Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розроблення програмного засобу управління роботизованим маніпулятором з використанням штучного інтелекту

(тема)

Виконав:
здобувач 2 року навчання,
групи КТРСм-23-2

Ахмад Фаді Халєд

Спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Тип програми Освітньо-професійна

Освітня програма Комп'ютеризовані та робототехнічні системи

Керівник к.т.н. Сичова О.В.

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2025р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР_____
(підпис)
« 25 » листопада 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ахмад Фаді Халєд
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного засобу управління роботизованим маніпулятором з використанням штучного інтелекту
Затверджена наказом по університету від 25 листопад 2024р. № 1239 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 22.01.2025 р.

3. Вихідні дані до роботи
3.1 Кількість ступенів свободи маніпулятора – 3.
3.2 Тип маніпулятора – ангулярний.
3.3 Спосіб визначення положення об'єкту на робочому полі – візуальний.
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз предметної області
4.2 Аналіз методів машинного навчання
4.3 Проектування автоматизованої системи управління роботизованим маніпулятором з використанням штучного інтелекту
4.4 Експериментальні дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Презентаційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

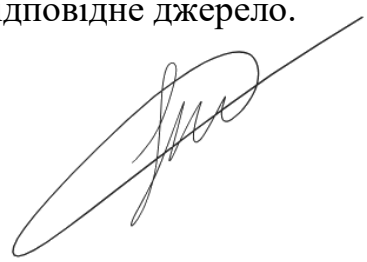
№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури за темою кваліфікаційної роботи	10.10.24 – 17.10.24	Виконано
2	Аналіз предметної області	18.10.24 – 30.10.24	Виконано
3	Аналіз методів машинного навчання	01.11.24 – 16.11.24	Виконано
4	Проектування автоматизованої системи управління роботизованим маніпулятором з використанням штучного інтелекту	17.11.24 – 10.12.24	Виконано
5	Експериментальні дослідження	11.12.24 – 20.12.24	Виконано
6	Оформлення пояснювальної записки	21.12.24 – 13.01.25	Виконано
7	Подання роботи на перевірку Інтернет-сервісом Unicheck	14.01.25 – 15.01.25	
8	Подання роботи на рецензію	16.01.25 – 17.01.25	
9	Подання роботи на підпис зав. кафедри	17.01.25 – 18.01.25	
10	Подання кваліфікаційної роботи в ЕК	22.01.25	

Дата видачі завдання 02.09.2024 р.

Здобувач _____ Ахмад Фаді Халєд
(підпис)

Керівник роботи _____ к.т.н. Сичова О.В.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.



«10» січня 2025 р. Ахмад Фаді Халєд

РЕФЕРАТ

Пояснювальна записка: 76 с., 29 рис., 3 дод., 15 джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, РОБОТИЗОВАНИЙ МАНІПУЛЯТОР, АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ, АЛГОРИТМ, ПРОГРАМА.

Об'єктом дослідження в даній роботі є процес автономного управління роботизованим маніпулятором.

Предмет дослідження – автоматизованої системи управляння роботизованим маніпулятором з елементами штучного інтелекту.

Метою цієї роботи є підвищення ефективності методу управління роботизованим маніпулятором за рахунок використання елементів ШІ в складі автоматизованої системи управляння.

Наведено основні компоненти системи автономного управління роботою маніпулятора та проведено аналіз переваг при застосуванні елементів штучного інтелекту при управлінні роботизованими маніпуляторами.

Побудована типова архітектура системи управління роботизованим маніпулятором з використанням ШІ та описано її складові. Дано визначення терміну «Машинне навчання» та проведено аналіз методів машинного навчання.

Виконано розроблення алгоритму роботи автоматизованої системи. Описано алгоритм роботи основної частини автоматизованої системи управління роботизованим маніпулятором.

Проведені експериментальні дослідження, що проведені в четвертому розділі кваліфікаційної роботи показала, що запропонований метод працює як для тестового зображення так і для потокового відео в реальному часі.

Також, отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме п. 9.2 «Забезпечити розширення використання електротранспорту та відповідної мережі інфраструктури».

ABSTRACT

Explanatory note: 76 p., 29 fig., 3 appendices, 15 sources.

ARTIFICIAL INTELLIGENCE, ROBOTIC MANIPULATOR, AUTOMATED CONTROL SYSTEM, ALGORITHM, PROGRAM.

The object of research in this work is the process of autonomous control of a robotic manipulator.

The subject of research is an automated control system for a robotic manipulator with elements of artificial intelligence.

The purpose of this work is to increase the efficiency of the method of controlling a robotic manipulator by using AI elements as part of an automated control system.

The main components of the autonomous control system for the manipulator are presented and the advantages of using artificial intelligence elements in controlling robotic manipulators are analyzed.

A typical architecture of a robotic manipulator control system using AI is constructed and its components are described. The term "Machine Learning" is defined and machine learning methods are analyzed.

The algorithm of the automated system has been developed. The algorithm of the main part of the automated control system for the robotic manipulator is described.

Experimental studies conducted in the fourth section of the qualification work showed that the proposed method works both for the test image and for streaming video in real time.

Also, the results of the work can be attributed to Sustainable Development Goal 9 "Industry, Innovation and Infrastructure", namely item 9.2 "Ensure the expansion of the use of electric transport and the corresponding infrastructure network".

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

МН – машинне навчання;

ШІ – штучний інтелект;

AI – artificial intelligence;

API – application programming interface;

DL – deep learning;

KNN – K-nearest neighbors;

OpenCV – open source computer vision library;

RBF – radial basis function;

SVM – support vector machine.

ВСТУП

Роботизовані маніпулятори є ключовими елементами автоматизації в багатьох галузях промисловості. Завдяки своєму широкому застосуванню – від збирання та обробки матеріалів до медичних операцій – вони значно підвищують продуктивність і точність виконання складних завдань. З розвитком технологій штучного інтелекту (ШІ) можливості роботизованих маніпуляторів значно розширилися. ШІ дозволяє роботу не лише виконувати запрограмовані операції, але й адаптуватися до нових умов, навчатися і приймати складні рішення в режимі реального часу.

Розробка програмного забезпечення для управління роботизованими маніпуляторами з використанням штучного інтелекту є важливим кроком у розвитку автоматизації. Завдяки використанню ШІ маніпулятори стають більш автономними, адаптивними та здатними вирішувати складні завдання в динамічних умовах. Впровадження таких технологій відкриває нові можливості для промисловості, медицини, логістики та інших галузей, де потрібна висока точність та швидкість виконання операцій.

Метою цієї роботи є підвищення ефективності методу управління роботизованим маніпулятором за рахунок використання елементів ШІ в складі автоматизованої системи управляння.

Об'єктом дослідження в даній роботі є процес автономного управління роботизованим маніпулятором.

Предмет дослідження – автоматизованої системи управляння роботизованим маніпулятором з елементами штучного інтелекту.

Для досягнення мети необхідно вирішити наступні завдання:

– виконати порівняльний аналіз сфер застосування роботизованих маніпуляторів;

– навести основні компоненти системи автономного управління роботою маніпулятора;

– провести аналіз переваг при застосуванні елементів штучного інтелекту при управлінні роботизованими маніпуляторами;

– провести аналіз методів машинного навчання;

– подобувати типову архітектура системи управління роботизованим маніпулятором з використанням ШІ та описано її складові;

– підібрати компоненти для побудови автоматизованої системи;

– виконати побудову алгоритму та розробити програму для проведення експериментальних досліджень;

– розробити програму для проведення експериментальних досліджень;

– виконати експериментальні дослідження для підтвердження правильності теоретичних рішень.

Роботу необхідно оформити згідно нормативної документації [1 – 6].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сфер застосування роботизованих маніпуляторів

Робот-маніпулятор складається з основних компонентів: маніпуляційних елементів, системи управління, засобів пересування та сенсорів. Маніпуляційний механізм включає рухомі елементи двох категорій:

– елементи, що забезпечують лінійні переміщення;

– елементи, що забезпечують обертальні рухи.

Деякі типи маніпуляторів обладнані захватними пристроями. Найбільш універсальні захоплювачі мають конструкцію, подібну до людської руки, і здійснюють захоплення за допомогою механічних пальців. Для роботи з однотипними деталями на виробництві розробляють спеціалізовані захватні системи. На маніпуляторах можуть бути встановлені спеціалізовані інструменти, такі як зварювальні апарати або викрутки (рис. 1.1) [7].

Рисунок 1.1 – Зварювальний промисловий робот

Область застосування промислових роботів характеризується в рамках проектів виробництва і виходячи з них отримується оптимальний варіант

задачі робота, їх кількість та інтеграції у виробничій процес.

Найбільш розповсюдженими функціями, що виконують роботи є:

– дугове або точкове зварювання;

– переміщення деталей (переміщення деталей с конвеєра на конвеєр;

– укладка деталей в тару, перенос деталей по приміщенню);

– лиття;

– штамповка;

– ковка;

– фарбування;

– контроль якості виробів.

Перевагами застосування промислових роботів є:

– економічна ефективність при заміні та скороченні промислових затрат на заробітні плати робочих за рахунок автоматизації їх робочих місць;

– оптимізована ефективність виробництва за рахунок безперервної роботи роботів с оптимальною швидкістю;

– обмеження роботи людини або її заміна в небезпечних умовах для життя за рахунок впровадження роботів;

– збільшення конкурентоспроможності;

– покращення якості продукції, що пов’язано з підвищенням точності виконування технологічних операцій.

Основними недоліками є:

– втрата робочих місць;

– збільшення інвестиційних затрат.

Автономне керування роботом здійснюється за допомогою алгоритму його роботи з вводу даних з його датчиків. Перевагами автономного керування є велика автономність робота та вирішення ним складних задач самостійно без участі людини, а недоліки залежать від якості програмування [8]. При автономному керуванні використовуються засоби розпізнавання образів для визначення положення об’єктів маніпуляції в робочій зоні (рис. 1.2).

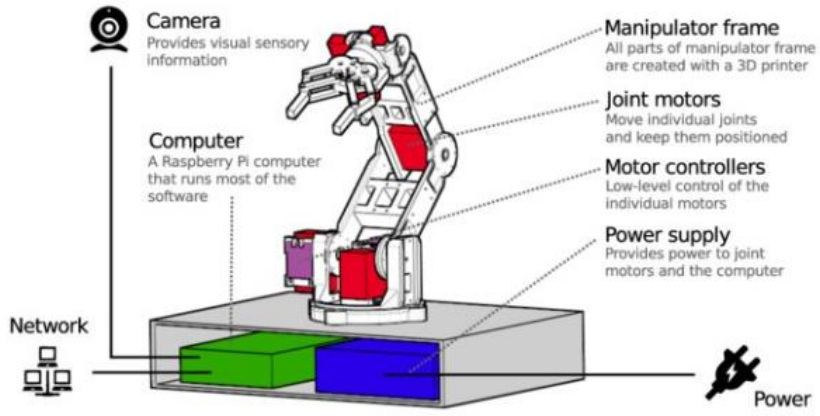


Рисунок 1.2 – Основні компоненти автономного управління роботою маніпулятора

Вебкамера використовується як один з датчиків, що дає інформацію про розташування об’єктів в робочій зоні. Поєднання з промисловою мережею дає можливість інтеграції робота в виробничий процес. Промислові мережі забезпечують високошвидкісний обмін даними між роботизованими маніпуляторами, контролерами та іншими елементами системи (датчиками, камерами, програмними серверами). Це дає можливість швидкого обміну інформацією про поточний стан обладнання, завдання та робочі параметри.

Завдяки мережам можливо дистанційне керування роботом та моніторинг його стану. Це важливо для великих промислових підприємств, де фізичний доступ до всіх частин системи може бути ускладнений.

Роботизовані маніпулятори генерують величезні обсяги даних, особливо якщо вони працюють із сенсорами або камерами. Хмарні обчислення дозволяють обробляти ці дані у великих масштабах та в реальному часі, використовуючи потужні віддалені сервери.

Хмара дає доступ до алгоритмів штучного інтелекту та машинного навчання, які допомагають аналізувати дані, приймати рішення та адаптувати поведінку маніпулятора до умов виробництва. Наприклад, на основі аналізу даних маніпулятор може вдосконалювати свої рухи, оптимізувати маршрути або виявляти несправності.

Віддалені обчислення на основі великих даних дозволяють дистанційно оновлювати програмне забезпечення роботизованого маніпулятора, забезпечуючи його найновішими функціями або виправленням помилок. Це зменшує час простою та забезпечує актуальність системи без фізичного втручання.

Хмарні обчислювальні платформи надають інструменти для аналітики, які допомагають виявляти вузькі місця у виробничому процесі, прогнозувати несправності або оцінювати ефективність роботи маніпуляторів.

Переважна кількість фірм виробників робототехніки розробляють власні мови програмування і засоби допоміжного програмного забезпечення. Фірми, що безпосередньо займаються впровадженням робототехніки у виробничі процеси (системні інтегратори), роблять основний упор на допоміжному програмному забезпеченні адаптованому до конкретних практичних умов, розробкам нових та модернізації старих технологій, впровадженні вимірюючих систем, що дозволяють підвищити точність та якість вироблюваної продукції (рис. 1.3).

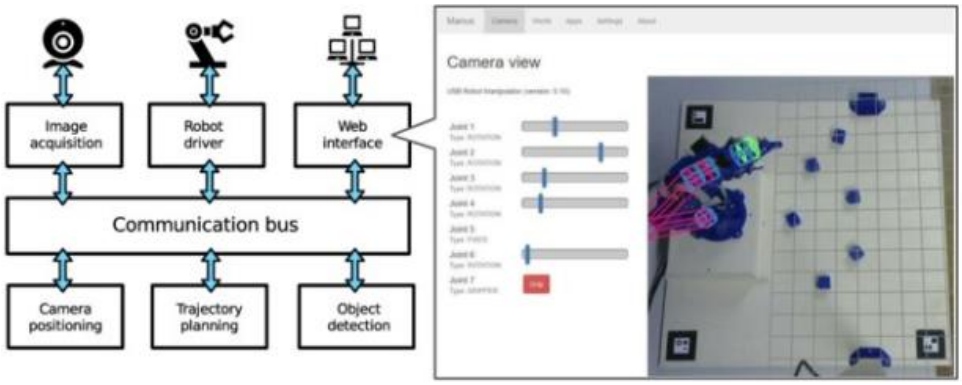


Рисунок 1.3 – Приклад роботи програми для управління роботизованим маніпулятором

Промислові роботи мають програмну оболонку, що інтегрує в собі різноманітні додаткові модулі розширень. Так, наприклад, існує можливість підключення модулів комунікацій з зовнішніми сенсорними пристроями:

система відео керування, система заміру прикладного навантаження, обертаючого моменту, що дає можливість реагування на зміну зовнішніх умов.

Часто контролер робота зв’язаний з програмованим логічним пристроєм, що відповідає за взаємодію робота і периферійного обладнання.

Промислові мережі забезпечують надійний зв’язок та інтеграцію роботизованих систем на місцевому рівні, а хмарні обчислення надають можливість розширеної обробки даних, навчання системи та підвищення ефективності. Разом ці технології створюють основу для автономного, гнучкого та інтелектуального управління роботизованими маніпуляторами в промислових умовах.

1.2 Аналіз переваг при застосуванні елементів штучного інтелекту при управлінні роботизованими маніпуляторами

Традиційні методи управління маніпуляторами ґрунтувалися на жорстко запрограмованих алгоритмах, які добре працювали в умовах передбачуваного середовища, але не могли адаптуватися до динамічних змін. ШІ, зокрема методи машинного навчання та глибокі нейронні мережі, дозволяють роботам вивчати нові сценарії на основі попередніх даних, робити висновки та приймати рішення самостійно. Це розширює можливості маніпуляторів, дозволяючи їм працювати в умовах неповної або неточної інформації, що важливо для багатьох сучасних виробничих процесів.

Основними напрямами, в яких штучний інтелект може поліпшити управління роботизованими маніпуляторами, є:

– оптимізація рухів і траєкторій;

– розпізнавання об'єктів і контроль середовища;

– адаптивне планування та прийняття рішень;

– інтерактивне навчання роботів.

