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

перший (бакалаврський)
(рівень вищої освіти)

Розроблення автоматизованого модуля керування роботом маніпулятором з використанням системи комп'ютерного зору з елементами штучного інтелеку
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТАКІТ 21-3
Катерина КАЗАНОВСЬКА
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник професор Сергій НОВОСЛЕОВ
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Казановська Катерина Андріївна, як здобувачка вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надававла і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовувала штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

"9" червня 2025 р. Катерина КАЗАНОВСЬКА

Харківський національний університет радіоелектроніки

Факультет АКТ

Кафедра КІТАР

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп’ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Казановській Катерині Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи Розроблення автоматизованого модуля керування роботом маніпулятором з використанням системи комп’ютерного зору з елементами штучного інтелекту
затверджена наказом університету від 19 травня 2025 р. № 390 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 11 червня 2025 р.
3. Вихідні дані до роботи програмна платформа .NET Framework, середовище розробки JetBrains Rider, середовище розробки Arduino IDE, відкрита бібліотека з засобами машинного навчання ML.NET, бібліотека комп’ютерного зору OpenCV
4. Перелік питань, що потрібно опрацювати в роботі Аналіз програмних засобів використання комп’ютерного зору та штучного інтелекту в процесах сортування, провести вибір компонентної бази розроблюваної системи, визначити необхідні засоби розробки програмного забезпечення, виконати побудову алгоритму та реалізацію визначеного функціоналу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
__Презентація PowerPoint на 13 слайдів_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	30.04.2025	
2	Аналіз літератури за темою	06.05.2025	
3	Вибір та обгрунтування компонентів системи	14.05.2025	
4	Збірка макету	16.05.2025	
5	Розроблення програмного забезпечення	01.06.2025	
6	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism	11.06.2025	
7	Подання роботи на рецензію	12.06.2025	
8	Подання роботи на підпис зав. кафедри	20.06.2025	
9	Подання кваліфікаційної роботи в ЕК	23.06.2025	

Дата видачі завдання __30__ __квітня_____20_25_ р.

Здобувач _____ _____Катерина КАЗАНОВСЬКА_____
(підпис)

Керівник роботи _____ _професор Сергій НОВОСЕЛОВ___
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 106 стор., 38 рис., 9 табл., 3 додатки, 39 джерел.

OPENCV, C#, КОМП'ЮТЕРНИЙ ЗІР, РОБОТ МАНІПУЛЯТОР, ARDUINO, ВИЯВЛЕННЯ ОБ'ЄКТІВ, СОРТУВАННЯ, КООРДИНАТИ ОБ'ЄКТА, ШТУЧНИЙ ІНТЕЛЕКТ

Мета роботи – підвищення ефективності процесу сортування об’єктів шляхом впровадження сиcетми комп’ютерного зору та штучного інтелекту для ідентифікації та класифікації об’єктів.

Об’єкт розробки – процес сортування продукції на основі рішень навченої моделі штучного інтелекту.

Предмет розробки – програмне забезпечення для виконання налаштування та контролю сортування продукції.

Проведено розробку структурної схеми автоматизованого модуля керування роботом маніпулятором з використанням системи комп’ютерного зору з елементами штучного інтелекту.

Визначено основні технології, що треба використовувати в розробці, складено алгоритм виконання операцій за яким буде виконано розробку програмного забезпечення.

Проведено підбір компонентів системи, обрано контролер, двигуни та деталі для макету робота маніпулятора; обрано камеру, характеристики котрої задовільняють визначені умови.

Розроблено програмне забезпечення для моніторингу та керування процесом сортування та ручним керуванням роботом. Зібрано макет для демонстрації роботи.

ABSTRACT

Explanatory note: 106 pages, 38 figures, 9 tables, 3 appendix, 39 sources.

OPENCV, C#, COMPUTER VISION, ROBOT MANIPULATOR, ARDUINO, OBJECT DETECTION, SORTING, OBJECT COORDINATES, ARTIFICIAL INTELLIGENCE

The purpose of the work is to develop software for configuring and controlling the sorting process by a robot manipulator with a computer vision module and elements of artificial intelligence.

The object of development is the process of sorting products based on the solutions of a trained artificial intelligence model.

The subject of development is software for configuring and controlling product sorting.

The structural diagram of the automated module for controlling the robot manipulator using a computer vision system with elements of artificial intelligence has been developed.

The main technologies to be used in the development have been determined, an algorithm for performing operations has been developed, according to which the software development will be performed.

The system components have been selected, the controller, engines and parts for the robot manipulator model have been selected; a camera has been selected, the characteristics of which satisfy the specified conditions.

Software has been developed for monitoring and controlling the sorting process and manual control of the robot. A model has been assembled for demonstrating the work.

ЗМІСТ

Перелік скорочень 7

Вступ 8

1 Аналіз технічного завдання 10

1.1 Призначення та функціональні вимоги до системи 10

1.2 Аналіз існуючих рішень автоматизованого сортування об’єктів роботами маніпуляторами 11

1.3 Аналіз методів комп’ютерного зору 17

1.4 Аналіз застосування штучного інтелекту на виробництві 20

1.5 Постановка задачі 23

2 Принцип роботи автоматизованої системи 25

2.1 Розробка структурної схеми взаємодії компонентів системи 25

2.2 Алгоритм обробки зображень і визначення координат об’єктів 28

2.3 Логіка класифікації та правила сортування об’єктів 35

2.4 Алгоритм руху маніпулятора та управління захватом 41

3 Вибір компонентів системи 44

3.1 Вибір камери та її параметрів 44

3.2 Вибір робота-маніпулятора та контролера керування 45

3.3 Вимоги до робочої поверхні 47

4 Розробка програмного забезпечення 49

4.1 Програмний стек та конфігурація проекту 49

4.2 Конфігураційний файл та модуль його обробки 50

4.3 Повідомлення системи для користувача 59

4.4 Модуль комп’ютерного зору 61
4.5 Розробка та навчання моделі штучного інтелекту для класифікації об’єктів 67
4.6 Модуль управління маніпулятором 72
4.7 Реалізація графічного інтерфейсу оператора 74
4.8 Прошивка камери 93
4.9 Результати випробувань 94
4.10 Розрахунок показників стійкості системи 96
5 Охорона праці 98
5.1 Робота з персональним комп’ютером 98
5.2 Робота з рухомими об’єктами 99
Висновки 100
Перелік джерел посилання 101
Додаток А Апробація результатів роботи 107
Додаток Б Код програми 115
Додаток В Демонстраційний матеріал 163

ПЕРЕЛІК СКОРОЧЕНЬ

КІТАР – комп’ютерно-інтегрованих технологій, автоматизації та робототехніки;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СБ – сортувальний бокс;

ХНУРЕ – Харківський національний університет радіоелектроніки;

ШІ – штучний інтелект;

FPS – Frames per second;

PLA – Polylactic Acid;

SCARA – Selective Compliance Assembly Robot Arm;

TSV – Tab-Separated Values.

ВСТУП

Ключовим критерієм успіху сучасного виробництва є ефективність технологічних процесів. Виробництва оптимізують процеси, щоб підвищити продуктивність, зменшити кількість дефектів та знизити собівартість продукції. Важливими чинниками також є стабільна робота обладнання та раціональне використання ресурсів й енергії. Застосування штучного інтелекту дозволяє ефективно вирішувати ці завдання.

Штучний інтелект (ШІ) здатний в короткий час аналізувати велику кількість даних, що значно зменшує кількість простоїв та пришвидшує прийняття рішень. Аналізуючи інформацію з датчиків, ШІ може приймати рішення щодо необхідності ремонту певного обладнання ще до виникнення негараздів, таким чином фахівці можуть виконувати ремонт в часи меншої завантаженості на виробництві та уникнути неочікуваного простою. Аналізуючи обороти продукції на складі можливе визначення зросту або падіння необхідності виготовлення, фасування певного типу продукції.

Залучення штучного інтелекту до технологічних процесів виробництва може значно підвищити точність сортування продукції, збільшення швидкості процесу та зниження витрат за рахунок зменшення потреби у ручній праці.

Метою роботи є підвищення ефективності процесу сортування об’єктів шляхом впровадження сисетми комп’ютерного зору та штучного інтелекту для ідентифікації та класифікації об’єктів.

Об’єктом розробки є процес сортування продукції на основі рішень навченої моделі штучного інтелекту.

Предметом розробки є програмне забезпечення для виконання налаштування та контролю сортування продукції.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих програмних рішень автоматизації роботи з використанням робота маніпулятора, комп’ютерного зору та штучного інтелекту;

– провести підбір елементної бази компонентів системи;

– розробити структурну схему взаємодії компонентів системи;

– розробити модулі використання ШІ, комп’ютерного зору та керування роботом;

– оформити кваліфікаційну роботу згідно ДСТУ 3008:2015 [1], а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» [2].

Проведені дослідження відповідають цілям сталого розвитку, зокрема Цілі сталого розвитку 9 – «Створення стійкої інфраструктури, сприяння всеохоплюючій і сталій індустріалізації та інноваціям» [3], оскільки спрямовані на впровадження інноваційних рішень в автоматизацію виробничих процесів, підвищення технологічного рівня сортувальних систем, оптимізацію ресурсів та розвиток інтелектуальних промислових технологій.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Призначення та функціональні вимоги до системи

Система призначена для автоматизації процесу сортування об’єктів на виробничому столі без участі оператора. Сортування об’єктів проводиться шляхом визначення на робочій поверхні об’єктів та їх координат, попередньо навчена модель штучного інтелекту проводить класифікацію кожного об’єкту на основі результату проводиться транспортування об’єкту до відповідної тари. Загальна схема розташування та взаємодії компонентів показана на рисунку 1.1.

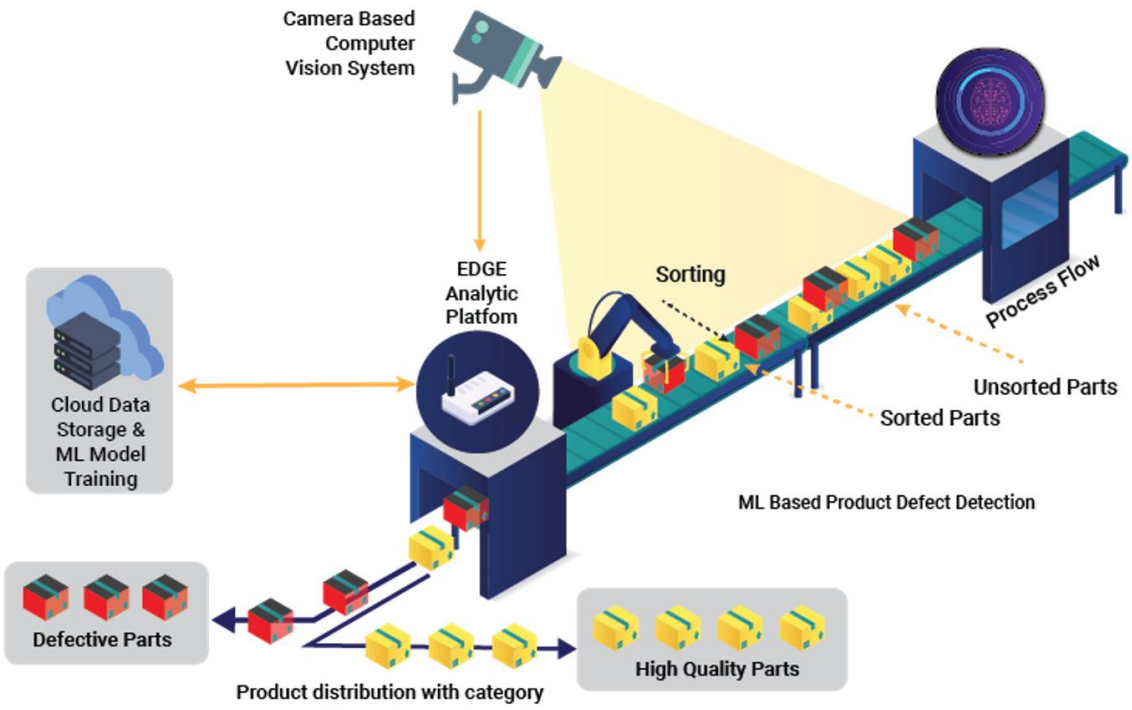


Рисунок 1.1 – Схема розташування та взаємодії компонентів системи сортування об’єктів [4]

Основними компонентами описаної системи є робот-маніпулятор, робоча поверхня, камера, модуль комп’ютерного зору, модель штучного

інтелекту. При потраплянні об’єкту в поле зору камери виконується захоплення кадру, який направляється до моделі штучного інтелекту на виконання перевірки наявності дефекту. В залежності від відповіді моделі, роботу-маніпулятору надаються інструкції стосовно сортування об’єкту.

Відповідно до опису призначення системи визначено, що мають бути виконані наступні функціональні вимоги:

– коректне приймання та обробка відеопотоку зі швидкістю не менше 30-ти кадрів на секунду;

– виконання сортування в реальному часі, не зупиняючи роботу конвеєрної стрічки;

– визначення координат з похибкою в межах 5-ти міліметрів;

– виконання класифікації об’єкту з впевненістю не менше 90-та відсотків;

– цикл сортування одного об’єкту не має перевищувати 5-ти секунд від початку руху до об’єкту з початкового положення робота до моменту відпускання об’єкту в кінцеву точку маршруту.

1.2 Аналіз існуючих рішень автоматизованого сортування об’єктів роботами маніпуляторами

Процес сортування призначений для відділення об’єктів за певними характеристиками, наприклад за кольором, формою, матеріалом, розміром тощо. Цей процес використовується для подальшої зручності зберігання продукції на складі, пакування, транспортування відсортованої продукції і тд.

Сортування є ручне та автоматизоване, в ручному задіяна людина, в автоматизованому – роботи, що керуються оператором або наданими оператором інструкціями. Для виконання повторюваних та однотипних дій часто використовуються роботи-маніпулятори через їх ефективність, робот не втомиться та ризик виникнення людського фактору відсутній,

сортування відбудеться строго у відповідності з наданими оператором інструкціями.

Промислові маніпулятори є зручним та гнучким інструментом для виконання різних завдань на виробництві. Конструкція робота маніпулятора нагадує руку людини та дозволяє відтворювати схожі до людських рухи (рис. 1.2), роботи-маніпулятори використовуються для таких завдань як перенесення вантажу, фарбування, зварювання тощо. Для виконання цих дій роботи-маніпулятори мають ряд датчиків, які дозволяють їм «відчувати» своє розташування у просторі та середовище навколо. Камери, сенсори, комп'ютерний зір, контролери – все це є компонентами для покращення та розширення можливостей робота-маніпулятора.

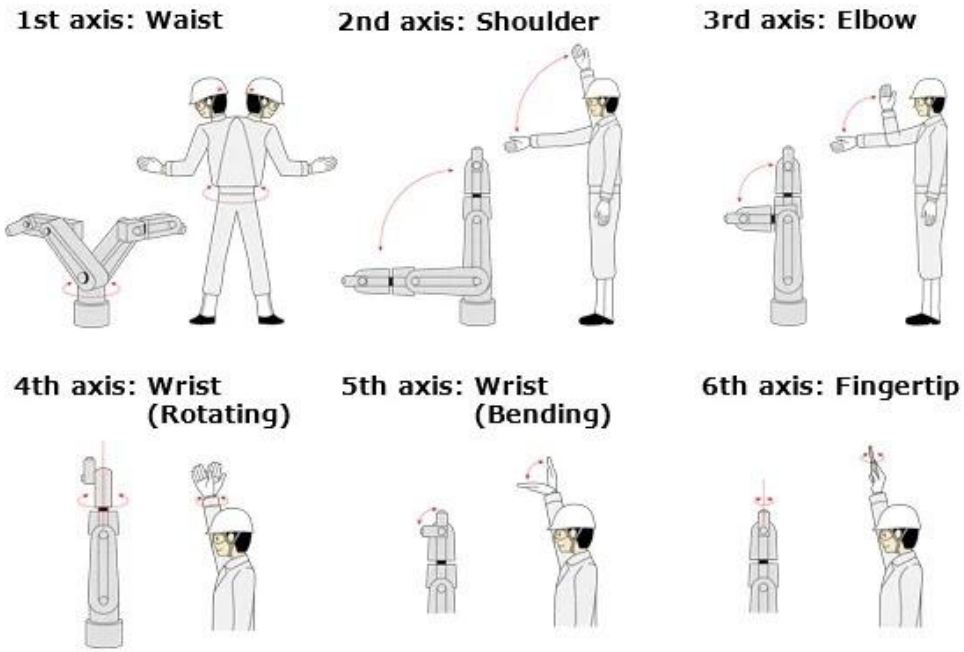


Рисунок 1.2 – Принцип роботи шестиосевого робота-маніпулятора порівняно з рухами людського тіла [5]

У промислових системах автоматизованого сортування об'єктів використовується декілька типів маніпуляторів, а саме наступні:

– шестикоординатні маніпулятори;

- дельта-роботи;
- SCARA-маніпулятори;
- декартові маніпулятори.

Декартові роботи також мають назву лінійних, оскільки вони рухаються по трьом лінійним осям X, Y та Z, дотримуючись декартової системи координат [6]. Конструкцію робота показано на рисунку 1.3.

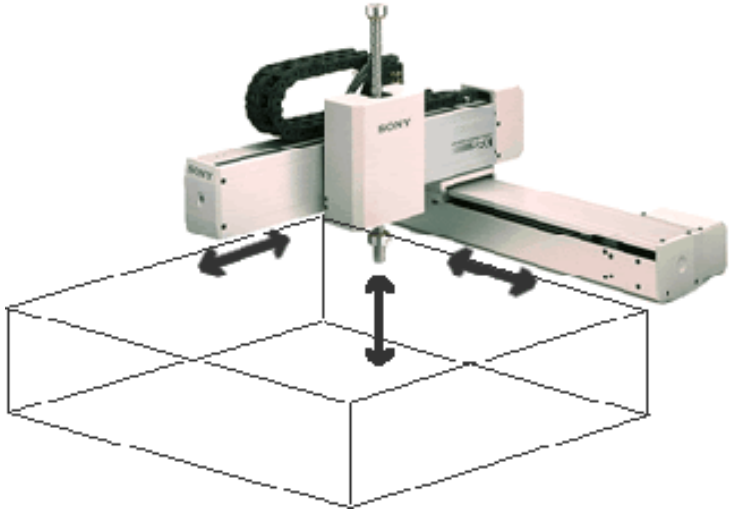


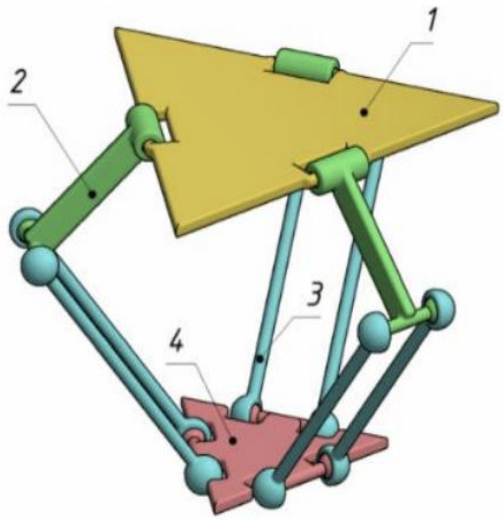
Рисунок 1.3 – Конструкція декартового робота [7]

Декартові роботи прості за конструкцією, налаштуванням та мають високу точність за рахунок чого активно використовуються в таких сферах як 3D-друк, захоплення та перенесення, перевірки якості. Працюють такі роботи в доволі обмеженому просторі через особливості конструкції, їх робоча площа квадратна або прямокутна, що може викликати складності для певних задач.

Дельта-роботи є маніпуляторами з паралельною структурою конструкції, високою швидкістю переміщення та ефективністю, конструкція робота показана на рисунку 1.4. Основа робота (1) є нерухомою, до неї шарнірно кріпляться три важелі (2), які з'єднуються через шарнірні з'єднання з трьома штангами (3). Інші кінці штанг

кріпляться до рухомої платформи (4), на якій встановлюється робочий орган [8].

Завдяки паралельній кінематичній схемі, платформа здатна швидко та точно переміщуватися в межах визначеної робочої зони. Дельта-роботи мають високу динамічність та точність позиціонування, вони мають низьку інерційність платформи, що дозволяє досягати високих прискорень та швидкостей. Дельта-роботи ідеально підходять для швидкого сортування легких об’єктів, конструкція робота не дозволяє виконати вантажопідйомність великою [8].



1 – основа, 2 – ричаг, 3 – штанга, 4 – платформа [8]

Рисунок 1.4 – Типова конструкція дельта робота

Роботи типу SCARA (Selective Compliance Assembly Robot Arm) – це промислові роботи маніпулятори, що мають невеликі розміри і займають небагато місця. Конструкція робота представлена на рисунку 1.5. Робот має дві горизонтальні осі обертання, що забезпечують рух робота по осям X та Y, вертикальна лінійна вісь здійснює рух захвату вздовж по осі Z.

Застосовуються маніпулятори типу SCARA в складанні електронних компонентів, сортуванні невеликих об’єктів, перенесення дрібних об’єктів.

Вантажопідйомність таких роботів невелика. Використання роботів такого типу обмежені за робочою зоною та гнучкістю конструкції.

Рисунок 1.5 – Конструкція робота типу SCARA [9]

Вертикальний шарнірний робот – це промисловий робот зі структурою послідовного з’єднання, має шість суглобів, тобто шість осей (рис. 1.6).

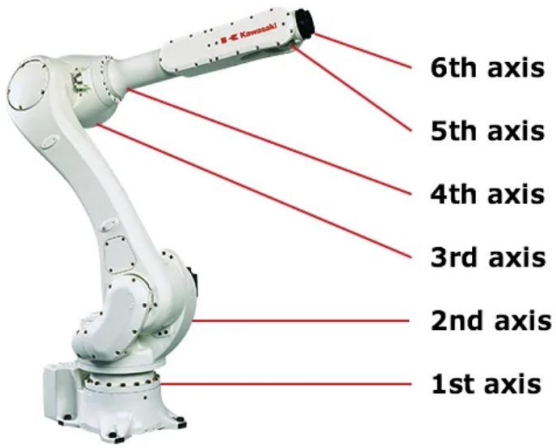


Рисунок 1.6 – Конструкція шестиосевого вертикального шарнірного маніпулятора [5]

Якщо розглянути можливі рухи робота з рухами людини (рис. 1.2), то видно, що з першої по третю осі це від талії до руки людини, з четвертої по шосту – від зап’ястя до кінчиків пальців [5].

Робочим органом прикріпленим до зап’ястя робота може бути інструмент підходящий під виконання поставленої задачі, це робить маніпулятор більш універсальним. Виходячи з матеріалу з якого виготовлений маніпулятор, його вантажопідйомність може бути високою.

На ринку автоматизованих систем представлено велику кількість компаній виробників різного роду роботизованих систем для задач підприємств. Найбільш відомими компаніями є FANUC, KUKA та Universal Robots.

Компанія FANUC займається розробкою рішень автоматизації процесів з 1956 року. FANUC перші в Японії побудували повністю автоматизоване виробництво, що було оснащене роботами, верстатами та числовим керуванням [10]. Наразі FANUC пропонує більше ніж 100 моделей роботів різних типів, які легко адаптуються до потреб виробництва. Вантажопідйомність роботів складає до 2,3 т, максимальний радіус робочої зони до 4,7 м [11].

Компанія FANUC має ряд серій роботів-маніпуляторів для різного роду задач. Серія LR Mate – це компактні рішення для швидкого переміщення та роботою з об’єктами до 14 кг. Роботи цієї серії підходять для багатьох галузей: від харчової до металообробної. Різниці між моделями серії в радіусі робочої зони, швидкістю зап’ястя та вантажопідйомністю. Роботів можна вдосконалювати, додаючи комп’ютерний зір, додаткове обладнання для зварювання, фарбування тощо [12].

Німецька компанія KUKA пропонує роботів KR AGILUS та KR DELTA. KR AGILUS – компактні шестикоординатні роботи, що поєднують високу точність та швидкість, ідеально підходять для сортування та точних операцій зі складання деталей. KR DELTA – це паралельні дельта-роботи з високою швидкістю роботи, що ідеально підходять для сортування дрібних об'єктів на високошвидкісних виробничих лініях. Дельта-роботи KR DELTA також мають окремо

роботів у гігієнічному дизайні, що надає можливість їх безпечного використання в харчовій, фармацевтичній промисловостях. Роботи мають вантажопідйомність від 2 до 1300 кг та досяжність від 500 до 4501 мм [13 - 14].

Universal Robots пропонує серію UR, яка включає колаборативні роботи. Основною перевагою цих маніпуляторів є їхня здатність безпечно працювати поруч із людиною без додаткових захисних бар'єрів. Вони мають інтуїтивно зрозуміле керування, гнучке програмування та швидку переналаштовуваність, що дозволяє активно використовувати їх для сортування та упаковки в умовах малого й середнього виробництва.

