
android.R.layout.simple_list_item_2, CLASSES);
    adapter.setDescriptionIds(DESCRIPTION_IDS);

    listView.setAdapter(adapter);
    listView.setOnItemClickListener(this);

    if (!allPermissionsGranted()) {
      getRuntimePermissions();
    }
  }

  @Override
  public void onItemClick(AdapterView<?> parent, View view, int
```

```
position, long id) {
    Class<?> clicked = CLASSES[position];
    startActivity(new Intent(this, clicked));
  }

  private String[] getRequiredPermissions() {
    try {
      PackageInfo info =
          this.getPackageManager()
              .getPackageInfo(this.getPackageName(),
PackageManager.GET_PERMISSIONS);
      String[] ps = info.requestedPermissions;
      if (ps != null && ps.length > 0) {
        return ps;
      } else {
        return new String[0];
      }
    } catch (Exception e) {
      return new String[0];
    }
  }

  private boolean allPermissionsGranted() {
    for (String permission : getRequiredPermissions()) {
      if (!isPermissionGranted(this, permission)) {
        return false;
      }
    }
    return true;
  }

  private void getRuntimePermissions() {
    List<String> allNeededPermissions = new ArrayList<>();
    for (String permission : getRequiredPermissions()) {
      if (!isPermissionGranted(this, permission)) {
        allNeededPermissions.add(permission);
      }
    }

    if (!allNeededPermissions.isEmpty()) {
      ActivityCompat.requestPermissions(
          this, allNeededPermissions.toArray(new String[0]),
PERMISSION_REQUESTS);
    }
  }

  private static boolean isPermissionGranted(Context context, String
permission) {
    if (ContextCompat.checkSelfPermission(context, permission)
        == PackageManager.PERMISSION_GRANTED) {
      Log.i(TAG, "Permission granted: " + permission);
```

```
      return true;
    }
    Log.i(TAG, "Permission NOT granted: " + permission);
    return false;
  }

  private static class MyArrayAdapter extends ArrayAdapter<Class<?>> {

    private final Context context;
    private final Class<?>[] classes;
    private int[] descriptionIds;

    MyArrayAdapter(Context context, int resource, Class<?>[] objects)
{
      super(context, resource, objects);

      this.context = context;
      classes = objects;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {
      View view = convertView;

      if (convertView == null) {
        LayoutInflater inflater =
            (LayoutInflater)
context.getSystemService(LAYOUT_INFLATER_SERVICE);
        view = inflater.inflate(android.R.layout.simple_list_item_2,
null);
      }

      ((TextView)
view.findViewById(android.R.id.text1)).setText(classes[position].getSi
mpleName());
      ((TextView)
view.findViewById(android.R.id.text2)).setText(descriptionIds[position
]);

      return view;
    }

    void setDescriptionIds(int[] descriptionIds) {
      this.descriptionIds = descriptionIds;
    }
  }
}
private void populateFeatureSelector() {
  Spinner featureSpinner = findViewById(R.id.feature_selector);
  List<String> options = new ArrayList<>();
```

```
  options.add(OBJECT_DETECTION);
  options.add(OBJECT_DETECTION_CUSTOM);
  options.add(CUSTOM_AUTOML_OBJECT_DETECTION);
  options.add(FACE_DETECTION);
  options.add(BARCODE_SCANNING);
  options.add(TEXT_RECOGNITION);
  options.add(IMAGE_LABELING);
  options.add(IMAGE_LABELING_CUSTOM);
  options.add(CUSTOM_AUTOML_LABELING);
  options.add(POSE_DETECTION);
  options.add(SELFIE_SEGMENTATION);

  // Creating adapter for featureSpinner
  ArrayAdapter<String> dataAdapter = new ArrayAdapter<>(this,
R.layout.spinner_style, options);
  // Drop down layout style - list view with radio button

dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dr
opdown_item);
  // attaching data adapter to spinner
  featureSpinner.setAdapter(dataAdapter);
  featureSpinner.setOnItemSelectedListener(
      new OnItemSelectedListener() {

        @Override
        public void onItemSelected(
            AdapterView<?> parentView, View selectedItemView, int pos,
long id) {
          selectedMode = parentView.getItemAtPosition(pos).toString();
          createImageProcessor();
          tryReloadAndDetectInImage();
        }

        @Override
        public void onNothingSelected(AdapterView<?> arg0) {}
      });
}

private void populateSizeSelector() {
  Spinner sizeSpinner = findViewById(R.id.size_selector);
  List<String> options = new ArrayList<>();
  options.add(SIZE_SCREEN);
  options.add(SIZE_1024_768);
  options.add(SIZE_640_480);

