Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра спеціалізованих комп'ютерних систем

РУСНАК Андрій Михайлович

# СИСТЕМА АВТОМАТИЗОВАНОГО УПРАВЛІННЯ ЗАХВАТОМ МАНІПУЛЯТОРА РОБОТА. / AUTOMATED CONTROL SYSTEM OF THE ROBOTIC MANIPULATPR GRIPPER.

спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
освітньо-професійна програма – Автоматизація та комп'ютерно-інтегровані технології

Випускна кваліфікаційна робота
здобувача першого (бакалаврського) рівня освіти

Виконав: студент групи АКІТ–41
А. М. Руснак
_____
Науковий керівник:
к.т.н., доцент Я. М. Николайчук
_____

Випускну кваліфікаційну роботу
допущено до захисту:
"____"_____2026 р.

Завідувач кафедри СКС
_____ А. І. Сегін

Тернопіль 2026

Факультет комп’ютерних інформаційних технологій
Кафедра спеціалізованих комп’ютерних систем
Освітній ступінь "бакалавр"
спеціальність:151 – Автоматизація та комп’ютерно-інтегровані технології
освітньо-професійна програма – Автоматизація та комп’ютерно-інтегровані технології

**ЗАТВЕРДЖУЮ:**
зав. кафедри СКС
_____ А. І. Сегін
"___"._____2025р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Руснаку Андрію Михайловичу
**(прізвище, ім’я по–батькові)**

**1. Тема кваліфікаційної роботи**
Система автоматизованого управління захватом маніпулятора робота / Automated control system of the robotic manipulator
керівник роботи д.т.н., професор Николайчук Я. М.
затверджено наказом по університету від « 1 » грудня 2025 р. № 894

**2. Строк подання студентом закінченої кваліфікаційної роботи**
25 травня 2026р.

**3. Вихідні дані до кваліфікаційної роботи:**
1. Технічні характеристики 6-осьового роботизованого маніпулятора відкритої архітектури (на базі Annin Robotics AR4).
2. Вимоги до швидкодії та точності системи комп'ютерного зору на базі нейронної мережі (архітектура YOLOv8) та алгоритмів контурного аналізу.
3. Вимоги до алгоритмів автоматичного виявлення помилок та системи замкненого контуру управління (Closed-loop control) роботизованим комплексом.

**4. Основні питання, які потрібно розробити**
1. Провести аналіз сучасних методів автоматизації маніпуляційних процесів та систем комп'ютерного зору.
2. Побудувати кінематичну модель маніпулятора (за методом Денавіта-Хартенберга) та математичний апарат перетворення просторових координат (матриця гомографії).
3. Розробити та навчити нейромережеву модель для розпізнавання об'єктів та обчислення їхньої просторової орієнтації (4D-вектор захоплення).
4. Розробити логіку автоматизованої системи управління на базі кінцевого автомата (FSM) для виправлення помилок маніпулювання.
5. Розробити програмно-апаратний комплекс сортування (на базі мікроконтролера Teensy 4.1 та мови Python) і провести натурні експериментальні дослідження.

**5. Перелік графічного матеріалу у роботі**

1.Загальна структурна схема системи автоматизованого управління маніпулятором._____

2. Кінематична схема 6-осьового маніпулятора AR4 та просторове перетворення координат._____

3. Блок-схема алгоритму комп'ютерного зору (YOLOv8 + OpenCV) та розрахунку вектора захоплення._____

4. Блок-схема алгоритму автоматичного повторного захоплення при випаданні об'єкта._____

5.Таблиця результатів експериментального тестування ефективності системи.___

**6. Консультанти розділів кваліфікаційної роботи**

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | Сегін А. І. | | |
| 2 | Сегін А. І. | | |
| 3 | Сегін А. І. | | |
| | | | |

**7. Дата видачі завдання** 01 грудня 2025р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назви етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз методів автоматизації маніпуляційних процесів, алгоритмів комп'ютерного зору та огляд літературних джерел. | 01.12.2025 р. – 15.01.2026 р. | виконано |
| 2 | Розробка математичного забезпечення, кінематичної моделі маніпулятора та підготовка/навчання нейронної мережі YOLOv8. | 16.01. 2026 р.– 15.03.2026р. | виконано |
| 3 | Апаратне складання експериментального стенда, розробка програмного забезпечення (Python) та проведення натурних досліджень. | 16.03.2026 р. – 30.04.2026 р. | виконано |
| 4 | Остаточне оформлення та подача кваліфікаційної роботи на перевірку щодо плагіату. | 1.05.2026 р. – 25.05.2026 р. | виконано |

Студент _____ Руснак А. М.
(підпис)

Керівник роботи _____ д.т.н. проф. Николайчук Я. М.
(підпис)

# АНОТАЦІЯ

**Руснак А. М.** Система автоматизованого управління захватом маніпулятора.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійною програмою – Автоматизація та комп'ютерно-інтегровані технології. – Західноукраїнський національний університет, Тернопіль, 2026.

Кваліфікаційну роботу присвячено розробці інтелектуальної системи автоматизованого управління (САУ) маніпулятором AR4, оснащеним захоплювачем адаптивного типу. У роботі проведено аналіз сучасних методів автоматизації розпізнавання об'єктів за допомогою нейронних мереж архітектури YOLO.

На основі проведених досліджень розроблено програмне та алгоритмічне забезпечення для автоматизованого визначення координат цілі та розрахунку траєкторії підходу. Математично обґрунтовано процес трансформації координат із системи технічного зору в робочий простір маніпулятора. Реалізовано спеціалізований алгоритм автоматичного виявлення помилок та повторного захоплення об'єкта у разі його втрати.

Впровадження розробленої інтелектуальної САУ дозволяє повністю автоматизувати процес сортування, забезпечити високу гнучкість системи при роботі з предметами довільної форми та підвищити надійність маніпуляційних операцій порівняно з системами жорсткого управління.

**Ключові слова:** автоматизація, маніпулятор AR4, захоплювач адаптивного типу, комп'ютерний зір, YOLO, інтелектуальне управління.

# ANNOTATION

**Rusnak A. M.** Automated control system of the manipulator gripper.

Research for obtaining a bachelor's degree in the specialty 151 – Automation and computer-integrated technologies, educational and professional program – Automation and computer-integrated technologies. – Western Ukrainian National University, Ternopil, 2026.

The qualification work is dedicated to the development of an intelligent automated control system (ACS) for the AR4 manipulator equipped with an adaptive type gripper. The work analyzes modern methods of automated object recognition using YOLO architecture neural networks.

Based on the research, software and algorithmic support for automated target coordinate determination and approach trajectory calculation have been developed. The process of coordinate transformation from the computer vision system to the manipulator's workspace is mathematically justified. A specialized algorithm for automated error detection and re-grasping of an object in case of its loss is implemented.

The implementation of the developed intelligent ACS allows for complete automation of the sorting process, ensuring high system flexibility when working with objects of arbitrary shapes and increasing the reliability of manipulation operations compared to rigid control systems.

**Keywords:** automation, AR4 manipulator, adaptive type gripper, computer vision, YOLO, intelligent control.

ЗМІСТ

# ВСТУП

**Актуальність теми.** На сучасному етапі розвитку промисловості автоматизація виробничих процесів є одним із пріоритетних завдань. Використання роботизованих маніпуляторів дозволяє значно прискорити роботу та підвищити її точність, проте більшість існуючих систем автоматизованого управління працюють за жорстко заданими алгоритмами. Це створює труднощі при роботі з об’єктами, що мають нестандартну форму або змінюють своє положення у просторі.

Для вирішення цих проблем необхідно впроваджувати системи технічного зору, які дозволяють маніпулятору «бачити» робочу зону та адаптуватися до змін. Використання сучасних алгоритмів на базі нейронних мереж YOLO дає змогу системі самостійно знаходити предмети на зображенні з камери та визначати їхні координати в реальному часі. Окрім цього, важливу роль відіграє використання захоплювачів адаптивного типу, які завдяки своїй гнучкості можуть надійно утримувати крихкі предмети та об'єкти складної форми. Особливо актуальною є розробка алгоритмів, що дозволяють маніпулятору AR4 самостійно виявляти помилки, такі як випадання предмета, та автоматично повторювати цикл захоплення. Це дозволяє зробити процес автоматизації більш надійним і автономним.

**Мета роботи.** Розробка інтелектуальної системи автоматизованого управління маніпулятором AR4 із застосуванням комп’ютерного зору та захоплювача адаптивного типу для забезпечення автономного циклу сортування об’єктів.