1.3 Аналіз методів комп'ютерного зору

Для виконання процесу сортування в сучасних автоматизованих системах використовуються методи комп'ютерного зору, щоб визначити положення, розмір, наявність дефекту об'єктів тощо. Класичними методами аналізу зображень є виділення контурів, сегментація зображення за кольором або яскравістю та детекція за допомогою ключових точок.

Робоча поверхня впливає на вибір методів обробки зображення, оскільки при чіткій та однорідній поверхні достатньо буде використовувати просте порогування та морфологічну обробку, при візуально навантаженій поверхні доведеться задіяти нейронні мережі.

Виявлення контурів об'єкта на зображенні може бути реалізовано за використання алгоритму Canny. Алгоритм Canny – це метод виявлення кордонів на зображенні, розроблений Джоном Канні у 1986 році. Спочатку зображення піддається згладжуванню фільтром Гаусса для зменшення шуму, потім виконується обчислення градієнтів зображення з використанням операторів Собеля у горизонтальному та вертикальному напрямках [15]. Результат виділення контурів за алгоритмом Canny показано на рисунку 1.7.

Алгоритм Canny є ефективним засобом виділення кордонів, що підтримує високу чутливість до контурів та мінімізує вплив шумів на зображення. Це робить його широко використовується в комп'ютерному зорі для завдань, таких як розпізнавання об'єктів, відстеження об'єктів, що рухаються, і виявлення контурів [15].

Сегментація зображення виконується для поділу зображення на сегменти, щоб надалі можна було провести аналіз кожного сегменту для визначення таких характеристик об’єкту як текстура, колір, форма тощо. Сегментація виконує чітке відділення об’єкту від фону. Приклад сегментації зображення показано на рисунку 1.8.

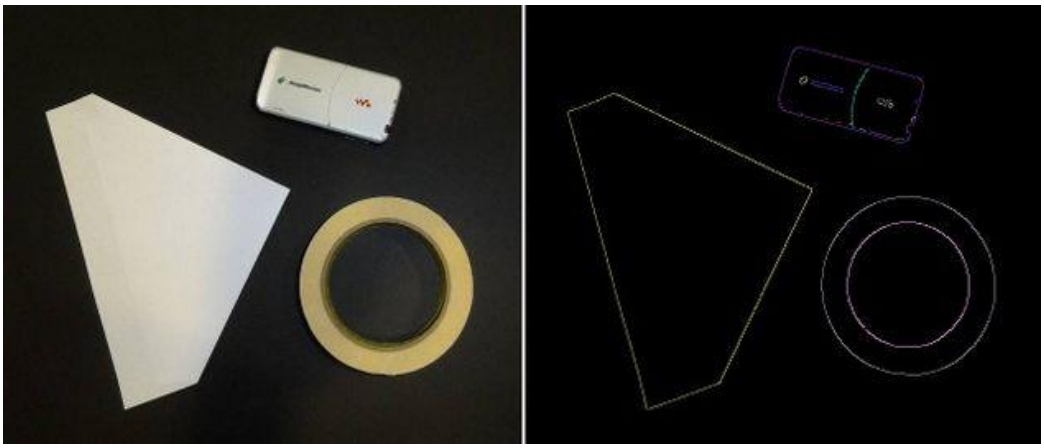


Рисунок 1.7 – Результат виявлення контурів за алгоритмом Canny [16]

Сегментація поділяється на такі типи як семантична, сегментація екземплярів та паноптична сегментація. Семантична сегментація виконує фарбування об’єктів відповідно до їх типу (всі три дерева буде позначено одного кольору, типу – дерева), а от сегментація екземплярів надає кожному об’єкту індивідуальну мітку («машина 1», «машина 2»). Паноптична сегментація поєднує в собі попередні дві, тобто визначає і тип, і індивідуальність.

Застосування сегментації зображення є актуальним для тих сфер, де необхідна чітка межа об’єктів, до таких сфер можна віднести медицину, автомобілі, робототехніку і тд.

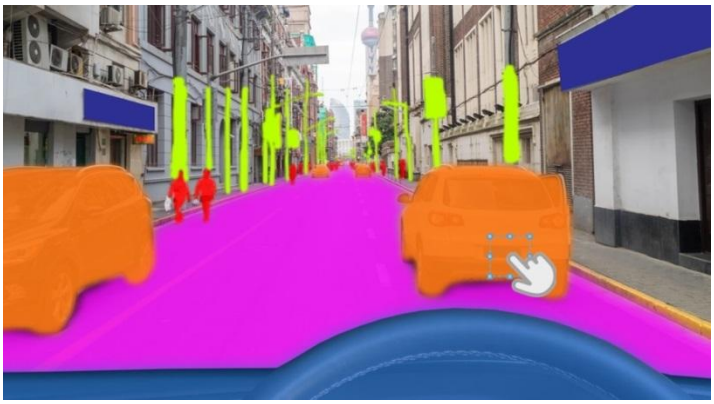


Рисунок 1.8 – Результат виконання сегментації зображення [17]

Оскільки під час процесу сортування також використовуються роботизовані системи виникає необхідність визначати координати об’єктів на робочій поверхні для подальшого перенесення об’єкту. Для визначення реальних координат об’єктів на робочій поверхні використовується система маркерів ArUco.

Система маркерів ArUco є ефективним інструментом для створення референцної координатної сітки. Маркери ArUco – це двовимірні бінарні шаблони, які мають унікальні ідентифікатори та легко визначаються навіть при зміні масштабу або кута зйомки (рис. 1.9).

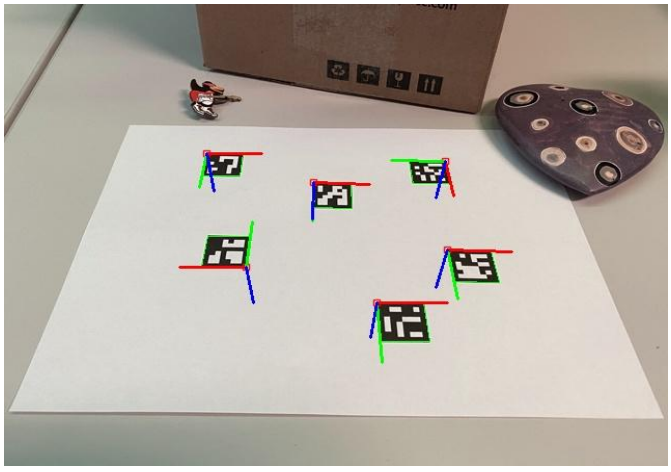


Рисунок 1.9 – Визначені маркери ArUco [18]

Кожен маркер це квадрат поділений на решітку з білих та чорних клітин, ці клітини в сукупності є бітовою матрицею маркера, певним

кодуванням завдяки якому можна визначити ідентифікатор маркеру. Існують спеціальні словники DICT_4x4_50, DICT_5x5_100, які визначають розмір бітової матриці маркеру та кількість унікальних маркерів в межах певного словника.

Для коректного виділення маркеру від фону є чорна смужка товщиною в одну клітинку по всьому периметру маркера, бітова матриця маркера знаходиться всередині цієї рамки.

Використовуючи маркери ArUco можна відновити реальні координати пікселя на зображенні через перспективну трансформацію. За використання такого методу, камеру можна розташовувати довільно, головне щоб маркери було чітко видно; також важливо друкувати маркери на матовій бумазі або уникати відблисків на маркері під час зйомки, оскільки це поскладнює визначення маркеру.

Найбільш популярною бібліотекою з реалізованими описаними алгоритмами та підтримкою роботи з маркерами ArUco є OpenCV (Open Source Computer Vision Library). Це відкрита бібліотека, дозволяє виконувати обробку зображень різними алгоритмами комп’ютерного зору. Бібліотека є кросплатформною, тобто може використовуватися на різних операційних системах; також є реалізації бібліотеки для різних мов програмування C++, Python, C# і тд.

1.4 Аналіз застосування штучного інтелекту на виробництві

Застосування штучного інтелекту на сьогодні зустрічається майже у всіх сферах нашого життя. В інтернет-магазинах нейронні мережі формують список товарів, які можуть зацікавити певного користувача відповідно до переглянутих ним товарів, пошуку в інтернеті, попередніх замовлень тощо. Платформи для прослуховування музики теж використовують таку технологію; на основі історичних даних нейронні мережі можуть робити прогноз вартості нерухомості і тд. Сучасне

виробництво не стало виключенням і все більше застосовується штучний інтелект для покращення процесів виробництва. На рисунку 1.10 показано приклад класифікації об’єктів на конвеєрній стрічці за допомогою моделі ШІ.

На виробництві ШІ може бути використаний для автоматизації обробки заявок клієнтів, класифікації відгуків клієнтів, автоматизації інвентаризації, визначення та класифікації дефектів тощо. Оскільки кожен з перелічених процесів відрізняється від інших, то існують різні типи моделей штучного інтелекту. Для автоматизації обробки заявок клієнтів типом є обробка документів, для класифікації відгуків клієнтів – класифікація категорій, для автоматизації інвентаризації – виявлення об’єктів [19].

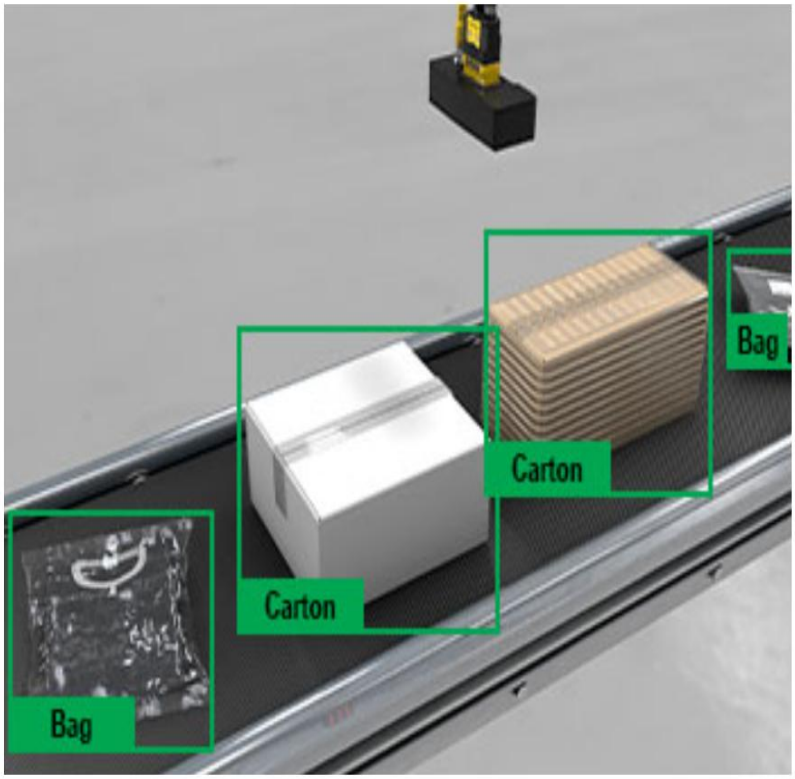


Рисунок 1.10 – Приклад конвеєрної стрічки з класифікованими об’єктами [20]

На рисунку 1.11 показано схему системи та її компонентів, що виконує перевірку якості пігулок при пакуванні. Представлений алгоритм може використовуватися не лише для перевірки дефектів, а й для виконання класифікації пігулок за типом, формою, розмірами і тд.

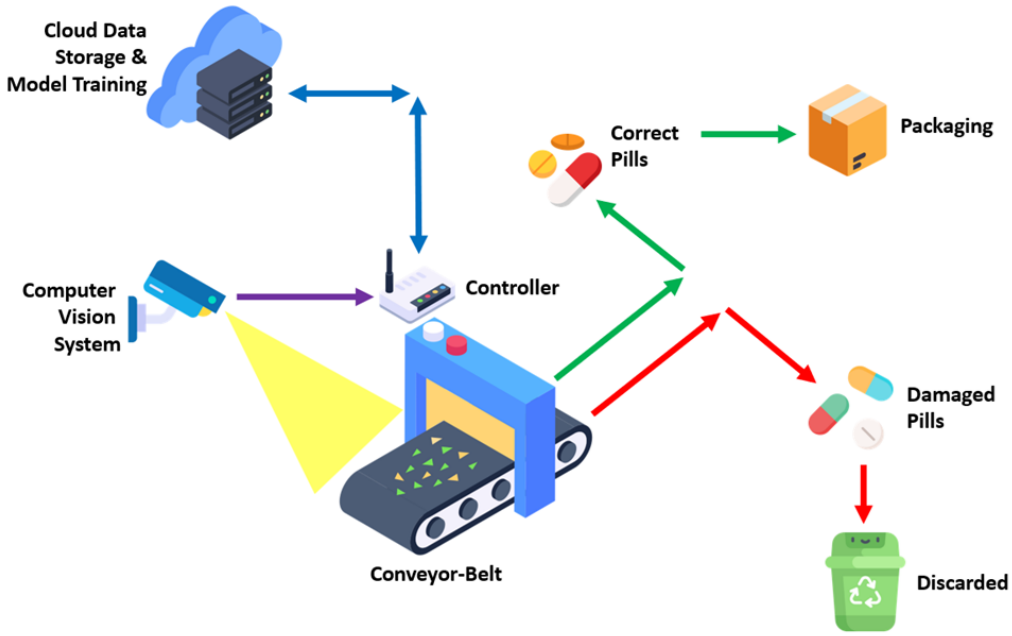


Рисунок 1.11 – Схема системи перевірки якості пігулок [21]

Перед використанням моделі штучного інтелекту необхідно провести навчання на основі маркованих відповідно до класифікації зображень, рекомендовано використовувати велику кількість зображень для навчання, щоб модель могла визначати об’єкти з високою точністю. У випадку з пігулками, може бути зазначено дві класифікації – пошкоджена пігулка та стороння речовина, зібравши набір зображень пігулок із зазначених класифікацій, проводиться навчання моделі. В результаті отримано файл моделі ШІ, який можна розгорнути на необхідній машині та використовувати для проведення класифікацій пігулок (рис. 1.12) [21].

Використання ШІ на виробництві здатносте прискорити обробку об’єктів для сортування, оскільки нейронні мережі здатні обробляти великі обсяги даних, адаптуючись до нових ситуацій без потреби ручного

програмування, та автоматизувати складні завдання, які раніше вимагали участі людини.

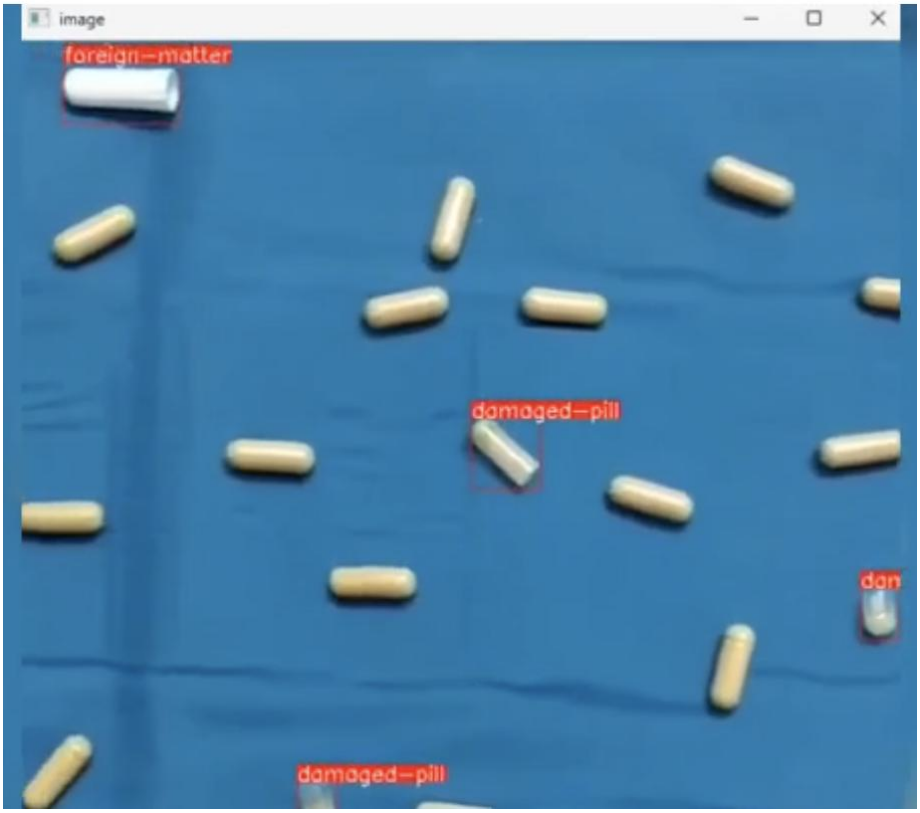


Рисунок 1.12 – Результат визначення пігулок з дефектами [21]

На відміну від класичних алгоритмів, ШІ-моделі, здатні розпізнавати об’єкти за нечіткими ознаками, враховувати контекст. Це дає змогу підвищити продуктивність, знизити кількість браку, оперативно впроваджувати нові класи виробів без змін у логіці керування та адаптувати систему до змін умов –наприклад, зміни освітлення, орієнтації об’єктів чи появи нових типів продукції.

1.5 Постановка задачі

Оскільки метою кваліфікаційної роботи є розробка автоматизованої системи керування роботом-маніпулятором, яка здійснює сортування об’єктів на основі алгоритмів комп’ютерного зору та рішень моделі

штучного інтелекту, необхідно виконати наступні завдання для досягнення поставленої мети:

– реалізувати зчитування відеопотоку з камери, встановленої над робочою поверхнею, та забезпечити його передачу до обчислювального модуля;

– здійснити детекцію ArUco-маркерів у межах зображення для побудови координатної системи на робочій поверхні та визначення реальних координат об’єктів;

– розробити алгоритм виділення об’єктів на зображенні та передання їх фрагментів до нейронної мережі;

– розробити модуль для створення та навчання моделі ШІ для класифікації об’єктів за заданими класами;

– реалізувати логіку керування роботом-маніпулятором на основі координат і класу об’єкта з метою переміщення його в задану зону;

– забезпечити інтеграцію всіх програмних та апаратних компонентів у єдину функціональну систему.

2 ПРИНЦИП РОБОТИ АВТОМАТИЗОВАНОЇ СИСТЕМИ

Перед розробкою системи виконується постановка задачі, формується технічне завдання, збирається комплект конструкторських документів. До конструкторських документів, що надається перед розробкою рішення можуть належати схемна документація, що включає в себе схеми різних типів, таких як структурна, функціональна, принципова тощо.

2.1 Розробка структурної схеми взаємодії компонентів системи

Структурна схема – це графічне зображення архітектури системи, де кожен її блок відображається графічним елементом, стрілками показано взаємні зв’язки між блоками. Блоками можуть виступати пункти управління, обчислювальні машини, щити тощо.

На рисунку 2.1 показано структурну схему автоматизованої системи сортування об’єктів з використанням комп’ютерного зору та штучного інтелекту.

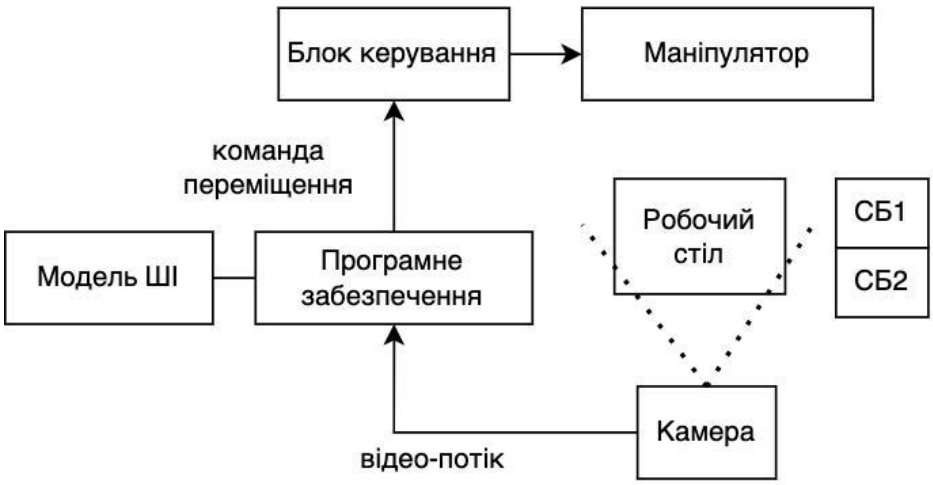


Рисунок 2.1 – Структурна схема системи

На рисунку 2.1 показано іменовані блоки елементів системи, де камера виконує захоплення кадрів робочого столу, для формування відео-потоку, що зчитується та обробляється програмним забезпеченням. Програмне забезпечення використовується для контролю та налаштування параметрів сортування об’єктів. За командою оператора виконується аналіз отриманого від камери зображення та визначення параметрів сортування, які оператор має підтвердити або відкоригувати. Після підтвердження параметрів сортування формуються та надсилаються команди для маніпулятора, який виконує переміщення об’єктів з робочого столу до відповідних сортувальних боксів (на рисунку 2.1 позначено як «СБ1» та «СБ2»).

Програмне забезпечення (ПЗ) структурно складається з декількох незалежних модулів. До цих модулів належать наступні:

– модуль розпізнавання зображення;

– модуль штучного інтелекту;

– модуль керування роботом;

– інтерфейс користувача;

– конфігураційний модуль.

Кожен модуль містить в собі ряд файлів з реалізацією певної логіки. Модуль розпізнавання зображення забезпечує підключення до відеокамери, отримання кадрів від камери, визначення ArUco маркерів, локалізацію об’єктів на зображенні робочої зони, визначення їх координат у реальному масштабі. Для реалізації модуля використовується бібліотека OpenCvSharp4 – це офіційна C#-обгортка над C++ бібліотекою OpenCv, яка надає повний набір функцій комп’ютерного зору.

Модуль штучного інтелекту призначено для створення, навчання, збереження, завантаження моделей штучного інтелекту; виконання класифікації об’єкту на зображенні за допомогою попередньо навченої моделі. Для реалізації модуля використовуються бібліотеки ML.NET та

TensorFlow, які забезпечують повний цикл роботи з моделями машинного навчання: підготовку, тренування, експорт і тд.

TensorFlow – це безкоштовна бібліотека програмного забезпечення з відкритим кодом для машинного навчання та штучного інтелекту. TensorFlow можна використовувати в багатьох мовах програмування таких як Python, C++, Java, C# та інші. Ця гнучкість дозволяє використовувати бібліотеку для написання програм в різних сферах [22].

Модуль керування роботом забезпечує управління роботизованим маніпулятором через G-коди, що передаються по послідовному порту. Модуль відповідає за формування та передачу команд до маніпулятора та контроль його стану. Передача команд виконується у вигляді текстових інструкцій.

Конфігураційний модуль забезпечує централізоване зберігання та доступ до параметрів, які необхідні для всіх модулів ПЗ. Для збереження даних використовується XML-файл, це дозволяє змінювати конфігурацію без зміни вихідного коду програмного забезпечення. Також, в модулі реалізовано можливість внесення змін з боку коду в процесі роботи.

Для реалізації інтерфейсу користувача перевага надається кросплатформеному фреймворку (це дозволить ПЗ працювати на різних операційних системах) AvaloniaUI з використанням патерну проектування MVC.

Патерн проектування – це шаблон, який заснований на загальних принципах та архітектурних рішеннях, які можна використовувати в різних ситуаціях. Тож патерн проектування допомагає програмісту проектувати програми більш організовано, роблять код перевикористовуваним і спрощують підтримку системи [23].

Патерн проектування MVC має три основні складові – Model, View, та Controller, які відповідають за бізнес-логіку, графічний інтерфейс та дані програми відповідно. В Model знаходиться основна логіка програми, проведення обчислень, комунікація з базою даних тощо. View – це

інтерфейс користувача, візуальні компоненти з якими користувач може взаємодіяти. Controller є прошарком між View та Model, він реагує на зміни цих двох шарів [24].

2.2 Алгоритм обробки зображень і визначення координат об’єктів

Одним з основних етапів сортування об’єктів маніпулятором за використання комп’ютерного зору та ШІ є обробка зображення, визначення об’єктів та їх координат на робочій поверхні. Для чого відведено окремий модуль в програмному забезпеченні. Цей етап матиме в загальному алгоритм дій як показано на рисунку 2.2.

Рисунок 2.2 – Алгоритм обробки зображення перед процесом сортування

Після запуску аналізу відбувається запит до камери, яка повертає JPEG-зображення. Отримане зображення треба декодувати у формат зрозумілий для OpenCV – Mat.

Mat – це двовимірний масив пікселів, де кожен піксель – це структура, яка зберігає значення кольору. Цей стандартний тип дозволяє зберігати зображення у вигляді матриці пікселів. Використання цього типу для збереження зображень дозволяє використовувати весь необхідний функціонал бібліотеки – від конвертації кольорів до пошуку контурів. На рисунку 2.3 показано як «бачитиме» цифровий пристрій автомобільне дзеркало, жовта таблиця – це значення кожного пікселя в відтінках сірого кольору [25]. Кожне число відповідає яскравості одного пікселя, з такою матрицею надалі виконується обробка зображення.