  // Creating adapter for featureSpinner
  ArrayAdapter<String> dataAdapter = new ArrayAdapter<>(this,
R.layout.spinner_style, options);
  // Drop down layout style - list view with radio button

dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dr
```

```
opdown_item);
  // attaching data adapter to spinner
  sizeSpinner.setAdapter(dataAdapter);
  sizeSpinner.setOnItemSelectedListener(
      new OnItemSelectedListener() {

        @Override
        public void onItemSelected(
            AdapterView<?> parentView, View selectedItemView, int pos,
long id) {
          selectedSize = parentView.getItemAtPosition(pos).toString();
          tryReloadAndDetectInImage();
        }

        @Override
        public void onNothingSelected(AdapterView<?> arg0) {}
      });
}

@Override
public void onSaveInstanceState(Bundle outState) {
  super.onSaveInstanceState(outState);
  outState.putParcelable(KEY_IMAGE_URI, imageUri);
  outState.putString(KEY_SELECTED_SIZE, selectedSize);
}

private void startCameraIntentForResult() {
  // Clean up last time's image
  imageUri = null;
  preview.setImageBitmap(null);

  Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
  if (takePictureIntent.resolveActivity(getPackageManager()) != null)
{
    ContentValues values = new ContentValues();
    values.put(MediaStore.Images.Media.TITLE, "New Picture");
    values.put(MediaStore.Images.Media.DESCRIPTION, "From Camera");
    imageUri =
getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_U
RI, values);
    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
  }
}

private void startChooseImageIntentForResult() {
  Intent intent = new Intent();
  intent.setType("image/*");
  intent.setAction(Intent.ACTION_GET_CONTENT);
  startActivityForResult(Intent.createChooser(intent, "Select
```

```
Picture"), REQUEST_CHOOSE_IMAGE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
  if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK)
{
    tryReloadAndDetectInImage();
  } else if (requestCode == REQUEST_CHOOSE_IMAGE && resultCode ==
RESULT_OK) {
    // In this case, imageUri is returned by the chooser, save it.
    imageUri = data.getData();
    tryReloadAndDetectInImage();
  } else {
    super.onActivityResult(requestCode, resultCode, data);
  }
}

private void tryReloadAndDetectInImage() {
  Log.d(TAG, "Try reload and detect image");
  try {
    if (imageUri == null) {
      return;
    }

    if (SIZE_SCREEN.equals(selectedSize) && imageMaxWidth == 0) {
      // UI layout has not finished yet, will reload once it's ready.
      return;
    }

    Bitmap imageBitmap =
BitmapUtils.getBitmapFromContentUri(getContentResolver(), imageUri);
    if (imageBitmap == null) {
      return;
    }

    // Clear the overlay first
    graphicOverlay.clear();

    // Get the dimensions of the image view
    Pair<Integer, Integer> targetedSize = getTargetedWidthHeight();

    // Determine how much to scale down the image
    float scaleFactor =
        Math.max(
            (float) imageBitmap.getWidth() / (float)
targetedSize.first,
            (float) imageBitmap.getHeight() / (float)
targetedSize.second);
```

```
    Bitmap resizedBitmap =
        Bitmap.createScaledBitmap(
            imageBitmap,
            (int) (imageBitmap.getWidth() / scaleFactor),
            (int) (imageBitmap.getHeight() / scaleFactor),
            true);

    preview.setImageBitmap(resizedBitmap);

    if (imageProcessor != null) {
      graphicOverlay.setImageSourceInfo(
          resizedBitmap.getWidth(), resizedBitmap.getHeight(), /*
isFlipped= */ false);
      imageProcessor.processBitmap(resizedBitmap, graphicOverlay);
    } else {
      Log.e(TAG, "Null imageProcessor, please check adb logs for
imageProcessor creation error");
    }
  } catch (IOException e) {
    Log.e(TAG, "Error retrieving saved image");
    imageUri = null;
  }
}

private Pair<Integer, Integer> getTargetedWidthHeight() {
  int targetWidth;
  int targetHeight;

  switch (selectedSize) {
    case SIZE_SCREEN:
      targetWidth = imageMaxWidth;
      targetHeight = imageMaxHeight;
      break;
    case SIZE_640_480:
      targetWidth = isLandScape ? 640 : 480;
      targetHeight = isLandScape ? 480 : 640;
      break;
    case SIZE_1024_768:
      targetWidth = isLandScape ? 1024 : 768;
      targetHeight = isLandScape ? 768 : 1024;
      break;
    default:
      throw new IllegalStateException("Unknown size");
  }

  return new Pair<>(targetWidth, targetHeight);
}

private void createImageProcessor() {
  if (imageProcessor != null) {
    imageProcessor.stop();
```