Основою програмного засобу для управління роботизованим маніпулятором є розробка програмної архітектури, яка поєднує в собі класичні

методи управління та алгоритми ШІ. Типова архітектура складається з таких модулів (рис. 1.4):

– модуль опитування сенсорів;

– модуль планування рухів;

– модуль прийняття рішень;

– модуль управління двигунами;

– модуль навчання.

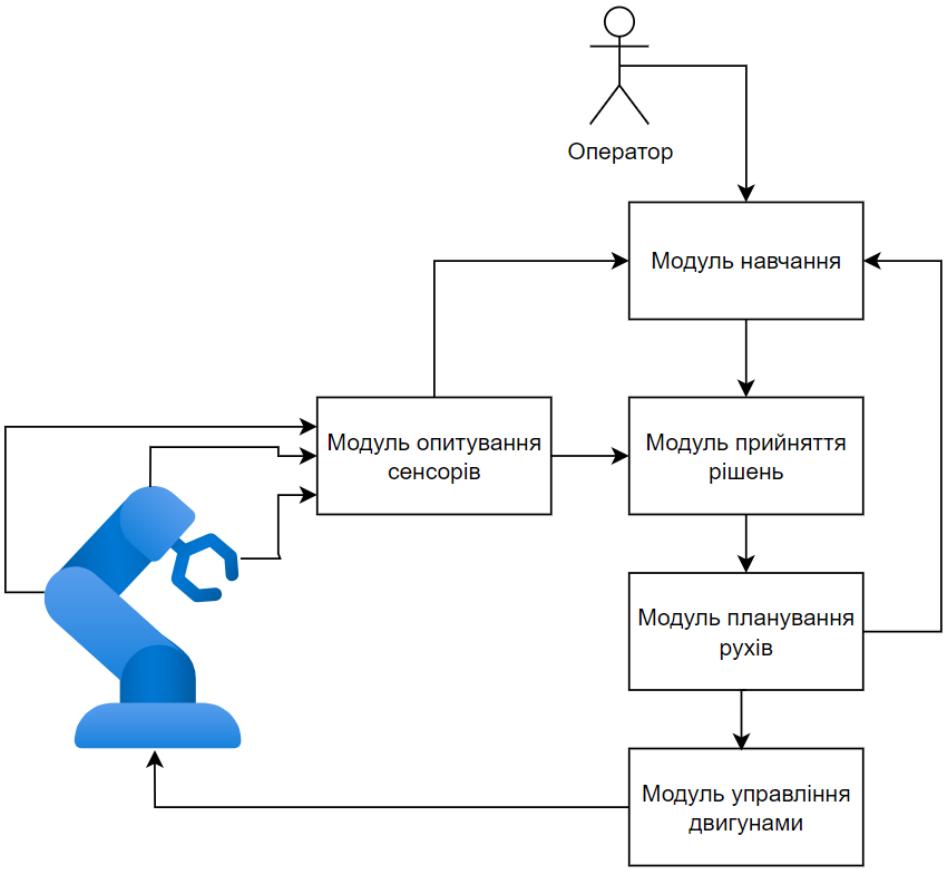


Рисунок 1.4 – Типова архітектура системи управління роботизованим маніпулятором з використанням ШІ

Модуль опитування сенсорів відповідає за зчитування інформації з сенсорів (камери, лідара, датчиків сили тощо) і передачу цих даних в обробку.

В системі інтелектуального управління роботом-маніпулятором "Модуль опитування сенсорів" виконує кілька важливих функцій,

забезпечуючи збір, обробку та аналіз даних, отриманих від різних сенсорів. Це дозволяє маніпулятору взаємодіяти з навколишнім середовищем та виконувати завдання з високою точністю й адаптивністю.

Модуль опитування сенсорів збирає інформацію з різних типів сенсорів, таких як:

– датчики положення (енкодери), що визначають поточну позицію кожного рухомого елемента маніпулятора;

– датчики сили та моменту, що вимірюють навантаження на маніпулятор під час виконання завдань, а це допомагає уникнути перевантаження або пошкодження;

– візуальні сенсори (камери) надають дані про об’єкти навколо, включаючи їх розташування, форму та орієнтацію;

– датчики наближення використовуються для виявлення об'єктів поблизу маніпулятора, що важливо для запобігання зіткненням.

Після обробки дані передаються в основну систему управління роботом, яка використовує їх для прийняття рішень. На основі інформації від сенсорів система може коригувати рухи маніпулятора, забезпечуючи точне позиціонування та виконання завдань. Отримана інформація також дозволяє реагувати на зміни в навколишньому середовищі (наприклад, обхід перешкод). Сенсори дозволяють адаптувати стратегії управління залежно від поточних умов (наприклад, зменшення швидкості при виявленні перешкод).

З рисунку 1.4 видно, модуль опитування сенсорів забезпечує постійний зворотний зв'язок між фізичними компонентами маніпулятора та його системою управління. Це дозволяє контролювати виконання команд та вносити необхідні корективи в реальному часі.

Модуль планування рухів використовує дані для визначення оптимальної траєкторії маніпулятора. Алгоритми машинного навчання можуть застосовуватися для адаптації до умов середовища.

Модуль планування рухів відіграє ключову роль в управлінні роботом-маніпулятором, забезпечуючи безпечне, ефективне та точне виконання його

завдань. Взаємодіючи з "модулем опитування сенсорів", модуль планування рухів обробляє інформацію про поточний стан маніпулятора та навколишнього середовища, щоб створити оптимальні траєкторії для переміщення та маніпуляцій.

Основні функції модуля планування рухів:

– розробка траєкторій руху;

– розрахунок кінематики;

– уникнення перешкод;

– адаптація до змін;

– координація дій;

– забезпечення плавності та безперервності руху.

Даний модуль планує траєкторії руху робота, враховуючи поточні позиції маніпулятора та необхідні точки досягнення. Він створює шлях, який забезпечить: оптимізацію переміщень, щоб мінімізувати час виконання завдань, а також уникнення зіткнень із об’єктами навколо маніпулятора. Також розраховується точне досягнення цілей, наприклад, планується позиціонування захватного пристрою для точного захвату об'єкта.

Модуль планування рухів використовує алгоритми прямої та зворотної кінематики для обчислення рухів ланок маніпулятора. Ці алгоритми дозволяють визначити, як повинні рухатися всі елементи маніпулятора для досягнення заданої кінцевої позиції.

Пряма кінематика визначає положення кінцевого ефектора на основі поточних кутів та переміщень рухомих ланок. Зворотна кінематика розраховує необхідні кути та переміщення для досягнення кінцевої точки.

Враховуючи дані з "Модуля опитування сенсорів", модуль планування рухів коригує траєкторії для уникнення зіткнень з об'єктами в робочій зоні маніпулятора. Він забезпечує безпечний рух, навіть якщо навколишнє середовище динамічно змінюється (наприклад, виявляються нові перешкоди або змінюється позиція об'єктів).

Якщо сенсори виявляють зміни в середовищі або непередбачені ситуації (наприклад, зміна положення об’єкта або перешкод), модуль планування рухів переглядає раніше створені траєкторії та адаптує їх. Це забезпечує можливість автономної роботи робота в непередбачуваних умовах.

У випадках, коли робот-маніпулятор взаємодіє з іншими роботами або системами, модуль планування рухів синхронізує його дії з іншими учасниками процесу. Це дозволяє уникнути конфліктів у рухах і забезпечити злагоджену роботу всіх елементів системи.

Крім всього іншого, модуль планування рухів забезпечує плавність переміщень, уникаючи різких ривків, що можуть призвести до зниження точності виконання завдань або пошкодження маніпулятора. Для цього використовуються методи інтерполяції, що дозволяють генерувати безперервні траєкторії руху.

Модуль прийняття рішень використовує алгоритми ШІ для аналізу ситуації і вибору дій (рис. 1.4). Він може працювати на основі нейронних мереж, що моделюють поведінку робота, враховуючи його попередній досвід.

Модуль управління перетворює обчислені команди в реальні дії маніпулятора. Цей модуль забезпечує зв'язок з апаратними частинами маніпулятора.

Модуль навчання забезпечує можливість автономного навчання робота шляхом імітації, отримання зворотного зв'язку та оптимізації дій на основі результатів виконання.

Алгоритми машинного навчання та глибокого навчання дозволяють роботу адаптувати свої дії, ґрунтуючись на історичних даних та власному досвіді. Для роботи маніпулятора алгоритми машинного навчання можуть використовуватися у таких випадках:

– контроль траєкторії;

– розпізнавання об'єктів;

– автономне навчання.

Використовуючи ШІ робот вчиться коригувати траєкторію руху відповідно до перешкод або змін у середовищі, щоб досягти більш плавного та точного виконання завдань. Використання алгоритмів комп'ютерного зору та розпізнавання об'єктів дозволяє маніпулятору автоматично розпізнавати об'єкти і взаємодіяти з ними.

Маніпулятор може навчатися на основі підкріплення (reinforcement learning), коли він отримує винагороду за правильні дії, що допомагає поліпшити продуктивність у складних і мінливих середовищах.

У рамках розробки програмного забезпечення ШІ використовується для адаптивного управління роботом. Це означає, що маніпулятор не просто виконує фіксовані команди, а здатний адаптуватися до умов середовища.

Наприклад, у разі зміни розташування об'єкта або виникнення несподіваних перешкод, ШІ може перебудувати траєкторію маніпулятора в реальному часі. Для цього можуть використовуватися методи глибинного навчання або методи навчання з підкріпленням. Ці підходи дозволяють роботу швидко навчатися на основі досвіду, знижуючи потребу в ручному налаштуванні кожного сценарію.

Розробка програмного забезпечення з використанням штучного інтелекту для роботизованих маніпуляторів зіштовхується з кількома викликами:

– для навчання ШІ потрібні великі набори даних, що вимагає ресурсів для збору та обробки інформації;

– алгоритми штучного інтелекту можуть бути обчислювально інтенсивними, що потребує потужного апаратного забезпечення;

– використання ШІ в реальному світі підвищує вимоги до безпеки та надійності роботи робота.

Проте перспективи розвитку є надзвичайно великими. Впровадження штучного інтелекту дозволяє маніпуляторам працювати в складних і непередбачуваних середовищах, підвищуючи їх гнучкість та ефективність.

1.3 Висновки по першому розділу

В результаті виконання першого розділу кваліфікаційної роботи виконано аналіз сфер застосування роботизованих маніпуляторів та показані переваги застосування штучного інтелекту в порівнянні з традиційними засобами керування. Розглянуті основні компоненти роботів маніпуляторів.

Хмарні обчислювальні платформи надають інструменти для аналітики, які допомагають виявляти вузькі місця у виробничому процесі, прогнозувати несправності або оцінювати ефективність роботи маніпуляторів.

Проаналізовано типову архітектуру системи управління роботизованим маніпулятором з використанням ШІ.

2 АНАЛІЗ МЕТОДІВ МАШИННОГО НАВЧАННЯ

2.1 Визначення терміну «Машинне навчання»

Машинне навчання (ML) – це пошук закономірностей у числах та використання цих закономірностей для побудови прогнозів. Машинне навчання дозволяє навчати модель, використовуючи ряди або послідовності 1 і 0, і вчитися цих даних так, щоб, отримавши нову послідовність, модель могла передбачити, яким буде результат. Навчання – це процес, в ході якого ML знаходить закономірності, можливі для використання з метою прогнозування майбутніх результатів, і саме звідси походить слово «навчання» в терміні «машинне навчання».

Машинне навчання та штучний інтелект є одними з найважливіших та найінноваційніших технологій сучасності. Вони лежать в основі багатьох передових розробок у галузі науки, техніки, медицини, бізнесу та багатьох інших сфер. Машинне навчання (МН) – це підрозділ штучного інтелекту (ШІ), який дозволяє комп'ютерам навчатися на основі даних, не будучи спеціально запрограмованими для виконання певних завдань. Штучний інтелект, у свою чергу, охоплює всі системи, які здатні імітувати людську інтелектуальну діяльність: розпізнавання мови, аналіз зображень, прийняття рішень, прогнозування та автоматизація різних процесів.

Машинне навчання використовує алгоритми, які навчаються на великих наборах даних, щоб зробити висновки та прогнози. Це досягається за допомогою різних моделей, таких як лінійна регресія, дерева рішень, нейронні мережі тощо. Один із ключових підходів до машинного навчання – це навчання з учителем, коли алгоритму надається набір даних з відповідними мітками, і він намагається побудувати модель, яка передбачатиме правильні відповіді для нових даних. Існує також навчання без учителя, де система намагається знайти структуру в даних без чітко визначених відповідей, і

навчання з підкріпленням, у якому агент вивчає стратегії дій через отримання нагород за правильні рішення.

Завдяки прогресу в обчислювальній потужності та доступності великих обсягів даних, нейронні мережі та глибоке навчання стали ключовими технологіями в ШІ. Глибоке навчання базується на багатошарових нейронних мережах, які здатні обробляти складні дані та виявляти приховані патерни. Це дозволило досягти значних успіхів у таких областях, як розпізнавання мови та зображень, автономні транспортні засоби, медична діагностика та багато іншого.

Машинне навчання і ШІ вже зараз активно використовуються в багатьох сферах життя. Наприклад, у фінансовій сфері ШІ допомагає прогнозувати ринки, виявляти шахрайство та аналізувати ризики. В медицині ці технології використовуються для аналізу медичних зображень, діагностики захворювань і навіть для розробки нових ліків. Виробничі компанії впроваджують ШІ для автоматизації процесів та покращення продуктивності, а в сфері розваг машинне навчання застосовується для створення рекомендаційних систем, таких як на платформах потокового відео або музики.

Однак із розвитком штучного інтелекту з’являються також і виклики. Одним із найсерйозніших є питання етики та приватності, оскільки алгоритми ШІ можуть використовуватися для контролю та маніпуляції великими обсягами даних про людей. Інший виклик полягає в потенційній заміні людей машинами в різних галузях, що може призвести до змін на ринку праці.

Відповідальне використання ШІ є однією з ключових тем сьогоднішнього часу. Для того, щоб забезпечити безпеку та справедливість у розвитку штучного інтелекту, важливо розробляти відповідні правила та регуляції, які допоможуть знизити ризики, пов'язані з його використанням.

Терміни «машинне навчання» і «штучний інтелект» (artificial intelligence, AI) сьогодні використовуються практично як взаємозамінні, проте насправді кожен із них має власне значення, як показано на рис. 2.1 [11].

З технічної точки зору машинне навчання – це підмножина штучного

інтелекту, яке включає в себе не тільки моделі машинного навчання, але й інші типи моделей, такі як експертні системи (системи, що приймають рішення на основі заданих вами правил) та системи навчання з підкріпленням, які навчаються поведінці, отримуючи винагороду за позитивні результати та штрафи за негативні. Прикладом системи навчання з підкріпленням є AlphaGo [11], яка стала першою комп'ютерною програмою, яка обіграла професійного гравця в го. Вона тренується на вже зіграних партіях та самостійно освоює стратегії перемоги.

Рисунок 2.1 – Взаємозв'язок між машинним навчанням, глибоким навчанням та штучним інтелектом

З практичної точки зору те, що більшість людей сьогодні називають штучним інтелектом, насправді є глибоким навчанням, яке є підмножиною машинного навчання. Глибоке навчання (Deep Learning, DL) – це машинне навчання з використанням нейронних мереж. (Існують форми глибокого навчання без використання нейронних мереж – одним із прикладів є глибокі машини Больцмана, але в переважній більшості випадків глибоке навчання

сьогодні здійснюється із застосуванням нейронних мереж.) Таким чином, моделі машинного навчання можна розділити на звичайні моделі, які використовують алгоритми навчання для моделювання закономірностей у даних, і моделі глибокого навчання на основі нейронних мереж.

Машинне навчання зосереджується на створенні систем або моделей, які можуть вчитися на даних і покращувати свою продуктивність у конкретних завданнях без необхідності явного програмування, змушуючи їх вчитися на минулому досвіді чи прикладах для прийняття рішень на нових даних. Це відрізняється від традиційного програмування, де люди-програмісти пишуть правила в коді, перетворюючи вхідні дані в бажані результати (рис. 2.2).