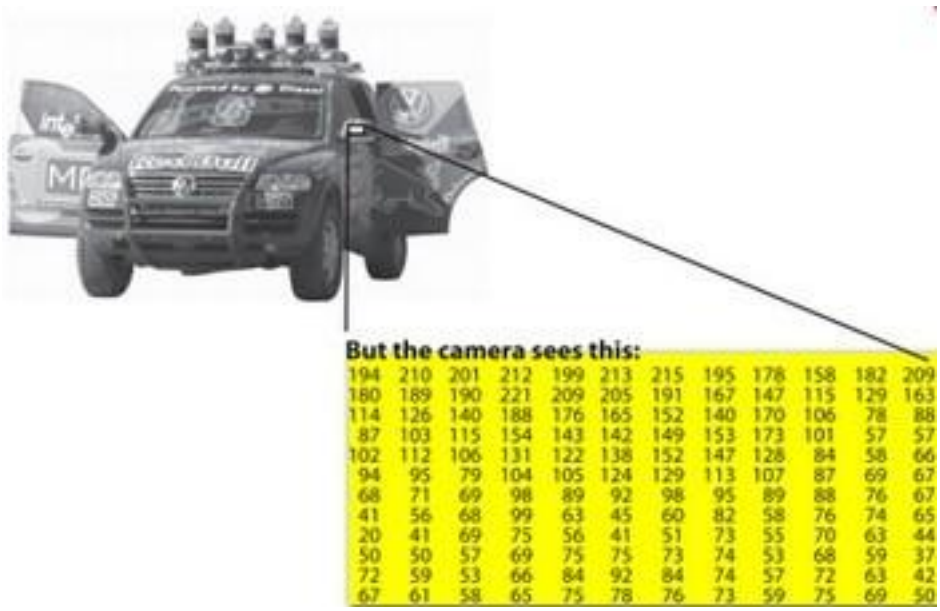


Рисунок 2.3 – Матриця пікселів зображення [25]

Робоча поверхня завжди має містити розміщені в кутах маркери ArUco, наявність чотирьох таких маркерів дозволяє точно визначити координати об’єктів, їх розміри. Використовуючи онлайн генератор, можна згенерувати декілька маркерів, роздрукувати їх та приклеїти до поверхні.

При генерації маркеру необхідно вказувати словник. Словник – це набір маркерів, які розглядаються в певній програмі. Основними властивостями словника є його розмір та розмір маркеру. Розмір словника – це кількість маркерів, які складають словник. Розмір маркера – це кількість бітів маркера. Таким чином, словник DICT_6X6_250 складається з 250 маркерів, розмір маркерів 6x6 біт [18]. В реалізації проекту використовуватиметься словник DICT_4X4_50, фізичний розмір такого маркеру складатиме 40x40 мм. На рисунку 2.4 показано робочу поверхню з визначеними маркерами ArUco.

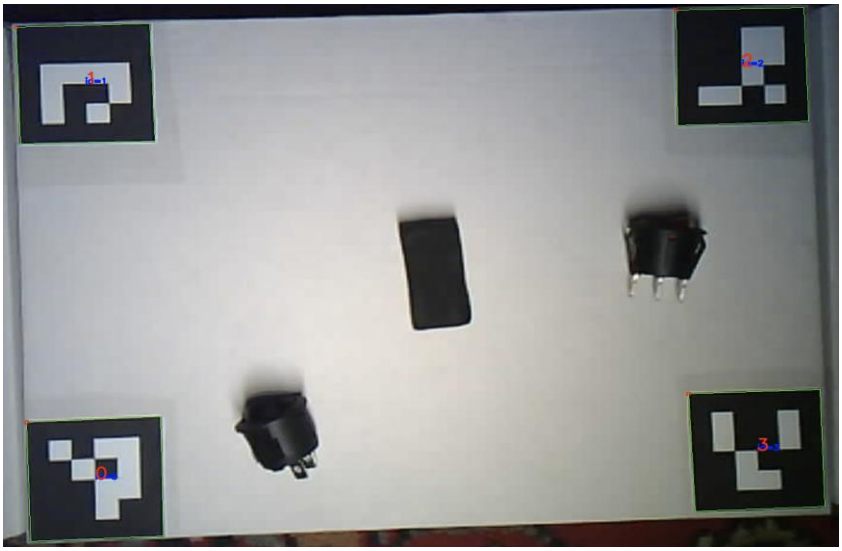


Рисунок 2.4 – Визначені маркери ArUco на зображені

В проекті центри маркерів ArUco виступатимуть опорними точками, ці точки визначають, де на зображені починається та закінчується область з якою треба працювати. Визначивши центра маркерів, треба встановити їх з реальними розмірами робочої поверхні (шириною та висотою), тоді кожен маркер буде позначати один з чотирьох кутів прямокутника. Для побудови перетворення встановлюється послідовність в якій точки зіставляються. Тобто, маркер у нижньому лівому куті – це лівий нижній кут прямокутника; маркер у верхньому лівому – лівий верхній кут прямокутника і тд.

Визначивши вихідні та цільові точки, визначається гомографія (проєктивне перетворення) – це відображення між проєктивними просторами, що зберігає інцидентність, тобто прямі переходять у прямі, точки на прямій залишаються на прямій, але паралельність, кути та довжини не зберігаються. У двовимірному випадку це перетворення набуває наступного вигляду [26]:

$$x' = Hx, \tag{2.1}$$

де $x = \begin{matrix} x \\ y \\ 1 \end{matrix}$ – однорідні координати точки у вхідному просторі,

$x' = \begin{matrix} x' \\ y' \\ w' \end{matrix}$ – однорідні координати відповідної точки у вихідному просторі,

$H \in R^{3\,x\,3}$ – матриця гомографії (невироджена, тобто det(H) ≠ 0), що визначена з точністю до ненульового множника.

Матриця H має наступний вигляд [26]:

$$H = \begin{matrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{matrix}. \tag{2.2}$$

В декартових координатах, після переходу з однорідних координат ($x' = \frac{x'}{w'}, y' = y'/w'$):

$$x' = \frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}},\ y' = \frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}}. \tag{2.3}$$

Матриця H має 9 параметрів, але з точністю до масштабного множника, тобто лише 8 незалежних ступенів вільності. Тому для

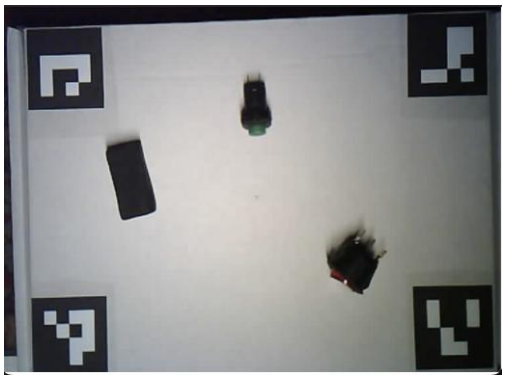
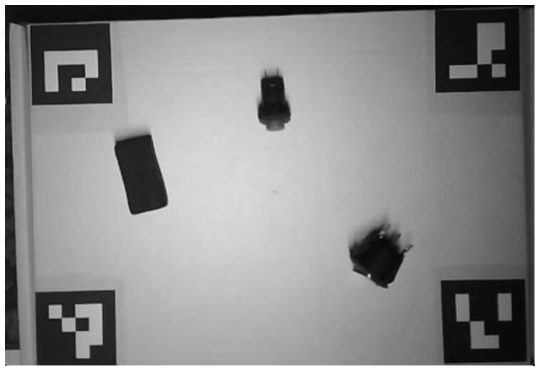
однозначного визначення гомографії необхідно мінімум 4 відповідні пари точок (по 2 координати кожна), що не лежать на одній прямій [26].

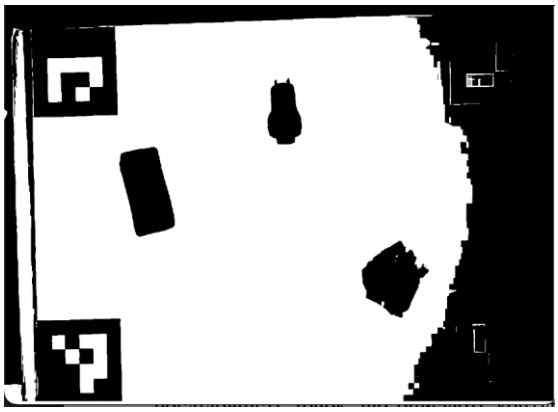
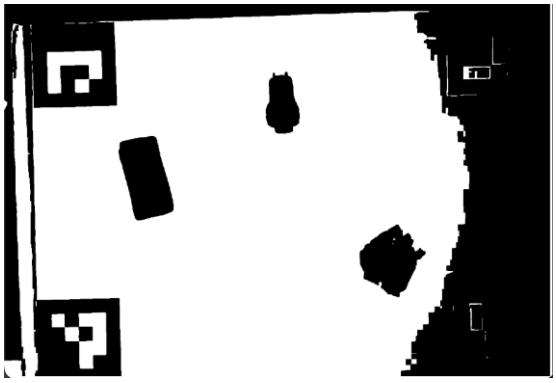
Оскільки, в проекті використовується робоча поверхня світлого та однорідного кольору, для визначення контурів об’єктів достатньо застосувати бінарну маску. Перетворивши зображення (рис. 2.5, а) у відтінки сірого (рис. 2.5, б) та використавши певний поріг виконується відділення об’єкту від фону. Кожен піксель зображення перевіряється відповідно до порогового значення, якщо піксель достатньо яскравий або темний, то він позначається як об’єкт, стає білого кольору; якщо ж піксель не пройшов порогове значення, то пізнається чорним кольором, тобто стає фоном. В результаті формується чорно-біле зображення, де об’єкт виділено білим кольором (рис. 2.5, в). Щоб прибрати дрібні плями всередині білих областей застосовуються морфологічні операції. Застосувавши морфологічну операцію закриття малі чорні отвори в білих ділянках зникнуть, що зробить контури чіткішими (рис. 2.5, г).

Отримавши чисту бінарну маску можна проводити аналіз об’єктів, для цього виконується «проходження» по краю кожної плями і фіксується послідовність точок, що описують контур плями. Отримавши список контурів всіх білих плям, проводиться фільтрація, щоб замалі контури, які не можуть бути об’єктами не враховувати в подальшому аналізі. Фільтрація може бути за розмірами або розташуванням.

В проекті треба виконати фільтрацію за площею та розміщенням, щоб виключити шуми та ділянки, що знаходяться занадто близько до краю та маркерів ArUco.

Перетворення координат у фізичну систему дозволяє перевести положення об’єкта, визначене у пікселях на зображенні, у реальні координати на площині робочої поверхні.

а)  б) 

в)  г) 

а) оригінал; б) відтінки сірого; в) порогова обробка; г) морфологічні операції

Рисунок 2.5 – Покрокова обробка зображення

Перспективна корекція – це процес відновлення прямокутної або фронтальної проекції об’єкта на зображенні, що було отримане з нахиленого ракурсу.

Така трансформація усуває спотворення, які можуть бути спричинені перспективою, та дозволяє «вирівняти» зображення так, наче воно було зняте перпендикулярно до площини об’єкта [26]. Цей етап виконується для того, щоб робот-маніпулятор міг точно направитися до об’єкту для виконання його переміщення. Приклад визначених координат об’єкту за використання перспективної корекції показано на рисунку 2.6.

Перспективна корекція – важлива операція в обробці зображень, вона дозволяє усунути перспективні спотворення та отримати фронтальний вигляд об’єкта [26]. Вона дозволяє врахувати викривлення через нахил камери, зміщення центру та неідеально умови розміщення.

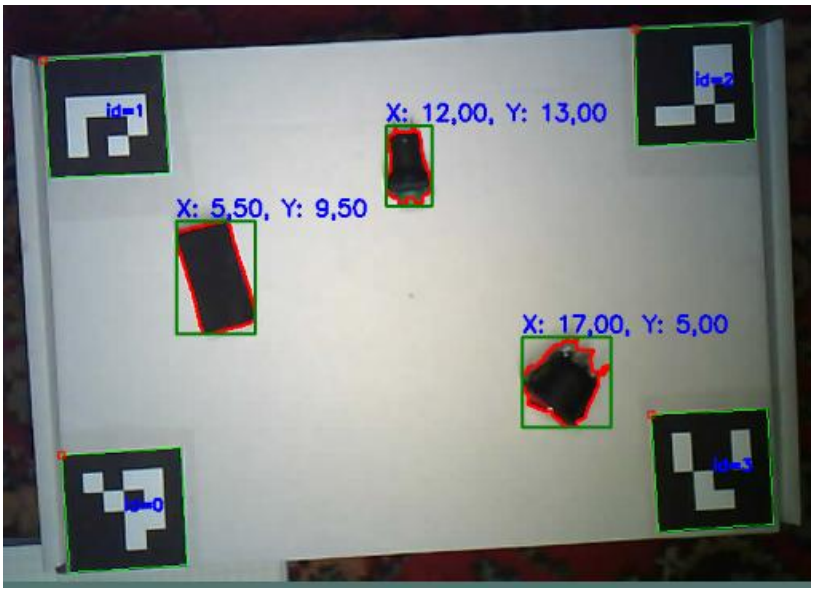


Рисунок 2.6 – Приклад анотованого зображення з контурами та підписом координат

Перспективна корекція, виконується за допомогою гомографії, описується наступним рівнянням [26]:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.4}$$

де (x, y) – координати точки на спотвореному зображенні,

$(\frac{x'}{w'}, \frac{y'}{w'})$ – координати тої самої точки після вирівнювання,

$H \in R^{3\,x\,3}$ – гомографічна матриця,

w’ – коефіцієнт однорідності, що враховує масштабні та перспективні спотворення [26].

Кожен знайдений об’єкт має бути вирізаний із загального зображення та збережений у відповідній директорії для подальшої класифікації, що виконується за використання ШІ.

Для відображення оператору інформації про результати обробки зображення, знайдених об’єктів та їх параметрів, треба сформувати відвідний список. Список має включати ідентифікатор об’єкту

(формується відповідно до порядку обробки об’єктів), координати центру об’єкту у фізичній площині та площу контуру об’єкта.

Описаний алгоритм забезпечує точну детекцію положення об’єктів на основі ArUco-маркерів з використанням перспективної корекції. Застосовуючи попередню обробку, фільтрацію та перетворення координат, можна отримати стабільні та масштабовані результати для подальшої класифікації та переміщення об’єктів роботом-маніпулятором.

2.3 Логіка класифікації та правила сортування об’єктів

Класифікація об’єктів виконується за рішенням попередньо натренованої моделі ШІ. Згідно з отриманим класом система приймає рішення щодо кінцевої зони сортування (сортувального боксу), куди повинен перемістити об’єкт робот-маніпулятор.

2.3.1 Загальна структура процесу класифікації та сортування

Алгоритм класифікації об’єктів виконується після обробки зображення робочої поверхні, визначення об’єктів та їх координат. На етапі обробки зображення робочої поверхні зображення кожного об’єкту врізається із загального.

Зображення кожного об’єкту подається на класифікацію, як окремий файл. Класифікація здійснюється за допомогою попередньо навченої моделі ШІ. На основі визначеного класу об’єкту формується та надається команда роботу-маніпулятору. На рисунку 2.7 показано блок-схему класифікації та сортування об’єктів.

Рисунок 2.7 – Блок-схема класифікації та сортування об’єктів

2.3.2 Опис моделі ШІ

Навчання моделі ШІ класифікувати зображення дуже важкий та затратний процес, який вимагає тисячі та мільйони якісних навчальних даних для тренування моделі та велику кількість годин для його проходження, саме тому доцільно використовувати попередньо навчену модель, як основу для подальших тренувань під конкретне завдання. Таким чином можна значно скоротити час на тренування моделі та знизити потреби до машини на якій це виконується.

Попередньо навчені моделі вміють класифікувати зображення за тисячами категорій, але в проекті важливо зберегти можливість користувачеві назначати категорії, які модель має визначати і не використовувати ніякі інші. Так як попередньо навчена модель проходила навчання на великому наборі навчальних даних вона містить у собі ознаки зображення, необхідні для ідентифікації зображень, тому саме ці ознаки

застосовуються для навчання нової моделі з набагато меншою кількістю класів [27].

TensorFlow – це гнучка платформа з відкритим кодом для машинного навчання, яка була розроблена Google. Фреймворк використовує орієнтований граф вузлів і ребер для представлення обчислень, де вузли представляють операції або функції, а ребра представляють вхідні та вихідні потоки. TensorFlow розроблений як масштабований і гнучкий, що дозволяє працювати на широкому діапазоні пристроїв і в розподіленому середовищі. TensorFlow використовується в широкому спектрі реальних програм, таких як класифікація зображень, обробка природної мови, аналіз часових рядів, генеративні моделі, навчання з підкріпленням і виявлення аномалій [22].

Програми на базі TensorFlow можуть виконуватися практично на будь-якій зручній цільовій системі, включаючи локальний ПК, хмарний кластер, телефони iOS та Android, центральні та графічні процесори. Моделі, згенеровані TensorFlow, можна встановити майже на будь-якій машині, на якій вони будуть використовуватися для прогнозування [28]. Така гнучкість дозволяє в майбутньому легко поширювати проект та масштабувати його виходячи з потреб підприємств.

TensorFlow використовує графову архітектуру. Граф збирає та пояснює всі послідовні обчислення, виконані під час навчання. Усі обчислення на діаграмі виконуються шляхом зв'язування тензорів. Тензори складаються з вузла, а також з ребра. Вузол виконує математичну дію та генерує вихідні кінцеві точки. Ребра описують вхідні/вихідні з'єднання між вузлами [28].

Входи та виходи в TensorFlow називаються тензорами – це n-вимірні вектори або матриці, що представляють усі форми даних. Кожне значення в тензорі має однаковий тип даних та відому (або частково відому) форму. Розмірність матриць або масиву – це форма даних. Тензор може бути

отриманий з необроблених даних або з результату обчислення, яке представлено на рисунку 2.8 [28].

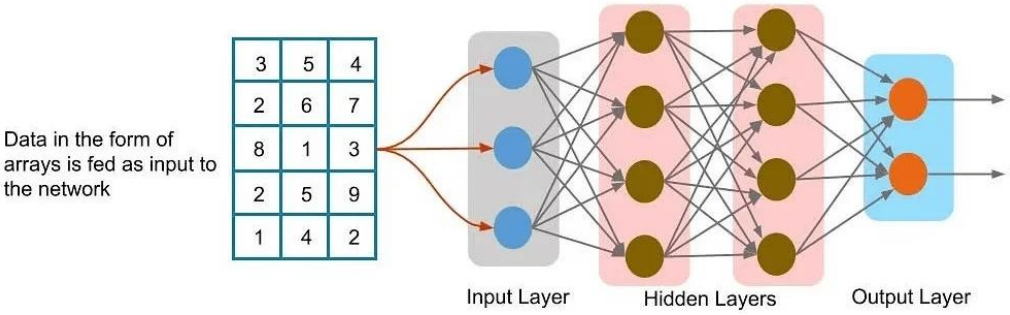


Рисунок 2.8 – Дані, збережені в тензорі, перетворені на результат [22]

Вхідні дані проходять через низку процедур, перш ніж стати виходом. Наприклад, на вхідні дані ми отримуємо велику кількість чисел, що вказують на біти зображення, а на виході отримуємо текст, наприклад, «це собака» [28].

Використовуючи бібліотеку ML.NET можна посилатися на вбудовану TensorFlow бібліотеку, яка дозволяє писати код, що завантажує уже існуючий TensorFlow файл навченої моделі. На основі якої буде створено нову модель [27].

Використовуючи попередньо навчену модель TensorFlow, для достагнення поставленого завдання, необхідно реалізувати наступний функціонал:

– створення моделі;

– тренування моделі;

– збереження моделі на пристрої;

– завантаження існуючої моделі для використання;

– використання моделі для класифікації зображення.

Алгоритм створення та тренування моделі на основі попередньо навченої моделі показано на рисунку 2.9.

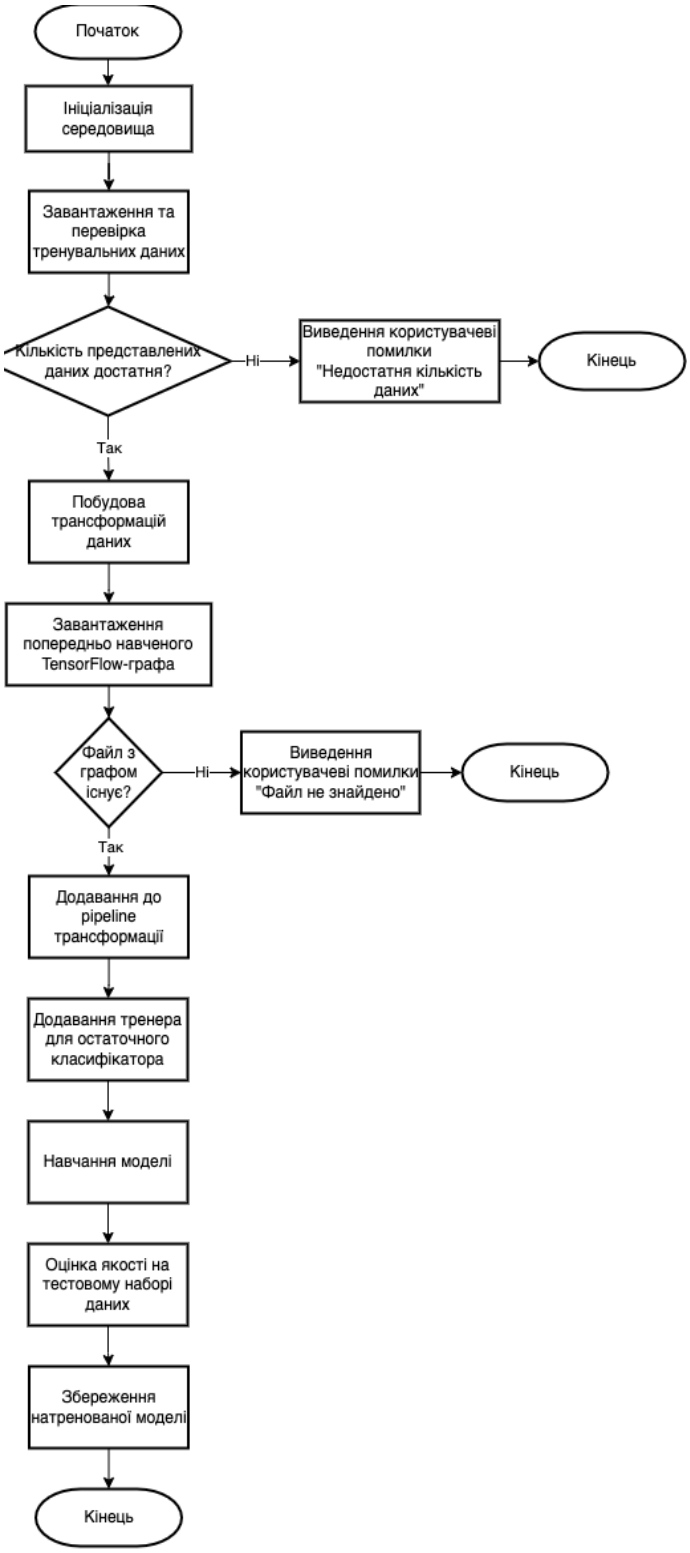


Рисунок 2.9 – Алгоритм створення та тренування моделі

Алгоритм класифікації зображення за використання моделі показано на рисунку 2.10. Шлях до директорії куди має бути збережено створену модель визначено в конфігураційному файлі.