**Завдання роботи:**

1. Проаналізувати сучасний стан автоматизації маніпуляційних систем та методи комп'ютерного зору для розпізнавання об'єктів.
2. Побудувати кінематичну модель маніпулятора AR4 та розробити математичні формули для перетворення координат із системи зору в робочий простір.

3. Описати принципи та алгоритмічні основи роботи нейронної мережі YOLO для автоматизації процесів ідентифікації об'єктів.
4. Розробити алгоритм автоматичного виявлення помилок захоплення та логіку повторного маніпулювання у разі втрати предмета.
5. Програмно реалізувати систему управління та провести експериментальне дослідження її ефективності.

**Об’єкт дослідження.** Процес автоматизованого управління роботами маніпуляторами із застосуванням інтелектуальних систем розпізнавання.

**Предмет дослідження.** Алгоритми автоматизованого керування роботом AR4 для захоплювача адаптивного типу, з використанням методів комп’ютерного зору на базі YOLOv8 та OpenCV .

**Практичне значення.** Полягає у розробці автономної системи управління, яка дозволяє маніпулятору самостійно орієнтуватися у робочому просторі та підлаштовуватися під хаотичне розташування деталей. Головною перевагою створеного комплексу є алгоритм автоматичного виявлення помилок: якщо деталь випадково вислизає із захоплювача, робот самостійно це фіксує та ініціює повторну спробу захоплення. Це значно зменшує час простою обладнання та зводить до мінімуму необхідність постійного контролю і ручного втручання людини-оператора у процес сортування.

**Напрями подальшого розвитку.** Перспективним вектором для вдосконалення системи є перехід від розпізнавання на площині до повноцінного 3D-зору. Застосування камер із датчиками глибини (RGB-D) дозволить маніпулятору сортувати деталі, які лежать хаотично одна на одній у контейнерах (задача «bin picking»). Крім того, логіку управління можна розширити для захоплення об'єктів безпосередньо з рухомого конвеєра, а не лише зі статичного столу.

# 1 АНАЛІЗ МЕТОДІВ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ РОЗПІЗНАВАННЯ ТА МАНІПЮЛЮВАННЯ ОБ’ЄКТАМИ

## 1.1 Сучасні методи автоматизації роботизованих маніпуляційних систем

Розвиток сучасного промислового виробництва та складської логістики нерозривно пов'язаний із процесами автоматизації. На сьогоднішній день використання роботизованих маніпуляторів є стандартом для виконання завдань, пов'язаних із переміщенням, сортуванням, пакуванням та складанням деталей. Проте підходи до управління такими системами зазнали суттєвих змін протягом останніх десятиліть.

Класичні системи автоматизованого управління (САУ) маніпуляторами базувалися виключно на жорсткому програмуванні траєкторій. У таких системах оператор або програміст задавав точні координати в просторі, а також кути повороту кожної осі робота. Маніпулятор виконував рухи за принципом переміщення від однієї фіксованої точки до іншої [1]. Цей метод є дуже ефективним для масового виробництва, де всі процеси суворо структуровані. Наприклад, якщо деталь завжди подається на конвеєр в одному і тому ж положенні, з точністю до міліметра, класичний робот може працювати роками без помилок.

Однак у реальних умовах, особливо на лініях сортування або при роботі з об'єктами різних габаритів, предмети рідко лежать ідеально рівно. Якщо об'єкт змістився, перекинувся або має нестандартну форму, жорстко запрограмований робот опустить свій захват у порожнє місце або, що ще гірше, розчавить деталь, намагаючись стиснути її у неправильній точці. Це призводить до зупинки всієї виробничої лінії та потребує ручного втручання оператора для перезапуску системи.

Для вирішення проблеми жорсткої прив'язки до координат сучасні методи автоматизації переходять до використання технологій комп'ютерного зору. Суть цього методу полягає в тому, що маніпулятор не отримує готових

координат заздалегідь. Замість цього система оснащується камерою, яка робить знімок робочої зони. Програмне забезпечення аналізує це зображення, знаходить об'єкт і самостійно розраховує його координати. Лише після цього система управління генерує траєкторію руху для моторів маніпулятора [2].

Для впровадження таких сучасних методів управління на реальних виробництвах зазвичай використовуються дорогі промислові комплекси від провідних виробників (наприклад, KUKA, ABB, Fanuc). Проте на етапі розробки алгоритмів, перевірки гіпотез або для дослідницьких цілей закупівля такого обладнання є економічно недоцільною. Тому в даній кваліфікаційній роботі для експериментальної перевірки теорії застосовується 6-осьовий маніпулятор відкритої архітектури Annin Robotics AR4 [7] (рисунок 1.1).


Рисунок 1.1 – Загальний вигляд 6-осьового маніпулятора відкритої архітектури Annin Robotics AR4

Цей робот є відкритим проектом (open-source), який був створений спеціально для освітніх закладів та інженерів. Його використання дозволяє з мінімальними бюджетними витратами побудувати реальний фізичний стенд з великою досяжністю самого робота, його усі параметри та розміри показано

на рисунку 1.2[8]. На базі AR4 можна повноцінно протестувати можливість виконання складних завдань, таких як розпізнавання об'єктів камерою та динамічне маніпулювання, не ризикуючи пошкодити дороге промислове обладнання. Відкрита архітектура дозволяє напряму підключатися до контролера робота і писати власні програмні скрипти, об'єднуючи управління моторами та дані з камери в єдину систему.

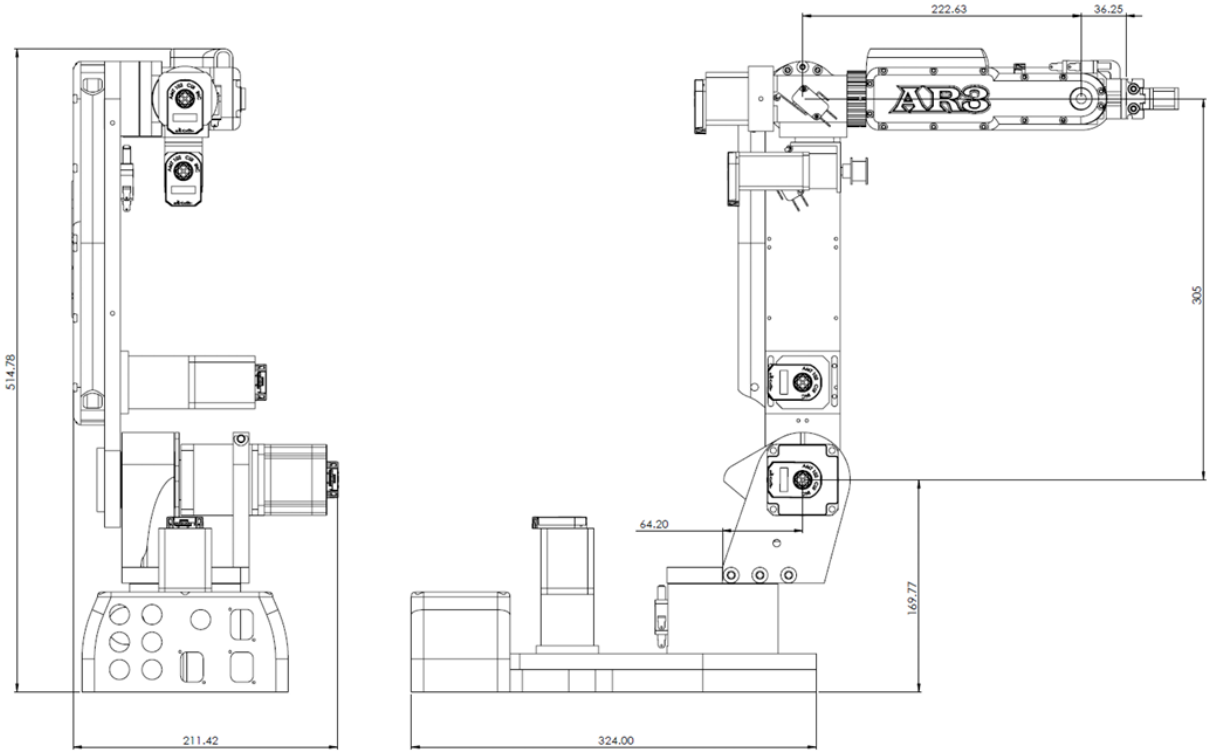


Рисунок 1.2 – Конструктивна схема з габаритними розмірами 6-осьового маніпулятора відкритої архітектури Annin Robotics AR4

Окрім методів розрахунку траєкторії, значної еволюції зазнали і кінцеві виконавчі механізми – захоплювачі. У традиційних промислових системах найчастіше використовуються жорсткі металеві або пластикові губки. Вони вимагають дуже точного розрахунку сили стискання: якщо сила замала, предмет випаде, якщо завелика – предмет може деформуватися. Для нашого експериментального стенду такий підхід є надто складним і негнучким. Тому

сучасний підхід до автоматизації передбачає використання захоплювачів адаптивного типу.