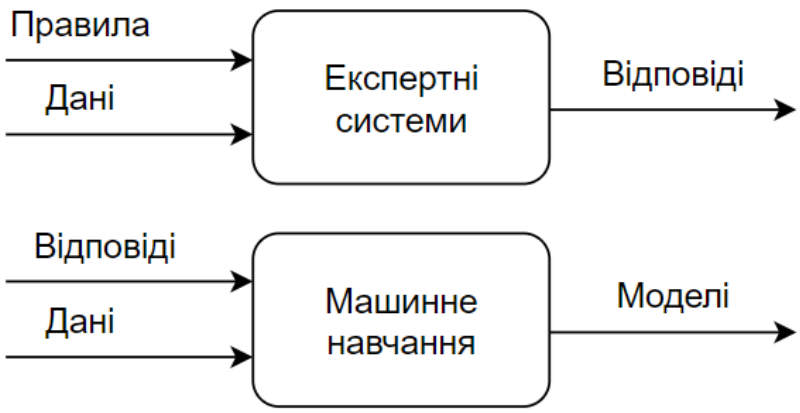


Рисунок 2.2 – Відмінності машинного навчання від традиційних експертних систем

Більшість моделей машинного навчання поділяються на дві великі категорії: моделі навчання з учителем (контрольованого) та без вчителя (неконтрольованого).

Мета моделей навчання з учителем – робити передбачення. Ви навчаєте їх за допомогою помічених даних, щоб вони могли приймати наступні вхідні дані та передбачати, які мітки будуть отримані. Більшість моделей ML, що використовуються сьогодні, є моделями контрольованого навчання.

Моделі навчання без вчителя, навпаки, не вимагають маркованих даних. Їхня мета – дати уявлення про існуючі дані або згрупувати дані за категоріями та відповідним чином класифікувати майбутні дані. Класичним прикладом неконтрольованого навчання є аналіз записів про придбані в компанії товари та покупці, які їх придбали, для визначення того, які клієнти можуть бути найбільш зацікавлені в новому продукті, що запускається, і подальшої побудови спрямованої на цих клієнтів маркетингової кампанії.

Спам-фільтр – це модель контрольованого навчання. Для її роботи потрібні позначені дані. Модель, яка сегментує клієнтів на основі доходів, кредитних балів та історії покупок, є неконтрольованою моделлю, і дані, які вона споживає, не обов'язково мають бути позначені.

Ще однією особливістю машинного навчання вважатимуться те, кожен алгоритм виконує зазвичай одне (чи кілька пов'язаних) завдань. Немає універсальних алгоритмів, які можуть вирішувати широке коло завдань. Це саме властиве людському інтелекту, але не штучному.

Процес підготовки та виконання машинного навчання а допомогою інструментів Microsoft ML показано на рисунку 2.3.

Діаграма, що представлена на рисунку 2.3 представляє структуру коду програми, а також ітераційний процес розробки моделі за допомогою Microsoft ML:

– на першому етапі виконується збирання та завантаження навчальних даних в об’єкт IDataView;

– далі вказується конвеєр операцій для вилучення функцій і застосування алгоритму машинного навчання;

– далі виконується навчання моделі;

– наступним кроком виконується оцінка моделі і повтор її для покращення;

– далі модель зберігається у двійковому форматі для використання в програмі;

– на передостанньому кроці виконується завантаження моделі назад в об’єкт ITransformer;

– наостанок робиться передбачення, викликаючи CreatePredictionEngine.Predict().

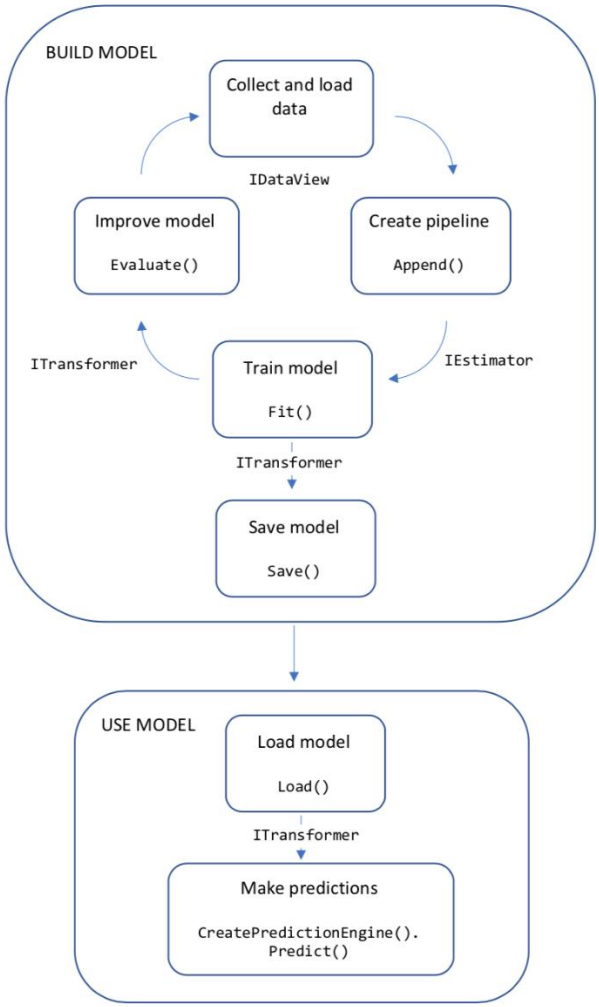


Рисунок 2.3 – Процес підготовки та виконання машинного навчання за допомогою інструментів Microsoft ML

Таким чином, ітераційний процес розробки моделі за допомогою Microsoft Machine Learning включає кілька етапів, які спрямовані на створення та вдосконалення моделі машинного навчання, що забезпечує високі результати в прогнозуванні або класифікації. Використання таких

інструментів як Azure Machine Learning або ML.NET дозволяє ефективно проводити ці етапи.

Перше, що потрібно зробити – чітко визначити завдання, яке буде вирішуватися за допомогою моделі. Це може бути класифікація, регресія, кластеризація або інша задача машинного навчання. На цьому етапі важливо визначити тип проблеми, метрику ефективності моделі та цільову змінну (label).

Підготовка даних є одним із найважливіших етапів у створенні моделі:

– збирання відповідних даних з наявних джерел, це можуть бути дані з баз даних, файлів або навіть API;

– видалення або заповнення пропущених значень, обробка дублікатів і помилкових записів;

– перетворення категоріальних змінних у числові за допомогою методів кодування (One-hot encoding, Label encoding), масштабування даних або нормалізація, а також створення нових ознак (feature engineering).

У Microsoft ML, зокрема у Azure Machine Learning, є вбудовані інструменти для підготовки даних, а ML.NET надає можливості для їх очищення і трансформації безпосередньо у вашому коді.

Виконання розділення даних на тренувальні та тестові набори відбувається поділенням їх на кілька наборів:

– тренувальний набір – використовується для навчання моделі;

– тестовий набір – використовується для оцінки якості моделі.

Можна також виділити валідаційний набір для додаткового налаштування моделі (cross-validation).

Microsoft ML надає зручні інструменти для такого поділу, наприклад, методи TrainTestSplit або функції крос-валідації в ML.NET.

На етапі вибору моделі обирається початкова модель, яка буде використовуватися для вирішення завдання. Microsoft ML підтримує різні типи моделей, такі як лінійні моделі, дерева рішень, випадкові ліси, градієнтний бустинг, нейронні мережі та інші. Вибір моделі залежить від

специфіки проблеми та даних. У Microsoft ML.NET або Azure Machine Learning є доступ до різних моделей через прості API або графічні інтерфейси.

Тренування моделі – це процес, під час якого обраний алгоритм на основі тренувальних даних навчається прогнозувати або класифікувати нові значення. Під час тренування модель намагається мінімізувати похибки на тренувальному наборі даних. У Microsoft ML.NET та Azure ML є готові інструменти для налаштування тренування, вибору гіперпараметрів та оптимізації моделей.

Після тренування необхідно оцінити ефективність моделі на тестових даних за допомогою відповідних метрик, таких як:

– середньоквадратична помилка (MSE), середня абсолютна похибка (MAE), коефіцієнт детермінації (R^2);

– точність (Accuracy), повнота (Recall), точність (Precision), F1-міра, площа під кривою (AUC);

У ML.NET доступні функції для обчислення цих метрик через відповідні класи (наприклад, BinaryClassificationMetrics або RegressionMetrics).

На основі результатів оцінки модель можна вдосконалити. Це можна зробити кількома способами:

– використання таких методів, як сітковий пошук (grid search) або випадковий пошук (random search) для пошуку оптимальних параметрів;

– поліпшення якості моделі через створення нових ознак або вибір найважливіших з існуючих;

– виконання зміни моделі, якщо початкова модель не дала хороших результатів, можна вибрати інший алгоритм, наприклад, перейти від лінійної регресії до градієнтного бустингу або нейронних мереж.

Після поліпшення моделі її знову оцінюють на тестових даних або за допомогою валідаційного набору. Мета цього етапу – впевнитись, що модель добре працює на невідомих їй даних і не є перенавченою.

Коли модель навчена та перевірена, її можна розгортати у виробничому середовищі. В Azure Machine Learning можна автоматизувати процес

деплойменту через вебсервіси або контейнери. ML.NET дозволяє інтегрувати модель безпосередньо у .NET-додатки.

Після розгортання моделі важливо відстежувати її роботу та ефективність. Дані можуть змінюватися з часом, і модель може втратити свою точність (явище «старіння моделі»). У такому випадку потрібно буде оновлювати модель новими даними або адаптувати її до нових умов.

2.2 Аналіз методів машинного навчання

2.2.1 Метод опорних векторів

Метод опорних векторів (англ. Support Vector Machine, SVM) – це потужний алгоритм машинного навчання, який використовується для задач класифікації та регресії [10]. SVM базується на побудові гіперплощини або множини гіперплощин у багатовимірному просторі, щоб розділити точки різних класів якомога далі одна від одної, забезпечуючи максимальний відступ (margin) між ними (рис. 2.4). Це один із найпоширеніших методів класифікації у випадках, коли дані важко лінійно розділити.

Метод опорних векторів прагне знайти таку гіперплощину, яка максимально розділяє два класи даних у просторі ознак. Якщо дані можуть бути лінійно розділені, SVM будує таку гіперплощину, що відстань до найближчих точок обох класів (опорних векторів) є максимальною. Ці найближчі точки і називаються опорними векторами.

Якщо дані не можуть бути лінійно розділені в початковому просторі, SVM використовує так звані ядрові функції (kernel functions), щоб перевести дані в простір вищої вимірності, де вони стають лінійно роздільними.

Розглянеом ключові поняття в SVM.

Гіперплощина – це простір, що розділяє дві різні класи. У двовимірному просторі це лінія, в тривимірному – площина, а у вищих вимірах – це багатовимірна гіперплощина. Метою SVM є побудова гіперплощини, що розділяє два класи з максимальним зазором (margin).

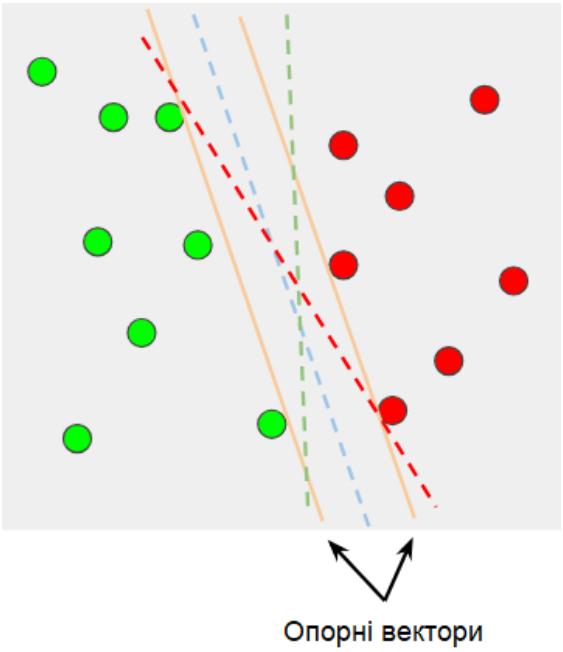


Рисунок 2.4 – Метод опорних векторів

На рисунку 2.5 наведено приклад, у якому беруть участь об'єкти двох типів. Лінія, що розділяє, задає межу, праворуч від якої – всі об'єкти типу brown (коричневий), а зліва – типу yellow (жовтий). Новий об'єкт, що потрапляє направо, класифікується як об'єкт класу brown або – як об'єкт класу yellow, якщо він розташувався ліворуч від прямої. І тут кожен об'єкт характеризується двома вимірами.

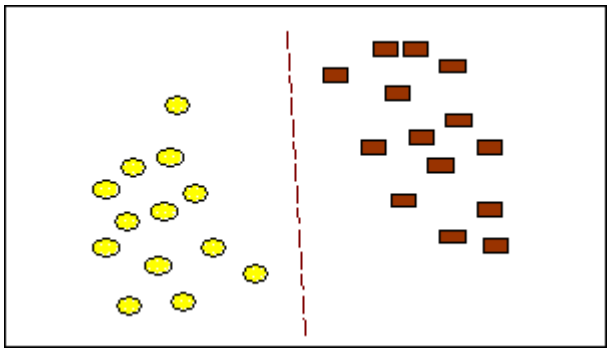


Рисунок 2.5 – Приклад розділення об’єктів двох типів

Відступ – це відстань між гіперплощиною та найближчими точками з кожного класу. Алгоритм SVM намагається максимізувати цей відступ, щоб

забезпечити стійкість моделі до нових, ще невідомих даних (рис. 2.6).

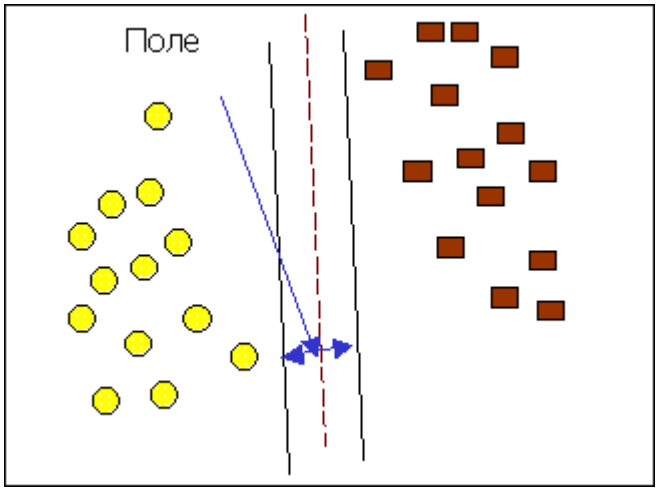


Рисунок 2.6 – Роздільна площина

Опорні вектори – це точки, які знаходяться на межі між двома класами і найближчі до гіперплощини (рис. 2.7). Вони визначають положення гіперплощини. Точки, які не є опорними векторами, не впливають на побудову цієї гіперплощини.

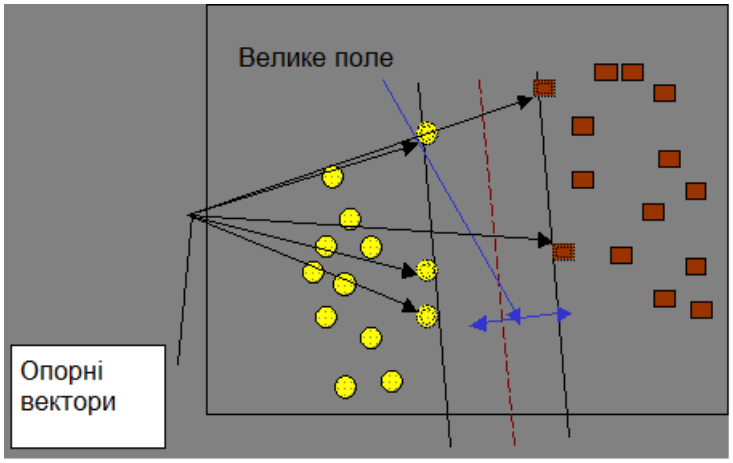


Рисунок 2.7 – Пошук зразків, що є на межі між двома класами

Метод опорних векторів відноситься до так званих методів навчання з учителем. Спочатку потрібно подати якусь навчальну вибірку, щоб навчити модель розрізняти класи (рис. 2.8).