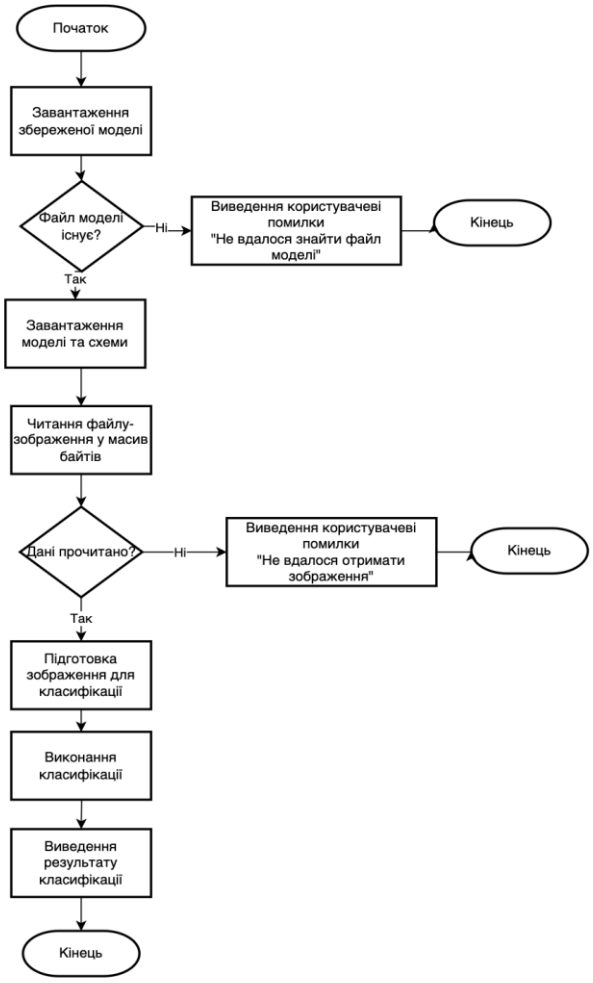


Рисунок 2.10 – Алгоритм використання моделі для класифікації зображення

2.3.3 Зв’язування класів із зонами сортування

Після проведення класифікації об’єктів необхідно визначити, куди повинен бути переміщений об’єкт. Користувач має обрати сортувальний бокс для кожного знайденого класу. Всього є чотири сортувальні бокси, що розміщені біля робочої поверхні, один з них має бути універсальним. В універсальний сортувальний бокс буде переміщено об’єкти, які було класифіковано з недостатнім відсотком впевненості, передбачається, що такі об’єкти надалі буде оглянуто людиною. Координати боксів та відсоток впевненості класифікації задаються в конфігураційному файлі.

2.4 Алгоритм руху маніпулятора та управління захватом

Керування роботом-маніпулятором відбувається за використання G-коду. З'єднання з роботом має бути через послідовний порт. Для виконання переміщення прошивка робота має підтримувати наступні команди:

– калібрування робота;

– лінійне переміщення на задані координати;

– закриття/відкриття захвату;

– відображення поточних координат робота.

На рисунку 2.11 показано алгоритм перенесення об'єкту з початкової позиції до сортувального боксу.

Модуль розпізнавання зображення визначає координати об'єкта на столі в системі координат робочої поверхні та, в подальшому, передає їх роботу-маніпулятору для виконання переміщення, але ці координати мають бути переведені в систему координат робота.

Система координати робочої поверхні по осі Xs рухається від нижнього лівого краю (від маркеру, що розташований знизу ліворуч) вправо, Ys від нижнього лівого краю вгору. Розміри робочої поверхні W – ширина, H – висота. Центр робочої поверхні (X_{sc}, Y_{sc}) визначається наступними формулами:

$$X_{sc} = \frac{W}{2}, \quad Y_{sc} = \frac{H}{2}. \tag{2.5}$$

Координати центрів маркерів ArUco визначаються за наступними формулами:

– лівий нижній маркер (id = 0):

$$\left(\frac{m}{2}, \frac{m}{2}\right); \tag{2.6}$$

– лівий верхній маркер (id = 1):

$$\left(\frac{m}{2}, H - \frac{m}{2}\right); \tag{2.7}$$

– правий верхній маркер (id = 2):

$$\left(W - \frac{m}{2}, H - \frac{m}{2}\right); \tag{2.8}$$

– правий нижній маркер (id = 3):

$$\left(W - \frac{m}{2}, \frac{m}{2}\right), \tag{2.9}$$

де m – сторона маркеру.

Початкова позиція робота – це координати робота після виконання операції калібрування. Робот має бути розміщений так, щоб в початковій (домашній) позиції захват робота знаходився над центром робочої поверхні. Початкові координати робота позначимо як X_{rh}, Y_{rh}, Z_{rh}.

Зсув по осях буде наступним:

$$X_{ofs} = X_{rh} - X_{sc}, \tag{2.10}$$

$$Y_{ofs} = Y_{rh} - Y_{sc}. \tag{2.11}$$

Відповідно з цими формулами можна розрахувати координати в системі робота за координатами в системі робочої поверхні:

$$X_{rob} = X_s + X_{ofs}, \tag{2.12}$$

$$Y_{rob} = Y_s + Y_{ofs}. \tag{2.13}$$

За наведеними формулами можна перевести координати об’єкту з однієї системи координат в іншу та направити до модуля керування маніпулятором для виконання операції переміщення об’єкту.

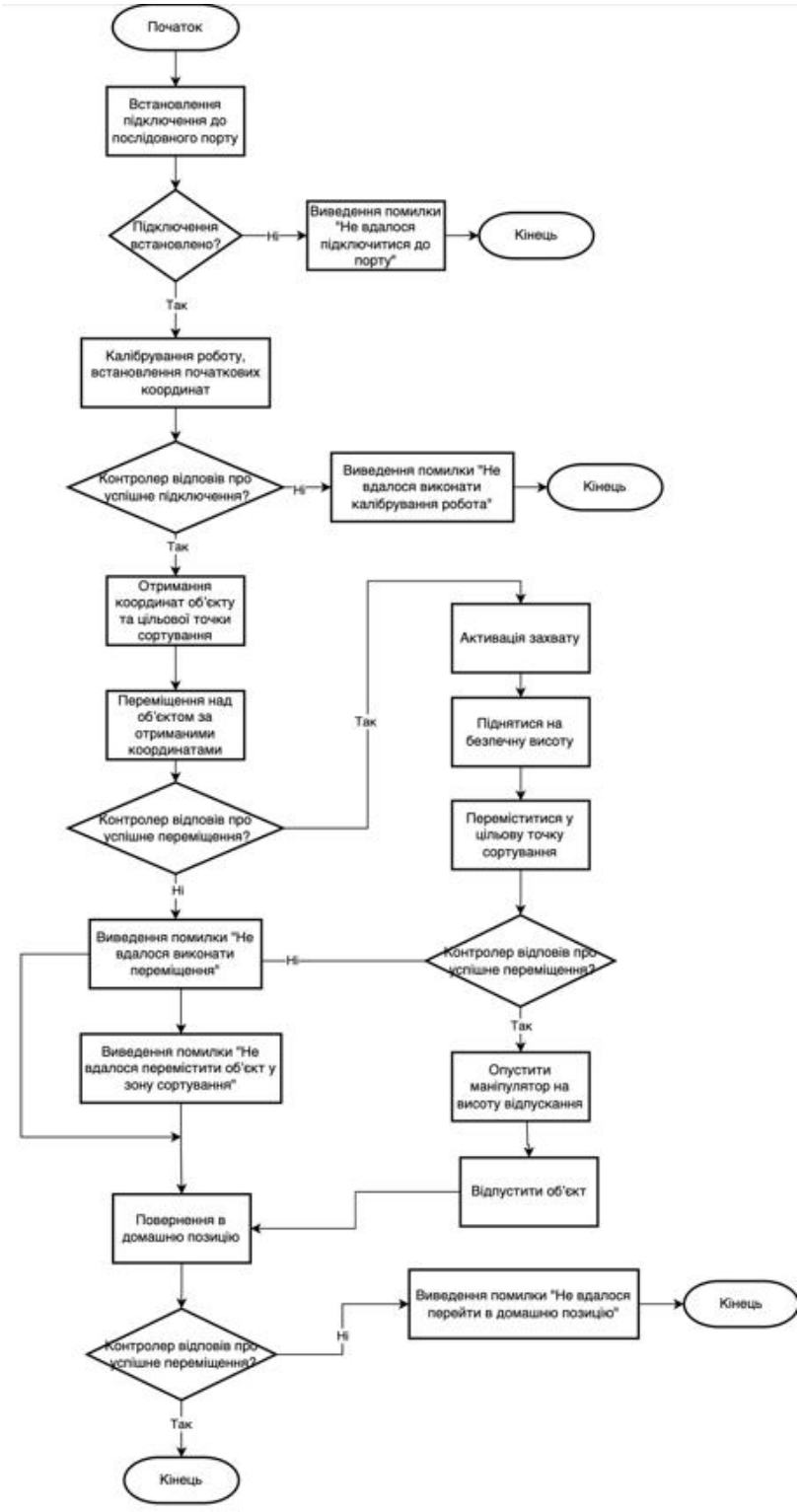


Рисунок 2.11 – Алгоритм перенесення об’єкту роботом-маніпулятором

3 ВИБІР КОМПОНЕНТІВ СИСТЕМИ

Для демонстрації роботи описаної системи з використанням системи комп’ютерного зору та елементами ШІ необхідно створити макет, який включатиме в себе: робот-маніпулятор, камеру та робочу поверхню. Аналіз існуючих пристроїв дозволить визначити, які пристрої та їх параметри найбільш пріоритетні для поставленої задачі.

3.1 Вибір камери та її параметрів

Модуль ESP32-CAM було обрано для реалізації потокового відео з камери через Wi-Fi. До плати ESP32-CAM потрібна камера, обрано OV2640, яка має такі параметри:

– 2-мегапіксельний сенсор;

– роздільну здатність: UXGA 1622x1200;

– формати відео:YUV422, YUV420, RGB565, RGB555 і 8-бітні стислі дані;

– фіксований кут огляду 62°;

– живлення 5В, споживання в активному режимі до 250 мА;

– розміри: 40x27 мм;

– швидкість передачі зображення: від 15 до 60 кадрів в секунду [29].

Підключається камера до плати з допомогою флекс-кабелю (шлейф) в FPC-роз'єм з 24-ма контактами (рис. 3.1) [29].

Використовується стандартна прошивка ESP-Camera, доопрацьована для автоматичного перезапуску при втраті зв’язку і налаштування через веб-інтерфейс. Прошивка включає HTTP-сервер для відеопотоку та API для отримання одного кадру у форматі raw або JPEG.

Рисунок 3.1 – Підключення камери до плати ESP32-CAM [29]

3.2 Вибір робота-маніпулятора та контролера керування

Для збирання макету автоматизованого модуля сортування було обрано відриту конструкцію 3D-друкованого маніпулятора «Community Robot Arm» від 20sffactory. Цей робот складається з трьох лінійних осей X, Y та Z, які приводяться до руху за використання крокових двигунів NEMA 17. В таблиці 3.1 наведено основні характеристики рокового двигуна [30].

Таблиця 3.1 – Основні характеристики крокового двигуна NEMA 17

Характеристика	Значення
Кількість фаз	2
Крок кута	1.8 град.
Температурний режим	до 80°C
Робоча температура довкілля	Від -20°C до +50°C
Вага	450 г
Момент утримання	мін. 70 Н·см

Документація до конструкції робота передбачає вибір, який контролер використовувати, представлено можливість використовувати Arduino або ESP32. Як контролер управління було обрано обрано плату Arduino Mega 2560, також використано насадку на плату Arduino Mega – RAMPS 1.4, яка має дискретні драйвери крокових двигунів (A4988). Порівняння контролерів, доступних для використання в представленій конструкції наведено в таблиці 3.2 [31 - 33].

Таблиця 3.2 – Характеристики контролерів

Характеристика	Arduino Mega 2560	Arduino Uno	ESP32 Wemos D1 R32
Флеш-пам’ять	256 Кб	32 Кб	4 Кб
Аналогові входи	16 шт	6 шт	6 шт
Цифрові входи/виходи	54 шт	14 шт	11 шт
Робоча напруга	5В - 12В	5В - 12В	5В
Тактова частота	16 МГц	16 МГц	160/240 МГц
Розмір	101.6 х 53 х 34 мм	68 х 53 х 15 мм	140 х 42 х 20 мм

Конструкція робота переважно складається з пластикових деталей, надрукованих на 3D-принтері, з пластику PLA. Після друку всіх деталей, необхідно перевірити якість друку, щоб переконатися, що деталі мають високу якість та не деформуються під час роботи маніпулятора. Збірка маніпулятора виконується за інструкцією відповідно обраному контролеру. Маніпулятор у готовому вигляді показано на рисунку 3.2.

Прошивка використовується та, що представлена в проекті, за яким виконувалась збірка робота – robotArm_v0.81. Основним функціоналом прошивки є:

– парсинг G-коду, прошивка підтримує такі команди, як G0, G1 (лінійні переміщення), G28 (калібрування), G92 (задання поточної позиції), M3/M5 (увімкнення/вимкнення серво-захвату);

– вибір кривої прискорення (flat, arctan, cosin) для плавності траєкторій;

– перевірка переповнення координат, запобігання перевищенню меж;

– виведення інформаційних повідомлень.

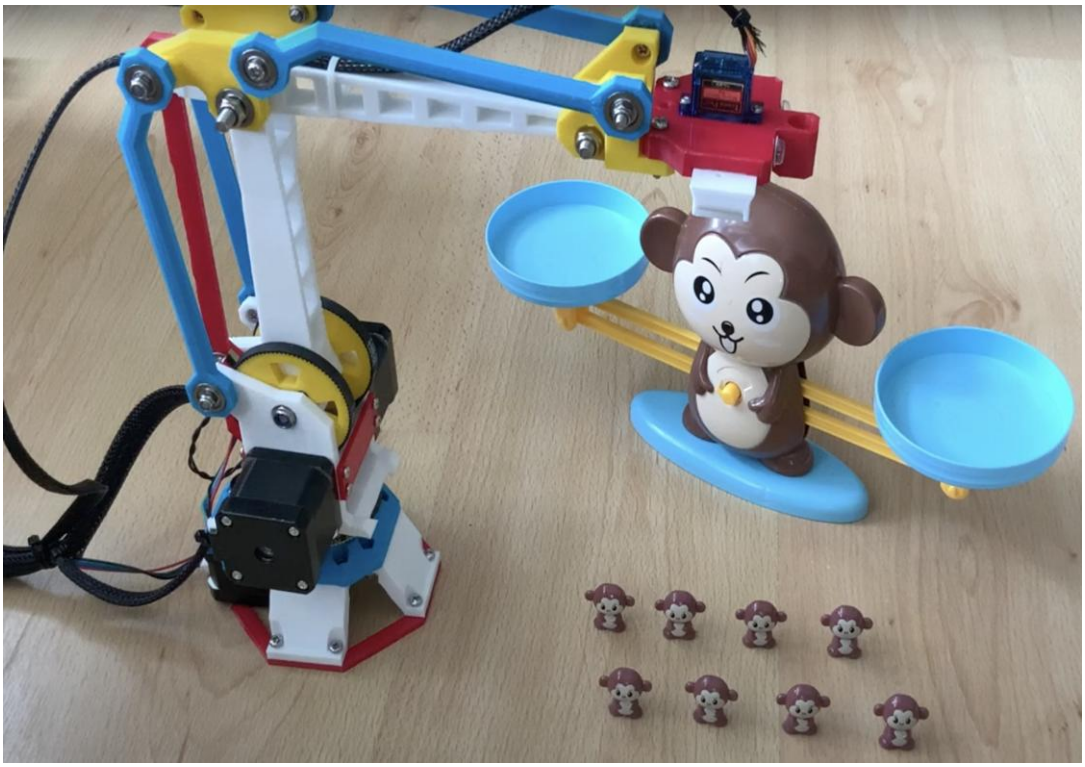


Рисунок 3.2 – Робот-маніпулятор [34]

Прошивка контролеру виконується в середовищі розробки Arduino IDE.

Поєднання такої апаратної та програмної частин забезпечують точність, достатню швидкість переміщення та гнучкість інтеграції з програмним забезпеченням системи.

3.3 Вимоги до робочої поверхні

Робоча поверхня напряму впливає на результати та процес роботи комп’ютерного зору та сортування. Саме на робочу поверхню спираються алгоритми комп’ютерного зору, калібрування ArUco-маркерів та

захоплення й перенесення об'єктів. Якщо робоча поверхня не відповідатиме певним критеріям, наприклад за розмірами, формою, кольором тощо, то це може призвести до зниження результатів точності координат, визначення об'єктів, а в подальшому і збоїв у роботі системи.

Тож вимоги до робочої поверхні, враховуючи алгоритми роботи комп'ютерного зору та робота-маніпулятора, будуть наступними, щоб гарантувати стабільність та коректні результати роботи системи:

– робоча поверхня має бути пласкою, прямокутною;

– матеріал має бути світло-сірим або білим за кольором, з матовим покриттям, щоб прибрати зайві відблиски поверхні та мати однорідну поверхню;

– у кожному куті має бути розташовано ArUco-маркер розміром 40х40 мм, вони мають бути зафіксовані, так щоб не зміщуватися під час роботи. ArUco-маркери теж мають бути матовими, щоб не було відблисків. Навколо контуру маркера має бути світла полоска ~3мм;

– конструкція має бути стабільною, треба забезпечити відсутність люфту та вібрацій;

– робоча поверхня має освітлюватися так, щоб світло рівномірно лягало на поверхню, щоб не було різких тіней;

– навколо робочої поверхні не має бути сторонніх предметів;

– перед проведенням процесу сортування необхідно провести візуальний огляд робочої поверхні та усунути невідповідності описаним вимогам, якщо такі виявлено.

Дотримання описаних вимог забезпечить точність детекції об'єктів та роботи системи в цілому.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Програмний стек та конфігурація проекту

Проект розробляється в середовищі розробки JetBrains Rider 2024.1 на базі .NET 8. Розробка виконується під операційною системою macOS Ventura 13.5.1. Мінімальні системні вимоги є наступними:

– процесор Intel Core i5;

– оперативна пам’ять від 8 Гб;

– вільне місце на накопичувачі (SSD/HDD) не менше 10 Гб;

– інтернет-зв’язок для завантаження залежностей, оновлень Rider і NuGet-пакетів.

Для реалізації проекту обрано мову програмування C# 12, у складі платформи .NET 8. Для структурування коду обрано патерн проектування Model-View-Controller (MVC). Інтерфейс користувача створення за використання фреймворку AvaliniaUI (версія 11.2.3).

Пакет NuGet – це ZIP-файл з розширенням .nupkg, який містить скомпільований код (DLL), інші файли, пов'язані з цим кодом, і описовий маніфест, що включає такі відомості, як номер версії пакета. Розробники, які мають код, до якого потрібно надати спільний доступ, створюють пакети та публікують їх на закритих або відкритих вузлах. Споживачі отримують ці пакети з відповідних вузлів, додають їх у свої проекти, а потім викликають функції пакета у коді свого проекту [36].

В проекті використано наступні NuGet пакети:

– Avalonia – базова бібліотека AvaloniaUI, яка надає кросплатформенний фреймворк для побудови графічного інтерфейсу;

– Avalonia.Controls.DataGrid – розширення для базової бібліотеки AvaloniaUI, яке додає компонент DataGrid, що використовується для відображення табличних даних;

– Microsoft.ML (4.0.1) – пакет ML.NET, який містить ядро платформи машинного навчання. З його допомогою можна створювати pipeline для обробки даних, налаштування трансформацій, тренування моделей;

– Microsoft.ML.ImageAnalytics (4.0.1) – розширення ML.NET для обробки зображень;

– Microsoft.ML.TensorFlow (4.0.1) – пакет для підключення та використання TensorFlow-моделі в ML.NET;

– Microsoft.ML.Vision (4.0.1) – API ML.NET для завдання transfer learning зображень, він забезпечує простий спосіб тренування власної мережі на основі попередньо навчених базових моделей;

– OpenCvSharp4 (4.10.0.20241108) – це .NET-обгортка для бібліотеки OpenCV, яка забезпечує всі стандартні методи для роботи з зображеннями, такі як конвертація у сірі відтінки, фільтрація, пошук контурів, детекція ArUco маркерів, побудова матриць перспективи, анотацію (малювання кіл, ліній, тексту);

– System.IO.Ports (9.0.3) – пакет, що надає клас SerialPort для роботи з послідовними COM-портами у .NET 6+;

– Emgu.CV (4.10.0.5680) – це .NET-обгортка для OpenCV (частково використовуються специфічні методи наприклад, конвертація між типами Mat і Bitmap для перегляду відео в AvaloniaUI).

4.2 Конфігураційний файл та модуль його обробки

Конфігураційний модуль призначено для читання, збереження та редагування конфігурації проекту RoboOptix. Конфігурація зберігається в XML файлі, в якому наведено ряд тегів для налаштування шляхів відносно яких будуть зберігатися певні дані, збереження класифікацій, налаштування адреси підключення до камери, розміри робочої поверхні, налаштування ШІ та координати сортувальних боксів. Модуль складається з трьох класів: XmlConfigReader, RoboOptixConfig та ConfigService.

Файл конфігурації має кореневий тег RoboOptix в якому розміщуються п’ять секцій відповідно до їх призначень:

– AI – містить вкладені теги для налаштування параметрів штучного інтелекту;

– Objects – містить список вкладених тегів Object кожний з яких описує класифікацію для штучного інтелекту;

– CameraIP – не містить вкладених тегів, значення цього тегу це IP адреса за якою виконується підключення до камери;

– Table – має два вкладені тега (Height та Width), які описують фізичні розміри робочої поверхні (висота та ширина, см);

– Boxes – містить список вкладених тегів Box з атрибутами, що описують сортувальний бокс.

Секція AI має наступний вигляд:

```
<AI>
<EnvPath>/Users/user/RiderProjects/RoboOptix.Desktop/assets/</EnvPath>

<ModelsPath>/Users/user/RiderProjects/RoboOptix.Desktop/assets/models/</Mod
elsPath>
<ImagesCount>5</ImagesCount>
<Model ConfidenceThreshold="50">
   <Name>baseModel</Name>
   <Description>This is base AI model</Description>
</Model>
</AI>
```

EnvPath – шлях до директорії в котрій знаходиться папки icons з іконками, що використовують в програмі, inception – зберігає Inception-модель у форматі .pb, яка використовується для побудови моделі під власний набір даних; ModelsPath – шлях до директорії models з моделлю ШІ та даними для тренування, тестування моделі ШІ; ImagesCount – мінімальна кількість картинок, яка має бути в тренувальному наборі для однієї класифікації; ConfidenceThreshold – атрибут тега Model, який містить значення мінімального порогу впевненості, який має бути для проведення успішної класифікації (результат класифікації нижче цього порогу вважається недостатнім); Name – вкладений тег тегу Model, який

зберігає назву моделі (назва відповідного файлу моделі ШІ); Description – опис моделі.

Секція Objects має наступний вигляд:

```
<Objects>
    <Object Id="f71e179e-a3ea-45c5-bf49-ce84be8206df" Status="Deleted"
Label="screw"      Path="screw/" />
    <Object Id="fedd708a-abad-407c-a7a4-eed773d3ff8d" Status="Trained"
Label="switch" Path="switch/" />
    <Object Id="26452b09-8348-4d25-a22c-fb81f4c3ed03" Status="Trained"
Label="button" Path="button/" />
 </Objects>
```

Тег Object описує клас та містить для цього ряд атрибутів: Id – ідентифікатор класу в системі, Status – статус класу (чи навчена модель визначати об’єкти цього класу), Label – назва класу, Path – шлях (відносно ModelsPath/training) до директорії в якій зберігаються зображення описуваного класу.

Навчальний набір даних – це сукупність вхідних даних (зображень) на яких проводиться навчання штучного інтелекту.

Статус класифікації (мітки) – це текстове позначення підготовленості та використання моделі ШІ для цього конкретного класу. Статус класифікації може мати наступні значення:

– Trained – модель ШІ навчена визначати об’єкт цього класу;

– Unknown – клас лише додано в систему, вказано його параметри;

– Added – клас додано до навчального набору;

– Delete – клас видалено з навчального набору;

– InTsv – проміжний статус, коли клас додано до навчання, але сам процес навчання ще не завершено та модель не може визначати об’єкти цього класу.

Секція CameraIP представляє рядок – тег та значення:

```
<CameraIP>http://192.168.0.104</CameraIP>
```

де http://192.168.0.104 – IP адреса для отримання відеопотоку з камери.

Секція Table має два вкладені теги для зберігання значень висоти та ширини робочої поверхні:

```
<Table>
      <Height>17.5</Height>
      <Width>24.3</Width>
</Table>
```

Значення Height – це фізична висота столу (в сантиметрах), Width – фізична ширина столу (в сантиметрах).

Секція Boxes має декілька вкладених тегів Box, що містять атрибути з описом сортувального боксу:

```
<Boxes>
    <Box Name="Box A" X="160" Y="144" Z="-40" />
    <Box Name="Box B" X="160" Y="224" Z="-40" />
    <Box Name="Default" X="-160" Y="134" Z="-60" />
</Boxes>
```

Атрибут Name – назва сортувального боксу (СБ), X – координата СБ по осі X в системі координат робота, Y – координата СБ по осі Y в системі координат робота, Z – координата СБ по осі Z в системі координат робота.

Для реалізації класів необхідно скористатися додатковими просторами імен, які наведено в таблиці 4.1.

Таблиця 4.1 – Додаткові простори імен для реалізації модуля конфігурації

Простір імен	Призначення
System.IO	Робота з файловою системою: перевірка наявності файлу, читання/запис XML через FileStream
System.Linq	LINQ-операції над колекціями (Any, FirstOrDefault, Last, тощо)
System.Xml.Serialization	Атрибути [XmlRoot], [XmlElement], [XmlAttribute] та серіалізація/десеріалізація XML

Серіалізація та десеріалізація – це процеси перетворення даних з одного формату в інший, зазвичай з об’єктів в рядок (серіалізація) та навпаки (десеріалізація). Це дозволяє зберігати або передавати дані в різних форматах, таких як текстові рядки, JSON, XML тощо [37].