Захоплювач адаптивного типу має гнучку конструкцію. При змиканні він не просто стискає об'єкт з двох боків, а пасивно обгортає його, самостійно підлаштовуючись під його контури [1]. Це дозволяє однією і тією ж програмою, без встановлення складних датчиків сили, надійно брати як тверді деталі правильної форми, так і предмети зі складною геометрією.

Для кращого розуміння відмінностей між традиційними та сучасними експериментальними методами автоматизації процесів маніпулювання, їх основні характеристики зведено у таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика методів управління маніпуляційними системами

| Характеристика | Класичний промисловий метод | Сучасний метод із комп'ютерним зором |
|---|---|---|
| Спосіб позиціонування | Жорсткі координати (фіксовані точки простору) | Динамічний розрахунок координат за допомогою камери |
| Гнучкість системи | Низька (потребує переналаштування при зміні позиції об'єкта) | Висока (алгоритм самостійно адаптується під ситуацію) |
| Тип захватних пристроїв | Жорсткі (пневматичні або механічні губки) | Захоплювачі адаптивного типу (гнучкі матеріали) |
| Реакція на помилки | Зупинка системи, потреба у втручанні оператора | Можливість програмного виявлення та автоматичного повторення |

## 1.2 Аналіз алгоритмів автоматизованого виявлення об’єктів на базі нейронних мереж

Перехід до гнучкої автоматизації та тестування теорії захоплення неможливий без надійного програмного забезпечення. Роботу потрібна

система, яка аналізуватиме зображення з камери і розумітиме, де саме знаходиться деталь і які її габарити.

Довгий час для вирішення цих завдань в автоматизації використовувалися класичні алгоритми комп'ютерного зору. Вони працювали на основі пошуку контурів, виділення країв або фільтрації пікселів за певним кольором [5]. Хоча ці алгоритми швидкі і не вимагають потужних обчислювальних ресурсів, вони є вкрай нестабільними в реальних умовах. Зміна освітлення в кімнаті, поява тіні, блік на деталі або часткове перекриття предмета іншим призводили до того, що програма давала збій.

Для того щоб експериментальна система працювала надійно, наразі використовують методи машинного навчання, а саме штучні нейронні мережі. Найбільшу ефективність у задачах аналізу зображень сьогодні демонструють згорткові нейронні мережі (CNN – Convolutional Neural Networks).

Щоб зрозуміти алгоритм роботи CNN, варто пам'ятати, що комп'ютер бачить фотографію не як цілісний образ, а як величезну матрицю чисел, де кожне число позначає колір та яскравість окремого пікселя. Згорткова нейромережа працює за математичним принципом згортки. Програма створює невелику числову матрицю, яка відіграє роль фільтра, і поступово ковзає цим фільтром по всьому зображенню, як показано на рисунку 1.3[6].

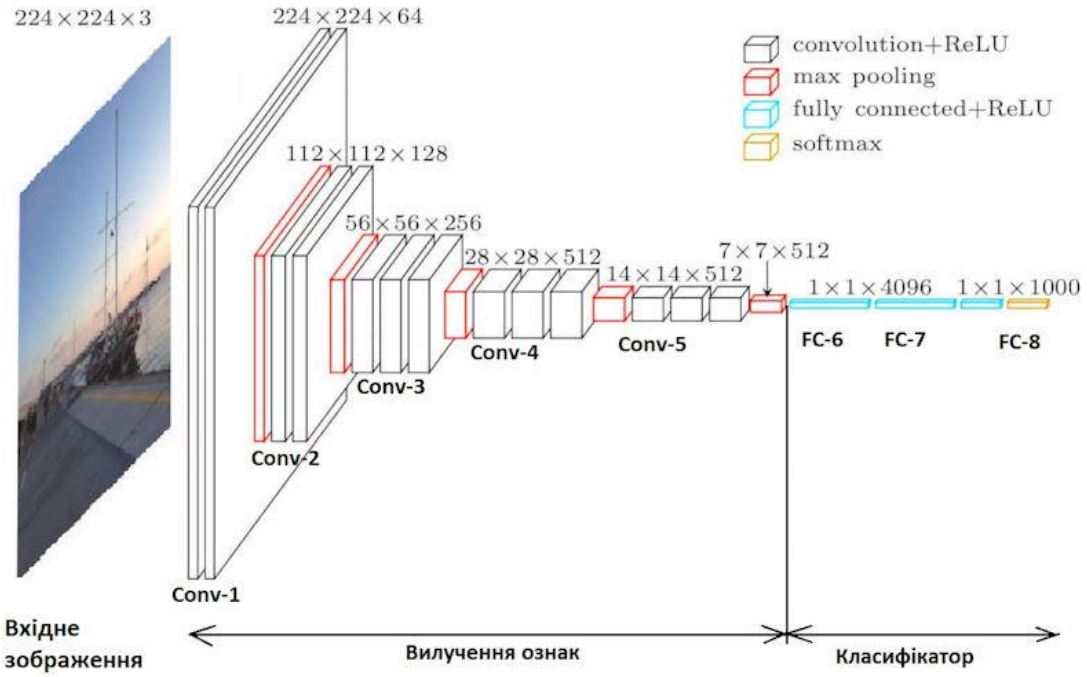


Рисунок 1.3 – Схема процесу згортки та застосування фільтрів у нейронній мережі

На перших рівнях обробки ці фільтри шукають найпростіші речі: прямі лінії, кути або різкі перепади контрасту. Далі ця інформація передається на наступні рівні мережі, де програма починає складати знайдені лінії у прості фігури, а потім — у складні об'єкти. Після кожного застосування фільтрів мережа математично стискає зображення. Цей процес називається підвибіркою[6]. Він необхідний для того, щоб відкинути зайвий фоновий шум і зменшити навантаження на процесор комп'ютера.

Однак, незважаючи на точність класичних згорткових мереж (наприклад, архітектури R-CNN), вони мають один критичний недолік — вони занадто повільні для управління роботами. Алгоритми попереднього покоління спочатку шукають на фотографії тисячі зон, де гіпотетично може бути предмет, і тільки потім аналізують кожну з них. На обробку одного кадру може піти кілька секунд, що робить динамічне управління маніпулятором неможливим.

Саме тому для нашого дослідницького проекту було обрано сучасну архітектуру YOLO (You Only Look Once) [2, 3, 4]. Цей алгоритм кардинально змінює підхід до розпізнавання і дозволяє обробляти відеопотік у режимі реального часу.

На відміну від своїх попередників, мережа YOLO не шукає окремі регіони на фото крок за кроком. Алгоритм бере все зображення цілком і накладає на нього фіксовану математичну сітку(рисунок 1.4). Потім нейронна мережа за один цикл обчислень аналізує всі комірки цієї сітки одночасно. Вона визначає, чи є в центрі якоїсь комірки потрібний нам об'єкт, і миттєво генерує навколо нього обмежувальну рамку (bounding box) [7, 8].