```
  }
  try {
    switch (selectedMode) {
      case OBJECT_DETECTION:
        Log.i(TAG, "Using Object Detector Processor");
        ObjectDetectorOptions objectDetectorOptions =

PreferenceUtils.getObjectDetectorOptionsForStillImage(this);
        imageProcessor = new ObjectDetectorProcessor(this,
objectDetectorOptions);
        break;
      case OBJECT_DETECTION_CUSTOM:
        Log.i(TAG, "Using Custom Object Detector Processor");
        LocalModel localModel =
            new LocalModel.Builder()

.setAssetFilePath("custom_models/object_labeler.tflite")
                .build();
        CustomObjectDetectorOptions customObjectDetectorOptions =

PreferenceUtils.getCustomObjectDetectorOptionsForStillImage(this,
localModel);
        imageProcessor = new ObjectDetectorProcessor(this,
customObjectDetectorOptions);
        break;
      case CUSTOM_AUTOML_OBJECT_DETECTION:
        Log.i(TAG, "Using Custom AutoML Object Detector Processor");
        LocalModel customAutoMLODTLocalModel =
            new
LocalModel.Builder().setAssetManifestFilePath("automl/manifest.json").
build();
        CustomObjectDetectorOptions customAutoMLODTOptions =

PreferenceUtils.getCustomObjectDetectorOptionsForStillImage(
                this, customAutoMLODTLocalModel);
        imageProcessor = new ObjectDetectorProcessor(this,
customAutoMLODTOptions);
        break;
      case FACE_DETECTION:
        imageProcessor = new FaceDetectorProcessor(this);
        break;
      case BARCODE_SCANNING:
        imageProcessor = new BarcodeScannerProcessor(this);
        break;
      case TEXT_RECOGNITION:
        imageProcessor = new TextRecognitionProcessor(this);
        break;
      case IMAGE_LABELING:
        imageProcessor = new LabelDetectorProcessor(this,
ImageLabelerOptions.DEFAULT_OPTIONS);
        break;
```

```
      case IMAGE_LABELING_CUSTOM:
        Log.i(TAG, "Using Custom Image Label Detector Processor");
        LocalModel localClassifier =
            new LocalModel.Builder()

.setAssetFilePath("custom_models/bird_classifier.tflite")
                .build();
        CustomImageLabelerOptions customImageLabelerOptions =
            new
CustomImageLabelerOptions.Builder(localClassifier).build();
        imageProcessor = new LabelDetectorProcessor(this,
customImageLabelerOptions);
        break;
      case CUSTOM_AUTOML_LABELING:
        Log.i(TAG, "Using Custom AutoML Image Label Detector
Processor");
        LocalModel customAutoMLLabelLocalModel =
            new
LocalModel.Builder().setAssetManifestFilePath("automl/manifest.json").
build();
        CustomImageLabelerOptions customAutoMLLabelOptions =
            new
CustomImageLabelerOptions.Builder(customAutoMLLabelLocalModel)
                .setConfidenceThreshold(0)
                .build();
        imageProcessor = new LabelDetectorProcessor(this,
customAutoMLLabelOptions);
        break;
      case POSE_DETECTION:
        PoseDetectorOptionsBase poseDetectorOptions =
            PreferenceUtils.getPoseDetectorOptionsForStillImage(this);
        Log.i(TAG, "Using Pose Detector with options " +
poseDetectorOptions);
        boolean shouldShowInFrameLikelihood =

PreferenceUtils.shouldShowPoseDetectionInFrameLikelihoodStillImage(thi
s);
        boolean visualizeZ =
PreferenceUtils.shouldPoseDetectionVisualizeZ(this);
        boolean rescaleZ =
PreferenceUtils.shouldPoseDetectionRescaleZForVisualization(this);
        boolean runClassification =
PreferenceUtils.shouldPoseDetectionRunClassification(this);
        imageProcessor =
            new PoseDetectorProcessor(
                this, poseDetectorOptions,
shouldShowInFrameLikelihood, visualizeZ, rescaleZ,
                runClassification, /* isStreamMode = */false);
        break;
      case SELFIE_SEGMENTATION:
        imageProcessor = new SegmenterProcessor(this, /* isStreamMode=
```