Щоб покращити написання коду зі сторони його чистоти та подальшої розробки для секцій конфігурації зроблено окремі класи, котрі будуть покращувати роботу програміста. Таким чином кожен клас буде відповідати лише за свій набір властивостей та зміни в ньому не «зламають» іншу логіку програми; це значно спрощує валідацію даних, оскільки передбачається що не можна створити декілька класифікацій з однаковою назвою (міткою), ідентифікатором тощо; кожен клас декорується атрибутами [XmlElement] чи [XmlAttribute], тож можна легко відслідковувати відповідність між XML і моделлю в коді.

Клас RoboOptixConfig відповідає кореневому елементу RoboOptix і містить властивості, які збирають усі підрозділи конфігурації: AI, Objects, CameraIP, Table та Boxes:

```
[XmlRoot("RoboOptix")]
public class RoboOptixConfig
{
    [XmlElement("AI")]
    public AIConfig AI { get; set; } = new AIConfig();

    [XmlArray("Objects")]
    [XmlArrayItem("Object")]
    public List<ObjectConfig> Objects { get; set; } = new();

    [XmlElement("CameraIP")]
    public string CameraIP { get; set; }

    [XmlElement("Table")]
    public TableConfig Table { get; set; } = new TableConfig();

    [XmlArray("Boxes")]
    [XmlArrayItem("Box")]
    public List<BoxConfig> Boxes { get; set; } = new List<BoxConfig>();
}
```

Властивість AI має тип AIConfig, який у свою чергу читає поля EnvPath, ModelsPath, ImagesCount та вкладений об’єкт ModelConfig, що відображає параметри моделі – атрибут ConfidenceThreshold й елементи Name та Description.

Клас TableConfig відповідає розділу Table і зчитує числові властивості Height та Width, які задають фізичні розміри столу.

Для кожного сортувального боксу призначено окремий об’єкт типу BoxConfig, що зчитує атрибути Name, X, Y, Z, що описують назву боксу та його координати у відповідних осях.

Колекція об’єктів для класифікації реалізована як список ObjectConfig, де кожен екземпляр відображає атрибути Id, Status, Label та Path, тобто ідентифікатор, статус, мітка та шлях.

Для перетворення XML у ці класи використовується клас XmlConfigReader, який через XmlSerializer для типу RoboOptixConfig відкриває файл і викликає Deserialize(), гарантуючи, що кожна секція та атрибут конфігурації потрапляють у відповідні властивості об’єктної моделі.

Публічний клас XmlConfigReader містить один статичний метод LoadConfig. Цей метод має сигнатуру public static RoboOptixConfig LoadConfig(string filePath): він приймає одиночний параметр filePath типу string, що є шляхом до КФ, який необхідно прочитати.

Якщо за вказаним шляхом нема файлу, то кидається виключення FileNotFoundException:

```
if (!File.Exists(filePath))
   throw new FileNotFoundException($"Файл не знайдено: {filePath}»);
```

Потім, якщо файл існує, створюється XmlSerializer для типу RoboOptixConfig, відкривається файл у режимі FileMode.Open і викликає Deserialize:

```
// Створення XmlSerializer
XmlSerializer serializer = new(typeof(RoboOptixConfig));
// Читання й десеріалізація
using FileStream fileStream = new(filePath, FileMode.Open);
return (RoboOptixConfig)serializer.Deserialize(fileStream) ?? new
RoboOptixConfig();
```

У разі успіху метод повертає об’єкт RoboOptixConfig, а якщо під час читання або десеріалізації сталася будь-яка помилка, ловить її і викидає як новий Exception із повідомленням:

```
        // Обробка помилок
        catch (Exception ex) {    throw new Exception($"Помилка читання XML: {ex.Message}", ex);     }
```

Публічний клас ConfigService відповідає за завантаження, валідацію та редагування XML-конфігурації. Він має приватне readonly поле _filePath, в якому зберігається шлях до файлу; публічну властивість Config типу RoboOptixConfig з приватним сеттером, куди записується результат десеріалізації; а також властивість ErrorMessage типу string, яка містить текст помилки при невдалому завантаженні або зміні конфігурації. Публічна властивість IsValid повертає true, якщо Config не рівний null. Частковий вміст класу наступний:

```
        public class ConfigService {
            private readonly string _filePath;
            public RoboOptixConfig? Config { get; private set; }
            public string? ErrorMessage { get; private set; }
            // конструктор класу
             public ConfigService(string filePath)
            {
                _filePath = filePath;
                try
                {
                    Config = XmlConfigReader.LoadConfig(_filePath);
                }
                catch (Exception ex)
                {
                    Config = null;
                    ErrorMessage = ex.Message; // Зберігаємо текст помилки
                }
            }
            public bool IsValid => Config != null;
            //     ...     (методи     AddObject,     UpdateObject,     DeleteObject, UpdateObjectStatus, SaveConfig)
        }
```

Конструктор класу зберігає шлях до файлу в приватну змінну _filePath, завантажує конфігурацію через XmlConfigReader.LoadConfig(…). Якщо десеріалізація пройшла без помилок, Config міститиме екземпляр RoboOptixConfig, якщо ж сталася будь-яка помилка (наприклад, файл не знайдено або некоректний XML), то вона зловиться в блоці catch, Config буде null, а в ErrorMessage збережеться текст виключення.

Метод public bool AddObject(ObjectConfig newObject) приймає об’єкт типу ObjectConfig для додавання його до переліку класифікацій. Метод

виконує перевірку, що Config завантажено, а також що серед списку об’єктів Config.Objects немає іншого елемента з таким самим Label (без урахування регістру). Якщо перевірки пройшли, то новий об’єкт додається до списку, викликається метод SaveConfig() для збереження змін у файлі і метод повертає true. У разі помилки повертає false, а ErrorMessage містить пояснення.

Метод public bool UpdateObject(ObjectConfig updatedObject) приймає об’єкт із оновленими даними, шукає в списку об’єктів Config.Objects елемент з тим самим Id, перевіряє унікальність нового Label серед інших об’єктів, оновлює властивості Label, Path і Status у знайденому об’єкті та викликає SaveConfig(). При успіху повертає true, інакше – false з повідомленням у ErrorMessage.

Метод public bool DeleteObject(Guid objectId) приймає ідентифікатор об’єкта, переконується, що дані КФ завантажено, знаходить елемент з переданим Id у списку об’єктів і видаляє його. Після видалення викликає SaveConfig() і повертає true. Якщо об’єкт не знайдено або Config дорівнює null, повертає false і встановлює ErrorMessage.

Метод public void UpdateObjectStatus(Guid objectId, string newStatus) приймає ідентифікатор і новий статус об’єкта, перевіряє наявність списку об’єктів, знаходить відповідний об’єкт, присвоює поле Status, а потім викликає SaveConfig(). Якщо об’єкт не знайдено або список порожній, просто виводить повідомлення через логування (Logger.WriteToLogs) без генерації винятків.

Метод public void SaveConfig() не приймає параметрів, він виконує перевірку завантаженості КФ, створює XmlSerializer для типу RoboOptixConfig, відкриває файл за допомогою FileStream у режимі Create і серіалізує об’єкт Config у XML. Успіх підкріплюється виведенням повідомлення, а помилки під час збереження ловляться та виводяться через логування.

У модулі конфігурації клас RoboOptixConfig із вкладеними об’єктами відображає структуру XML-файлу, а метод XmlConfigReader.LoadConfig завантажує його в об’єктну модель. ConfigService у конструкторі викликає цей метод, зберігає результат у властивості Config, визначає валідність через IsValid та надає можливість додавати, оновлювати й видаляти записи в пам’яті з автоматичним збереженням змін назад у файл через метод SaveConfig.

Завантаження конфігурації виконується в класі App, в точці входу додатку (метод OnFrameworkInitializationCompleted) відбувається створення ConfigService, перевірка коректності завантаження та ініціалізація головного вікна. Це зроблено для того, щоб в разі неправильно заповненого КФ програма не давала доступу до свого функціоналу. Конфігураційний файл має мати назву «roboOptixConfig.xml» та знаходитися біля виконавчого файлу програми:

```
public partial class App : Application
{
    // Статичне поле, щоб можна було дістати конфіг із будь-якої частини програми
    public static ConfigService ConfigService { get; set; } = null!;

    public override async void OnFrameworkInitializationCompleted()
    {
        // Створення сервісу та зчитування файлу
        ConfigService = new ConfigService("roboOptixConfig.xml");
        ...
    }
}
```

Після створення ConfigService обов’язково проводиться перевірка що сервіс вважає КФ валідним (IsValid) та що об’єкт Config не дорівнює null:

```
if (!ConfigService.IsValid)
{
    // Якщо файл не було знайдено або XML некоректний,
    // показати повідомлення і вийти з програми
    await MessageBoxManager
        .GetMessageBoxStandard(new MessageBoxStandardParams
        {
            ContentTitle   = "Помилка конфігурації",
            ContentMessage = ConfigService.ErrorMessage,
            Icon           = Icon.Error,
```

```
            ButtonDefinitions = ButtonEnum.Ok
            })
            .ShowAsync();
        Environment.Exit(1);
        return;
    }

    if (ConfigService.Config == null)
    {
        // Додаткова перевірка на випадок, якщо ConfigService.IsValid == true,
        // але сам Config залишився null
        await MessageBoxManager
            .GetMessageBoxStandard(new MessageBoxStandardParams
            {
                ContentTitle   = "Помилка",
                ContentMessage = "Конфігурація не була завантажена!",
                Icon           = Icon.Error,
                ButtonDefinitions = ButtonEnum.Ok
            })
            .ShowAsync();
        Environment.Exit(1);
        return;
    }
```

Після успішного завантаження конфігурації створюється головне вікно:

```
    if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime desktop)
{
        desktop.MainWindow = new MainWindow();        }
    base.OnFrameworkInitializationCompleted();
```

Отримати доступ до всіх налаштувань конфігурації можна з будь-якої частини програми, використавши наступний вираз:

```
private RoboOptixConfig?  _config = App.ConfigService.Config;
```

4.3 Повідомлення системи для користувача

Користувач програми має знати в який момент роботи яка дія виконується, яка помилка була при виконанні функцій тощо. Саме для цього в програмі зроблено клас, який призначений для централізованого логування повідомлень із будь-якої частини програми. Весь текст, що відображає виконувані процеси, має виводитися у спеціальний графічний елемент розмітки користувацького інтерфейсу, але це надає деякого мінуса

– він може бути використаний тоді, коли головне вікно ще не створено (наприклад, під час десеріалізації КФ).

Через такий недолік клас Logger реалізовано таким чином, що його можна використовувати в будь-який момент із будь-якої частини програми ще до того моменту, коли головне вікно й елемент управління для виводу логів стануть доступними в UI. Він містить приватне статичне поле _pendingMessages, яке є буфером рядків, що накопичуються до появи вікна.

Для передачі повідомлення в клас логування треба викликати метод Logger.WriteToLogs, передавши параметром рядок повідомлення. Метод спершу перевіряє, чи вже створено екземпляр MainWindow і чи не дорівнює властивість Logs цього вікна null. Ця перевірка виконується наступним чином:

```
      if (Application.Current.ApplicationLifetime is
IClassicDesktopStyleApplicationLifetime       desktop
          && desktop.MainWindow is MainWindow mainWindow
          && mainWindow.Logs != null)
      { … }
```

Якщо всі умови виконані, тобто головне вікно й текстове поле для логів вже існують, із буфера спершу виводяться всі накопичені рядки:

```
      foreach (var pending in _pendingMessages)
          mainWindow.Logs.Text += pending + Environment.NewLine;
      _pendingMessages.Clear();
```

Лише після цього нове повідомлення дописується безпосередньо в mainWindow.Logs.Text із переходом на новий рядок, а каретка (властивість CaretIndex) встановлюється в int.MaxValue, щоб автоматично прокрутити текстове поле до кінця; якщо ж вікно ще не створене або всередині нього немає елемента Logs, повідомлення просто додається в _pendingMessages і чекатиме своєї черги.

Такий підхід гарантує, що всі повідомлення, викликані в ранніх етапах запуску програми (ще до появи головного вікна) не загубляться, а

будуть відображені відразу після того, як MainWindow.Logs стане доступним. Таким чином, Logger забезпечує прозоре накопичення і вивід логів у єдиному текстовому полі без необхідності перевіряти наявність UI-елемента в кожному місці, де формується повідомлення.

4.4 Модуль комп’ютерного зору

Модуль комп’ютерного зору у проекті є окремим сервісом, який використовує бібліотеку OpenCvSharp4. Його завданням є оброблення вхідного зображення, детекція ArUco маркерів, визначення координат робочої поверхні, пошук контурів об’єктів, фільтрація зайвих контурів та перетворення знайдених контурів у фізичну систему столу.

Збереження результатів вимірювань одного об’єкта на робочій поверхні виконується за допомогою класу ObjectMeasurement. Для забезпечення оновлення UI необхідно реалізувати двосторонню прив’язку через інтерфейс INotifyPropertyChanged, це дозволить сповіщувати інтерфейс користувача при будь-яких змінах у відповідних полях класу. Основні властивості класу наступні:

– ObjectId (тип даних int) – порядковий номер (ідентифікатор) об’єкта;

– CenterX, CenterY (тип даних float) – координати центру об’єкта в сантиметрах;

– ContourArea (тип даних double) – площа контуру в пікселях;

– Classification (тип даних string) – рядок із результатом класифікації об’єкта (міткою);

– Confidence (тип даних double) – впевненість моделі у віднесенні до класу (мітки);

– SortBox (тип даних SortBoxModel) – сортувальний бокс.

Механізм сповіщення про зміни має наступний вигляд:

```
public event PropertyChangedEventHandler? PropertyChanged;
```

```
protected void OnPropertyChanged(string propertyName)
    => PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
```

Щоб застосувати цей механізм необхідно створювати властивості наступним чином:

```
private float _centerX;
public float CenterX     {
    get => _centerX;
    set      {
        if (_centerX != value)        {
            _centerX = value;
            OnPropertyChanged(nameof(CenterX));        }
    }
}
```

Усі властивості з приватними полями викликають OnPropertyChanged, щоб оновити відображення у UI (наприклад, у таблиці).

Клас ArUcoObjectMeasurementProcessor призначено для аналізу вхідного зображення, визначення ArUco-маркерів, побудови перспективної трансформації до фізичної системи координат робочої поверхні, пошуку контурів об’єктів, їх вимірювання та збереження зображення кожного окремого об’єкту, щоб в подальшому передавати його на аналіз моделі ШІ та відображати в користувацькому інтерфейсі.

Додаткові пакети NuGet, які використано в класі ArUcoObjectMeasurementProcessor наведено в таблиці 4.2.

Таблиця 4.2 – Додаткові пакети NuGet/простори імен для класу ArUcoObjectMeasurementProcessor

Пакет/Простір імен	Призначення
OpenCvSharp4	Основна бібліотека, C#-обгортка OpenCV для декодування JPEG/PNG у Mat, конвертації кольору, морфології, пошуку контурів, побудови порогів тощо
OpenCvSharp4.runtime.win	Модуль ArUco для детекції і малювання маркерів, обчислення перспективної трансформації між піксельними та фізичними координатами

Продовження таблиці 4.2

Пакет/Простір імен	Призначення
Avalonia.Media.Imaging	Надає класи та методи для конвертації OpenCvSharp Mat або Bitmap у Avalonia.Media.Imaging.Bitmap для відображення кадрів у UI
System.Net.Http	Стандартний .NET-пакет з HttpClient для запитів до IP-камери
System.Drawing.Common	Забезпечує клас System.Drawing.Bitmap для представлення кадрів у пам’яті та передачі їх у callback відеопотоку

Клас має ряд конфігурованих параметрів, тобто в окремі властивості класу винесено ті дані, що використовуються в методах для обробки зображень, налаштувань параметрів визначення маркерів і тд.

Конфігуровані параметри наступні:

– OutputFolder (тип даних string) – папка для збереження вирізаних об’єктів;

– TableWidth, TableHeight (тип даних float) – реальні розміри столу в сантиметрах;

– MinAreaThreshold (тип даних double, пікселі) – мінімальна площа, нижче якої контур ігнорується;

– MarkerDistanceThreshold (тип даних double, пікселі) – мінімальна відстань від маркера, щоб фільтрувати контури, що навколо маркерів;

– MarkerDictionaryName (тип даних PredefinedDictionaryName) – словник ArUco маркерів.

Публічний конструктор класу не має параметрів, він встановлює стандартні значення для конфігурованих параметрів.

Основний публічний метод Image приймає масив байт, декодує його в Mat, робить детекцію маркерів, будує перспективну трансформацію, шукає та фільтрує контури, обчислює для кожного об’єкта центр у фізичних одиницях і збирає їх у список List<ObjectMeasurement>, а також повертає анотоване зображення і кількість знайдених маркерів.

Детекція маркерів здійснюється викликом методу:

```
CvAruco.DetectMarkers(frame, _dictionary, out Point2f[][] corners, out int[] ids, _detectorParams, out Point2f[][] rejected),
```

де frame – зображення типу Mat, _dictionary – словник ArUco, corners – вихідний параметр, масив кутів кожного маркера, ids – масив їхніх id, _detectorParams – об’єкт DetectorParameters, rejected – вихідний параметр відкинутих кандидатів маркерів.

Якщо метод повертає ids == null або ids.Length == 0, то він малює на центрі кадру повідомлення “Markers not found” за допомогою методу Cv2.PutText і повертає порожній список, якщо маркери знайдені, то їх межі малюються викликом наступного методу:

```
CvAruco.DrawDetectedMarkers(frame, corners, ids),
```

де corners і ids означають відповідно масив кутів знайдених маркерів і їхні ідентифікатори.

Після детекції формується словник центрів, будується матриця перспективної трансформації через метод Cv2.GetPerspectiveTransform і застосовується фільтрація контура за площею й відстанню до маркерів, перетворення координат контуру у фізичні одиниці через методи Cv2.PerspectiveTransform та BoundingRect, а також анотація зображення – малювання контурів червоним, прямокутників зеленим і підписів синім (рис. 4.1).

Рисунок 4.1 – Зображення отримане в результаті його обробки методом Image

Приватний метод SaveCroppedObject приймає Mat, прямокутник Rect та індекс об’єкта, розширює прямокутник на певну кількість пікселів, потім вирізає цю область у новий Mat і зберігає її у файл з іменем object_{index}.png у папці, що вказано в конфігурованих параметрах класу.

Метод ProcessImage є спрощеною версією Image: він лише декодує вхідні байти, виконує визначення маркерів викликом методу CvAruco.DetectMarkers, малює знайдені маркери через метод CvAruco.DrawDetectedMarkers та повертає оброблений кадр і кількість id без подальшої обробки контурів.

Клас ComputerVisionProcessor забезпечує підключення до камери, захоплення відеопотоку, отримання окремих кадрів та управління процесом відеострімінгу. Цей клас оголошений публічним і реалізує інтерфейс IDisposable, щоб коректно звільняти ресурси після завершення роботи з камерою. Клас має приватне поле HttpClient для HTTP-запитів до IP-камери, властивість CameraStatusMessage з приватним сеттером, що через подію StatusChanged повідомляє про зміни стану, та властивість VideoRunningStatus з подією VideoRunningChanged для індикації запущеного відеопотоку.

Також, клас має токен скасування асинхронної операції CancellationTokenSource _cts. Він потрібен для того, щоб мати можливість достроково зупинити асинхронний цикл захоплення відеокадрів. Поки відеопотік працює, у методі StartVideoStreamAsync виконується нескінченний цикл із затримкою через Task.Delay(frameDelay, _cts.Token) і на випадок, коли треба зупини відеопотік методом StopVideoStream(), встановлюється VideoRunningStatus в false і викликається _cts.Cancel(), це знімає блокування Task.Delay за допомогою переданого токена і змушує цикл завершитися негайно, не чекаючи закінчення інтервалу очікування.

Конструктор приймає рядок ipCameraUrl з адресою звернення до якої надає зображення робочої поверхні, задає його в readonly поле і встановлює таймаут для _httpClient:

```
public ComputerVisionProcessor (string ipCameraUrl)    {
    _ipCameraUrl = ipCameraUrl;
    // таймаут для HTTP запитів (3 секунд)
    _httpClient.Timeout = TimeSpan.FromSeconds(3);
}
```

Публічний асинхронний метод CaptureFrameBytesAsync без параметрів завантажує кадр як byte[] через метод GetByteArrayAsync і кидає виняток, якщо масив порожній:

```
public async Task<byte[]> CaptureFrameBytesAsync()
{
    byte[] imageBytes = await _httpClient.GetByteArrayAsync(_ipCameraUrl)
        .ConfigureAwait(false);
    if (imageBytes == null || imageBytes.Length == 0)
        throw new Exception("Не вдалося отримати кадр. Отримано порожній
масив байтів.");
    return imageBytes;
}
```

Метод StartVideoStreamAsync приймає делегат Action<Bitmap> onFrameReceived, який викликатиметься щоразу коли отримано новий кадр і необов’язковий параметр frameDelay (стандартне значення якого 30). Параметр frameDelay відповідає за затримку між обробкою двох послідовних кадрів відео. Значення вказується в мілісекундах, стандартно це 30 мс, що дає приблизно 33 кадри в секунду. Ця затримка потрібна, щоб не завантажувати камеру та CPU необмежено швидким циклом запитів і обробки, а підтримувати стабільний і передбачуваний Frames per second (FPS).

Метод запускає цикл, у якому за допомогою HttpClient завантажує байти, декодує їх у Mat через метод Cv2.ImDecode, конвертує в Bitmap через приватний статичний MatToBitmap, викликає onFrameReceived(bitmap) та чекає вказаний інтервал. Якщо виникає помилка, то метод обробляє її через подію VideoStreamError і встановлює VideoRunningStatus у false.

Для перевірки доступності камери існує асинхронний метод CheckCameraConnectionAsync, який через метод GetByteArrayAsync

виконує спробу отримати байти, оновлює CameraStatusMessage і повертає true або при помилці задає повідомлення помилки й викидає виключення.

Метод StopVideoStream публічний, змінює VideoRunningStatus на false і скасовує CancellationTokenSource, а виклик методу Dispose звільняє ресурси HttpClient та CancellationTokenSource.

Програмний код описаних методів модуля комп’ютерного зору наведено в додатку А.

4.5 Розробка та навчання моделі штучного інтелекту для класифікації об’єктів

Щоб реалізувати модуль штучного інтелекту треба створити два прості класи-схеми та клас-менеджер. Визначити структуру для навчальних та тестових даних, створити методи для створення, збереження моделі та виконання передбачення зображення. Всі класи буде виділено в окремий простір імен – AIManager, для спрощення подальшого використання доступних методів/властивостей модуля.

У просторі імен AIManager клас ModelInput визначає вхідну структуру для навчальних та тестових даних. Декоратори [LoadColumn(0)] і [LoadColumn(1)] вказують ML.NET брати перший стовпець Tab-Separated Values (TSV) як ImagePath (шлях до файлу зображення) і другий – як Label (текстова мітка). Клас ModelOutput наслідує ModelInput і додає масив оцінок і властивість PredictedLabelValue, у якій зберігається результат класифікатора. Саме ці два класи використовуються при завантаженні даних у IDataView і при отриманні результатів із PredictionEngine<ModelInput, ModelOutput>.

Tab-Separated Values – це простий текстовий формат, у якому кожен рядок містить кілька полів, розділених певним роздільником.

Файл baseModel.tsv (назва файлу має відповідати назві моделі ШІ), матиме два стовпці розділені пробілом, де зліва вказуєте шлях до кожного файлу, а справа його мітка:

```
switch/2.png switch
switch/3.png switch
switch/1.png switch
button/8.png button
button/9.png button
button/14.png button
```

Менеджер AIManager інкапсулює створення та збереження моделі, він містить MLContext _mlContext, який ініціалізується в конструкторі класу та необхідний для будь-яких операцій ML.NET.