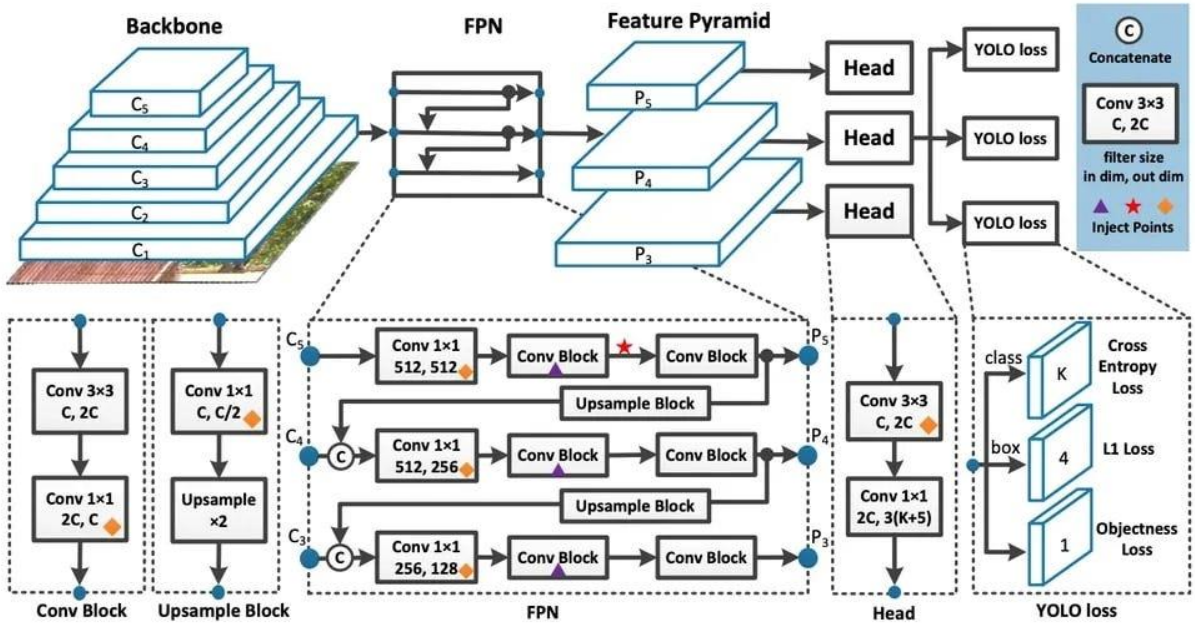


Рисунок 1.4 – Принцип розбиття зображення на сітку та виявлення об'єктів за алгоритмом YOLO

Окрім визначення координат цієї рамки (центр по осях X та Y, а також її ширина та висота), алгоритм видає показник впевненості (Confidence Score). Цей показник демонструє, наскільки нейронна мережа впевнена у своєму рішенні. Завдяки тому, що всі обчислення відбуваються за один прохід, час

розпізнавання одного кадру скорочується до мілісекунд. Це дозволяє камері передавати координати системі управління маніпулятора AR4 практично миттєво. Таким чином, система здатна вчасно розрахувати правильний вектор руху та виконати захоплення деталі, навіть якщо її положення трохи змінилося.

## 1.3 Особливості застосування захоплювачів адаптивного типу в автоматизації

У будь-якій системі автоматизованого маніпулювання кінцевий виконавчий пристрій (захоплювач) відіграє критично важливу роль. Саме він безпосередньо контактує з об'єктом, і від його конструкції залежить успішність виконання всієї операції. Традиційно в промисловості найпоширенішими є жорсткі паралельні або кутові захоплювачі, виготовлені з металу чи твердого пластику (рисунок 1.5) [5, 6]. Вони приводяться в дію пневматичними циліндрами або кроковими двигунами і працюють за принципом лещат, просто стискаючи деталь з двох боків.

Такий підхід ідеально підходить для роботи з міцними деталями правильної геометричної форми (металеві циліндри, куби, пластмасові блоки), коли їхні координати заздалегідь відомі з високою точністю. Однак при сортуванні предметів різної форми, крихких виробів або об'єктів з нерівною поверхнею класичні захоплювачі стикаються з серйозними проблемами [10]. Якщо програма подасть занадто мале зусилля, деталь вислизне під час переміщення маніпулятора. Якщо ж зусилля буде завеликим, жорсткі губки просто розчавлять або подряпають предмет. Для вирішення цієї проблеми виробникам доводиться встановлювати дорогі тензометричні датчики (датчики сили), що значно ускладнює систему керування і робить її дорожчою.

Рисунок 1.5 – Приклади традиційних жорстких механічних захоплювачів

Враховуючи ці недоліки, у сучасній робототехніці активно розвивається використання захоплювачів адаптивного типу (напрямок Soft Robotics). Їхня головна відмінність полягає у матеріалах та біонічних принципах роботи. Замість жорстких шарнірів у них використовуються гнучкі полімери (силікон, поліуретан, гума, TPU пластик), а сама конструкція дозволяє пристрою пасивно деформуватися під час контакту з ціллю [1].

Одним із найпопулярніших механізмів адаптивного захоплення є використання біонічного ефекту Fin Ray (плавника риби). Суть цього ефекту полягає в тому, що коли на гнучку клиноподібну структуру здійснюється боковий тиск, вона не відхиляється вбік, як звичайний важіль. Навпаки — вона згинається назустріч джерелу тиску, охоплюючи його [1]. Завдяки цьому захоплювач не просто тисне на деталь точково, а м'яко обгортає її по всьому контуру, як зображено на рисунку 1.6.

Рисунок 1.6 – Принцип обгортання об'єкта захоплювачем адаптивного типу

Для нашої експериментальної установки на базі маніпулятора AR4 застосування адаптивного захоплювача є не просто покращенням, а технічною необхідністю, що тісно пов'язана з алгоритмами комп'ютерного зору. Як було описано в попередньому підрозділі, нейронна мережа YOLO будує навколо знайденого об'єкта прямокутну обмежувальну рамку і видає координати її центру [7]. Проте геометричний центр цієї 2D-рамки не завжди збігається з реальним центром мас об'єкта, особливо якщо деталь має дуже асиметричну форму.

Якби експериментальна система використовувала жорсткий захоплювач, така алгоритмічна похибка в розрахунку координат навіть на кілька міліметрів призвела б до того, що маніпулятор зачепив би деталь краєм губки і перекинув би її. Захоплювач адаптивного типу повністю нівелює такі похибки позиціонування. Навіть якщо маніпулятор опуститься трохи зміщено відносно ідеального центру, гнучкі пальці під час закриття самі відцентрують об'єкт і надійно зафіксують його завдяки великій площі контакту та збільшеній силі тертя.

Щоб наочно продемонструвати переваги обраного нами підходу, порівняємо характеристики класичних та адаптивних пристроїв. Вони наведені у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз традиційних та адаптивних захоплювачів

| Характеристика | Традиційний жорсткий захоплювач | Захоплювач адаптивного типу |
|---|---|---|
| Матеріали конструкції | Метал, алюміній, твердий пластик | Гнучкі полімери, силікон, TPU |
| Площа контакту з деталлю | Точкова або лінійна (мінімальна) | Поверхнева (повне обгортання по контуру) |
| Вимоги до точності позиціонування | Дуже високі (долі міліметра) | Низькі (система прощає алгоритмічні похибки камери) |
| Робота з крихкими об'єктами | Потребує складних і дорогих датчиків контролю сили | Безпечна завдяки пасивній деформації самого матеріалу |
| Універсальність | Низька (підходить переважно для деталей одного типу) | Висока (один пристрій здатен брати об'єкти різних форм) |

Використання адаптивного захоплювача дозволяє значно спростити програмне забезпечення для управління маніпулятором. Відпадає необхідність у розробці складних математичних моделей для розрахунку зусилля затискання та точного 3D-моделювання центру мас кожного окремого предмета. Дослідницька система стає набагато стійкішою до просторових похибок, що неминуче виникають під час трансформації координат із плоского двовимірного зображення камери у тривимірний робочий простір робота AR4.

# 2 МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАХВАТОМ РОБОТА

## 2.1 Кінематичне моделювання 6-осьового маніпулятора AR4

Для побудови ефективної системи автоматизованого управління будь-яким роботизованим комплексом першочерговим завданням є опис його математичної моделі. Основою такої моделі є кінематичний аналіз. Кінематика маніпулятора описує геометричний зв'язок між кутами повороту кожного з його шести вузлів (моторів) та фінальним положенням кінцевого робочого органа (захоплювача) у тривимірному просторі [11].

Кінематичне моделювання в робототехніці традиційно поділяють на два базові типи задач:

1. Пряма задача кінематики (ПЗК). Полягає у визначенні координат захвату $(X, Y, Z)$ та його просторової орієнтації, виходячи із заданих (відомих) значень кутів повороту кожного з шести суглобів маніпулятора.
2. Зворотна задача кінематики (ЗЗК). Це обернений процес: знаходження необхідних значень кутів повороту моторів для того, щоб вивести захоплювач у конкретну цільову точку простору (наприклад, туди, де лежить деталь).

Маніпулятор AR4 має послідовну кінематичну структуру з шістьма ступенями вільності (рисунок 2.1). Для систематичного математичного опису таких багатоланкових просторових механізмів у світовій практиці використовується метод параметрів Денавіта-Хартенберга (Denavit-Hartenberg, D-H). Згідно з цим методом, за кожною рухомою ланкою робота закріплюється локальна система координат [9].