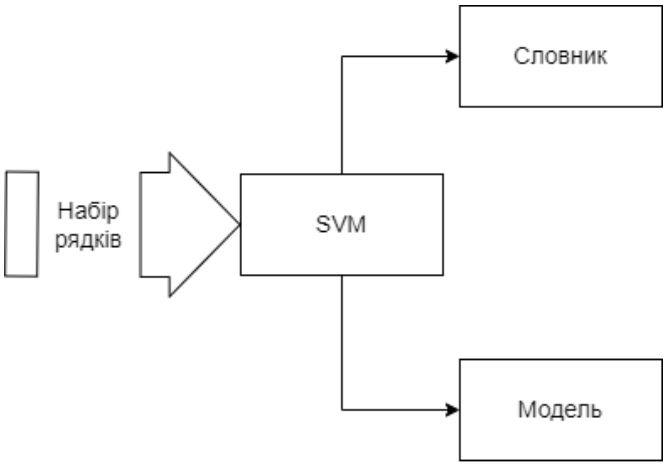


Рисунок 2.8 – Процес навчання

В результаті виконання навчання отримуємо готову до розпізнавання модель і словник (рис. 2.9).

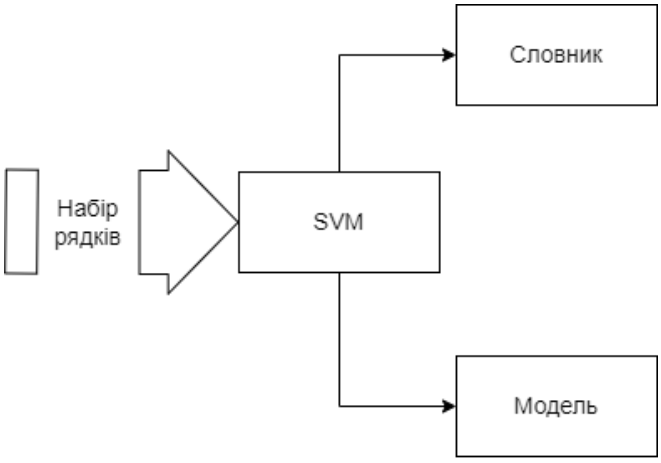


Рисунок 2.9 – Процес класифікації

Для класифікації даних використовується модель і словник, отримані на етапі навчання.

Ядро – у випадку, коли дані не можуть бути лінійно розділені в початковому просторі, SVM використовує ядрові методи для перетворення даних в інший простір більшої вимірності, де вони можуть стати лінійно роздільними. Найбільш поширеними ядровими функціями є:

– лінійне ядро (Linear kernel);

– поліноміальне ядро (Polynomial kernel);

– радіально-базисне ядро (Radial Basis Function, RBF);

– сигмоїдне ядро (Sigmoid kernel).

Можна виділити два етапи роботи SVM:

– навчання;

– класифікація.

На етапі навчання SVM бере набір навчальних даних і намагається знайти гіперплощину, яка найкраще розділяє класи, максимізуючи відступ. Якщо дані нелінійні, використовується ядро для перетворення в інший простір.

На етапі класифікації, після навчання нові дані класифікуються залежно від того, на якій стороні гіперплощини вони знаходяться. Якщо точка знаходиться з одного боку гіперплощини, вона належить до одного класу, з іншого до другого класу.

Параметрами налаштування є:

– C (параметр регуляризації);

– Gamma (γ).

C (параметр регуляризації) – це параметр, що контролює чутливість до неправильних класифікацій. Велике значення C намагається мінімізувати кількість помилок, але може призвести до перенавчання. Маленьке значення C дозволяє допускати помилки, але збільшує відступ.

Gamma (γ) – це параметр для RBF-ядра, що визначає, як далеко розповсюджується вплив однієї навчальної точки. Велике значення gamma може призвести до перенавчання.

Переваги методу опорних векторів:

– висока ефективність у випадку невеликих або середніх за розміром наборів даних;

– ефективність у випадках, коли кількість вимірів (ознак) значно більша за кількість спостережень;

– підтримка як лінійних, так і нелінійних розділень за допомогою ядрових функцій;

– стійкість до перенавчання, особливо при високих значеннях відступу.

Недоліки методу опорних векторів:

– SVM може бути неефективним для великих наборів даних, оскільки алгоритм повільний при великій кількості спостережень;

– труднощі у виборі правильної ядрової функції для складних задач;

– може бути чутливим до шуму в даних, особливо коли дані не роздільні (сильно перекриваються).

2.2.2 Лінійна регресія (Linear Regression)

Лінійна регресія – це один з найпростіших і найпоширеніших методів машинного навчання, який використовується для передбачення числових значень [10]. Основна ідея лінійної регресії полягає в знаходженні математичної залежності між незалежними змінними (факторами, ознаками) та залежною змінною (цільовою величиною) за допомогою побудови лінії, яка найкраще описує ці взаємозв'язки. Приклад лінійної регресії показано на рисунку 2.10.

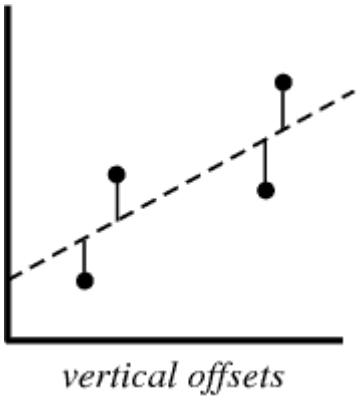


Рисунок 2.10 – Приклад лінійної регресії

Лінійна регресія шукає таку лінійну функцію, яка найкраще відображає залежність між незалежними змінними та цільовою величиною. Ця лінійна функція має вигляд:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n, \qquad (2.1)$$

де y – передбачене значення залежної змінної (цільова змінна);

b_0 – вільний член (перетин із віссю y);

$b_1, b_2, \cdots, b_n$ – коефіцієнти регресії (вагові коефіцієнти для кожної ознаки);

$x_1, x_2, \cdots, x_n$– незалежні змінні (фактори, ознаки).

Метою лінійної регресії є вибір таких коефіцієнтів $b_0, b_1, \cdots, b_n$, які мінімізують різницю між фактичними значеннями залежної змінної та передбаченими значеннями моделі. Ця різниця зазвичай вимірюється за допомогою суми квадратів залишків (RSS – Residual Sum of Squares):

$$RSS = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \tag{2.2}$$

де y_i – фактичне значення залежної змінної;

$\hat{y}_i$ – передбачене значення залежної змінної.

Вагові коефіцієнти $b_1, b_2, \cdots, b_n$ показують, як сильно кожна незалежна змінна впливає на залежну змінну. Якщо коефіцієнт позитивний, це означає, що зі збільшенням відповідної незалежної змінної залежна змінна також зростає, і навпаки.

R-квадрат (R^2) – це метрика, яка показує, наскільки добре модель лінійної регресії пояснює варіацію залежної змінної. R^2 змінюється від 0 до 1, де 1 означає, що модель повністю пояснює варіацію цільової змінної, а 0 означає, що модель не дає корисної інформації.

Одним з припущень лінійної регресії є те, що залишки (різниці між фактичними та передбаченими значеннями) мають нормальний розподіл. Це допомагає забезпечити правильність роботи моделі та точність її прогнозів.

Іншим припущенням є лінійність взаємозв'язку між незалежними змінними та цільовою змінною. Лінійна регресія підходить, коли залежність між змінними є лінійною.

Важливо, щоб залишки були незалежними одне від одного. Це означає,

що значення одного залишку не повинно залежати від значення іншого залишку.

2.2.3 Логістична регресія (Logistic Regression)

Логістична регресія – це статистичний метод, який використовується для розв'язання задач класифікації, де цільова змінна є категоріальною, тобто приймає обмежену кількість значень (найчастіше два значення – для бінарної класифікації) [9, 10]. На відміну від лінійної регресії, яка прогнозує безперервні значення, логістична регресія моделює ймовірність того, що зразок належить до певного класу.

Логістична регресія базується на тому, що лінійна модель не підходить для класифікації, оскільки може передбачати значення поза межами діапазону [0, 1], що неприйнятно для ймовірностей. Тому для перетворення результатів лінійної моделі в ймовірності використовується логістична функція або сигмоїда:

$$p = \frac{1}{1+e^{-z}}, \tag{2.3}$$

де p – це передбачена ймовірність належності до класу 1;

$z = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n$ – лінійна комбінація незалежних змінних;

b_0 – вільний член (зміщення);

$b_1, b_2, \cdots, b_n$– коефіцієнти регресії.

Значення функції логістичної регресії завжди перебуває в діапазоні від 0 до 1, що дає можливість інтерпретувати результат як ймовірність належності до певного класу.

Приклад на рисунку 2.11 показує порівняння лінійної та логістичної регресійних моделей на однакових наборах даних.

Причина з якої логістична регресія має вигин – це той факт, що для її обчислення не використовуючи лінійне рівняння. Натомість логістична

регерсійна модель будується на використанні сигмоїди (також званою логістичною функцією, тому що вона використовується в логістичній регресії).

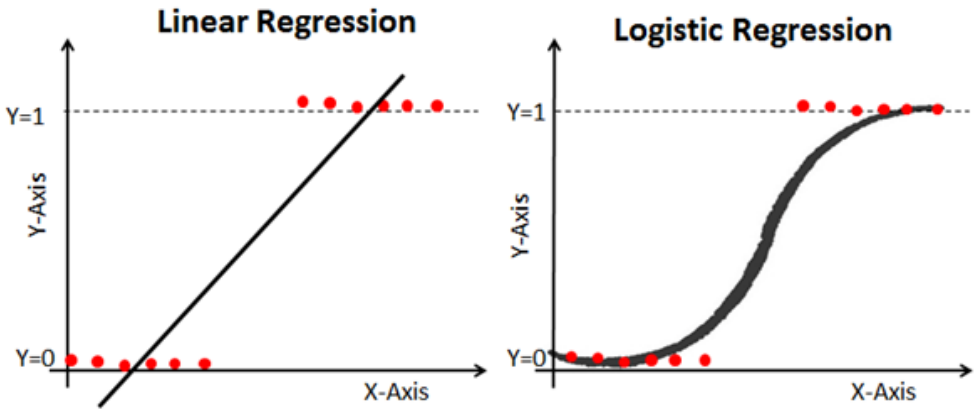


Рисунок 2.11 – Різниця між лінійною та логістичною регресійною моделлю

Логістична регресія передбачає лінійну залежність між незалежними змінними та логарифмом відношення ймовірностей класів (log-odds), що може бути недостатнім для складних, нелінійних даних.

Хоча логістичну регресію можна розширити для багатокласової класифікації (через методи «один проти всіх» або «softmax»), вона не є найефективнішим методом для таких задач.

Перевагами логістичної регресії є:

– коефіцієнти моделі можуть інтерпретуватися як вплив кожної незалежної змінної на ймовірність належності до певного класу;

– метод легко реалізується та є ефективним для задач бінарної класифікації;

– логістична регресія є відносно швидким методом навіть для великих наборів даних.

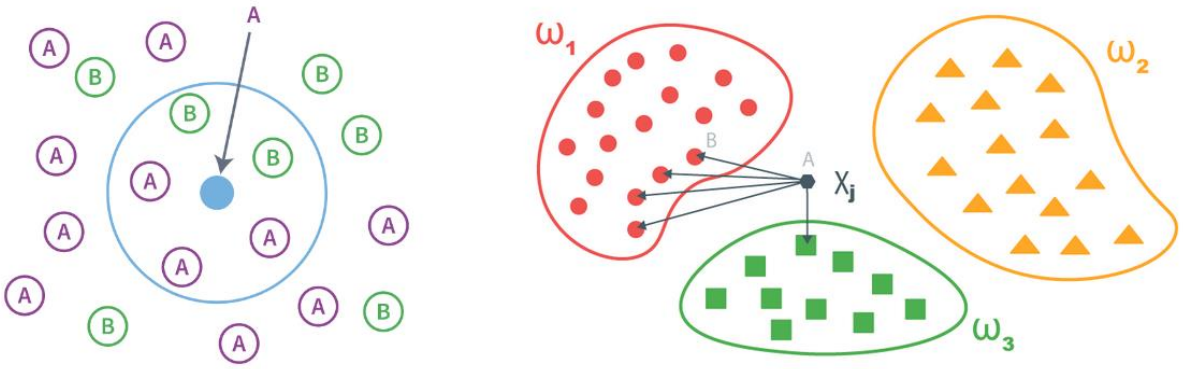
2.2.4 Алгоритм k-найближчих сусідів (K-Nearest Neighbors)

Алгоритм k-найближчих сусідів (K-Nearest Neighbors, KNN) – це один із найпростіших і водночас ефективних методів машинного навчання, що

використовується для розв'язання задач класифікації та регресії [10]. KNN базується на принципі порівняння нових даних із вже наявними навчальними зразками та прийняття рішення на основі їх схожості.

Алгоритм KNN класифікує об'єкт, знаходячи k найближчих сусідів цього об'єкта в багатовимірному просторі ознак і визначаючи клас нового об'єкта за більшістю класів серед його сусідів. У випадку регресії результатом буде середнє значення відповідей найближчих сусідів.

Вважається, що досліджуваний об'єкт належить до того класу, сумарна вага якого має мінімальне значення (рис. 2.12, а).



а) б)

Рисунок 2.12 – Класифікація об’єктів за допомогою KNN алгоритму

Параметр *k* визначає кількість найближчих сусідів, які будуть враховуватись для прийняття рішення. Це важливий параметр, оскільки від нього залежить точність і стабільність роботи алгоритму.

Математична основа алгоритму базується на гіпотезі компактності, згідно з якою, якщо міру подібності між об'єктами визначено правильно, то подібні об'єкти частіше належать до одного класу, ніж до різних. Водночас межі між класами є простими, а самі класи утворюють компактні області, локалізовані в просторі об'єктів (рис. 2.12, б).

Опишемо математичний апарат даного алгоритму. Вихідними даними

алгоритму є:

– навчальна вибірка X^m, що складається з об'єктів $\{x_1, x_2, \dots x_m\}$ з однаковим набором атрибутів. Для всіх об'єктів повинні бути відомі значення атрибутів та приналежність їх до класів $\{y_1, y_2, \dots y_m\}$. Атрибути можуть бути числовими або категоріальними. Таким чином, навчальна вибірка задана так:

$$X^m = \{(x_1, y_1); (x_2, y_2); \dots (x_m, y_m)\}; \tag{2.4}$$

– метрика $\rho(x, x')$ розрахунку відстаней між об'єктами;

– об'єкт u, для якого необхідно визначити приналежність до одного із класів;

– розраховані відстані сортуються в порядку збільшення значення:

$$\rho(u, x_{1;u}) < \rho(u, x_{2;u}) < \dots < \rho(u, x_{m;u}), \tag{2.5}$$

де $x_{1;u}$ – об'єкт навчальної вибірки, що є і-им сусідом по відношенню до дослідного об'єкту.

У результаті сортування породжується нова нумерація об'єктів навчальної вибірки. Тоді результат роботи алгоритму k-найближчих сусідів по відношенню до дослідного об'єкту u можна представити так:

$$a(u) = \arg\max \sum_{i=1}^{m} [y(x_{i;u}) = y] w(i, u), \tag{2.6}$$

де $y(x_{i;u})$ – мітка класу i-ого сусіда дослідного об'єкта;

$w(i, u)$ – вагова функція, для оцінки ступеня впливу i-го об'єкта на результату класифікації об'єкта u.

Від способу завдання вагової функції залежить реалізація алгоритму kNN:

– $w(i, u) = [i = 1]$ – метод найближчого сусіда;

– $w(i, u) = [i \le k]$ – метод k-найближчих сусідів;

– $w(i, u) = [i \le k] w_i$ – метод k зважених найближчих сусідів.