Метод TrainModel приймає корінь директорії із зображеннями, шляхи до TSV-файлів із мітками для тренувальної та тестової вибірки, шлях, куди зберегти модель, а також два прогрес-репортери IProgress для моніторингу відсотка виконання та текстових повідомлень:

```
public ITransformer TrainModel(
    string imagesFolder,
    string trainTagsTsv,
    string testTagsTsv,
    string saveModelPath,
    out IEnumerable<ModelOutput> imagePredictionData,
    IProgress<int> progressReporter,
    IProgress<string> logReporter)
```

Спочатку метод повідомляє про побудову конвеєра трансформацій і викликає приватний метод BuildPipeline, який завантажує зображення, змінює їх розміри, нормалізує пікселі, застосовує попередньо натреновану TensorFlow-модель і мапить рядкові мітки у числові ключі перед передачею на тренер LbfgsMaximumEntropy:

```
IEstimator<ITransformer> pipeline = BuildPipeline(imagesFolder);
```

Далі метод TrainModel завантажує тренувальні дані через метод LoadFromTextFile, проводить навчання моделі, проводить тестове передбачення, відбирає результати, обчислює метрики якості (LogLoss і

PerClassLogLoss) та логує їх. Наприкінці він зберігає натреновану модель у вказане розташування. Ця логіка програмно виглядає наступним чином:

```
        // Завантаження тренувальних даних
        IDataView trainingData = _mlContext.Data.LoadFromTextFile<ModelInput>(
            path: trainTagsTsv, hasHeader: false, separatorChar: ' ');
        // Навчання моделі
        ITransformer newModel = pipeline.Fit(trainingData);
        //Завантаження тестових даних
        IDataView testData = _mlContext.Data.LoadFromTextFile<ModelInput>(
            path: testTagsTsv,
            hasHeader: false,
            separatorChar: ' ');
        // Прогнозування для тестових даних
        imagePredictionData                                                    =
_mlContext.Data.CreateEnumerable<ModelOutput>(predictions,       reuseRowObject:
true);
        // Обчислення метрик якості
        var metrics = _mlContext.MulticlassClassification.Evaluate(
            predictions,
            labelColumnName: "LabelKey",
            predictedLabelColumnName: "PredictedLabel");
        // Збереження моделі
        _mlContext.Model.Save(newModel, trainingData.Schema,
            Path.Combine(saveModelPath, "baseModel.zip"));
```

Кожен крок методу TrainModel фіксується в текстовому (наприклад, logReporter.Report($"LogLoss: {metrics.LogLoss:F4}")) та чисельному (progressReporter.Report(5)) вигляді. Текст може бути відображено в користувацькому інтерфейсі у відповідному текстовому елементі, а чисельне значення для відображення прогресу лінією.

Метод ClassifySingleImage призначений для того, щоб узявши завантажену модель і шлях до одного зображення, повернути результат класифікації саме цього кадру:

```
        public    ModelOutput    ClassifySingleImage(ITransformer    model,    string
predictSingleImage)
```

де model – екземпляр конвеєра (ITransformer), predictSingleImage – рядок із шляхом до файлу зображення, яке потрібно класифікувати.

У методі створюється об’єкт вхідного типу, ініціалізується PredictionEngine<ModelInput, ModelOutput> – сервіс, що упаковує трансформації та класифікатор у конвеєр та дає змогу застосовувати їх до поодиноких об’єктів ModelInput. В результаті отримується екземпляр

ModelOutput, в якому поля Score містять масив довірчих показників для кожного класу, а PredictedLabelValue – рядок із тією міткою, яку модель вважає найбільш імовірною:

```
var imageData = new ModelInput {    ImagePath = predictSingleImage   };
_mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(model)
ModelOutput prediction = predictor.Predict(imageData);
```

В завершенні методу повертається значення ModelOutput prediction.

Метод BuildPipeline призначено для побудови послідовності перетворень даних та навчальну частину конвеєра ML.NET, повертаючи об’єкт типу IEstimator<ITransformer>, готовий до навчання:

```
private IEstimator<ITransformer> BuildPipeline(string imagesFolder)
```

де imagesFolder – це шлях до кореневої папки, яка містить підпапки з зображеннями для кожної категорії.

Всередині методу спершу викликається метод LoadImages щоб за шляхом в ModelInput.ImagePath, завантажити відповідні бінарні дані зображення до стовпця під назвою "input":

```
_mlContext.Transforms.LoadImages(
    outputColumnName: "input",
    imageFolder: imagesFolder,
    inputColumnName: nameof(ModelInput.ImagePath))
```

Далі через метод .Append(_mlContext.Transforms.ResizeImages(...)) розмір кожного кадру приводиться до стандартних 224×224 пікселів, використовуючи константи з InceptionSettings. Потім наступне перетворення для нормалізації пікселів, воно перетворює кожне зображення на числовий вектор, віднімаючи середнє значення (117) і, залежно від ChannelsLast, розставляючи порядок каналів RGB:

```
.Append(_mlContext.Transforms.ExtractPixels(
    outputColumnName: "input",
    interleavePixelColors: InceptionSettings.ChannelsLast,
    offsetImage: InceptionSettings.Mean))
```

Після цього йде інтеграція попередньо навченого TensorFlow-графа Inception:

```
.Append(_mlContext.Model.LoadTensorFlowModel(_inceptionTensorFlowModel)
    .ScoreTensorFlowModel(
        outputColumnNames: new[] { "softmax2_pre_activation" },
        inputColumnNames: new[] { "input" },
        addBatchDimensionInput: true))
```

цей елемент бере вхідний стовпець "input" і передає його TensorFlow-моделі, отримуючи на виході передактиваційні значення шару softmax2_pre_activation. Далі рядкові мітки зі стовпця "Label" перетворюються в числові ключі за допомогою:

```
.Append(_mlContext.Transforms.Conversion.MapValueToKey(
    outputColumnName: "LabelKey",
    inputColumnName: "Label"))
```

і цей ключований стовпець використовують у тренері:

```
.Append(_mlContext.MulticlassClassification.Trainers.LbfgsMaximumEntropy(
    labelColumnName: "LabelKey",
    featureColumnName: "softmax2_pre_activation"))
```

так навчається мультікласова модель на векторі ознак, що вийшов із TensorFlow, а потім перетворюються числові прогнози назад у текстові мітки через наступний запис:

```
.Append(_mlContext.Transforms.Conversion.MapKeyToValue(
    "PredictedLabelValue",
    "PredictedLabel"))
```

щоб у кінцевому ModelOutput з’явилось зрозуміле поле PredictedLabelValue. Наприкінці .AppendCacheCheckpoint(_mlContext) додає кешування, яке прискорює повторні проходи по даних. У результаті BuildPipeline повертає повністю сконфігурований пайплайн із завантаженням зображень, їх обробкою, TensorFlow-скорингом, тренером і зворотними перетвореннями – готовий до виклику методу .Fit(trainingData).

Метод LoadModel призначений для того, щоб відновити раніше навчену та збережену модель із файлу й одразу отримати її схему даних:

public ITransformer LoadModel(string modelPathAndName, out DataViewSchema modelSchema)

де modelPathAndName – містить шлях до ZIP-файлу, у якому модель була збережена викликом _mlContext.Model.Save(…). Метод повертає конвеєр трансформацій й алгоритм класифікації (типу ITransformer), який можна використовувати для передбачень, і змінну modelSchema, куди ML.NET запише структуру вхідних і вихідних колонок, задекларованих у час тренування.

Внутрішня логіка методу складається лише з одного виклику, де MLContext відкриває файл, завантажує інтерналізовану послідовність трансформацій і тренера, відновлюючи весь пайплайн так, як він був під час збереження:

```
return _mlContext.Model.Load(modelPathAndName, out modelSchema);
```

Після цього повернутий ITransformer можна передавати у метод ClassifySingleImage.

Таким чином, LoadModel спрощує перехід від фази навчання до фази інференсу: жоден додатковий код для відновлення налаштувань трансформацій чи відновлення ваг мережі не потрібен – ML.NET сам відновлює все з збереженого файлу і віддає готовий до роботи пайплайн.

Програмний код модуля наведено в додатку А.

4.6 Модуль управління маніпулятором

Модуль управління роботом відповідає за встановлення зв’язку з апаратною частиною маніпулятора, надсилання й обробку G- та M-команд по послідовному порту, а також збір і парсинг відповідей пристрою. Він інкапсулює відкриття, налаштування та закриття порту, забезпечуючи надійний обмін даними з контролером робота.

Клас RobotController у просторі імен RobotController відповідає за низькорівневу взаємодію з роботизованим маніпулятором по послідовному

порту. Конструктор класу приймає назву порту (portName) та швидкість зв’язку в бодах (baudRate), відкриває порт із 8 бітами даних, без парності, з одним стоп-бітом, встановлює таймаути читання/запису по 2000 мс, чекає 2 секунди для ініціалізації плати й автоматично викликає HomeAll() для калібрування маніпулятора. Після успішного відкриття через наступну подію:

```
public event Action<string>? StatusChanged;
```

розсилається рядок "Підключено".

Публічна властивість IsConnected дозволяє перевірити, чи порт наразі відкритий, вона повертає булеве значення, якщо true – порт відкрито, false – закрито. Публічна властивість PortName типу рядок повертає назву підключеного порту.

Щоб закрити з’єднання, реалізовано IDisposable: у методі Dispose() якщо порт відкритий, він закривається й через StatusChanged("Відключено") сповіщається про розрив зв’язку, а потім звільняються ресурси _serialPort. Фіналізатор ~RobotController() гарантує вихід у випадку, якщо Dispose() не був викликаний вручну.

Метод static string[] GetAvailablePorts() повертає всі доступні назви серійних інтерфейсів через SerialPort.GetPortNames(), що полегшує динамічне формування списку у UI.

Універсальний метод SendCommand(string command) перевіряє, що порт відкритий, очищає вхідний буфер, нормалізує закінчення рядка "\r\n" й записує команду в порт:

```
if (!command.EndsWith("\r\n")) command += "\r\n";
_serialPort.Write(command);
```

На основі методу SendCommand побудовані конкретні дії:

– MoveLinear(double x, double y, double z, double feedRate = 0) формує G-команду формату "G0 X{x}Y{y}Z{z}" і викликає SendCommand;

– HomeAll(), GrabObject() та ReleaseObject() надсилають відповідно команди "G28", "M3" та "M5".

Метод ReportCoordinates() служить для отримання поточного положення маніпулятора: він надсилає команду "M114", призначає коротший ReadTimeout (200 мс), у циклі читає усі рядки з порту через метод ReadLine() до викидання TimeoutException, збираючи їх у List<string>, після чого парсить останній елемент і повертає об’єкт типу RobotResponse через наступний метод:

```
return RobotResponse.Parse(lastLine);
```

Клас RobotResponse відповідає за розбір і форматування рядків логів або відповідей від контролера. Він містить такі властивості як Timestamp, Level, Message, Command і координати X, Y, Z, E. Розбір реалізовано через компільований регулярний вираз Regex LineRegex, що групує час у форматі HH:mm:ss.fff, рівень повідомлення, основний текст, а також необов’язковий блок координат у квадратних дужках.

Метод static RobotResponse Parse(string line) перевіряє вхідний рядок, парсить час через DateTime.ParseExact (або встановлює поточний, якщо префікс відсутній), виділяє рівень й текст, шукає в ньому G-команду, конвертує числові групи в тип double? і повертає заповнений екземпляр. Альтернативний метод static bool TryParse(string line, out RobotResponse? result) виконує те саме, але без викидання виключення, просто повертає false при невдачі.

Для протилежного перетворення об’єкта в рядок є метод ToLogLine(), який збирає рядок із тимчасовим префіксом, рівнем, повідомленням, опціональною командою та наявними координатами в зручному форматі.

Програмний код модуля наведено в додатку А.

4.7 Реалізація графічного інтерфейсу оператора

Графічний інтерфейс оператора – це єдина точка входу для моніторингу роботи всіх підсистем та прямого керування роботом у

режимі реального часу. Він призначений для того, щоб людина могла візуально стежити за потоком з камери, бачити результати детекції й вимірювань, переглядати класифікацію об’єктів, отримувати зворотний зв’язок від контролера та відправляти команди до маніпулятора без жодного ручного втручання у код чи консоль.

4.7.1 Головне вікно програми

Головним вікном програми є MainWindow, вигляд вікна показано на рисунку 4.2.

Головне вікно має ряд елементів для моніторингу та налаштування процесу сортування. Шапка вікна має дві кнопки «Почати» та «Зупинити» для запуску та негайної зупинки процесу сортування; інформаційні поля підключення робота та камери та їх індикатори; кнопка «Перевірити підключення» для примусової перевірки з’єднання з камерою. Перший блок вікна має два прямокутними для відображення відеопотоку з камери та візуального результату аналізу зображення робочої поверхні; кнопку «Почати аналіз» для запуску процесу аналізу робочої поверхні. Другий блок містить таблицю з результатами проведеного аналізу робочої поверхні та вікно для налаштування СБ для знайдених класів об’єктів. В нижній частині вікна знаходиться багаторядкове текстове поле в який виводяться всі діагностичні чи інформаційні повідомлення за час роботи додатка.

Праворуч від основної частини вікна розташовано бічну панель, яку можна розгорнути та приховати за допомогою кнопки-тумблера, вони містить три пункти навігації: кнопка із іконкою розпізнавання призначена для переходу до вікна управління об’єктами, кнопка із зображенням маніпулятора відкриває вікно керування роботом, а кнопка із іконкою «AI» веде у налаштування штучного інтелекту.

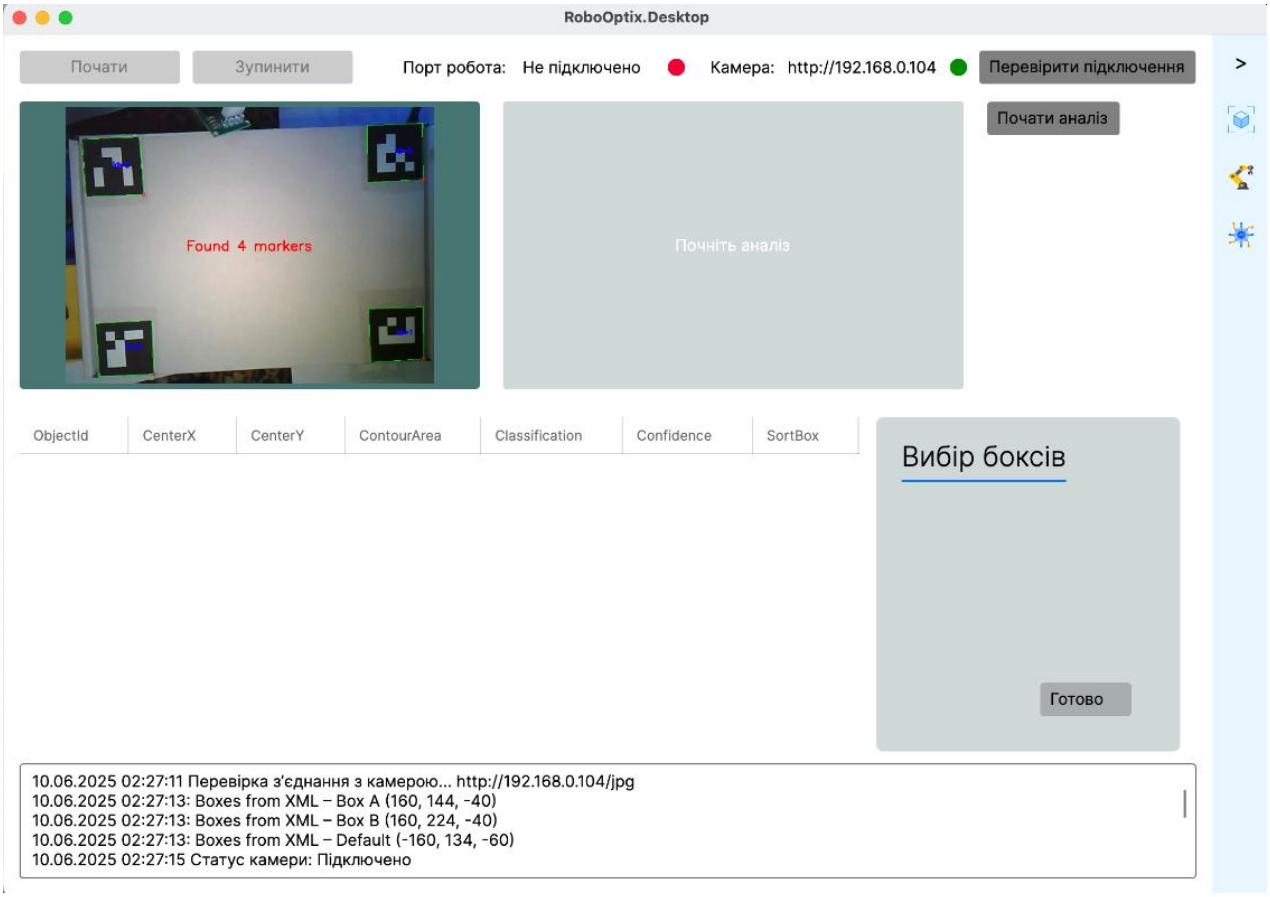


Рисунок 4.2 – Вигляд головного вікна програми

Біндінг у контексті UI – це механізм, який автоматично синхронізує значення властивостей у коді з елементами інтерфейсу. Замість ручного оновлення кожного елемента після зміни даних достатньо задати прив’язку (Binding) між властивістю та UI-елементом.

У фреймворці Avalonia binding робиться через прив’язку властивостей UI-елементів до властивостей об’єкта-моделі (DataContext). Через встановлення DataContext вікна на екземпляр класу та вказання у XAML необхідних значень:

```
<TextBlock Text="{Binding RobotPortDisplay}" />
```

де за допомогою {Binding PropertyName} виконується одностороння прив’язка (OneWay). Тепер при зміні властивості RobotPortDisplay в DataContext оновиться інтерфейс користувача.

В таблиці 4.3 наведено поля, що використовуються в класі MainWindow, їх призначення та короткий опис.

Таблиця 4.3 – Властивості класу MainWindow

Назва	Тип та контекст використання	Призначення
_visionProcessor	ComputerVisionProcessor, приватне поле	Використовується для отримання відеокадрів з IP-камери, обробки статусу й помилок відеопотоку
_arucoProcessor	ArUcoObjectMeasurementProcessor, приватне поле	Використовується для детекції ArUco-маркерів, вимірювання контурів й формування списку знайдених об’єктів
_aiManager	AIManager.AIManager, приватне поле	Використовується для класифікації вирізаних об’єктів
_config	RoboOptixConfig, приватне поле	Глобальна конфігурація програми
_ipCameraUrl	string, приватна властивість	URL MJPEG-потоку камери (Config.CameraIP + "/jpg")
_tableWidth	float, приватне поле	Ширина столу, зчитана з конфігурації
_tableHieght	float, приватне поле	Висота столу, зчитана з конфігурації
_isCameraConnected	bool, приватне поле	Використовується як внутрішній флаг, чи вдалося підключитися до камери
_isRobotConnected	bool, приватне поле	Використовується як внутрішній флаг, чи підключений робот
_robotSubscribed	bool, приватне поле	Вказує, чи вже підписалися на подію App.Robot.StatusChanged, щоб не підписуватися двічі
_cancelSorting	bool, приватне поле	Флаг, який сигналізує циклу сортування зупинитися при натисканні Stop
_isSortingInProgress	bool, приватне поле	Використовується як допоміжний флаг, щоб запобігти повторному запуску сортування

Продовження таблиці 4.3

Назва	Тип та контекст використання	Призначення
FoundObjects	ObservableCollection<ObjectMeasurement>, для інтерфейсу	Колекція знайдених та виміряних об’єктів; біндиться до таблиці DataGrid для відображення списку об’єктів
SortBoxes	ObservableCollection<SortBoxModel>, для інтерфейсу	Колекція доступних сортувальних боксів; джерело для ComboBox при призначенні боксів класифікаціям
UniqueClassifications	ObservableCollection<ObjectMeasurement>, для інтерфейсу	Колекція унікальних міток знайдених об’єктів; біндиться до ItemsControl у вкладці “Вибір боксів” для призначення
canStart	bool, для прив’язки IsEnabled кнопки Start	Керує доступністю кнопки Start: активна, коли підключена камера, робот і призначені бокси для знайдених об’єктів
CanStop	bool, для прив’язки IsEnabled кнопки Stop	Керує доступністю кнопки Stop у UI: активна, коли робот підключений та процес сортування запущено
IsVideoStreamOn	bool, для прив’язки в інтерфейсі	Відображає стан відеопотоку: впливає на колір індикатора та видимість відео
IsVideoStreamOff	bool, для прив’язки в інтерфейсі	Інверсія IsVideoStreamOn, використовується для затемнення відео та показу повідомлення
RobotPortDisplay	string, для відображення в інтерфейсі	Повертає назву порту робота або “Не підключено”; відображається поруч із індикатором роботи
CameraUrlDisplay	string, для відображення в інтерфейсі	Повертає URL камери або “Не підключено”; відображається поруч із індикатором камери

В таблиці 4.4 наведено методи класу MainWindow з описом їх призначення, програмний код класу наведено в додатку А.

Таблиця 4.4 – Методи класу MainWindow

Метод	Сигнатура	Призначення та короткий опис
SubscribeToRobotStatus	public void SubscribeToRobotStatus()	Виконує підписку на подію StatusChanged в App.Robot, якщо ще немає підписки. Оновлює індикатор та властивість RobotPortDisplay при зміні стану з’єднання
Конструктор класу	public MainWindow()	Ініціалізує всі сервіси (модуль комп’ютерного зору, ШІ, конфігурації), зчитує конфіг, підписується на події статусів камери й робота, готує колекції FoundObjects, UniqueClassifications, SortBoxes, завантажує модель ШІ і встановлює DataContext. Після відкриття вікна запускає перевірку з’єднання з камерою (якщо успішно підключено, то відразу показує відеопотік)
FoundObjectsCollectionChanged	private void FoundObjectsCollectionChanged(object? sender, NotifyCollectionChangedEventArgs e)	Обробляє зміни у списку знайдених об’єктів, підписуючи або відписуючи PropertyChanged у кожного елементу, і викликає UpdateCanStartStop() для оновлення доступності кнопок
ObjectMeasurementPropertyChanged	private void ObjectMeasurementPropertyChanged(object? sender, PropertyChangedEventArgs e)	Відстежує зміну властивості SortBox у будь-якого елементу списку знайдених об’єктів і при її оновленні викликає UpdateCanStartStop() для оновлення статусу кнопок
UpdateCanStartStop	private void UpdateCanStartStop()	Перевіряє три умови (що підключено робот, камеру, і всі знайдені об’єкти мають призначений SortBox) для встановлення CanStart, а також оновлює CanStop
LoadSortBoxesFromConfig	private void LoadSortBoxesFromConfig()	Зчитує з ConfigService.Config.Boxes конфігурацію СБ і додає їх до колекції SortBoxes, яка використовуються у UI для призначення знайденим об’єктам
ToogleMenuBtnClick	private void ToogleMenuBtnClick(object? sender, RoutedEventArgs e)	Обробник натискання кнопки відкриття/закриття бічної панелі. Перемикає панель і змінює символ кнопки між « > » та « < »

Продовження таблиці 4.4

Метод	Сигнатура	Призначення та короткий опис
ConvertBitmapToByteArray	public static byte[] ConvertBitmapToByteArray(Bitmap bitmap)	Зберігає System.Drawing.Bitmap у MemoryStream у форматі PNG і повертає масив байтів для подальшої обробки
ConvertMatToAvaloniaBitmap	private Bitmap ConvertMatToAvaloniaBitmap(Mat mat)	Записує OpenCvSharp.Mat у MemoryStream у форматі BMP і створює з потоку Avalonia.Bitmap для відображення в UI
CheckCameraConnectionAsync	private async Task CheckCameraConnectionAsync()	Асинхронно звертається до _visionProcessor.CheckCameraConnectionAsync(), оновлює _isCameraConnected і, за потреби, лог-файли через Logger
StartVideoStream	private void StartVideoStream()	Робить «fire-and-forget» виклик _visionProcessor.StartVideoStreamAsync(OnFrameReceived) для запуску потоку кадрів без блокування UI і показує/ховає елемент VideoFrame залежно від властивості VideoRunningStatus
OnFrameReceived	private void OnFrameReceived(Bitmap frame)	Делегат для отримання нових кадрів: конвертує Bitmap у байти, проганяє через _arucoProcessor.ProcessImage для обробки зображення, отримує анотований Mat, конвертує його в Avalonia.Bitmap і призначає VideoFrame.Source
CheckCameraConnection_Click	private async void CheckCameraConnection_Click(object? sender, RoutedEventArgs e)	Обробник натискання кнопки «Перевірити підключення»: викликає асинхронний метод CheckCameraConnectionAsync(), щоб перевірити доступність камери та оновити статус
ClassifySingleImageOnClick	private async void ClassifySingleImageOnClick(object? sender, RoutedEventArgs e)	Очищує список знайдених об’єктів, захоплює кадр, відображає його в інтерфейсі, викликає _arucoProcessor.Image для визначення контурів, сортує результати, додає їх до списку знайдених об’єктів, потім класифікує кожен вирізаний об’єкт через_aiManager.ClassifySingleImage, заповнює Classification та Confidence, оновлює таблицю та панель унікальних класифікацій

Продовження таблиці 4.4

Метод	Сигнатура	Призначення та короткий опис
UpdateUniqueClassifications	private void UpdateUniqueClassifications()	Метод очищує список UniqueClassifications і заново заповнює його одним «шаблонним» ObjectMeasurement для кожної унікальної мітки з FoundObjects, щоб UI відобразив відповідні ComboBox для вибору СБ
OnClosed	protected override void OnClosed(EventArgs e)	Перехоплює закриття вікна: зупиняє відеопотік (_visionProcessor.StopVideoStream()) та звільняє ресурс (_visionProcessor.Dispose())
StartSortingProcess	private async void StartSortingProcess(object? sender, RoutedEventArgs e)	Обробник натискання «Почати»: блокує кнопку, скидає _cancelSorting, потім послідовно для кожного об’єкту зі списку знайдених об’єктів формує команди роботу (MoveLinear, GrabObject, ReleaseObject) згідно з координатами та призначеним СБ. Після завершення чи примусового переривання оновлює кнопки CanStart/CanStop
DoneBtnClick	private void DoneBtnClick(object? sender, RoutedEventArgs e)	Після вибору боксів у вкладці «Вибір боксів» підтверджує призначення: для кожної унікальної класифікації присвоює обраний СБ всім об’єктам у списку, що мають цю мітку
ObjectManageMenuBtnClick	private void ObjectManageMenuBtnClick(object? sender, RoutedEventArgs e)	Відкриває модальне вікно ObjectManagerView для керування списком об’єктів класів
ArmManageMenuBtnClick	private void ArmManageMenuBtnClick (object? sender, RoutedEventArgs e)	Відкриває вікно ArmManage для налаштування параметрів роботизованого маніпулятора
AIManageMenuBtnClick	private void AIManageMenuBtnClick(object? sender, RoutedEventArgs e)	Відкриває вікно AIManage для керування моделлю ШІ
ObjectsTableDoubleTapped	private async void ObjectsTableDoubleTapped(object? sender, TappedEventArgs e)	Обробник подвійного клацання по рядку таблиці: відкриває ObjectDetailsWindow для редагування полів об’єкту, зберігає зміни назад в колекцію

Кінець таблиці 4.4

Метод	Сигнатура	Призначення та короткий опис
StopSortingProcess	private void StopSortingProcess(object? sender, RoutedEventArgs e)	Обробник натискання «Зупинити»: встановлює _cancelSorting в значення true, викликає UpdateCanStartStop(), а також зупиняє відеопотік через метод _visionProcessor.StopVideoStream()

Головне вікно програми дає оператору змогу повністю контролювати весь цикл сортування та класифікації об’єктів у реальному часі. Спершу можна перевірити та встановити з’єднання з IP-камерою відповідною кнопкою та побачити статус камери та робота за кольором індикаторів.