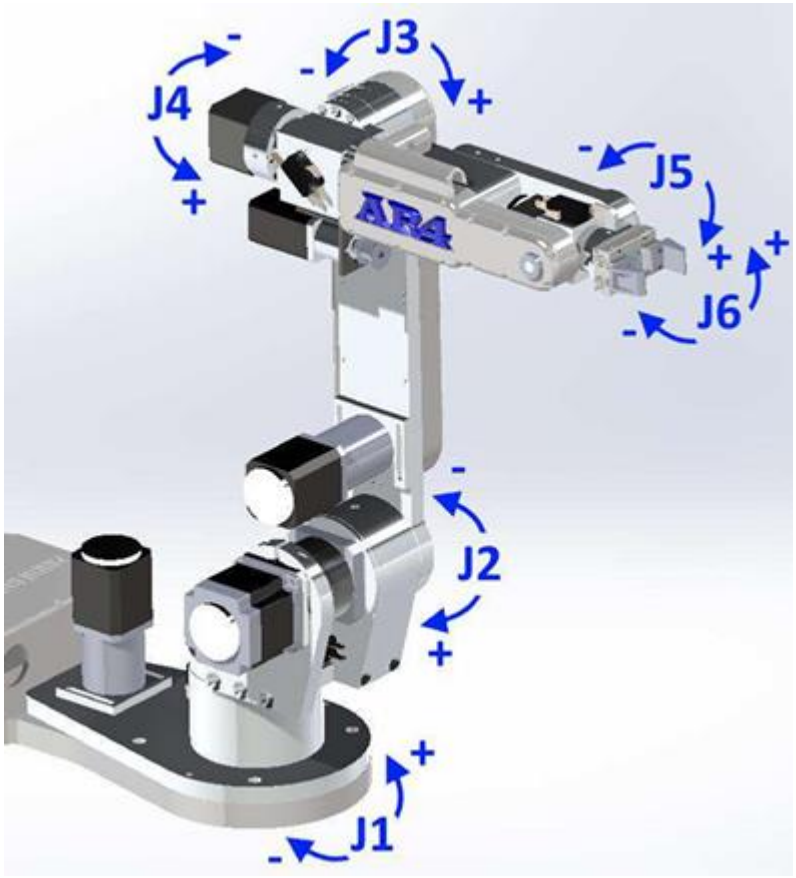


Рисунок 2.1 – Схема розподілу осей та систем координат маніпулятора AR4

Метод Денавіта-Хартенберга описує кожну ланку $i$ за допомогою чотирьох ключових геометричних параметрів:

- $a_i$ (довжина ланки) – відстань між осями обертання, виміряна вздовж осі $X_i$.

  $\alpha_i$ (кут повороту ланки) – кут перекосу між осями $\alpha_i$ та $Z_i$.
- $d_i$ (зміщення ланки) – відстань зсуву між ланками вздовж осі $Z_{\{i-1\}}$.
- $\theta_i$ (кут з'єднання) – змінний кут повороту самого мотора навколо осі $Z_{\{i-1\}}$.

Геометричні параметри для маніпулятора AR4, розраховані на основі його офіційної технічної документації та креслень [12], наведені у таблиці 2.1.

Таблиця 2.1 – Параметри Денавіта-Хартенберга для маніпулятора AR4

| Ланка (i) | $a_i$ (мм) | αi (град) | di (мм) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 (Основа) | 64.2 | 90 | 169.77 |
| 2 (Плече) | 305 | 0 | 0 |
| 3 (Лікоть) | 0 | 90 | 0 |
| 4 (Зап'ястя 1) | 0 | -90 | 222.63 |
| 5 (Зап'ястя 2) | 0 | 90 | 0 |
| 6 (Зап'ястя 3) | 0 | 0 | 36.25 |

Математично перехід від системи координат попередньої ланки до наступної описується матрицею однорідної трансформації $T_i^{\{i-1\}}$. Ця матриця має загальний вигляд, визначений правилами лінійної алгебри для просторових перетворень [9, 10]:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Для знаходження фінального положення захоплювача відносно нерухомої основи робота необхідно послідовно перемножити матриці трансформації для всіх шести ланок:

$$T_6^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \cdot T_5^4 \cdot T_6^5$$

Для підтвердження практичної працездатності кінематичної моделі та верифікації алгоритмів управління виконано чисельний розрахунок просторового положення робочого органу маніпулятора AR4 з урахуванням геометричних параметрів встановленого двопальцевого м'якого захоплювача. В якості цільового об'єкта маніпулювання визначено пластиковий кубик з габаритними розмірами 20x20x20 мм. На основі аналізу геометрії робочого простору експериментального стенда для об'єкта задано реалістичні координати у зоні гарантованої досяжності: X = 250 мм та Y = 250 мм відносно початку координат бази маніпулятора. Задамо вектор узагальнених координат (кутів повороту осей), який забезпечує точне позиціонування м'якого

захоплювача над центром об'єкта: q1 = 45, q2 = 30, q3 = 15, q4 = 0, q5 = 90, q6 = 0 градусів.

Згідно з конструктивними параметрами встановленого виконавчого пристрою, його довжина становить 165.5 мм. Для забезпечення надійного захоплення кубика по всій його висоті та запобігання вислизанню, кінчики еластичних губок м'якого захоплювача повинні опуститися майже до самої поверхні столу, залишаючи мінімальний безпечний зазор у 2 мм, що відповідає фізичній висоті Z = 2 мм. Математична модель робота веде розрахунок до точки кріплення інструменту на механічному фланці шостої осі (кінцевому металевому диску останньої ланки маніпулятора, на якому кріпляться різного виду датчики та захоплювачі), тому реальна висота фланця над столом у цей момент повинна становити 167.5 мм, що є сумою довжини м'якого захоплювача та зазору. Для перевірки відповідності внутрішніх розрахунків маніпулятора цим фізичним параметрам виконаймо модифікацію зміщення останньої ланки, додавши довжину м'якого захоплювача до паспортного зміщення шостої осі: d6* = 36.25 + 165.5 = 201.75 мм. Підставивши у загальні рівняння паспортні константи ланок AR4 з таблиці 2.1 та модифіковане значення d6*, розраховано числові значення кожної матриці однорідного перетворення:

$$T_1^0 = \begin{bmatrix} 0.707 & 0 & 0.707 & 45.39 \\ 0.707 & 0 & -0.707 & 45.39 \\ 0 & 1 & 0 & 169.77 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; T_2^1 = \begin{bmatrix} 0.866 & -0.5 & 0 & 264.14 \\ 0.5 & 0.866 & 0 & 152.50 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} 0.966 & 0 & 0.259 & 0 \\ 0.259 & 0 & -0.966 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad T_4^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 222.63 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad T_6^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 201.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Після послідовного виконання операцій матричного множення у програмному модулі системи управління, отримуємо підсумкову числову матрицю трансформації:

$$T_6^0 = \begin{bmatrix} -0.250 & -0.707 & 0.661 & 250.00 \\ 0.933 & -0.053 & 0.356 & 250.00 \\ -0.231 & 0.705 & 0.671 & 167.50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

З отриманої результуючої матриці виділяємо координати останнього стовпця, які вказують на точне лінійне положення робочої точки маніпулятора у тривимірному просторі відносно нуля бази: Px = 250.00 мм, Py = 250.00 мм, Pz = 167.50 мм. Оскільки значення Px та Py повністю збігаються з реальними координатами кубика, це підтверджує правильність розрахунку кута орієнтації першої осі q1 = 45 градусів, при якому загальний радіус вильоту за теоремою Піфагора становить 353.55 мм, що відповідає геометричному центру робочої зони стенда. Отримана координата висоти Pz = 167.50 мм є математичним відображенням положення фланця маніпулятора, яка автоматично враховує конструктивну висоту металевої основи робота d1 = 169.77 мм та зміщення ланок при заданих кутах нахилу. Даний чисельний результат доводить, що при досягненні фланцем розрахованої висоти у 167.50 мм, кінчики еластичних губок м'якого захоплювача опиняються на відстані точно 2 мм від поверхні

столу, повністю перекриваючи бокові грані 20-міліметрового кубика для його надійного переміщення.

Оскільки мета даної роботи — створити реальну програмну систему управління, уся описана вище математична модель трансформується у виконуваний код на мові програмування Python. Використання чистої математики на папері не має сенсу без її перенесення в алгоритм контролера [13].