Застосування алгоритму k-найближчих сусідів у класифікаційних задачах супроводжується низкою викликів. По-перше, важливо правильно підібрати параметр k. Наприклад, при k = 1 класифікація стає дуже чутливою до шуму в навчальних даних, тоді як при $k = m$ алгоритм втрачає адаптивність і часто повертає один і той самий результат (клас). Оптимальне значення k у кожному конкретному випадку можна визначити лише експериментально.

По-друге, виникає потреба у фільтрації навчальної вибірки від зайвих даних. Це пов'язано з тим, що інформативний об'єкт може бути оточений надмірно схожими об'єктами того ж класу. Така надлишковість збільшує час обчислень під час класифікації та негативно впливає на точність роботи алгоритму k-найближчих сусідів.

По-третє, важливим є правильний вибір метрики для визначення схожості між об'єктами. Використання різних метрик може призводити до істотних відмінностей у результатах класифікації.

В якості метрики можуть використовуватися, наприклад:

– евклідова відстань:

$$\rho(X,Y) = \sqrt{\sum_i (x_i - y_i)^2}\,; \tag{2.7}$$

– зважена евклідова відстань:

$$\rho(X,Y) = \sqrt{\sum_i w_i (x_i - y_i)^2}\,; \tag{2.8}$$

– Манхеттенська відстань:

$$\rho(X,Y) = \sum_i |x_i - y_i|\,; \tag{2.9}$$

– відстань Махаланобіса:

$$\rho(X,Y) = (x_i - y_i)^T K^{-1}(x_i - y_i), \quad (2.9)$$

де K – коваріаційна матриця.

Визначити яка з метрик дозволить забезпечити максимальну точність можливо тільки експериментальним шляхом.

Переваги алгоритму KNN:

– алгоритм легко зрозуміти та реалізувати, особливо для невеликих наборів даних;

– можна використовувати для задач як класифікації, так і регресії;

– KNN здатний виявляти нелінійні кордони між класами, на відміну від лінійних моделей.

Недоліки KNN:

– алгоритм потребує обчислення відстаней до кожного зразка, що робить його неефективним для великих наборів даних, оскільки пошук найближчих сусідів має часову складність $O(n)$, де n – кількість зразків;

– якщо в навчальній вибірці є викиди, вони можуть суттєво вплинути на результат, особливо при малому значенні;

– алгоритм є «повільним», тобто не будує явно моделі під час навчання, а зберігає всі навчальні зразки, що може займати багато пам'яті.

2.3 Висновки по другому розділу

В результаті виконання другого розділу кваліфікаційної роботи виконано аналіз методів машинного навчання. Дано визначення терміну «Машинне навчання», показано взаємозв'язок між машинним навчанням, глибоким навчанням та штучним інтелектом. Проаналізовані відмінності машинного навчання від традиційних експертних систем.

Проаналізовано процес підготовки та виконання завдання машинного навчання на прикладі інструментів Microsoft ML.

Проведено огляд таких методів, як метод опорних векторів, лінійної регресії, логістичної регресії та алгоритму k-найближчих сусідів. Показані їх позитивні боки та недоліки.

3 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ РОБОТИЗОВАНИМ МАНІПУЛЯТОРОМ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Розроблення архітектури автоматизованої системи

Архітектура автоматизованої системи показана на рисунку 3.1.

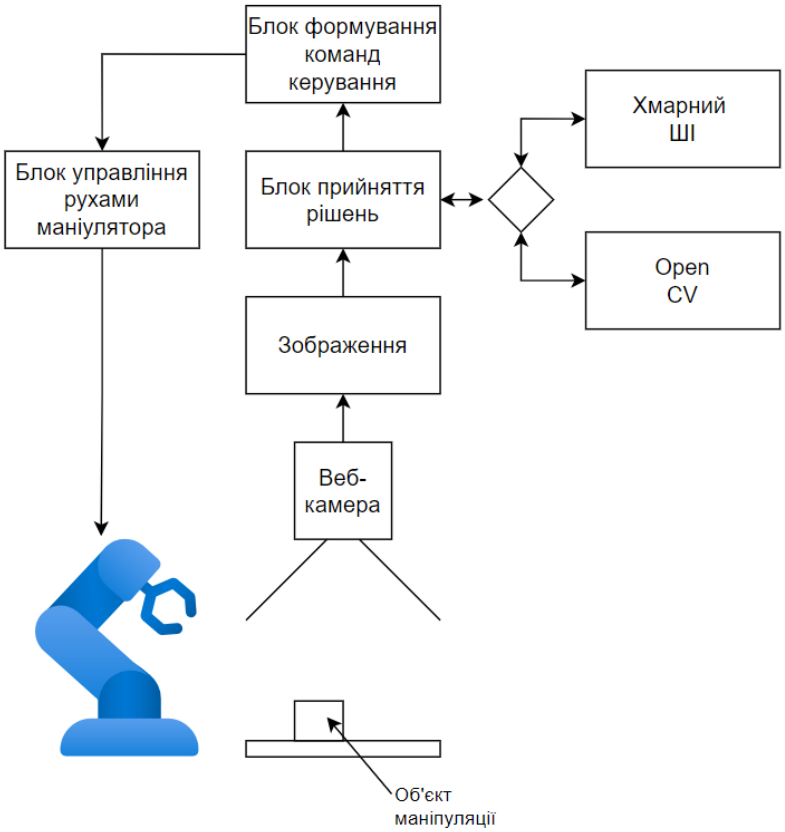


Рисунок 3.1 – Архітектура автоматизованої системи

Автоматизована система управління роботизованим маніпулятором складається з багатьох компонентів. Вона забезпечує обробку зображень і прийняття рішень завдяки поєднанню локальних обчислень на основі бібліотеки OpenCV та хмарного середовища, що працює з використанням

технологій розпізнавання зображень зі штучним інтелектом. Кожен блок системи забезпечує роботу роботизованого маніпулятора, починаючи з отримання зображення і закінчуючи виконанням фізичних дій пристрою (маніпулятора). До складу автоматизованої системи входять:

– блок отримання зображення на основі веб-камери;

– блок обробки зображення;

– блок прийняття рішень;

– блок формування команд керування роботизованим маніпулятором;

– блок аналізу зображення на основі OpenCV;

– блок аналізу зображення на основі хмарного сервісу зі штучним інтелектом;

– блок управління рухами роботизовано го маніпулятора;

– роботизований маніпулятор, або його цифровий двійник.

Блок отримання зображення на основі веб-камери відповідає за збирання інформації про навколишнє середовище. Основним елементом для отримання зображення є веб-камера, яка захоплює зображення простору, де знаходиться об'єкт маніпуляції роботизованого маніпулятора. Отримане зображення передається до наступного блоку для обробки.

Блок обробки зображення проводить початкову обробку зображень для виділення необхідної інформації. Основним завданням є перетворення необроблених даних із камери в структури, які можуть бути проаналізовані та використані іншими блоками системи. Процеси обробки можуть включати фільтрацію шумів, сегментацію об'єктів, виявлення контурів та інші методи попередньої обробки зображень.

Центральним елементом системи є блок прийняття рішень, який координує роботу всіх інших блоків. Після отримання попередньо обробленого зображення, блок прийняття рішень визначає подальші дії системи. Цей блок управляє інформаційними потоками на основі попередньої обробки зображення засобами локального блоку OpenCV. Якщо в результаті попередньої обробки зображення не було знайдено координат об’єкту

маніпуляції то зображення передається на блок аналізу зображення на основі хмарного сервісу зі штучним інтелектом.

OpenCV (Open Source Computer Vision Library) – бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом [11]. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Бібліотека OpenCV в даній архітектурі використовується для базових операцій з обробки зображень. У цьому блоці зображення обробляються за допомогою алгоритмів комп'ютерного зору, таких як розпізнавання об'єктів, виявлення країв та інших. Якщо аналіз на цьому етапі дозволяє ідентифікувати об'єкт маніпуляції і його координати, то блок прийняття рішень передає їх на наступний етап – блок формування команд керування.

Якщо локальні засоби аналізу зображення не змогли визначити координати об'єкта, то зображення передається на аналіз до хмарного сервісу, який використовує штучний інтелект [13]. Це може бути ресурс chatGPT, або Copilot. Хмарні платформи, що використовують глибоке навчання та нейронні мережі, мають вищу обчислювальну потужність і здатні краще справлятися з більш складними задачами обробки зображень [14]. Наприклад, вони можуть розпізнавати об'єкти, що перекриваються, або працювати з великим обсягом даних.

Після того як були отримані координати об'єкта, цей блок генерує відповідні команди для керування роботизованим маніпулятором. Він перетворює координати та параметри об'єкта в конкретні дії: переміщення маніпулятора, захоплення об'єкта тощо.

Блок управління рухами роботизованого маніпулятора відповідає за безпосереднє управління рухами маніпулятора. Він приймає команди з блоку формування команд і перетворює їх в механічні рухи маніпулятора, керуючи

приводами та механізмами. Це може бути реальний фізичний маніпулятор або його цифровий двійник, який моделюється для тестування та налагодження системи.

Роботизований маніпулятор виконує основну функцію – маніпулювання об'єктами в фізичному світі. Якщо система працює з цифровим двійником, модель маніпулятора у віртуальному середовищі допомагає проводити симуляції, прогнозувати результати і налагоджувати систему без потреби використовувати реальне обладнання.

3.2 Розроблення алгоритму роботи автоматизованої системи

На рисунку 3.2 показано алгоритм роботи основної частини автоматизованої системи управління роботизованим маніпулятором.

На початку роботи програми виконується налаштування інтерфейсу захвату відеозображення. Дана функція виконується на основі бібліотеки OpenCV.

Захоплення відеозображення за допомогою бібліотеки OpenCV є одним із основних процесів обробки відеоданих, що застосовується для створення систем комп'ютерного зору, аналізу зображень, відеоспостереження та інших задач. Принцип захоплення відео базується на використанні спеціального класу OpenCV під назвою VideoCapture, який дозволяє отримувати відео з різних джерел: веб-камер, IP-камер або збережених відеофайлів. Процес починається з ініціалізації об'єкта VideoCapture, в який передається ідентифікатор пристрою або шлях до відеофайлу. Якщо використовується веб-камера, ідентифікатор зазвичай представляє собою номер камери (наприклад, 0 для першої камери).

Після ініціалізації OpenCV відкриває з'єднання з джерелом відео, і система починає захоплювати послідовні кадри відеопотоку. Кожен кадр можна отримати за допомогою методу read(), який зчитує один відеокадр і зберігає його у вигляді зображення у форматі Mat (матриця зображення

OpenCV). Ця матриця може бути оброблена різними інструментами бібліотеки: виконуються операції з перетворення, фільтрації, розпізнавання об'єктів, а також аналізу руху та інших завдань. Важливим аспектом є правильне управління частотою кадрів, оскільки програма повинна бути здатна обробляти відеопотік у реальному часі.

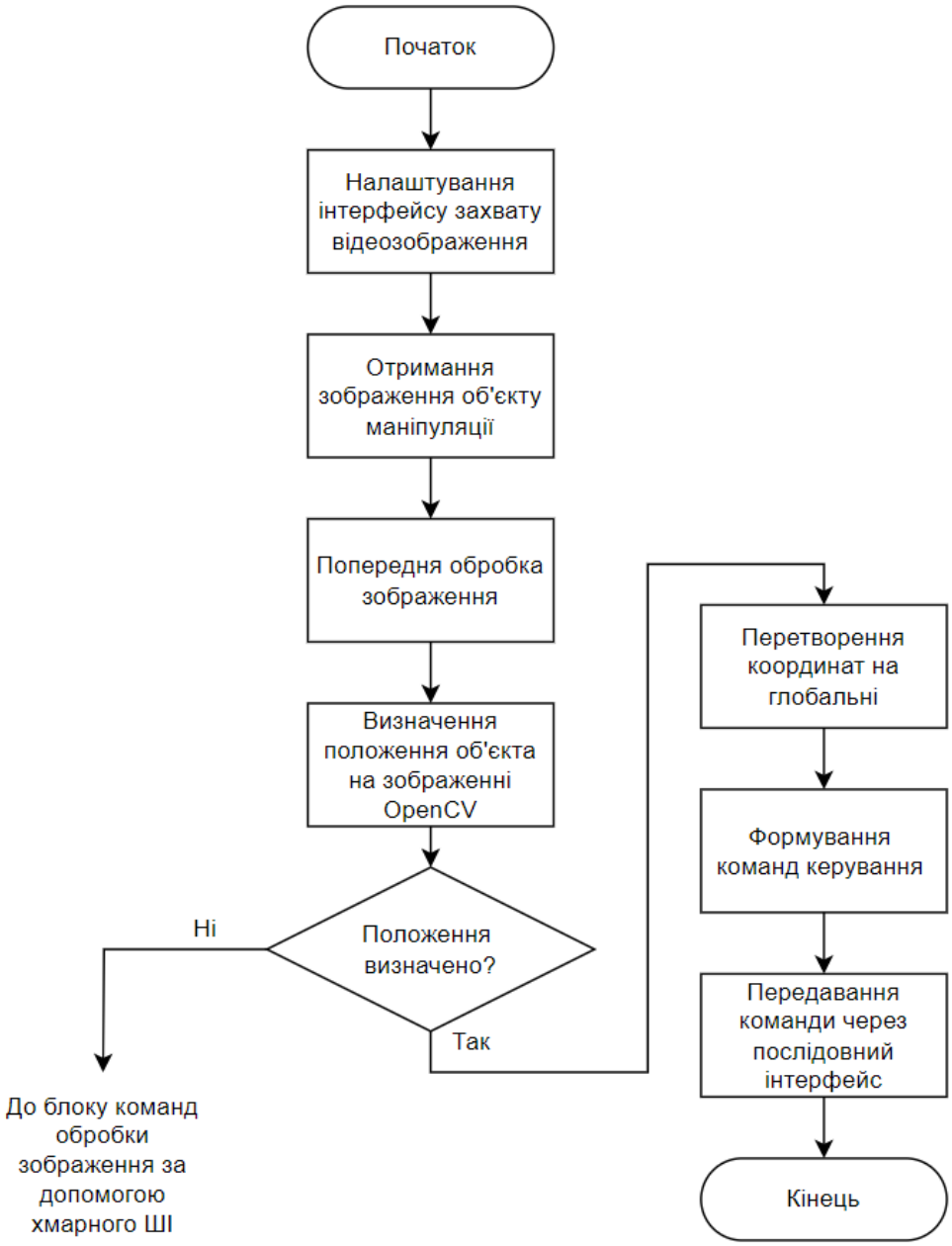


Рисунок 3.2 – Алгоритм роботи основної частини автоматизованої системи управління роботизованим маніпулятором

Захоплення відео в OpenCV також передбачає обробку помилок, наприклад, перевірку, чи успішно відкрито джерело відео, чи доступний кадр, що зчитується. Після завершення роботи з відеопотоком необхідно закрити з'єднання з відеопристроєм, використовуючи метод release(), що звільняє ресурси і дозволяє коректно завершити роботу з камерою або файлом.

Після ініціалізації відеокамери відбувається отримання зображення та його попередня обробка. Процес попередньої обробки зображення для подальшого пошуку круглих об'єктів та визначення їх координат складається з кількох етапів, кожен з яких дозволяє підготувати зображення до розпізнавання. Для цього використовуються інструменти комп'ютерного зору, зокрема бібліотека OpenCV. Цей процес зазвичай включає декілька етапів.

На першому етапі виконується конвертація в відтінки сірого. Оскільки кольори часто не є важливими для виявлення форм об’єктів, зображення конвертується в відтінки сірого для спрощення обчислень і зменшення впливу кольорових шумів. Це робиться за допомогою функції CvInvoke.CvtColor, яка перетворює кольорове зображення (BGR) у монохромне (градації сірого).

Далі, щоб уникнути впливу дрібних шумів і артефактів на результати пошуку, зображення піддається розмиттю. Для цього зазвичай використовується гаусове розмиття (GaussianBlur), яке згладжує різкі переходи між пікселями і допомагає зменшити шумові елементи на зображенні.