Після запуску вікна відеопотік автоматично запускається й оператор бачить пряму трансляцію робочої області: ця область затемнюється з надписом «Камера не підключена» допоки камера не відповідає на запити.

Після натискання кнопки «Почніть аналіз» в правому прямокутнику показується захоплений кадр робочої поверхні, обробку якого буде виконано. Кожен знайдений об’єкт буде показано на повний розмір правого прямокутника протягом двох секунд, в цей же час в таблиці наповниться рядок даними про цей об’єкт. Після обробки всіх знайдених об’єктів в прямокутник виводиться початковий кадр, але з вже обведеними об’єктами та вказаними для них координатами (рис. 4.3).

Після проведення аналізу в інтерфейсі оператор має таблицю з рядками, що описують кожен знайдений об’єкт, якщо оператор хоче змінити клас, координати або призначити вручну сортувальний бокс, то він може це зробити у додатковому вікні, яке відкривається після подвійного кліку по рядку відповідного об’єкту. Додаткове вікно змін показано на рисунку 4.4. В цьому вікні оператор може: змінити клас об’єкту, його координати, призначити сортувальний бокс, додати зображення об’єкту до навчального набору вказаного класу.

Якщо після виконання класифікації об’єкт має низьке значення впевненості (нижче ніж вказано в КФ), то під час обробки йому автоматично призначається СБ «Default». Після проведення аналізу в блоці “Вибір боксів”, оператор має призначити кожному типу об’єкту цільовий контейнер із попередньо завантаженого списку боксів. Натисканням кнопки «Готово» запускається процес призначення об’єктам сортувальних боксів.

Після підтвердження налаштувань можна натиснути кнопку “Почати” (рис. 4.3), тоді робот послідовно підійде до кожного знайденого об’єкта, захопить його, перемістить у заданий бокс та повернеться в початкову позицію. У процесі сортування кнопку “Зупинити” можна натиснути в будь-який момент, щоб негайно зупинити рухи та відеопотік.

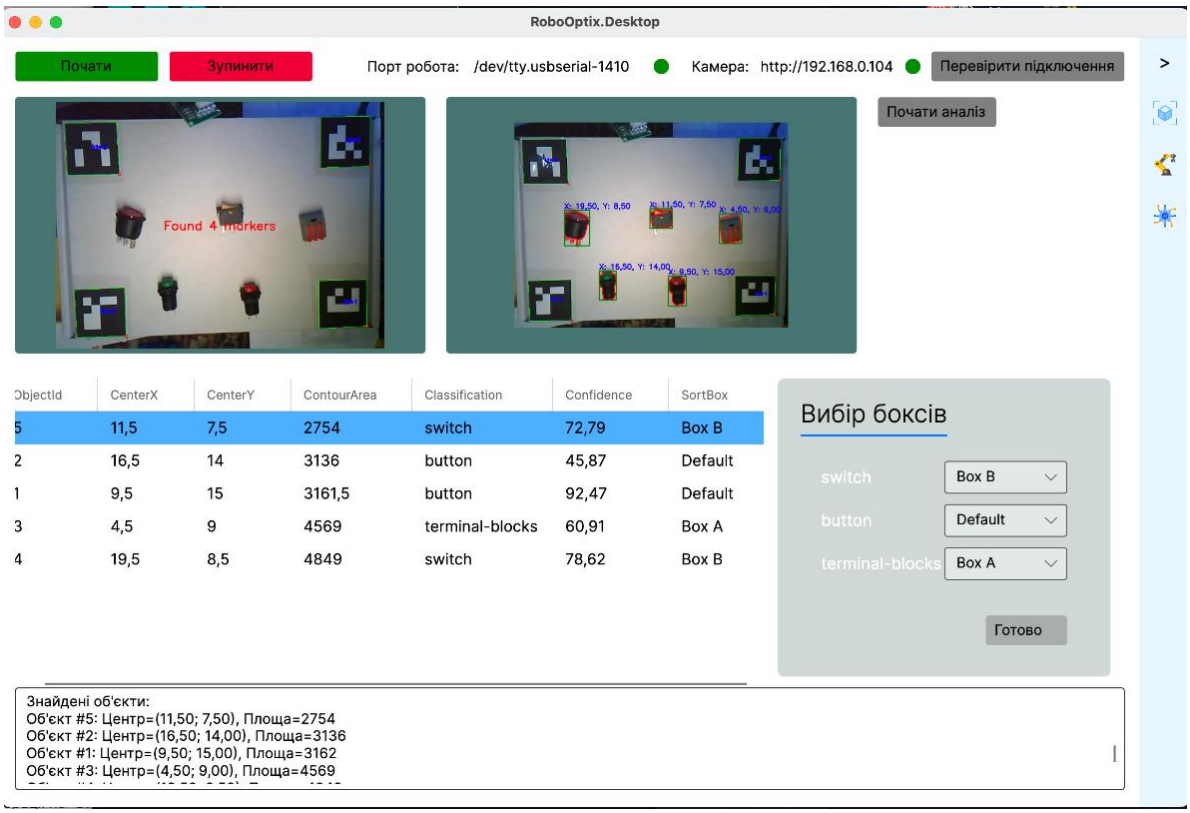


Рисунок 4.3 – Вигляд інтерфейсу після проведення аналізу робочої поверхні

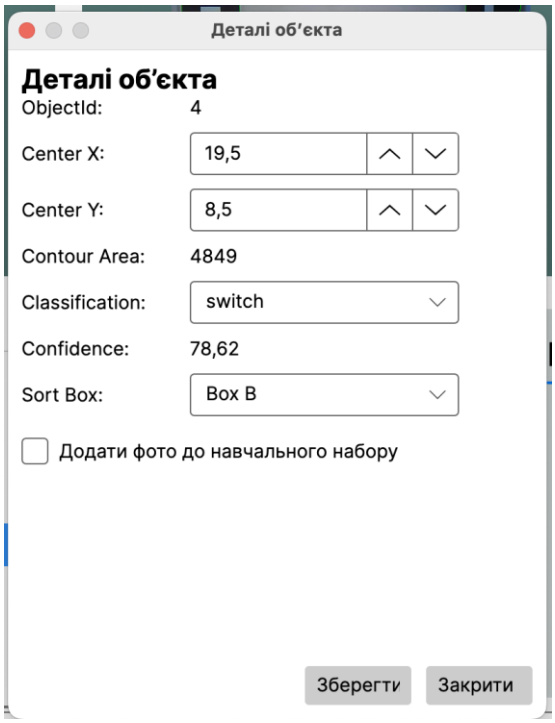


Рисунок 4.4 – Додаткове вікно для зміни параметрів знайденого об’єкту

Внизу вікна є журнал всіх дій і повідомлень системи – від статусу камери та робота до результатів класифікації й будь-яких помилок, що виникають під час роботи (рис. 4.3). А з бічного меню оператор може перейти до окремих модулів керування: об’єктами, налаштування робота та ШІ. Таким чином, MainWindow надає інтуїтивний і водночас гнучкий інтерфейс.

4.7.2 Вікно керування класами

У вікні керування об'єктами ObjectManagerView оператор може переглядати, додавати, видаляти та редагувати список класів об’єктів, що використовуються в системі для навчання ШІ. Зовнішній вигляд вікна показано на рисунку 4.5.

Вікно має панель для фільтрування списку класів та сам список. Оператор може ввести частину ідентифікатора або назву класу в текстове поле, обрати статус у випадному списку та після натискання кнопки «Фільтрувати» в списку відобразяться ті класи, що підходять під введені

параметри. В нижній частині вікна є кнопки «Додати», «Редагувати», «Видалити», «Додати до навчального набору» та «Видалити з навчального набору».

Кнопка «Видалити» стає доступною при виділенні певного рядку таблиці, після її натискання з’являється повідомлення (рис. 4.6) з уточненням кількості рядків для видалення, в якому є кнопки, де користувач має підтвердити або відхилити операцію видалення.

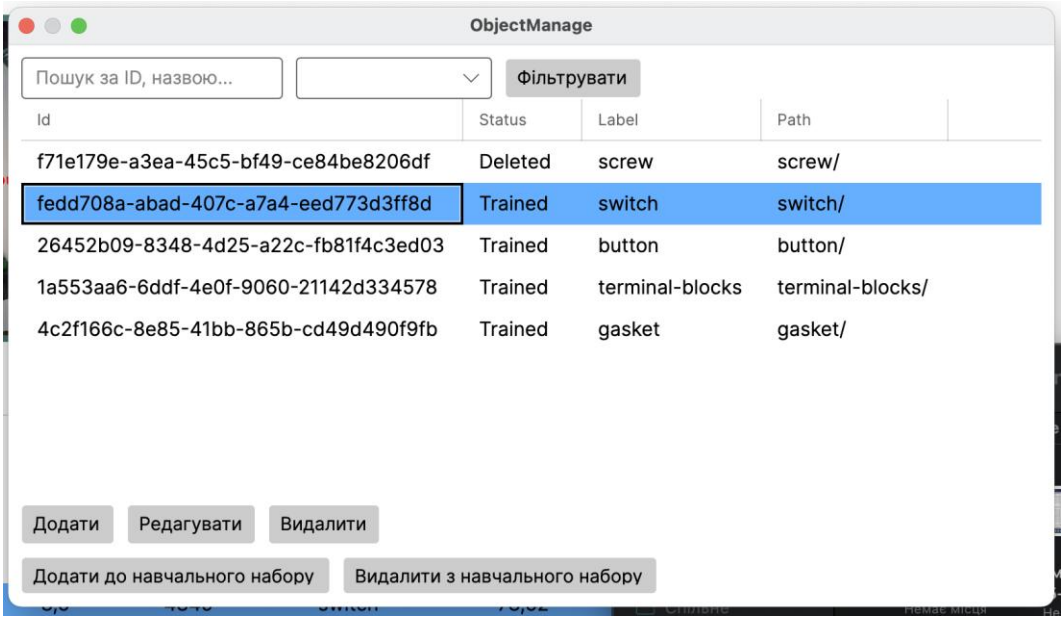


Рисунок 4.5 – Вигляд вікна керування класами

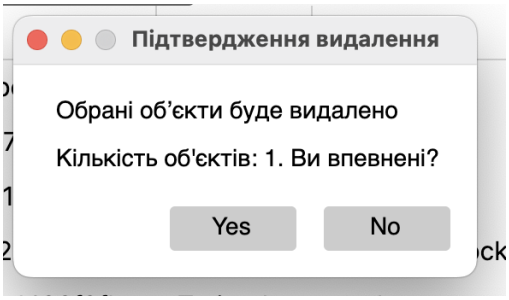


Рисунок 4.6 – Вікно підтвердження видалення записів

Кнопка «Додати» призначена для додавання нового запису про клас, натискання кнопки приводить до відкриття додаткового вікна.

Кнопка «Редагувати» стає доступною при виділенні певного рядку таблиці, після її натискання з’являється додаткове вікно для внесення змін запису. Кнопка «Додати до навчального набору» та «Видалити з навчального набору» стають доступними при виділенні певного рядку таблиці, їх натискання призводить до змін статусу обраного запису на Added або Deleted, що відповідає доданому або видаленому з навчального набору.

Усі зміни записів синхронізується з модулем конфігурації ConfigService, тому оператор може вносити зміни в записи, а інші частини програми завжди матимуть актуальну інформацію про доступні класи та їх статус.

У вікні додавання нового класу AddObjectWindow (рис. 4.7) оператор має вказати назву класу (мітку), та шлях до директорії (треба обрати з діалогу, що відкривається кнопкою «Огляд») в якій знаходиться набір зображень відповідного класу. Натиснувши кнопку «ОК» до списку буде додано новий клас, кнопка «Скасувати» скасує додавання. Після успішного виконання цієї операції в загальному списку класів вікна ObjectManagerView буде створено новий рядок з описом класу. Якщо виникне помилка, то з’явиться вікно з текстом помилки (рис. 4.8).

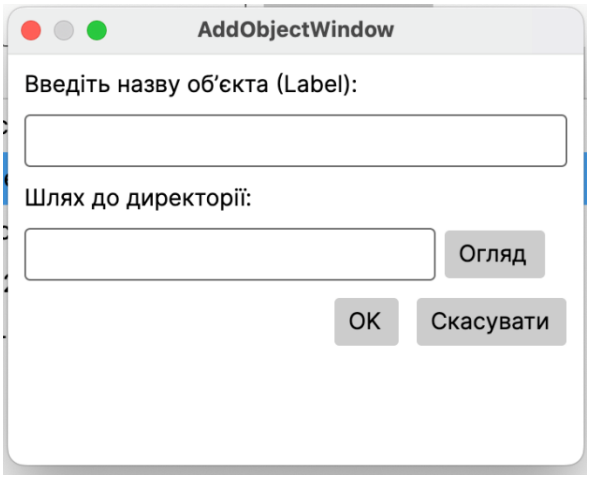


Рисунок 4.7 – Вікно додавання нового класу

Клас AddObjectWindow має дві властивості ObjectLabel та ObjectPath, які містять назву класу, яку оператор ввів у поле Label та відносний шлях до папки з фотографіями для навчання цього класу. Для кнопок створено відповідні обробники подій натискання кнопки. Натискання на кнопку «Огляд» обробляє метод BrowseButton_Click, який відкриває діалог вибору папки (меотд OpenFolderDialog – це системний діалог, що дозволяє користувачу обрати папку). Після вибору встановлює обраний шлях у текстове поле. Для кнопки «ОК» – обробник OkButton_Click, який перевіряє коректність введених даних, тобто що поле Label не пусте, його унікальність серед існуючих, наявність і мінімальну кількість зображень у вказаній папці, правильність шляху відносно ModelsPath/training. Якщо всі умови виконані, то обробник події додає новий запис через метод модуля конфігурації ConfigService.AddObject(obj) і закриває вікно із результатом true.

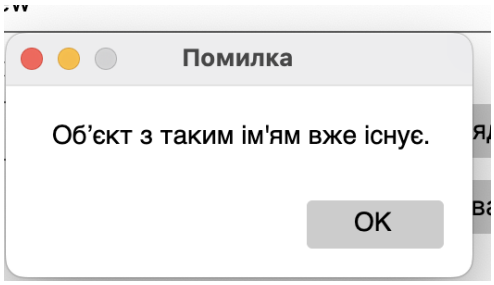


Рисунок 4.8 – Вікно з текстом помилки

Натискання кнопки «Редагувати», яка доступна лише якщо обрано рядок в таблиці, призводить до відкриття додаткового діалогового вікна (рис. 4.9), клас якого – EditObjectWindow. Цей клас має три властивості: ObjectLabel, ObjectPath та ObjectId, що містять дані – назва класу, шлях та ідентифікатор. Ідентифікатор незмінний, тобто користувач може лише переглянути значення цього поля, він використовується для пошуку необхідного для зміни запису в конфігураційному файлі.

Коли оператор у вікні редагування класу натискає «Переглянути фото» для обраної директорії, програма створює новий екземпляр вікна PhotoGalleryWindow, передаючи в конструктор шляхи до всіх .png і .jpg файлів у цій папці.

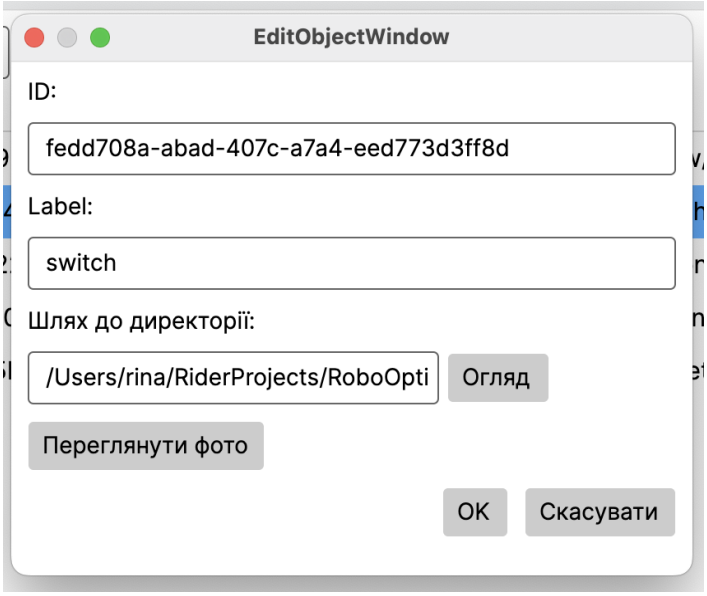


Рисунок 4.9 – Вікно редагування запису класу

У конструкторі класу кожен шлях перевіряється на існування, після чого будується мініатюрне зображення розміром 150×150 пікселів. Ці Image-елементи автоматично розміщуються в гнучкому WrapPanel, що дозволяє бачити усі фото в одну стрічку із перенесенням на наступний рядок при нестачі ширини екрану. Обгорнутий у ScrollViewer, WrapPanel дає змогу прокручувати вікно й переглянути всі фотографії в обраній папці.

PhotoGalleryWindow є простим і зручним інструментом швидкого візуального огляду зразків зображень для переліку об’єктів, не потребуючи зовнішніх переглядачів або додаткових діалогів.

Вікно редагування дає змогу безпомилково оновлювати назву й шлях кожного типу об’єкта перед подальшим тренуванням або класифікацією.

4.7.3 Вікно керування роботом

Вікно «ArmManage» дає оператору можливість обрати та встановити з'єднання з послідовним портом робота, змінювати порт для підключення, налаштувати координати для ручного керування роботом, керувати механізмом захоплення та відпускання об’єкта.

Метод InitializeRobot у класі App відповідає за створення нового або оновлення вже існуючого екземпляра контролера роботизованого маніпулятора на основі обраного користувачем послідовного порту. Він оголошений як статичний, щоб бути доступним з будь-якої частини програми без потреби створювати новий об’єкт App.

Першим кроком метод InitializeRobot перевіряє, чи вже існує раніше створений об’єкт Robot: якщо так, то він викликає у нього метод Dispose(), щоб закрити відкритий серійний порт і коректно звільнити всі ресурси, після цього виконується створення нового екземпляра RobotController.RobotController, якому передається назва порту (portName) і опційно швидкість з’єднання (baudRate, за замовчуванням 115200 бод). Після виходу з InitializeRobot в статичному полі App.Robot зберігається новий об’єкт, який готовий приймати подальші команди руху та захоплення з будь-якої точки програми. Це забезпечує безперервність роботи: попереднє з’єднання закривається належним чином, а новий контролер одразу доступний для всіх викликів, що надалі надходять від оператора чи логіки сортування.

4.7.4 Вікно налаштування та навчання ШІ

Вікно «AIManage» надає оператору засоби для налаштування параметрів моделі, формування та запуску навчання, а також тестування моделі на вибраній директорії з фотографіями.

Реалізує вікно клас AIManage, який має ряд властивостей та методів. Властивості класу AIManage наведено в таблиці 4.5, методи в таблиці 4.6.

Таблиця 4.5 – Властивості класу AIManage

Властивість	Тип	Призначення
Model	ModelConfig	Зберігає поточну конфігурацію AI-моделі (ім’я, опис, поріг впевненості), двосторонньо зв’язана з елементами інтерфейсу
TrainedObjects	ObservableCollection<ObjectConfig>	Перелік об’єктів із статусом Trained, що відображається в розділі «Робота з моделлю»
InTsvObjects	ObservableCollection<ObjectConfig>	Перелік об’єктів зі статусом InTsv, які вже включені у TSV для навчання
ForTrainingObjects	ObservableCollection<ObjectConfig>	Узагальнений список об’єктів для донавчання (сукупність Trained, InTsv та Added)
TestResults	ObservableCollection<TestResultModel>	Колекція результатів тестування (шлях до фото, передбачений клас, відсоток упевненості), що прив’язується до таблиці

Таблиця 4.6 – Методи класу AIManage

Метод	Сигнатура	Опис
refreshObjList	private void refreshObjList()	Очищає і заново наповнює колекції TrainedObjects, InTsvObjects та ForTrainingObjects згідно з поточним статусом кожного класу у конфігурації
EnsureTsvFileExistsAndClear	private async Task<bool> EnsureTsvFileExistsAndClear(string tsvFile, IProgress<string> logReporter)	Перевіряє наявність TSV-файлу, створює його, якщо відсутній, або очищає, якщо існує; повертає false при помилці, щоб зупинити процес
ProcessTrainingObjects	private async Task<bool> ProcessTrainingObjects(string tsvFile, IProgress<string> logReporter)	Для кожного об’єкта у списку ForTrainingObjects метод перевіряє, що відповідна папка існує та містить достатню кількість зображень, записує рядки у TSV-файлу та змінює статус об’єкта на InTsv. Повертає false при будь-якій помилці

Продовження таблиці 4.6

Метод	Сигнатура	Опис
RunTraining	private async Task<ITransformer> RunTraining(string tsvFile, IProgress<int> progressReporter, IProgress<string> logReporter)	У фоновому потоці метод викликає метод TrainModel модуля ШІ, передаючи шлях до папки з навчальними даними та TSV-файли, а також репортери прогресу й логу. Повертає навчену модель ITransformer
StartTrainingButton_OnClick	private async void StartTrainingButton_OnClick (object? sender, RoutedEventArgs e)	Основний обробник кнопки «Почати навчання»: створює репортери, очищає/підготовлює TSV, обробляє об’єкти, запускає тренування, оновлює статуси, оновлює прогресбар і журнал, показує повідомлення про успіх або помилку
UpdateTrainingObjectsStatusToTrained	private void UpdateTrainingObjectsStatus ToTrained()	Після успішного тренування змінює статус усіх об’єктів списку ForTrainingObjects на Trained через метод оновлення конфігурації
TestResultsDataGrid_DoubleTapped	private async void TestResultsDataGrid_Double Tapped(object? sender, TappedEventArgs e)	При подвійному кліку по рядку результатів тестування відкриває вікно TestResultDetailsWindow, передаючи йому вибраний TestResultModel для детального перегляду
SaveChangesBtnClick	private async void SaveChangesBtnClick(object? sender, RoutedEventArgs e)	Записує змінені значення опису та порогу впевненості у конфігурацію, викликає метод _configService.SaveConfig() і показує повідомлення про успішне (або невдале) збереження конфігурації у XML-файлі
SelectTestDirectoryButton_OnClick	private async void SelectTestDirectoryButton_O nClick(object? sender, RoutedEventArgs e)	Відкриває діалог вибору папки з тестовими зображеннями, перевіряє її існування й наявність файлів .png/.jpg, очищає попередні результати й запускає асинхронну класифікацію кожного зображення, вимірюючи загальний час

Вигляд вікна «AIManage» показано на рисунках 4.10 та 4.11.