```
*/ false);
        break;
      default:
        Log.e(TAG, "Unknown selectedMode: " + selectedMode);
    }
  } catch (Exception e) {
    Log.e(TAG, "Can not create image processor: " + selectedMode, e);
    Toast.makeText(
            getApplicationContext(),
            "Can not create image processor: " + e.getMessage(),
            Toast.LENGTH_LONG)
        .show();
  }
}
```

Сторінка інтерфейсу

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10dp"
    android:fillViewport="true">

    <LinearLayout
        android:id="@+id/Linear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fillViewport="true"
        android:orientation="vertical">

        <TextView
            android:id="@+id/vehicleNumberTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/fareTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/workTimeTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:textSize="24sp" />
```

```
        <TextView
            android:id="@+id/intervalTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/shortDescriptionTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:textSize="24sp" />
    </LinearLayout>
</ScrollView>
```

Висновки до розділу 4

У наш час мобільні пристрої досить потужні, вони мають щось схоже на очі і можуть розпізнавати предмети та отримувати деяку інформацію на основі цього предмета.

Прототип це чорновий варіант програми, створений здебільш для того, щоб перевірити влучність концепції, архітектури, функціональних та технологічних рішень, а також для того, щоб продемонструвати робочу програму.

Можна побачити, як працює додаток, зчитує текст, який він бачить, зчитує номери транспортних засобів і видає на екран мобільного пристрою інформацію про маршрут, інформацію бере з API, також наданий код класів.

ВИСНОВКИ

Значна тема – комп'ютерний зір. Була досліджена концепція комп'ютерного зору та застосування в реальному світі, щоб зрозуміти актуальність цієї галузі, прикладів його використання, причини, що стосуються питання, чому люди потребують цього, і чому важливо розширити область комп'ютерний зору.

КЗ – це дисципліна, який витягує інформацію з зображень, а зображення можуть бути різними за типом. Можуть бути фотографії, відео, набори фотографій або зображення для медичних цілей і інших галузей. Останнім часом алгоритми, які працюють з зображеннями та об'єднати кольорову інформацію про точки та їх просторове положення, стали дуже популярними, оскільки датчики, які отримують таку інформацію, стали набагато доступними та потужними, і це стало набагато простіше для роботів або комп'ютерів, щоб зрозуміти світ.

Ідея розробленого додатку полягає в тому, що люди не завжди точно знають, як дістатися до потрібного місця, наприклад, якщо вони іноземці і нічого не знають про маршрути місцевих транспортних засобів, тому за допомогою програми можна зрозуміти, чи може конкретний транспортний засіб доставити користувача до потрібного місця.

Загалом, у процесі втілення даного дипломного проекту у життя, було отримано розуміння сучасного стану концепції комп’ютерного зору та його частини – оптичного розпізнавання символів та їх завдань та проблем. Також була отримана інформація про концепції, необхідні для реалізації програми.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Додатки ком'ютерного зору/Оптичне розпізнавання символів. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/cloud/blog/optical-character-recognition

2. Система виявлення та зчитування штрих коду. [Електронний ресурс]. – Режим доступу: https://opr.crimea.ru/uk/content/get/id/116

3. Область комп'ютерного зору в медицині. [Електронний ресурс]. – Режим доступу: https://www.axxonsoft.com/ua/products/intellect/additional_modules/lpr/

4. Ком'ютерний зір. [Електронний ресурс]. – Режим доступу: https://www.ibm.com/topics/computer-vision

5. Що таке ком'ютерний зір. [Електронний ресурс]. – Режим доступу: https://www.v7labs.com/blog/what-is-computer-vision

6. Індетифікація, виявлення. [Електронний ресурс]. – Режим доступу: https://www.arhivinfo.ru/1-101938.html

7. Довідник по форматам файлів Word, Excel і PowerPoint. [Електронний ресурс]. – Режим доступу: ela.kpi.ua/bitstream/123456789/25816/1/Vedmedenko_magistr.docx

8. Довідник по форматам файлів Word, Excel і PowerPoint. [Електронний ресурс]. – Режим доступу: scs.kpi.ua/sites/default/files/files/2017/Бакалавр...ія про розробку.docx

9. Google КЗ. [Електронний ресурс]. – Режим доступу: masters.donntu.org/2018/fknt/medvedev/diss/indexu.htm