Один із важливих етапів – виявлення країв об'єктів. Це дозволяє підкреслити контури круглих об'єктів на зображенні. Для цього можна використовувати оператор Кенні (Canny Edge Detector), який виділяє області різких змін яскравості і дозволяє чіткіше побачити межі об'єктів.

Після того, як зображення підготовлене, можна застосувати алгоритм HoughCircles для виявлення кіл. Цей метод аналізує зображення та шукає круглі форми на основі параметрів радіусу та контрасту між об'єктом і фоном. Алгоритм використовує простір Хафа для визначення можливих центрів кіл і їх радіусів.

Налаштування параметрів HoughCircles відбувається наступним чином:

– налаштування dp – параметр, що визначає розмірність зображення, на якій шукаються кола (наприклад, 1 означає оригінальну роздільність);

– налаштування minDist – мінімальна відстань між центрами виявлених кіл;

– налаштування param1 і param2 – параметри для налаштування чутливості алгоритму (значення для порогових фільтрів);

– налаштування minRadius і maxRadius – мінімальний і максимальний радіус кіл, які потрібно знайти.

Після виявлення кіл алгоритм повертає масив даних, в якому для кожного кола зберігаються координати його центру (x, y) та радіус. Ці координати можна використовувати для подальшого управління роботизованим маніпулятором або інших дій.

На фінальному етапі можна нанести виявлені кола на оригінальне зображення для візуалізації результатів або передати координати круглих об'єктів до блоку управління маніпулятором для виконання фізичної маніпуляції.

Якщо в процесі локальної обробки зображення засобами OpenCV не було виявлено об’єкту маніпуляції (це може бути в результаті поганого освітлення, або зашумленого фону), автоматизована система перемикається на використання хмарного середовища, що працю за принципом штучного інтелекту.

На рисунку 3.3 показано алгоритм роботи автоматизованої системи в режимі використання хмарного середовища зі штучним інтелектом.

Після попереднього оброблення зображення перетворюється в стандартний формат, який можна легко передати через API або інтерфейс ChatGPT. Наприклад, зображення може бути збережене у форматах PNG або JPEG для мінімізації розміру файлу без значної втрати якості. У C# це можна зробити за допомогою бібліотеки System.Drawing, яка дозволяє зберігати зображення у потрібний формат.

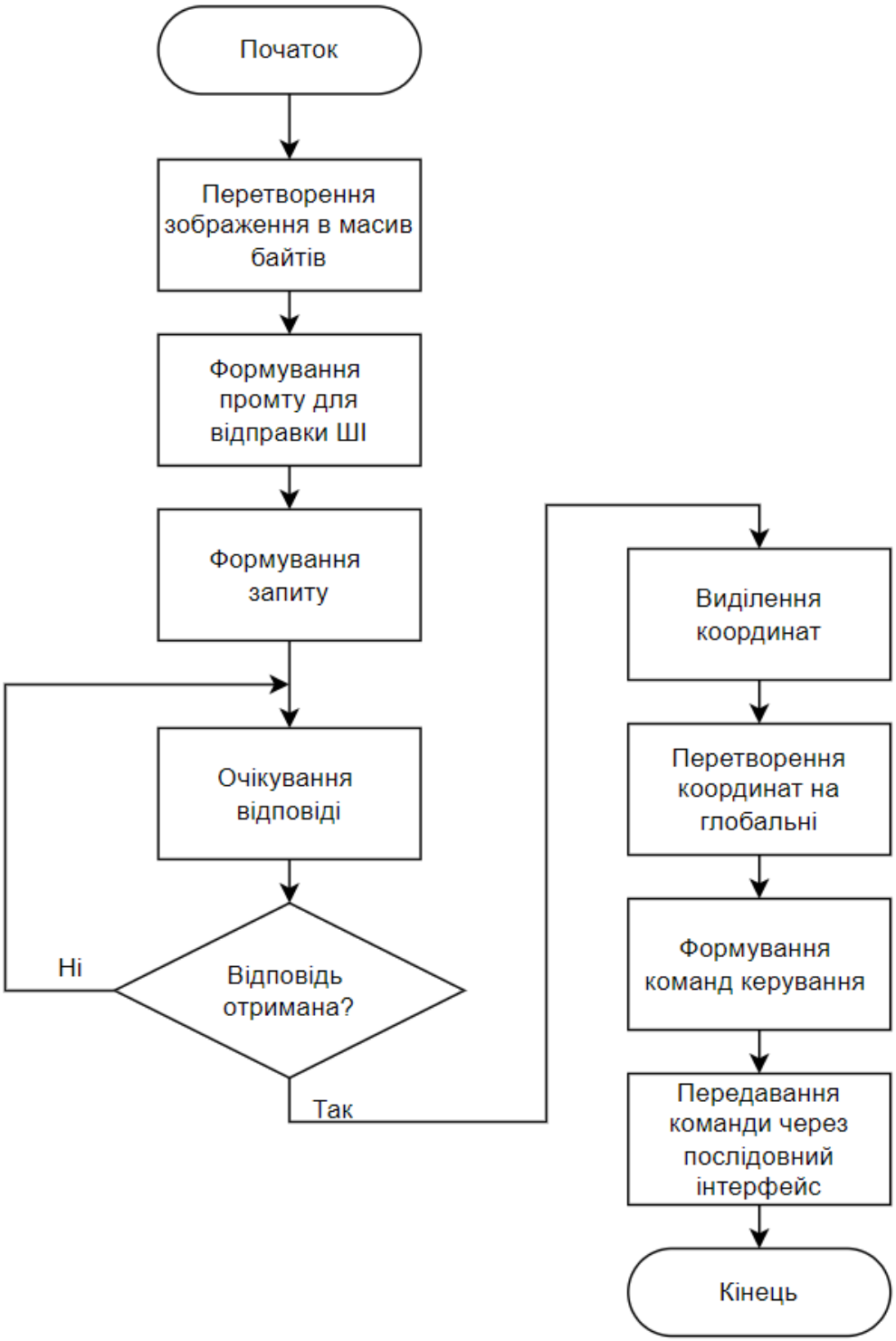


Рисунок 3.3 – Алгоритм роботи автоматизованої системи в режимі використання хмарного середовища зі штучним інтелектом

Після цього важливо переконатися, що розмір і роздільність зображення підходять для аналізу. Великі зображення можуть уповільнювати процес аналізу, тому краще попередньо їх зменшити або оптимізувати. Це можна

зробити за допомогою бібліотеки OpenCV або стандартних засобів C#, таких як методи класу Bitmap.

Далі необхідно перетворити зображення у формат, який можна передати в, наприклад, ChatGPT. Оскільки ChatGPT працює з текстовими запитами, зображення потрібно перетворити в Base64 рядок. Це кодування дозволяє представити будь-які дані, включаючи зображення, у вигляді тексту. У C# для цього можна використовувати метод Convert.ToBase64String(), який перетворює байти зображення в текстовий рядок. Отримане зображення у вигляді Base64-рядка можна включити в текстовий запит, який надсилається в ChatGPT.

Окрім кодування зображення, важливо також надати деталі щодо формату або розміру зображення, щоб модель штучного інтелекту мала достатньо інформації для коректного аналізу. Після того, як ChatGPT отримує закодоване зображення, воно може бути проаналізоване за допомогою алгоритмів, які визначають координати об'єктів маніпуляції.

Насамкінець, після аналізу зображення в ChatGPT, модель може повернути результат у вигляді текстового опису координат розташування об'єкта. Ці координати потім можна використовувати у програмі на C# для управління роботизованим маніпулятором чи іншими подальшими діями.

Після отримання відповіді від хмарного середовища, або від локальної підсистеми аналізу та розпізнавання зображення, необхідно виконати перетворення координат на тлі зображення з відеокамери в реальні координати розташування об’єкта маніпуляції.

Процес перетворення координат полягає у врахуванні масштабування, орієнтації та розташування камери відносно реального простору. Зазвичай, камера забезпечує двовимірне зображення об'єкта в пікселях, проте для управління роботизованим маніпулятором потрібно перетворити ці координати у тривимірну систему, яка враховує реальні відстані між об'єктом, камерою і маніпулятором.

Першим кроком у цьому процесі є калібрування камери. Калібрування

дозволяє визначити параметри, такі як фокусна відстань, розташування лінзи, та положення камери в просторі. За допомогою OpenCV або інших інструментів комп'ютерного зору можна виконати калібрування камери, яке дасть можливість перетворити координати пікселів на зображенні у метричні координати.

Другий етап – це визначення відношення між координатами пікселів на зображенні та реальними координатами об'єкта в просторі. Якщо камера розміщена на фіксованій відстані від робочого простору маніпулятора, використовується проста пропорційна залежність, де координати пікселів можуть бути перетворені у реальні координати через масштабування. Для цього потрібно знати реальні розміри сцени, яку охоплює камера (наприклад, ширину і висоту робочої поверхні). На основі цього розраховується коефіцієнт масштабування, який застосовується для перетворення координат з пікселів у реальні одиниці вимірювання (наприклад, міліметри або сантиметри).

Третій етап – це врахування перспективних спотворень, які виникають через те, що камера може бути розташована під кутом до робочої поверхні. Для цього застосовують методи геометричних перетворень, такі як гомографія, яка дозволяє перетворити перспективні координати зображення у реальні координати на площині. За допомогою OpenCV можна виконати ці перетворення, використовуючи функції для пошуку та застосування матриці гомографії.

Четвертий етап полягає у перетворенні двовимірних координат у тривимірну систему координат, якщо роботизований маніпулятор працює у тривимірному просторі. Для цього потрібно знати висоту об'єкта або застосовувати методи стереозору (якщо використовується більше однієї камери), або сенсори глибини (як, наприклад, камери типу Kinect).

Після перетворення координат на основі цих факторів отримані значення можуть бути передані роботизованому маніпулятору, щоб той міг точно розташуватися у відповідній точці простору для виконання необхідної дії з об'єктом.

3.3 Висновки по третьому розділу кваліфікаційної роботи

В результаті виконання третього розділу кваліфікаційної роботи виконано розроблення архітектури автоматизованої системи роботизованим маніпулятором з використанням штучного інтелекту. Описано склад розробленої системи. Вона забезпечує обробку зображень і прийняття рішень завдяки поєднанню локальних обчислень на основі бібліотеки OpenCV та хмарного середовища, що працює з використанням технологій розпізнавання зображень зі штучним інтелектом.

Виконано розроблення алгоритму роботи автоматизованої системи. Описано алгоритм роботи основної частини автоматизованої системи управління роботизованим маніпулятором, де задачами системи є отримання зображення з веб-камери, попередня обробка та локальне розпізнавання об’єкту маніпуляції. Якщо в процесі локальної обробки зображення засобами OpenCV не було виявлено об’єкту маніпуляції (це може бути в результаті поганого освітлення, або зашумленого фону), автоматизована система перемикається на використання хмарного середовища, що працю за принципом штучного інтелекту.

В даній частині кваліфікаційної роботи також описано алгоритм роботи автоматизованої системи в режимі використання хмарного середовища зі штучним інтелектом.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Розроблення програми для виконання експериментальних досліджень

Програма для виконання експериментальних досліджень написана в Microsoft Visual Studio 2022 з використанням мови програмування C#. Основною особливістю розробки програми для розпізнавання зображення є інтеграція бібліотеки OpenCV або Emgu CV (C#-обгортки OpenCV), яка дозволяє виконувати широкий спектр операцій з обробки та аналізу зображень. Emgu CV підтримує різні формати зображень, методи обробки зображень, такі як фільтрація, сегментація, а також алгоритми машинного навчання, що дозволяють розпізнавати об'єкти на зображенні.

Перше, що потрібно враховувати при розробці програми на C# для розпізнавання зображення, це необхідність додавання відповідних бібліотек, таких як Emgu CV, через систему керування пакетами NuGet. Це дозволяє легко додати залежності та отримати доступ до основних функцій OpenCV через C#.

Для захоплення зображень з веб-камери або файлів використовується клас VideoCapture, який дозволяє отримувати відеопотік або окремі кадри з камери в реальному часі. Після отримання кадру його можна перетворити в об'єкт типу Mat, що дозволяє виконувати подальші операції з обробки зображень.

Після отримання зображення наступним кроком є його обробка для виявлення об'єкта маніпуляції. Один з основних методів – це попередня обробка зображення, яка включає конвертацію в відтінки сірого, зменшення шуму за допомогою фільтрів (наприклад, GaussianBlur), а також застосування алгоритмів для виділення контурів або певних фігур. Для виявлення круглих

об'єктів можна використовувати метод HoughCircles, який дозволяє знаходити круги на зображенні на основі градієнтів і радіусів об'єктів.

Visual Studio 2022 також забезпечує зручне середовище для налагодження програм, включаючи можливості для відстеження стану змінних та покрокового виконання коду. Це особливо важливо при розробці алгоритмів для розпізнавання об'єктів, оскільки процес обробки зображення часто потребує експериментування з параметрами фільтрів, порогів і метрик.

Використовуючи Windows Forms або WPF (Windows Presentation Foundation), можна створювати інтуїтивно зрозумілі графічні інтерфейси для взаємодії з користувачем, де будуть відображатися результати аналізу зображень або дозволяти користувачу завантажувати нові зображення для обробки.

Робота з великими зображеннями або відеопотоками може бути ресурсомісткою, тому необхідно враховувати ефективність використання пам'яті та швидкість обробки. Для цього можна використовувати багатопоточність або асинхронне програмування, яке підтримується в C# і дозволяє виконувати обробку зображення паралельно з іншими завданнями програми.

На рисунку 4.1 показано інтерфейс розробленої програми. В даному інтерфейсі передбачено:

– п’ять кнопок керування процесом розпізнавання;

– область відображення відеопотоку з веб-камери;

– область моделювання рухів роботизованого маніпулятора;

– область виведення координат розташування об’єкту маніпуляції.

Кнопки керування виконують наступні функції:

– включення та управління відеозахватом (кнопка «Відео»);

– кнопка зупинки відеопотоку та перемикання на поле для моделювання розташування об’єкту захоплення на координатній площі;

– кнопка очищення поля для моделювання розташування об’єкту захоплення на координатній площі;

– кнопка для запуску процесу моделювання рухів маніпулятора к визначеним координатам об’єкту.

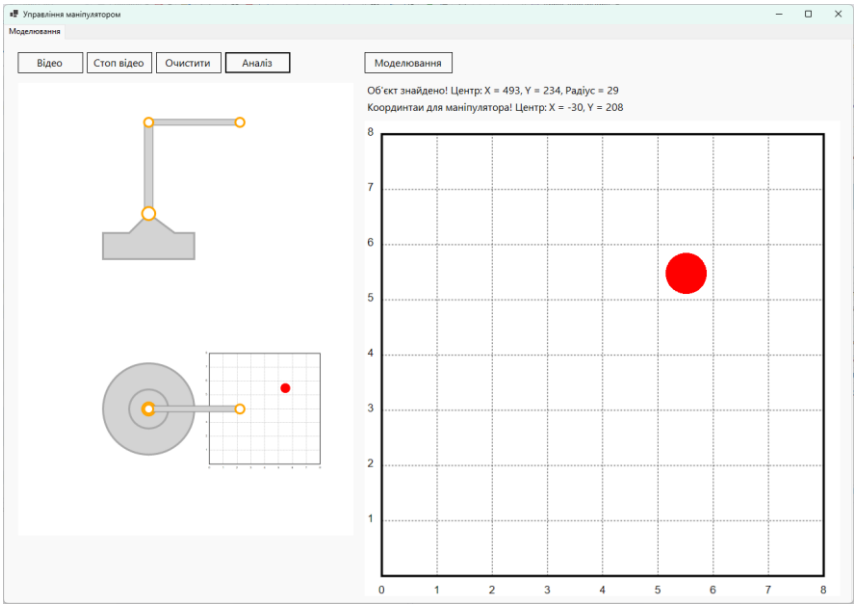


Рисунок 4.1 – Інтерфейс розробленої програми

Область відображення відеопотоку з веб-камери також використовується, як поле для моделювання розташування об’єкту захоплення на координатній площі. Переключення режимів роботи відбувається при керуванні включення відеозахвату потоку з веб-камери.