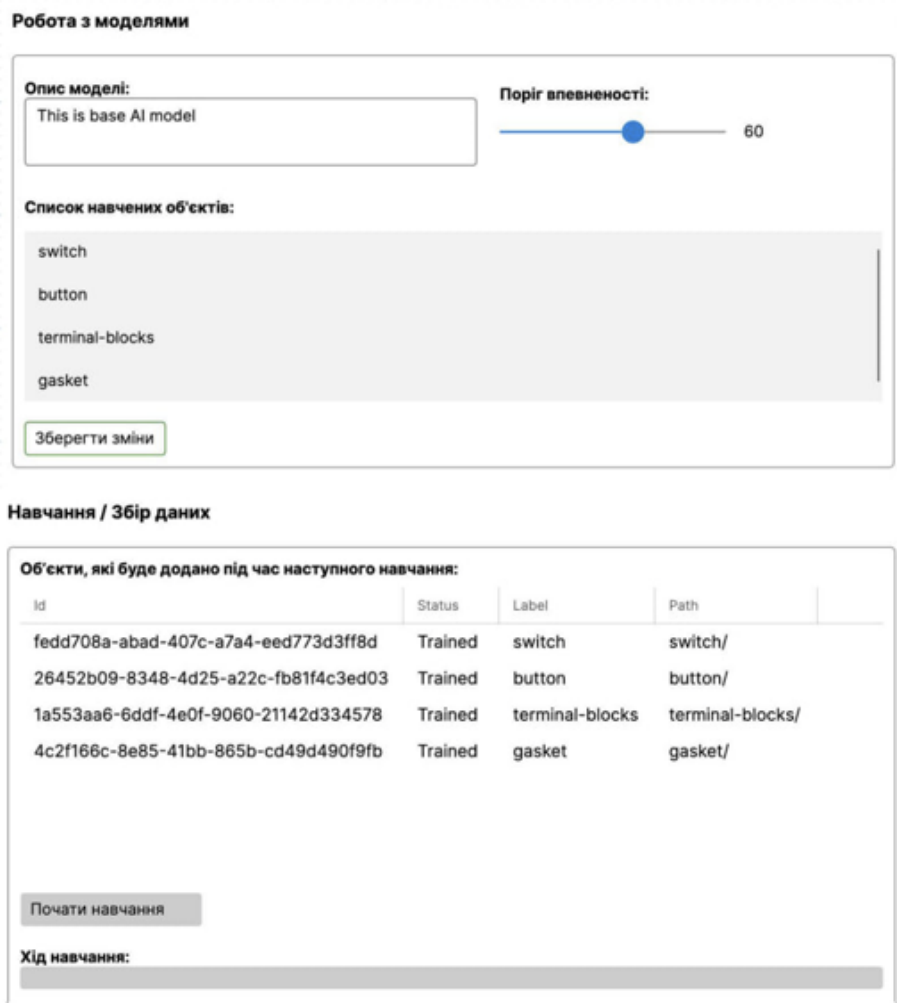


Рисунок 4.10 – Вікно налаштування та навчання ШІ

Рисунок 4.11 – Вікно налаштування та навчання ШІ

4.8 Прошивка камери

Камера для використання програми має бути прошита спеціальною прошивкою для ESP32-CAM, вона налаштовує модуль камери OV2640, підключається до Wi-Fi і потім запускає HTTP-сервер, який вміє видавати як потокове MJPEG-зображення, так і окремі JPEG-кадри.

У setup() відбувається початкова ініціалізація:

– Serial.begin() для налагодження;

– cam.init(…) для налаштування камери;

– підключення до Wi-Fi з відображенням IP-адреси;

– реєстрація HTTP-маршрутів (/mjpeg, /, /jpg) та запуск сервера;

– за потреби запуск RTSP-серверу й створення екземпляра OV2640Streamer.

У функції loop() на вході викликається server.handleClient(), щоб обробити HTTP-запити, а також перевіряються й обслуговуються сесії RTSP, виконуючи відеопотік кадрів з фіксованою затримкою для підтримки обраного FPS.

Після завантаження скетчу в середовищі Arduino IDE та перезавантаження модуля в Serial Monitor буде виведено точки підключення до Wi-Fi та IP-адресу.

Щоб переглянути відеопотік треба відкрити у браузері http://<IP>/mjpeg для потокового MJPEG або http://<IP>/ для аналогічного потоку. Одне зображення з камери JPEG можна отримати за http://<IP>/jpg.

Усі ресурси (Wi-Fi, серійний порт камери) закриються автоматично при виключенні пристрою.

Таким чином, цей код перетворює ESP32-CAM на універсальний HTTP відеосервер із мінімальними налаштуваннями. Код наведено в додатку А.

4.9 Результати випробувань

Щоб підтвердити, що система дійсно відповідає поставленим задачам та мета роботи досягнута необхідно провести ряд експериментів. Проведення експериментів допомагає також визначити наявність помилок, непередбачуваних подій та неточностей у розрахунках.

Перед початком роботи з програмою необхідно провести навчання моделі ШІ класифікувати об’єкти, які в подальшому буде використано для експериментів. Для проведення експериментів буде використано такі електричні елементи як з’єднувач (рис. 4.12, а), кнопка (рис. 4.12, б) та перемикач (рис. 4.12, в). Провівши навчання моделі на зображеннях, що було скачено з інтернету, можна провести тестування для перегляду результатів класифікації зображень тестового набору.

Після успішного проведення навчання необхідно підготувати об’єкти для виконання сортування, поклавши об’єкти на робочу поверхню. Підключивши камеру та робота, проводиться аналіз робочої поверхні та зазначення сортувальних боксів (рис. 4.13). Координати об’єктів було визначено вірно.

а) б) в)

а) з’єднувач; б) кнопка; в) перемикач

Рисунок 4.12 – Електричні компоненти для проведення тестування

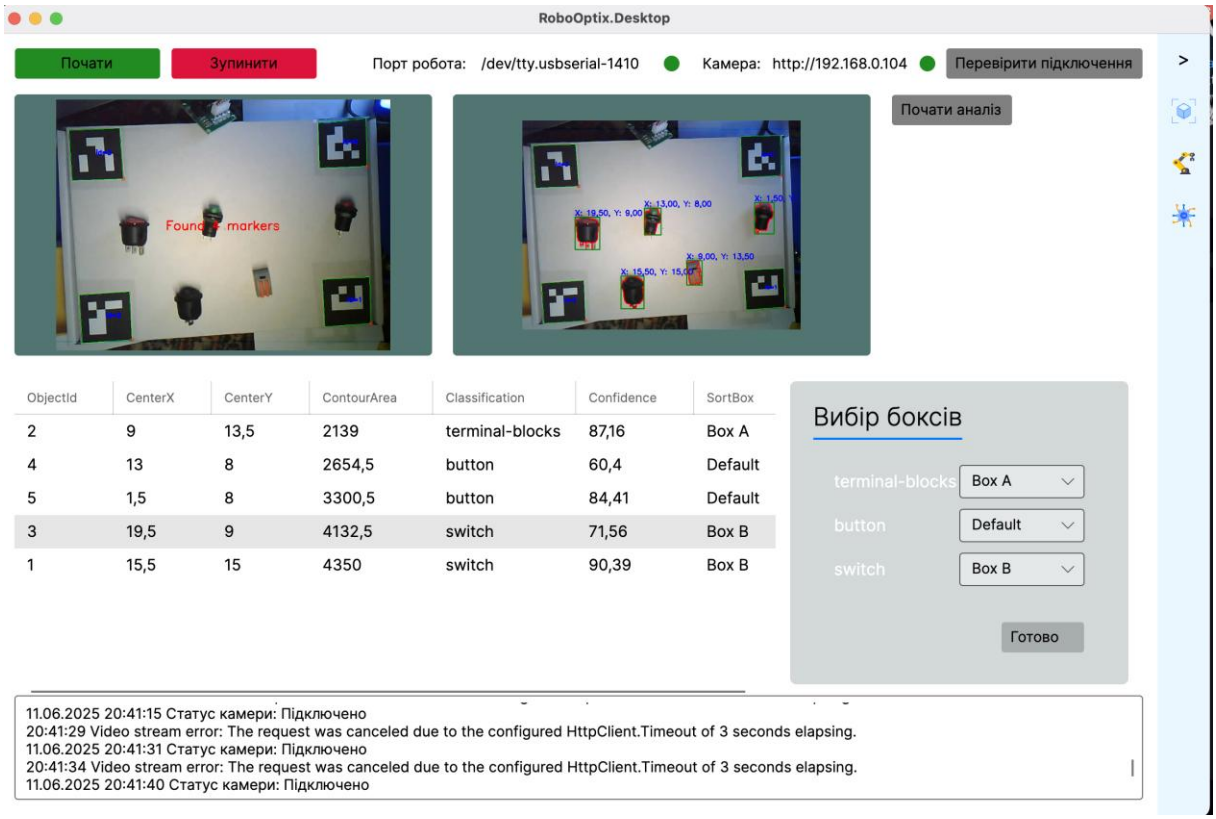


Рисунок 4.13 – Вікно програми при підготовці до процесу сортування

Запустивши процес сортування, було отримано результат: два перемикача було перенесено до сортувального боксу з назвою «Box B», дві кнопки було перенесено до сортувального боксу з назвою «Default», з’єднувач перенесено до сортувального боксу з назвою «Box A».

Проведення експерименту показало, що координати було визначено вірно, оскільки робот зміг захопити та перенести об’єкти до відповідних сортувальних боксів. В ході проведення експерименту один раз було втрачено зв’язок з камерою, відновити довелося вручну, кнопкою «Перевірити підключення».

Проведено ще декілька циклів сортування для перевірки стійкості системи до навантаження. В результаті цих число циклів сортування було перенесено 50 об’єктів, відеопотік було втрачено 2 рази, загальний час сортування склав 9 хилин та 5 секунд, всі координати було визначено вірно, всі об’єкти було перенесено відповідно до зазначених сортувальних

боксів, 12 об’єктів було класифіковано невірно. Похибка класифікації об’єктів складає 24%.

Оскільки, в попередніх циклах було проведено визначення об’єктів та деяка їх частина була класифікована невірно, зображення цих об’єктів було додано до навчального набору та проведено повторне навчання. Після проведення повторних циклів сортування похибка визначення класу об’єкта знизилася до 4%.

Отже, з отриманих результатів можна визначити, що мета була досягнена, оскільки визначення координат об’єктів не мало значних помилок, робот виконав перенесення всіх об’єктів відповідно до призначених сортувальних боксів, втрати зв’язку з камерою були нечастими та швидко відновлювалися, після проведення додаткового навчання похибка класифікації значно зменшилася, а значить, при великій кількості різноманітних навчальних даних модель може виконувати класифікацію майже безпомилково.

4.10 Розрахунок показників стійкості системи

Для визначення стійкості системи буде використовуватися лінійна САУ, яка складається з трьох об’єктів: сенсора (камера), комп’ютера та робота маніпулятора. Для представлення об’єктів у вигляді передатних функцій буде достатньо використовувати аперіодичні ланки першого порядку [37]. Для сенсора передавальна функція буде:

$$W_{\text{сен}}(s) = \frac{1}{0.033s+1}. \quad (4.1)$$

Для комп’ютера передавальна функція буде:

$$W_{\text{ком}}(s) = \frac{1}{0.05s+1}. \quad (4.2)$$

Для робота передавальна функція буде:

$$W_{\text{роб}}(s) = \frac{1}{0.7s+1}. \quad (4.3)$$

Передавальна функція розімкненої системи буде:

$$W_{\text{роб}}(s) = \frac{1}{(0.033s+1)\,(0.05s+1)(0.7s+1)} \quad (4.4)$$

Для визначення стійкості системи використовується критерій Гурвіца, за яким всі визначники матриці Гурвіца мають бути додатніми. Треба скласти характеристичне рівняння системи виду $a_0\lambda^n + a_1\lambda^{n-1} + \ldots + a_n$:

$$0{,}001155\lambda^3 + 0{,}05975\lambda^2 + 0{,}783\lambda + 1 = 0.$$

Визначники будуть наступними:

$$a_0 = 0{,}001155 > 0;$$

$$\Delta_2 = a_1a_2 - a_0a_3 = 0.0456 > 0;$$

$$\Delta_3 = a_3\Delta_2 = 0.0456 > 0.$$

Оскільки всі визначники Гурвіца більше нуля, то систему можна вважати стійкою.

5 ОХОРОНА ПРАЦІ

5.1 Робота з персональним комп’ютером

Робота за комп’ютером належить до категорії робіт з підвищеним напруженням зору та статичним навантаженням на опорно-руховий апарат. Для забезпечення безпеки та збереження здоров’я користувача необхідно дотримуватись певних вимог охорони праці.

При розміщенні елементів робочого місця слід враховувати: робочу позу користувача, простір для розміщення користувача, можливість огляду елементів робочого місця, можливість огляду простору поза межами робочого місця, можливість робити записи, розміщувати на робочому столі документацію та матеріали, які використовує користувач. Розміщення елементів робочого місця не має заважати рухам та переміщенню для експлуатування персонального комп’ютера (ПК) [38].

Тривалість безперервної роботи за комп’ютером не повинна перевищувати 2 години. Після кожної години роботи необхідно робити перерву 15 хвилин для відпочинку очей і зміни положення тіла [38].

ПК треба встановлювати на рівній твердій поверхні, не дозволено встановлювати ПК та оргтехніку на хитких підставках чи на похилій поверхні, впритул до стіни, перегородки тощо. Не можна допускати загородження вентиляційних отворів ПК сторонніми предметами. Розетка біля ПК має бути в доступному місці, щоб в аварійних випадках можна було своєчасно його відімкнути. Не можна допускати ушкодження чи модифікування шнура живлення, заборонено ставити важкі речі на шнур живлення, тягнути чи надмірно перегинати його, скручувати та перев’язувати шнур живлення вузлом [38].

Робота з комп’ютером потребує особливої уваги до безпеки праці, оскільки тривале використання ПК призводить до статичного

навантаження на опорно-руховий апарат, зорового перенапруження, нервової напруги та ризику електробезпекових інцидентів, тому важливо передбачати перерви та забезпечувати справність електрообладнання.

5.2 Робота з рухомими об’єктами

Використання автоматизованих технічних засобів і промислових роботів значно підвищує продуктивність праці, але водночас створює додаткові ризики для персоналу. Тому на підприємстві, де діють рухомі об'єкти (маніпулятори, мобільні платформи, конвеєри тощо), необхідно суворо дотримуватись вимог охорони праці.

Кожен працівник має проходити навчання інструктаж перед тим, як приступити до роботи з рухомими об’єктами. Інструктаж має проводитися як первинний, так і періодичний [39].

Робоча зона має бути обмежена фізичним бар’єром. Обладнання має бути оснащенним спеціальним режимом роботи для уповільнення швидкості переміщення, щоб знизити ризик працівнику під час наближення до обладнання. Треба забезпечити на виробництві проведення регулярного технічного обслуговування зменшить ризики непередбачуваних поломок техніки [39].

Працівники мають носити засоби індивідуального захисту такі як технічні жилети, спеціальний робочий одяг та взуття [39].

Робота поруч із рухомими роботами пов’язана з небезпекою ураження або травм через зіткнення, несподівані рухи чи несправність систем. Важливими є наявність огороджень, сигналізації, забезпечення технічного обслуговування й чітке регламентування допуску працівників.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено програмне забезпечення для налаштування та керування процесом сортування роботом-маніпулятором з модулем комп’ютерного зору та елементами штучного інтелекту. Для демонстрації роботи розробленого програмного забезпечення було зібрано макет, який включає в себе робот маніпулятор та камеру.

Розроблено алгоритм класифікації зображень, що використовує попередньо навчену модель штучного інтелекту, використовуючи додаткові бібліотеки. Середня точність класифікації на тестовому наборі даних складає 86% впевненості.

Проведені тестування роботи макету показали стабільну роботу системи при навантаженні до 30 об’єктів поспіль без втрати точності та перебоїв зв’язку.

Використання розробленного програмного забезпечення дозволяє автоматизувати процес сортування об’єктів та покращити його шляхом впровадження штучного інтелекту для визначення типів об’єктів за певними класам.

Матеріали за тематикою роботи було опубліковано у статті.

Проведені дослідження відповідають цілям сталого розвитку, зокрема Цілі сталого розвитку 9 – «Створення стійкої інфраструктури, сприяння всеохоплюючій і сталій індустріалізації та інноваціям» [3].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. - 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В Сичова. – Харків: ХНУРЕ, 2022. – 55 с.

3. United Nations. The 17 Sustainable Development Goals. United Nations, 2025, Режим доступу: sdgs.un.org/goals (дата звернення: 05.06.2025).

4. Malche T. Automated Sorting with Computer Vision. Roboflow Blog. URL: https://blog.roboflow.com/automated-sorting-with-computer-vision/ (date of access: 22.04.2025).

5. The Ins & Outs of Industrial Robot Arms | Industrial Robots by Kawasaki Robotics. Kawasaki Robotics. URL: https://kawasakirobotics.com/blog/the-ins-outs-of-industrial-robot-arms/ (date of access: 22.04.2025).

6. Cartesian Robots Guide: Structure, Applications, And Benefits. EngiTech. URL: https://engitech.in/cartesian-robots-guide/#what-are-cartesian-robots-h-1 (date of access: 22.04.2025).

7. Cartesian Robots. E-MotionSupply. URL: https://www.e-motionsupply.com/category_s/4194.htm (date of access: 22.04.2025).

8. Конференції Державного університету «Житомирська політехніка». URL: https://conf.ztu.edu.ua/wp-content/uploads/2022/02/133.pdf (дата звернення: 22.04.2025).

9. DIY SCARA Robot Arduino Professional Level | MULTAN ELECTRONICS. Your Source for Arduino, Sensors, 3D Printers, and More | MULTAN ELECTRONICS. URL: https://www.multanelectronics.com/product/professional-diy-scara-robot-build-your-own-arduino-based-robot/ (date of access: 22.04.2025).

10. Історія компанії FANUC - FANUC. FANUC - World market leader in factory automation. URL: https://www.fanuc.eu/ua/uk/про-нас/історія-fanuc (дата звернення: 22.04.2025).

11. Промислові роботи FANUC для інтелектуальної автоматизації - FANUC. FANUC - World market leader in factory automation. URL: https://www.fanuc.eu/ua/uk/роботи (дата звернення: 22.04.2025).

12. Промислові роботи FANUC серії LR Mate - FANUC. FANUC - World market leader in factory automation. URL: https://www.fanuc.eu/ua/uk/роботи/robot-filter-page/серія-lrmate (дата звернення: 22.04.2025).

13. Industrial robot | KUKA AG. KUKA AG. URL: https://www.kuka.com/en-se/products/robotics-systems/industrial-robots (date of access: 22.04.2025).

14. KR DELTA robot in Hygienic Design | KUKA AG. KUKA AG. URL: https://www.kuka.com/en-se/products/robotics-systems/industrial-robots/kr-delta-robot-hm (date of access: 22.04.2025).

15. Методичні вказівки до лабораторних робіт з дисципліни „Гнучкі комп'ютеризовані виробничі комплекси” для студентів усіх форм навчання спеціальності 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка, / Упоряд. Гурін Д. В., Євсєєв В.В., − Харків: ХНУРЕ. − 2024. − 56 с.

16. Weiterleitungshinweis. Google. URL: https://www.google.com/url?sa=i&url=https://boofcv.org/index.php?title=Example_Canny_Edge&psig=AOvVaw0WMCwSGiDeHmYYOwMVd9s

U&ust=1745437573544000&source=images&cd=vfe&opi=89978449&ved=0CBcQjhxqFwoTCICpwYG07IwDFQAAAAAdAAAAABAQ (date of access: 22.04.2025).

17. Semantic segmentation in AI, principle and applications. Innovatiana - Data Labeling Outsourcing with impact!. URL: https://en.innovatiana.com/post/semantic-segmentation-in-ai (date of access: 22.04.2025).

18. OpenCV: Detection of ArUco Markers. OpenCV documentation index. URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (date of access: 22.04.2025).

19. Моделі ШІ та бізнес-сценарії. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/ai-builder/model-types (дата звернення: 22.04.2025).

20. Malche T. Automated Sorting with Computer Vision. Roboflow Blog. URL: https://blog.roboflow.com/automated-sorting-with-computer-vision/#how-to-build-an-automated-sorting-system (date of access: 22.04.2025).

21. Malche T. Building a Computer Vision Assisted Pill Inspection System. Roboflow Blog. URL: https://blog.roboflow.com/computer-vision-pill-inspection-quality-assurance/ (date of access: 22.04.2025).

22. Що таке Tensorflow? oksim. URL:https://www.oksim.ua/2023/08/07/shho-take-tensorflow/#Для_чого_використовується_TensorFlow (дата звернення: 09.05.2025).

23. Патерни проектування та переваги їх використання. FoxmindEd. URL: https://foxminded.ua/paterny-proektuvannia/ (дата звернення: 09.05.2025).

24. MVC: шаблон проектування архітектури додатку | студія Brander. Створення і розробка інтернет-магазинів від Brander. URL: https://brander.ua/technologies/mvc (дата звернення: 09.05.2025).

25. OpenCV: Mat - The Basic Image Container. OpenCV documentation index. URL:https://docs.opencv.org/4.x/d6/d6d/tutorial_mat_the_basic_image_container.html (date of access: 12.05.2025).

26. Методичні рекомендації для проведення практичних занять з навчальної дисципліни «ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ В АВТОМАТИЗОВАНИХ ТА ІНФОРМАЦІЙНИХ СИСТЕМАХ» для здобувачів вищої освіти освітнього ступеня «магістр» спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» освітньо-професійна програма «Комп‘ютеризовані інформаційновимірювальні системи» / Упоряд.: Ю.О. Подчашинський, О.О. Лугових. Житомир, Житомирська політехніка, 2020. – 65с.

27. Tutorial: ML.NET classification model to categorize images - ML.NET. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/dotnet/machine-learning/tutorials/image-classification (date of access: 14.05.2025).

28. BasuMallick C. What Is TensorFlow? Meaning, Working, and Importance. spiceworks. URL: https://www.spiceworks.com/tech/devops/articles/what-is-tensorflow/ (date of access: 14.05.2025).

29. Матвієнко С. Програмування в ARDUINO камери в ESP32-CAM – IT Master - електроніка та програмування. Головна – IT Master - електроніка та програмування. URL: https://itmaster.biz.ua/electronics/esp32/esp32-cam-camera-arduino.html (дата звернення: 16.05.2025).

30. Кроковий двигун NEMA 17 (42 мм), 1,8 град/крок, 70 Н·см: ціна, купити в Україні. серводвигуни від PROMZONA.VN.UA - 520620415. PROMZONA.VN.UA - Електротехніка, КВПіА, привідна техніка, промислова автоматика, товари, ціна, купити, в Україні. URL:

https://promzona.vn.ua/ua/p520620415-shagovyj-dvigatel-nema.html (дата звернення: 16.05.2025).

31. Arduino Mega 2560 R3 (CH340) плата мікроконтролера. Arduino KIT. URL: https://arduinokit.com.ua/ua/p1080109370-arduino-mega-2560.html (дата звернення: 16.05.2025).

32. Плата Arduino Uno R3 CH340 (Ревізія 2020 ATmega328P). Arduino KIT. URL: https://arduinokit.com.ua/ua/p1137698825-plata-arduino-uno.html (дата звернення: 16.05.2025).

33. Плата Arduino з WiFi модулем ESP8266 (WeMos D1) . Arduino KIT. URL: https://arduinokit.com.ua/ua/p1387844419-plata-arduino-wifi.html (дата звернення: 16.05.2025).

34. GitHub - 20sffactory/community_robot_arm: Repository of Community Robot Arm Documents. GitHub. URL: https://github.com/20sffactory/community_robot_arm/tree/master (date of access: 16.05.2025).

35. What is NuGet and what does it do?. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/nuget/what-is-nuget (date of access: 05.06.2025).

36. Що таке serialize та deserialize в JavaScript. IT Blog. URL: https://it-blog.in.ua/shcho-take-serialize-ta-deserialize-v-javascript (date of access: 05.06.2025).

37. Теорія автоматичного управління (збірник задач): навчальний посібник / І. Ш. Невлюдов, О. В. Токарєва. Харків: ХНУРЕ, 2020. 240 с.
35. Perišić, N., & Jovanović, R. (2023). Control of direct current motor by using artificial neural networks in Internal model control scheme. FME Transactions, 51(1), 109-116.

38. Інструкція з охорони праці при роботі на персональному комп’ютері (зразок). Календар бухгалтера. URL:

https://services.uteka.ua/ua/publication/zrazky-34-trudovi-vidnosyny-ta-oplata-pratsi-138-instrukciya-po-oxrane-truda-pri-rabote-na-personalnom-kompyutere-obrazec (дата звернення: 06.06.2025).

39. Про затвердження Правил охорони праці під час експлуатації вантажопідіймальних кранів, підіймальних пристроїв і відповідного обладнання. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0244-18#Text (дата звернення: 05.06.2025).