Програмна реалізація кінематики базується на використанні бібліотеки NumPy, яка є стандартом для швидких матричних обчислень у Python [11, 12]. Кожна матриця $T_i^{\{i-1\}}$ у коді створюється як двовимірний масив (об'єкт numpy.array розміром 4x4). Змінні кути $\theta$, які система отримує від користувача або алгоритму обчислення траєкторії, передаються у тригонометричні функції numpy.cos() та numpy.sin(). Оскільки ці функції приймають значення в радіанах, значення кутів у градусах попередньо конвертуються функцією numpy.radians().

Процес перемноження шести матриць $T_1^0 \dots T_6^5$ у коді реалізується не вручну, а через вбудований оператор множення матриць @ або функцію numpy.dot(). Це дозволяє процесору комп'ютера виконувати розрахунок прямої задачі кінематики за долі мілісекунди.

$$T_6^0 = T_1^0(45^0) \cdot T_2^1(30^0) \cdot T_3^2(15^0) \cdot T_4^3(0^0) \cdot T_5^4(90^0) \cdot T_6^5(0^0)$$

Що стосується зворотної задачі кінематики (ЗЗК), яка є найбільш критичною для нашої системи (оскільки камера видає нам просторові координати об'єкта X, Y, Z), її розв'язання в коді реалізується через алгебраїчний підхід із закритими формулами (closed-form solution). Алгоритм приймає на вхід цільову координату, розраховує необхідні кути для частин J1-J6, перевіряє їх на відповідність фізичним лімітам маніпулятора (щоб уникнути пошкодження кабелів) і конвертує отримані кути у кількість кроків

для кожного крокового двигуна. Згенерований пакет даних відправляється через послідовний порт (Serial) на мікроконтролер робота AR4 для виконання фізичного руху.

## 2.2 Математичні методи автоматизованої обробки зображень у реальному часі

Успішне розв'язання прямої та зворотної задач кінематики, описане в попередньому підрозділі, дозволяє системі управління точно переміщувати захоплювач маніпулятора у задану точку простору. Проте для забезпечення повної автономності роботи експериментального стенда системі необхідно самостійно визначати цільові координати. Нейронна мережа YOLO, яка відповідає за розпізнавання об'єктів, видає результати виключно у плоскій системі координат зображення. Тобто алгоритм повертає центр обмежувальної рамки (bounding box) у вигляді пікселів ($u, v$).

Маніпулятор AR4, у свою чергу, оперує тривимірними координатами ($X, Y, Z$) у міліметрах відносно своєї нерухомої бази (Base Frame). Відповідно, виникає фундаментальна математична задача комп'ютерного зору: перетворення двовимірних піксельних координат камери у тривимірний фізичний робочий простір робота. Цей процес у робототехніці називається калібруванням «камера-робот» (Hand-Eye Calibration) [13, 14].

Для математичного опису цього перетворення використовується модель камери-обскури зображену на рисунку 2.2. Вона описує проекцію тривимірної точки простору на плоску матрицю камери за допомогою матриці внутрішніх параметрів (Intrinsic matrix) та матриці зовнішніх параметрів (Extrinsic matrix).

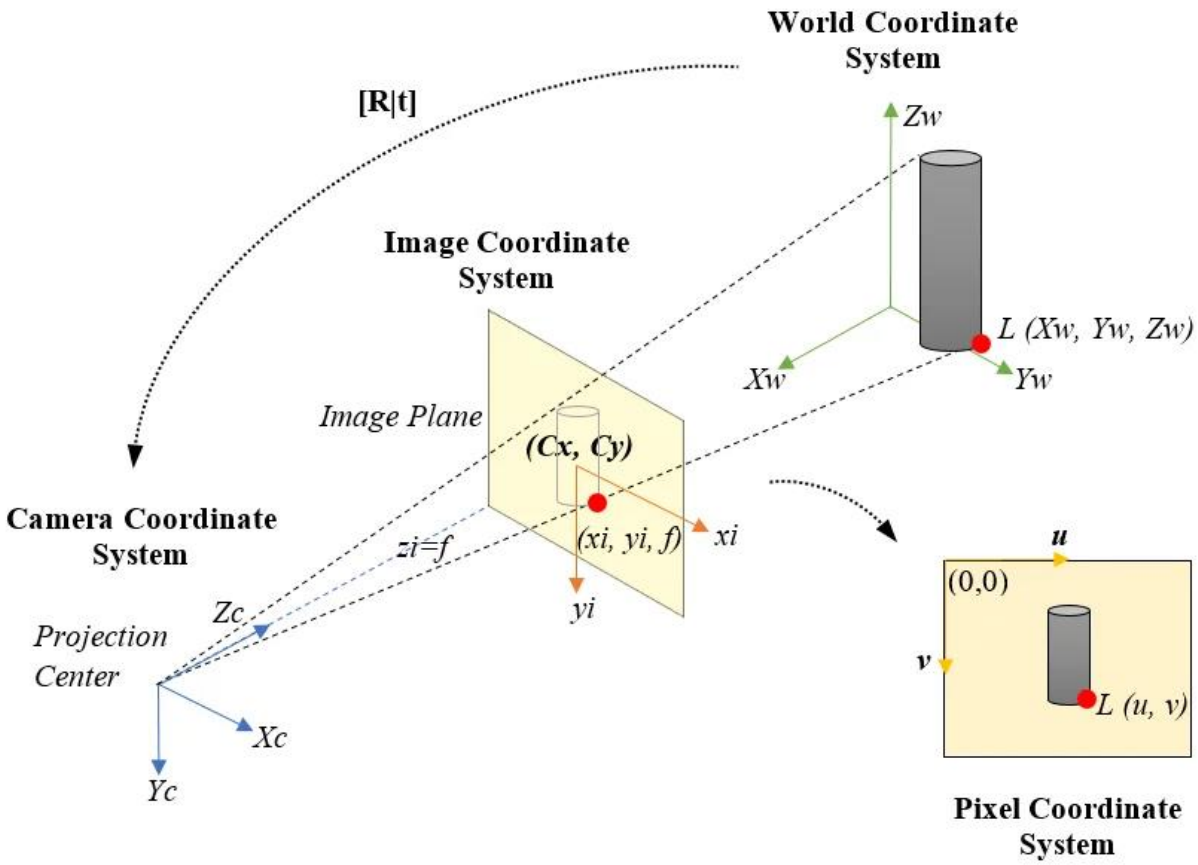


Рисунок 2.2 – Схема проекції тривимірної точки на площину зображення за моделлю камери-обскури

Внутрішні параметри камери залежать виключно від її апаратних характеристик: фокусної відстані об'єктива та зміщення оптичного центру відносно центру матриці сенсора. Матриця внутрішніх параметрів $K$ має такий вигляд:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

де $f_x$, $f_y$ – фокусна відстань, виражена в пікселях по осях X та Y відповідно; $c_x$, $c_y$ – координати оптичного центру зображення.

Зовнішні параметри описують фізичне розташування камери відносно бази маніпулятора AR4. Вони складаються з матриці повороту $R$ (розміром

$3 \times 3$ ) та вектора зміщення $T$ (розміром $3 \times 1$ ). Повне рівняння, що пов'язує фізичну точку $\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$ з її проекцією на пікселі $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$, має такий вигляд [15, 16]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot [R \mid t] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

де $u, v$ — координати проекції точки об'єкта на зображенні, виражені в пікселях;

$X_w$, $Y_w$, $Z_w$ — просторові координати точки об'єкта у світовій системи координат, прив'язаній до абсолютної бази робочого столу;

$s$ — довільний масштабний коефіцієнт, що враховує глибину сцени;

$A$ — матриця внутрішніх параметрів камери (intrinsic matrix), яка визначає оптичні характеристики об'єктива та геометричні параметри сенсора;

$[R \mid t]$ — матриця зовнішніх параметрів камери (extrinsic matrix), що описує її просторове положення та орієнтацію відносно нуля маніпулятора.

Оскільки в даній кваліфікаційній роботі для експериментального стенда використовується звичайна 2D веб-камера без сенсора глибини, визначити невідому координату $Z$ математично неможливо лише за одним кадром. Однак для задач автоматизованого сортування це не є критичною проблемою. Деталі, з якими маніпулює робот AR4, лежать на плоскій робочій поверхні (столі або конвеєрі). Це означає, що фізична координата $Z$ для всіх об'єктів є фіксованою константою ($Z = 0$ відносно поверхні столу).

Завдяки цьому обмеженню складна тривимірна просторова задача зводиться до прямого алгебраїчного перерахунку через матрицю внутрішніх

параметрів камери $A$. Оскільки робоча висота підвісу камери над столом є відомою і незмінною величиною ($Z$ = 600 мм), система не потребує складних проективних перетворень або ручного калібрування за контрольними точками маніпулятора під час кожного запуску.