На рисунку 4.2 показано приклад функції запуску захвату відеозображення. Ця функція забезпечує управління відеозахопленням: початок, паузу і відновлення відео за допомогою однієї кнопки, змінюючи її текст відповідно до поточного стану. Функція викликається при натисканні кнопки button4 і відповідає за запуск та призупмнення захоплення відео з веб-камери, використовуючи бібліотеку Emgu.CV.

Спочатку перевіряється, чи ініціалізовано об'єкт _capture, який використовується для захоплення відео з камери. Якщо він ще не ініціалізований (тобто рівний null), створюється новий об'єкт VideoCapture. У

цьому ж блоці додається обробник події ImageGrabbed, що викликає метод ProcessFrame кожного разу, коли захоплюється новий кадр відео.

```
private void button4_Click(object sender, EventArgs e)
{
    if (_capture == null)
    {
        try
        {
            // Ініціалізуємо захоплення відео з веб-камери
            _capture = new VideoCapture();
            _capture.ImageGrabbed += ProcessFrame; // Додаємо обробник для кожного кадру
        }
        catch (Exception ex)
        {
            MessageBox.Show("Помилка доступу до камери: " + ex.Message);
        }
    }

    if (_capture != null)
    {
        if (_captureInProgress)
        {
            // Якщо відео вже йде, зупиняємо його
            _capture.Pause();
            button4.Text = "Старт";
        }
        else
        {
            // Запускаємо відео
            _capture.Start();
            button4.Text = "Пауза";
        }

        _captureInProgress = !_captureInProgress;
    }
}
```

Рисунок 4.2 – Приклад методу запуску функції захвату відеозображення

Якщо при ініціалізації або доступі до веб-камери виникає помилка, вона ловиться в блоці catch, і користувачу виводиться повідомлення з описом помилки через MessageBox.Show. Це дозволяє сповістити користувача про будь-які проблеми з підключенням до камери.

Далі перевіряється, чи активно відеозахоплення (через змінну _captureInProgress). Якщо відео вже йде (тобто _captureInProgress має значення true), воно зупиняється за допомогою методу Pause(), і текст на кнопці змінюється на "Старт". В іншому випадку, якщо відео не активне, викликається метод Start(), щоб розпочати захоплення відео, і текст кнопки змінюється на "Пауза".

Наприкінці, змінна _captureInProgress інвертується (міняється на протилежне значення), щоб відобразити поточний стан відеозахоплення. Якщо відео було зупинено, вона стає false, а якщо відео було розпочато – true.

Розпізнавання об’єктів на зображенні виконується за допомогою наступної функції:

```
private void DetectCircles(string imagePath)
{
    // Завантаження зображення за допомогою Emgu.CV
    Mat img = CvInvoke.Imread(imagePath, ImreadModes.Color);
    Mat grayImg = new Mat();
    // Конвертуємо зображення в відтінки сірого
    CvInvoke.CvtColor(img, grayImg, ColorConversion.Bgr2Gray);
    // Застосовуємо розмиття для зменшення шуму
    CvInvoke.GaussianBlur(grayImg, grayImg, new Size(9, 9), 2, 2);
    // Масштабування зображення до реального розміру
    float scaleFactor = 730.0f / pictureBox1.Width;
    int realWidth = (int)(img.Width * scaleFactor);
    int realHeight = (int)(img.Height * scaleFactor);
    CvInvoke.Resize(img, img, new Size(realWidth, realHeight));
    // Використовуємо метод Хаффа для пошуку кругів
    CircleF[] circles = CvInvoke.HoughCircles(grayImg, HoughModes.Gradient, 1.0, 20.0, 100, 30, 10, 100);
    // Виводимо центри кругів на зображення
    foreach (var circle in circles)
    {
        Point center = new Point((int)circle.Center.X, (int)circle.Center.Y);
        int radius = (int)circle.Radius;
        // Малюємо центр кола на зображенні
        CvInvoke.Circle(img, center, radius, new MCvScalar(0, 0, 255), 3);
        // Виводимо координати центру кола та його радіус
```

```
            Console.WriteLine($"Коло знайдено! Центр: X = {center.X}, Y = {center.Y}, Радіус = {radius}");
            label1.Text = String.Format("Об'єкт знайдено! Центр: X = {0}, Y = {1}, Радіус = {2}",
                center.X, center.Y, radius);
            //Конвертування координат
            var (xConverted, yConverted) = ConvertCoordinates(center.Y, center.X);
            label2.Text = String.Format("Координтаи для маніпулятора! Центр: X = {0}, Y = {1}",
                (int)xConverted, (int)yConverted);
            X_kinematic = (int)xConverted;
            y_kinematic = (int)yConverted;
            pictureBox1.Invalidate();
        }
        // Показуємо оброблене зображення в PictureBox
        Image<Bgr, byte> image = img.ToImage<Bgr, byte>();
        pictureBox1.Image = img.ToImage<Bgr, byte>().ToBitmap();
    }
```

Головною функцію даного методу є CvInvoke.HoughCircles (рис. 4.3).

```
CircleF[] circles = CvInvoke.HoughCircles(
    grayImg,             // 1. Вхідне зображення (відтінки сірого)
    HoughModes.Gradient, // 2. Метод Хафа, використовується для пошуку кіл (часто HoughModes.Gradient)
    1.0,                 // 3. Роздільна здатність радіуса. Якщо 1.0, то така сама, як у вхідного зображення.
    20.0,                // 4. Мінімальна відстань між центрами знайдених кіл.
    100,                 // 5. Поріг для визначення країв (значення верхнього порога для методу Canny).
    30,                  // 6. Поріг центрування. Нижній поріг для знаходження центрів кіл.
    10,                  // 7. Мінімальний радіус для знайдених кіл.
    100                  // 8. Максимальний радіус для знайдених кіл.
);
```

Рисунок 4.3 – Метод CvInvoke.HoughCircles

У методі CvInvoke.HoughCircles бібліотеки Emgu CV, який реалізує алгоритм пошуку кругів за допомогою методу Хафа, параметри мають таке

значення:

– grayImg – вхідне зображення (відтінки сірого);

– HoughModes.Gradient – метод Хафа, використовується для пошуку об’єктів;

– 1.0 – роздільна здатність радіуса. Якщо 1.0, то така сама, як у вхідного зображення;

20.0 – мінімальна відстань між центрами знайдених об’єктів;

100 – поріг для визначення країв (значення верхнього порога для методу Canny);

30 – поріг центрування. Нижній поріг для знаходження центрів об’єктів;

10 – мінімальний радіус для знайдених об’єктів;

100 – максимальний радіус для знайдених об’єктів;

);

Для роботи з хмарним середовищем використовується наступний програмний код:

```
static string SendOpenAIRequest(string apiKey, string base64Image)
{
   // URL API OpenAI
   string apiUrl = "https://api.openai.com/v1/chat/completions";
   // Створення клієнта для відправки запиту
   var client = new RestClient(apiUrl);
   // Створення запиту
   var request = new RestRequest()
      .AddHeader("Authorization", $"Bearer {apiKey}")
      .AddHeader("Content-Type", "application/json");
   // Формування запиту у форматі JSON
   var payload = new
   {
      model = "gpt-4o-mini", //"gpt-4o",
```

```
    messages = new[]
    {
    new
    {
        role = "user",
        content = new object[]
        {
            new
            {
                type = "text",
                text = "Find the coordinates of the round    " +
                 "red object in the image. " +
                "Provide the information in the form: " +
                "center: x=...; y=..."
            },
            new
            {
                type = "image_url",
                image_url = new
                {
                    url = $"{base64Image}"
                }
            }
        }
    }
},
    max_tokens = 300
};
// Серіалізація запиту у JSON
string jsonPayload = JsonSerializer.Serialize(payload);
```

```
        // Додаємо тіло запиту у JSON форматі
        request.AddStringBody(jsonPayload, DataFormat.Json);
        // Виконуємо запит
        RestResponse response = client.Post(request);
        // Повертаємо відповідь
        return response.Content;
    }
}
```

Даний код дозволяє передати зображення та текстовий запит до OpenAI API, а потім отримати відповідь, що містить координати об’єкта на зображенні, що корисно для розпізнавання об’єктів або інших завдань аналізу зображень.

Цей код реалізує функцію для відправлення запиту до OpenAI API за допомогою бібліотеки RestSharp, яка використовується для роботи з HTTP-запитами. Мета цієї функції – передати зображення у форматі base64 разом з текстовим запитом, щоб модель OpenAI могла обробити зображення та повернути координати об’єкта, що на ньому розташований.

Спочатку встановлюється URL для API OpenAI, що вказує на ендпоінт для отримання відповідей від моделі GPT. Далі створюється об'єкт клієнта RestClient, який використовується для відправлення запиту на сервер OpenAI. Запит налаштовується з двома важливими заголовками:

– перший – для авторизації за допомогою API-ключа;

– другий – для вказівки, що тіло запиту має бути у форматі JSON.

Тіло запиту складається з двох основних частин. Перша частина – це текстовий запит, що вказує моделі на необхідність знайти координати круглого червоного об'єкта на зображенні. Друга частина – це саме зображення, яке передається у форматі base64, де зображення конвертується в рядок типу {base64Image}.

Після цього об'єкт запиту серіалізується у формат JSON, використовуючи метод JsonSerializer.Serialize. Потім цей JSON додається до

запиту як тіло. Далі виконуються операції по відправці запиту на сервер через метод client.Post(request).

Після того, як сервер обробить запит, результат повертається у вигляді відповіді через об'єкт RestResponse, а відповідь передається у функцію у вигляді рядка через response.Content.

4.2 Експериментальні дослідження

Виконаємо перевірку обраного методу управління роботизованим маніпулятором з використанням штучного інтелекту.

Перший експеримент проведемо з використанням поля для моделювання розташування об’єкту захоплення на координатній площі. На рисунку 4.4 показано розташування тестового об’єкта на полі та початкове розташування маніпулятора.

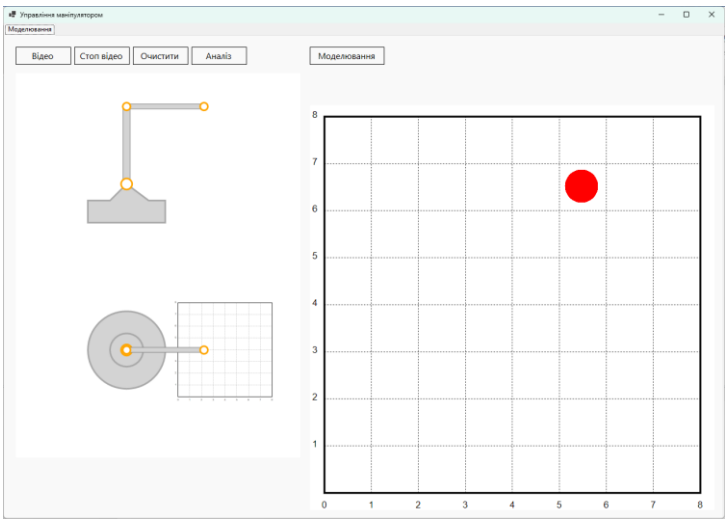


Рисунок 4.4 – Розташування тестового об’єкта на полі та початкове розташування маніпулятора

Після натискання на кнопку «Аналіз» програма запускає алгоритм пошуку об’єкту маніпуляції на тестовому полі. Результат пошуку

відображається у верхній області форми (рис. 4.5).

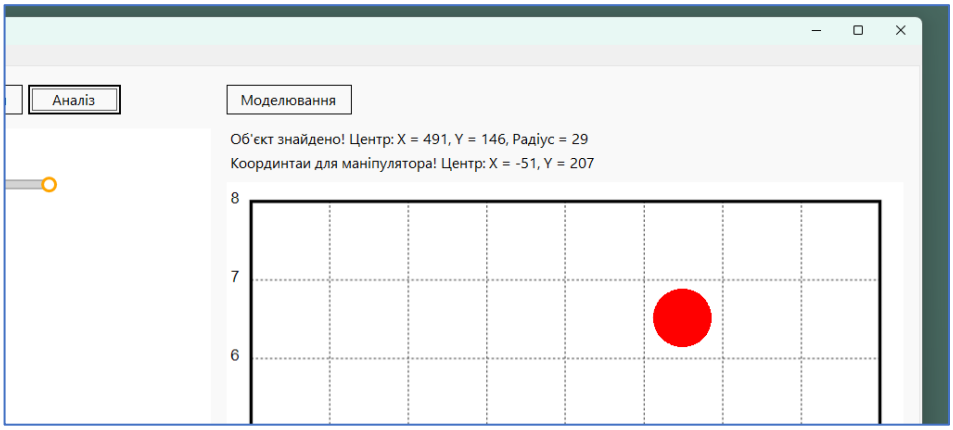


Рисунок 4.5 – Відображення координат центру знайденого об’єкту

Далі запускаємо моделювання переміщення цифрового двійника роботизованого маніпулятора для переміщення робочого органу в центр розташування об’єкту маніпуляції (рис. 4.6).

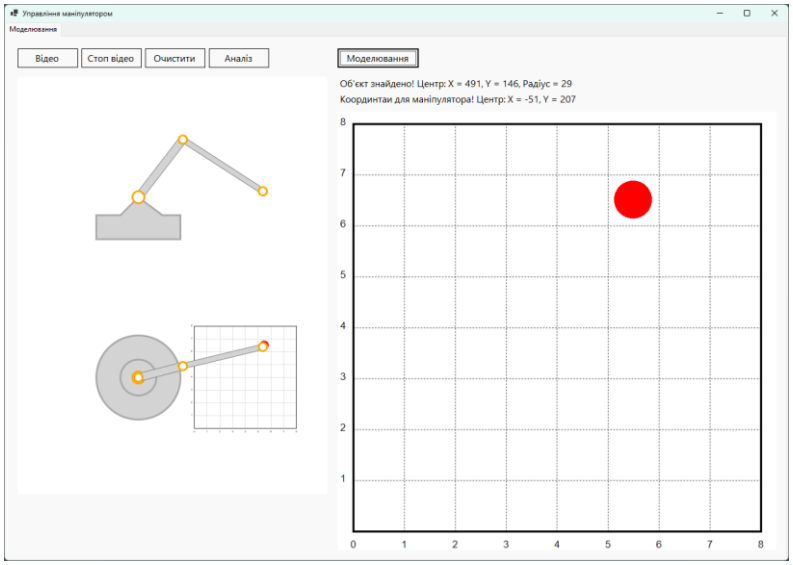


Рисунок 4.6 – Моделювання переміщення робочого органу маніпулятора в задану точку

На рисунку 4.7 показана траєкторія руху маніпулятора. На рисунку 4.8

показано похибку позиціонування робочого інструменту маніпулятора відносно місця розташування об’єкта.

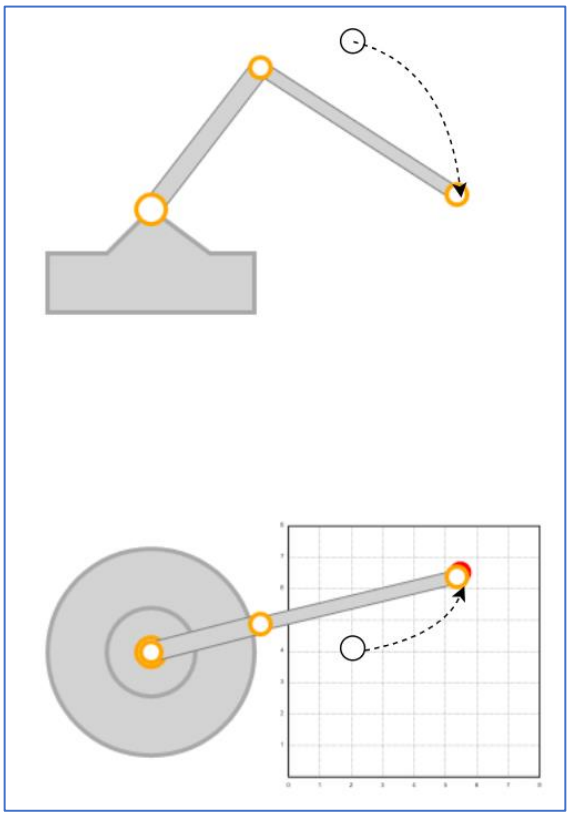


Рисунок 4.7 – Траєкторія руху маніпулятора

В масштабі вікна візуалізації похибка становить 3 мм. Дана подія пов’язана з алгоритмом перетворення визначених координат в реальні координати переміщення маніпулятора.