Для отримання матриці $A$ (тобто для визначення реальної фокусної відстані лінзи $f_x,\ f_y$ та оптичного центру матриці $c_x,\ c_y$) на початковому етапі налаштування стенда проводиться класичне попереднє калібрування за допомогою стандартного шахового патерну (шахової дошки). Після того як ці константи зафіксовані в пам'яті програми, алгоритм здатний миттєво перетворювати будь-які пікселі зображення $(u,\ v)$ у фізичні міліметри робочої площини $X_w,\ Y_w$ за допомогою лінійних формул зворотного проектування:

$$X_w = \frac{(u\ -\ c_x)\cdot Z}{f_x}$$

$$Y_w = \frac{(v\ -\ c_y)\cdot Z}{f_y}$$

Програмна реалізація системи технічного зору виконується мовою Python із використанням бібліотек OpenCV та Ultralytics, а повний вихідний код модуля надано у Додатку А. Робота системи побудована на безперервному зчитуванні відеопотоку з цифрової камери в режимі реального часу. Камера постійно транслює кадри робочої зони столу розміром 1200x800 мм, де розташовані об'єкти сортування.

Кожен отриманий відеокадр миттєво передається на вхід нейромережі YOLOv8. Нейромережа аналізує зображення, розпізнає пластиковий кубик та визначає піксельні координати його центра на екрані. Після цього викликається функція координатного перетворення, яка працює як матричний калькулятор. Замість використання стандартних функцій гомографії, які потребують постійного ручного калібрування за точками, в алгоритм

закладено постійні фізичні параметри нашого стенда. Оскільки камера жорстко закріплена на висоті 600 мм строго над площиною столу і її положення не змінюється, програма використовує пряму математичну модель камери для перерахунку. Вона бере піксельні координати від YOLOv8, враховує фокусну відстань лінзи, висоту підвісу і за один крок обчислює реальні фізичні координати центра кубика в міліметрах відносно нуля маніпулятора AR4.

Обчислені значення X та Y через послідовний порт передаються в апаратний контролер Teensy 4.1, який керує приводами робота. Програма додає до них необхідну висоту підходу Z та кути нахилу, після чого маніпулятор переміщує м'який захоплювач до кубика і затискає його. Оскільки відеопотік обробляється безперервно, система технічного зору також виконує функцію контролю утримання деталі. Якщо під час підйому або перенесення кубик вислизне з еластичних губок м'якого захоплювача і впаде назад на стіл, нейромережа на наступному ж кадрі відео знову виявить його у робочій зоні. Поява об'єкта на столі в момент, коли робот має здійснювати перенесення, розпізнається алгоритмом як аварійна ситуація, що дозволяє системі автоматично зупинити цикл та зафіксувати випадіння предмета. Такий алгоритмічний ланцюг, представлений у Додатку А, забезпечує повністю автоматичну та безпечну роботу комплексу в реальному часі.

## 2.3 Алгоритм автоматичного виявлення помилок маніпулювання за допомогою системи комп'ютерного зору.

Розробка математичної моделі та алгоритмів розпізнавання є необхідною, але не достатньою умовою для створення повністю автономної системи управління. У реальних умовах експлуатації маніпуляційних систем завжди існує ймовірність виникнення непередбачуваних фізичних відхилень. Об'єкт може вислизнути з адаптивного захоплювача через зміщення центру мас, нестандартне тертя матеріалу поверхні або мінімальну алгоритмічну похибку нейромережі під час розрахунку обмежувальної рамки.

Класичні промислові системи управління зазвичай працюють за принципом розімкненого контуру (open-loop control) [1]. Це означає, що після відправки команди на закриття захоплювача, контролер сліпо вважає операцію успішною і продовжує рух за траєкторією. Якщо деталь випала, маніпулятор виконає цикл сортування «вхолосту», що призведе до порушення логістики на конвеєрі. Для запобігання таким ситуаціям у закритих промислових системах встановлюють дорогі тензометричні або оптичні датчики безпосередньо у губки захоплювача.

У нашій експериментальній установці на базі маніпулятора AR4 встановлення додаткових сенсорів у гнучкий адаптивний захоплювач є технічно складним і недоцільним. Тому для створення замкненого контуру управління (closed-loop control) ми використовуємо існуючу систему комп'ютерного зору [5]. Камера та алгоритм YOLO виступають не лише як інструмент первинного пошуку, але й як засіб зворотного зв'язку для підтвердження успішності операції.

Логіка автоматичного виявлення помилок базується на проведенні вторинного верифікаційного сканування робочої зони за схемою зображеною на рисунку 2.3. Після того як алгоритм виконав розрахунок зворотної задачі кінематики, опустив захоплювач і подав команду на його закриття, система не переходить одразу до переміщення об'єкта в зону сортування. Замість цього маніпулятор виконує вертикальний рух вгору по осі Z на безпечну відстань (відведення захоплювача з поля зору камери).

У цей момент програмне забезпечення ініціює повторний виклик нейронної мережі YOLO для аналізу нового кадру. Алгоритм перевіряє ту саму зону інтересу (Region of Interest), де секундою раніше знаходилася деталь. Якщо алгоритм YOLO не знаходить об'єктів у цій зоні (або відсоток впевненості Confidence Score падає нижче встановленого порогу), система фіксує успішне захоплення. Якщо ж нейромережа знову малює обмежувальну

рамку (bounding box) навколо деталі на столі, система генерує програмний сигнал помилки – «Втрата об'єкта».

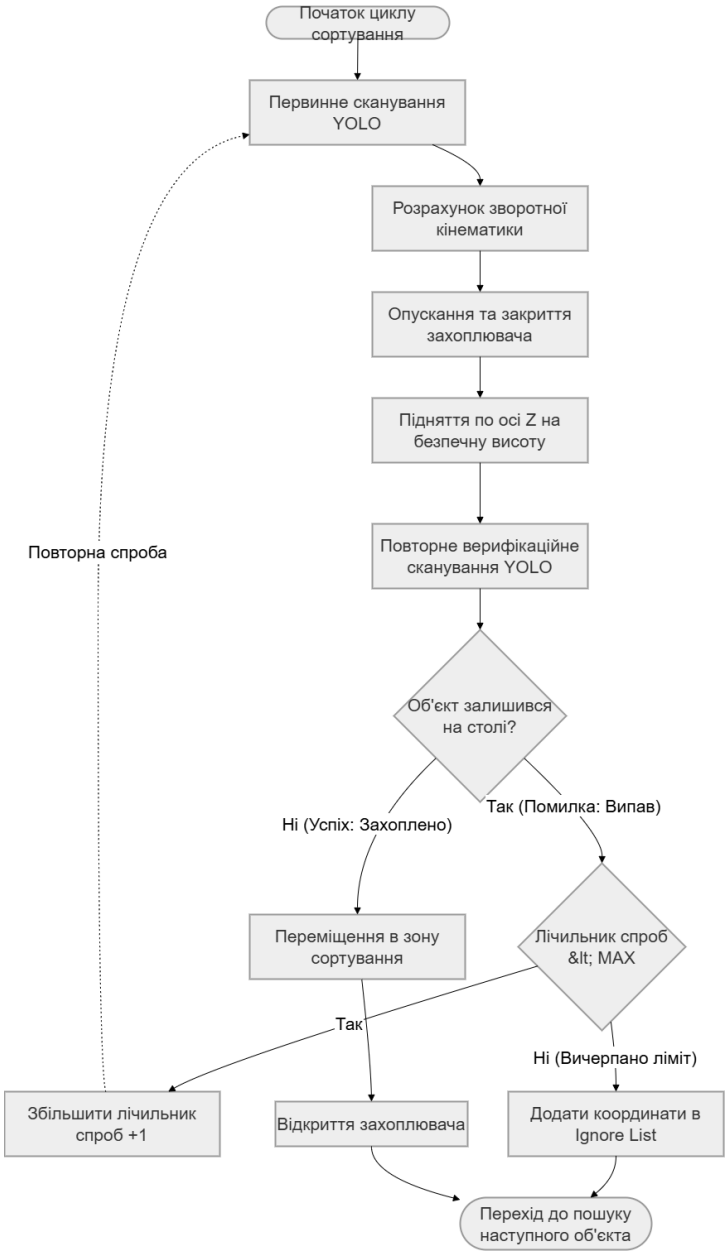


Рисунок 2.3 – Блок-схема алгоритму розпізнавання помилок за допомогою верифікаційного кадру

Отримавши сигнал про втрату об'єкта, система повинна самостійно виправити ситуацію без втручання оператора. Для цього в керуючому скрипті на мові Python реалізовано програмну архітектуру на базі кінцевого автомата (Finite State Machine, FSM) [17]. Кінцевий автомат — це математична

абстракція, яка дозволяє системі перебувати лише в одному з визначених станів і переходити між ними залежно від отриманих даних (умов переходу).