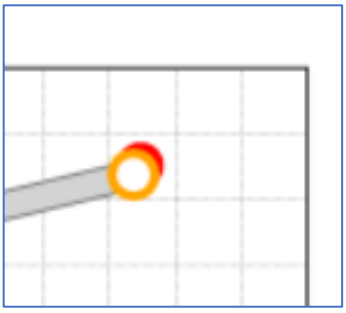


Рисунок 4.8 – Похибка позиціонування робочого інструменту маніпулятора

Для перетворення координат застосовується наступна функція:

```
public (double, double) ConvertCoordinates(double x, double y) {
// Вхідні координати мають діапазон від 50 до 665
double xInputMin = 50;
double xInputMax = 665;
double yInputMin = 50;
double yInputMax = 665;
// Вихідні координати мають діапазон для x від -75 до 75,
// а для y від 100 до 250
double xOutputMin = -75;
double xOutputMax = 75;
double yOutputMin = 100;
double yOutputMax = 250;
// Лінійне масштабування для x
double xOutput = (x - xInputMin) / (xInputMax - xInputMin) * (xOutputMax - xOutputMin) + xOutputMin;
// Лінійне масштабування для y
double yOutput = (y - yInputMin) / (yInputMax - yInputMin) * (yOutputMax - yOutputMin) + yOutputMin;
// Повертаємо результат
return (xOutput, yOutput); }
```

Ця функція відповідає за перетворення вхідних координат у заданому діапазоні в нові координати, які мають інший діапазон. Вхідні координати для x і y знаходяться в діапазоні від 50 до 665, а вихідні координати для x будуть знаходитися в діапазоні від -75 до 75, а для y – в діапазоні від 100 до 250.

Такі значення обумовлені розмірами тестового поля та поля візуалізації цифрового двійника роботизованого маніпулятора. Також маніпулятор має нульове значення координати x по центру робочого поля. Переміщення вліво має негативні значення, а вправо – позитивні.

Процес перетворення координат виконується через лінійне масштабування. Спочатку функція визначає межі вхідних координат (xInputMin, xInputMax, yInputMin, yInputMax) та межі вихідних координат (xOutputMin, xOutputMax, yOutputMin, yOutputMax). Потім за допомогою формули лінійного масштабування для кожної координати (x і y) здійснюється їх перерахунок із старого діапазону в новий.

Для обчислення нових координат x і y використовуються такі формули:

$$x_{out} = \frac{(x - x_{in\_min})}{(x_{in\_max} - x_{in\_min})} \cdot (x_{out\_max} - x_{out\_min}) + x_{out\_min}, \qquad (4.1)$$

де x – визначені координати розташування об’єкту;

x_{in_min} – мінімальна можлива координата *x* тестового поля;

x_{in_max} – максимальна можлива координата *x* тестового поля;

x_{out_min} – мінімальна можлива координата *x* віртуального поля;

x_{out_max} – максимальна можлива координата *x* віртуального поля.

Аналогічно розраховується координата y:

$$y_{out} = \frac{(y - y_{in\_min})}{(y_{in\_max} - y_{in\_min})} \cdot (y_{out\_max} - y_{out\_min}) + y_{out\_min}, \qquad (4.2)$$

де y – визначені координати розташування об’єкту;

y_{in_min} – мінімальна можлива координата *y* тестового поля;

y_{in_max} – максимальна можлива координата *y* тестового поля;

y_{out_min} – мінімальна можлива координата *y* віртуального поля;

y_{out_max} – максимальна можлива координата *y* віртуального поля.

На рисунку 4.9 показано розпізнавання положення об’єкта на зображенні з веб-камери із застосуванням хмарного середовища на основі ШІ.

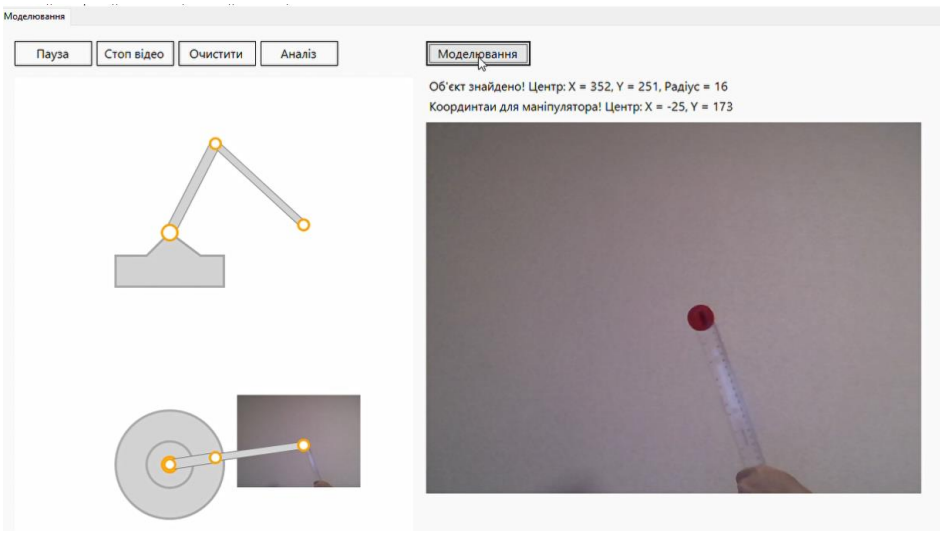


Рисунок 4.9 – Розпізнавання положення об’єкта на зображенні з веб-камери із застосуванням хмарного середовища на основі ШІ

На рисунку 10 показана послідовніть визначення координат в реальному часі.

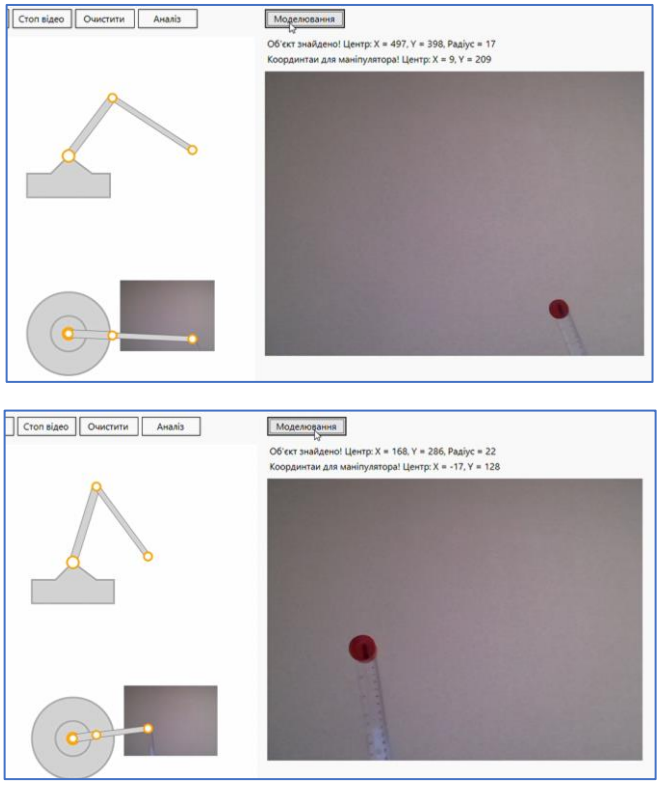


Рисунок 4.10 – Послідовніть визначення координат в реальному часі

4.3 Розрахунок штучного освітлення приміщення

Лабораторія, де виконується розробка програмного забезпечення, має наступні характеристики:

– площа приміщення – 70 м^2 (10 м × 7 м);

– висота – 3,5 м;

– кількість робочих місць – 7 р.м .;

– обладнання – стіл з ПК і периферією (7 шт).

Приміщення, відповідно до ДНАОП 0.00-1.31-99, має забезпечувати 6 м^2 площі і 20 м^3 обсягу на одне окреме робоче місце з ЕОМ. Площа приміщення 70 м^2 і обсяг 245 м^3, на кожне робоче місце припадає 8,8 м^2 площі і обсяг 35 м^3, тобто вимога виконана.

Приміщення з ЕОМ повинні мати природне і штучне освітлення відповідно до ДБН В.25-28-2006 «Природне і штучне освітлення» [15]. Природне світло повинно проникати через бічні світлові прорізи, зорієнтовані, як правило, на північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5%.

Розрахункова формула методу проводиться за (4.3):

$$W = \frac{W_{\Sigma}}{S}, \tag{4.3}$$

де W – питома потужність, Вт/м^2;

S – площа приміщення, м^2;

W_{Σ} – загальна потужність освітлювальної установки Вт, яка розраховується за (4.4):

$$W_{\Sigma} = W_{св} \cdot n_{св}, \tag{4.4}$$

де $W_{св}$ – потужність одного освітлювача, Вт;

$n_{св}$ – кількість освітлювачів у приміщенні

$$W_{\sum} = 100 \cdot 5 = 500 \text{ Вт},$$

$$W = \frac{500}{70} = 7{,}14 \text{ Вт/м}^2.$$

Питомій потужності 7,14 Вт/м^2 відповідає освітленість в 250 лк при мінімальній допустимої освітленості 300 лк.

Отже, для створення сприятливих зорових умов в лабораторії необхідно збільшити кількість світильників або замінити лампи в світильниках на більш потужні або використати сучасні світлодіодні лампи зі збереженням тих же показників освітлення при зменшенні споживаної потужності.

4.4 Висновки по четвертому розділу

Таким чином, проведені експериментальні дослідження, що проведені а четвертому розділі кваліфікаційної роботи показала, що запропонований метод працює як для тестового зображення так і для потокового відео в реальному часі.

Розроблене програмне забезпечення використовує бібліотеки OpenCV для локального розпізнавання зображення та хмарне середовище Open AI для розпізнавання зображення з нечіткими границями та забрудненим фоном. Програма для виконання експериментальних досліджень написана в Microsoft Visual Studio 2022 з використанням мови програмування C#.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблена автоматизована система управління роботизованим маніпулятором з використанням штучного інтелекту.

Виконано аналіз сфер застосування роботизованих маніпуляторів та показані переваги застосування штучного інтелекту в порівнянні з традиційними засобами керування. Розглянуті основні компоненти роботів маніпуляторів. Проаналізовано типову архітектуру системи управління роботизованим маніпулятором з використанням ШІ.

Виконано аналіз методів машинного навчання. Дано визначення терміну «Машинне навчання», показано взаємозв'язок між машинним навчанням, глибоким навчанням та штучним інтелектом. Проаналізовані відмінності машинного навчання від традиційних експертних систем.

Проаналізовано процес підготовки та виконання завдання машинного навчання на прикладі інструментів Microsoft ML. Проведено огляд таких методів, як метод опорних векторів, лінійної регресії, логістичної регресії та алгоритму k-найближчих сусідів. Показані їх позитивні боки та недоліки.

Виконано розроблення архітектури автоматизованої системи роботизованим маніпулятором з використанням штучного інтелекту. Описано склад розробленої системи. Вона забезпечує обробку зображень і прийняття рішень завдяки поєднанню локальних обчислень на основі бібліотеки OpenCV та хмарного середовища, що працює з використанням технологій розпізнавання зображень зі штучним інтелектом.

Виконано розроблення алгоритму роботи автоматизованої системи. Описано алгоритм роботи основної частини автоматизованої системи управління роботизованим маніпулятором, де задачами системи є отримання зображення з веб-камери, попередня обробка та локальне розпізнавання об’єкту маніпуляції. Якщо в процесі локальної обробки зображення засобами

OpenCV не було виявлено об’єкту маніпуляції (це може бути в результаті поганого освітлення, або зашумленого фону), автоматизована система перемикається на використання хмарного середовища, що працю за принципом штучного інтелекту.

Проведені експериментальні дослідження, що проведені в четвертому розділі кваліфікаційної роботи показала, що запропонований метод працює як для тестового зображення так і для потокового відео в реальному часі.

Розроблене програмне забезпечення використовує бібліотеки OpenCV для локального розпізнавання зображення та хмарне середовище Open AI для розпізнавання зображення з нечіткими границями та забрудненим фоном. Програма для виконання експериментальних досліджень написана в Microsoft Visual Studio 2022 з використанням мови програмування C#.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008–2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Документація. – Введ. 2015-06-22. - К.: Держстандарт України, 2015. - 31 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2023. 55 с..

3. Кафедра технології та автоматизації виробництва РЕЗ та ЕОЗ [Електронний ресурс] – Режим доступу: www/ URL: https://tapr.nure.ua/wp-content/uploads/2020/12/buklet_17-30.pdf

4. Основи наукових досліджень: Навч. посібник / І.Ш. Невлюдов, Ю.М. Олександров, А.О. Андрусевич, О.О. Чала. – Кривий Ріг: Криворізький коледж НАУ, 2019. – 396 с.

5. Положення про кваліфікаційну роботу здобувача вищої освіти на другому (магістерському) рівні [Електронний ресурс] : Наказ ХНУРЕ від 06 травня 2021 р. № 143. – Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/143-vid-06.05.2021-pro-vvedennja-v-diju-rishennja-vchenoi-radi-universitetu.pdf.

6. Положення про протидію академічному плагіату в Харківському національному університеті радіоелектроніки [Електронний ресурс] : наказ ректора ХНУРЕ від 28.04.2017 р. № 290 // Нормативно-правова база ХНУРЕ : офіційний веб-портал. – Режим доступу: https://nure.ua/universytet/normativno-pravova-baza#id13. – Станом на 18.09.2024. – Назва з екрану.

7. 10 Industrial Robotic Arms Manufacturers | Blog | DIY Robotics. DIY Robotics. URL: https://diy-robotics.com/blog/10-industrial-robotic-arms-manufacturers/ (date of access: 27.10.2024).

8. Čehovin Zajc, Luka & Rezelj, Anže & Skočaj, Danijel. (2017). Open-Source Robotic Manipulator and Sensory Platform. 10.1007/978-3-319-62875-2_22

9. Asongo, A.I, et. al. International Journal of Engineering Research and Applications. ISSN: 2248-9622, Vol. 11, Issue 8, (Series-II) August 2021, pp. 55-64.

10. Introduction to Artificial Intelligence and Machine Learning. Community.aws. URL: https://community.aws/content/2drbbXokwrIXivItJ8ZeCk3gT5F/introduction-to-artificial-intelligence-and-machine-learning (date of access: 03.01.2025).

11. OpenCV – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення: 05.01.2025).

12. Основи розробки кросплатформного програмного забезпечення на Avalonia : Навчальний посібник / С. П. Новоселов, О. В. Сичова. – Харків: Видавництво Іванченка І. С., 2024. – 267 с. ISBN 978-617-8332-57-0. DOI: 10.30837/978-617-8332-57-0.

13. Невлюдов І. Ш. Застосування цифрових двійників технічних засобів автоматизації для розроблення програмно-технічних комплексів АСУ ТП : Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. – Харків: Видавництво Іванченка І. С., 2023. – 267 с. ISBN 978-617-8059-95-8, DOI: 10.30837/978-617-8059-95-8.

14. Невлюдов І. Ш. Технологія програмування промислових контролерів в інтегрованому середовищі CODESYS : навч. посіб. / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2019. – 264 с. : іл. –ISBN 978-966-659-265-4. DOI: 10.30837/978-966-659-265-4.

15. Стиценко Т.Є., Пронюк Г.В., Сердюк Н.М., Хондак І.І. Безпека життєдіяльності : Навчальний посібник / Т.Є Стиценко, Г.В. Пронюк, Н.М. Сердюк, І.І. Хондак. – Харків: ХНУРЕ, 2018. – 336 с.