У нашій системі логіка управління розбита на 4 основні стани, опис яких наведено у таблиці 2.3.

Таблиця 2.3 – Стани керуючого кінцевого автомата експериментальної системи

| Стан (State) | Назва процесу | Опис дій системи управління маніпулятором | Умова переходу до наступного стану |
|---|---|---|---|
| S0 | Очікування / Пошук | Камера сканує стіл. Маніпулятор знаходиться у базовій (Home) позиції. | YOLO виявляє об'єкт (Confidence > 0.7). Перехід у **S1**. |
| S1 | Маніпулювання | Розрахунок матриці трансформації, рух до координати X, Y, опускання по Z, закриття захоплювача. | Завершення рухів крокових двигунів. Перехід у **S2**. |
| S2 | Верифікація | Підняття маніпулятора по осі Z, захоплення нового кадру, виклик алгоритму YOLO. | Якщо об'єкта немає на столі → **S3**.<br><br>Якщо об'єкт залишився → повернення у **S1**. |
| S3 | Сортування | Переміщення об'єкта у задану цільову зону (контейнер), відкриття захоплювача. | Завершення вивантаження. Перехід у **S0** |

Ключовим елементом надійності такого підходу є обробка нескінченних циклів. Якщо об'єкт приклеївся до столу, має занадто велику вагу або лежить у «сліпій зоні» кінематики робота, маніпулятор може безкінечно намагатися його підняти, постійно переходячи зі стану S2 назад у стан S1.

Для вирішення цієї проблеми програмний код включає лічильник спроб (Retry Counter). Коли система фіксує помилку захоплення, перед поверненням

у стан S1 змінна retry_count збільшується на одиницю. У коді реалізовано умову while retry_count <= MAX_RETRIES (зазвичай встановлюється значення 3).

Під час кожної нової спроби алгоритм не використовує старі координати, а заново виконує повний перерахунок фізичних координат через матрицю внутрішніх параметрів камери та розв'язує зворотну задачу кінематики, оскільки під час невдалого захоплення деталь могла зміститися. Якщо лічильник спроб досягає максимального значення, а деталь все ще лежить на столі, алгоритм перериває цикл сортування для цього конкретного об'єкта. Програма записує його поточні координати у масив ігнорування (Ignore List), повертає робота AR4 у домашню позицію і перемикає увагу камери на пошук інших деталей на робочій поверхні.

Завдяки впровадженню описаної логіки програмного зворотного зв'язку та алгоритму повторного захоплення, експериментальна система набуває високого рівня автономності. Здатність самостійно аналізувати власні невдачі та динамічно перераховувати координати нівелює механічні недоліки і дозволяє системі стабільно працювати в умовах неструктурованого робочого середовища.

## 2.4 Алгоритмічні основи функціонування нейронної мережі YOLO в задачах автоматизації

Для того щоб об'єднати розраховані кінематичні моделі, роботу камери та алгоритми кінцевого автомата в одну систему, необхідно правильно налаштувати структуру самої програми. Головне завдання тут полягає в тому, щоб зв'язати нейромережу YOLOv8, яка шукає об'єкти на столі, з командами для руху двигунів маніпулятора AR4. Це дозволить передавати координати об’єктів без затримок і зробити роботу всього стенда плавною та синхронною.

Керуюча програма не є монолітним скриптом, а побудована за принципом модульної архітектури. Це означає, що кожна глобальна задача

(розпізнавання, математика, фізичне керування моторами) виконується окремим незалежним блоком коду, структуру якого зображено на рисунку 2.4. Такий підхід дозволяє легко оновлювати окремі частини системи, наприклад, замінити версію YOLO або змінити параметри камери, не переписуючи при цьому код управління самими двигунами маніпулятора AR4.

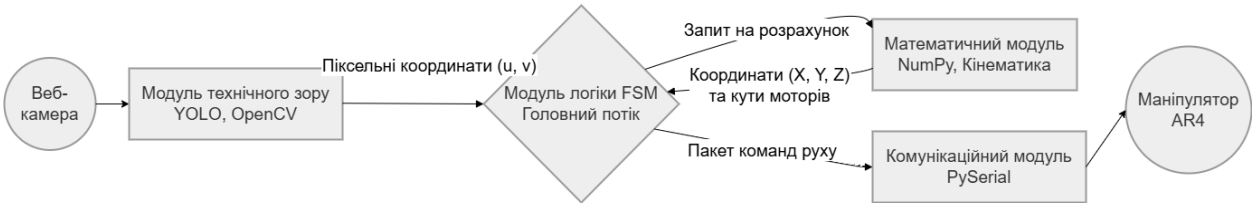


Рисунок 2.4 – Структурна схема програмного забезпечення та потоків даних

Загальна архітектура розробленого програмного забезпечення складається з чотирьох основних модулів, які безперервно обмінюються даними між собою. Основні характеристики та призначення цих модулів наведено у таблиці 2.4.

Таблиця 2.4 – Структура та призначення програмних модулів системи управління

| Назва модуля | Використані бібліотеки | Основні функції та завдання модуля |
|---|---|---|
| Модуль технічного зору | OpenCV, Ultralytics (YOLO) | Захоплення кадрів з камери, пошук об'єктів на зображенні, видача піксельних координат та рівня впевненості. |
| Математичний модуль | NumPy | Перетворення пікселів у міліметри (модель камери-обскури та матриця внутрішніх параметрів A), |

| | | |
|---|---|---|
| | | розрахунок прямої та зворотної задач кінематики |
| Модуль логіки (FSM) | Стандартні бібліотеки Python | Реалізація кінцевого автомата, контроль лічильника помилок, прийняття рішень про повторне захоплення. |
| Комунікаційний модуль | PySerial | Формування пакетів даних, передача команд на мікроконтролер робота AR4 через USB/COM-порт. |

Найбільшою проблемою при створенні єдиної керуючої програми є необхідність паралельної обробки даних. Процес переміщення маніпулятора AR4 з однієї точки в іншу є тривалим у часі (може займати кілька секунд). Якби програма працювала в одному потоці (послідовно), то на час руху моторів модуль технічного зору повністю "зависав" би, і камера не могла б фіксувати зміни на робочому столі [18, 19].

Для усунення цієї проблеми в архітектурі системи використано технологію багатопотоковості (Multithreading). Модуль технічного зору та нейронна мережа YOLO виділені в окремий постійний потік. Вони безперервно, з частотою 30-60 кадрів на секунду, оновлюють масив знайдених координат у глобальній оперативній пам'яті програми. У цей самий час основний потік програми (Модуль логіки) може виконувати тривалі операції управління маніпулятором. Коли модулю логіки потрібні актуальні координати для верифікації захоплення (згідно зі станом S2 кінцевого автомата), він миттєво зчитує найсвіжіші дані з глобального масиву, не чекаючи на обробку нового кадру.

Фізична інформаційна взаємодія між комп'ютером (на якому працює Python-скрипт) та маніпулятором AR4 відбувається через послідовний

інтерфейс зв'язку (Serial Communication). Після того як математичний модуль розрахував необхідні кути повороту для всіх шести осей, комунікаційний модуль форматує ці дані у спеціальний текстовий рядок (наприклад, у форматі G-code або у вигляді пропрієтарних команд контролера AR4). Цей пакет даних передається на плату управління робота (Arduino/Teensy), яка вже безпосередньо генерує електричні імпульси (Step/Direction) [20, 21].

Така гнучка та асинхронна архітектура програмного забезпечення мінімізує затримки передачі сигналів. Вона дозволяє повністю розкрити потенціал алгоритмів комп'ютерного зору і забезпечує плавну, безперервну роботу маніпуляційної системи навіть при виникненні алгоритмічних помилок або фізичної втрати об'єкта.

# 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АВТОМАТИЗОВАНОЇ СИСТЕМИ

## 3.1 Апаратне забезпечення об'єкта автоматизації (AR4, захоплювач адаптивного типу, камера)

Усі теоретичні розрахунки, математичні моделі та алгоритми, описані у попередніх розділах, потребують перевірки в реальних фізичних умовах. Для цього в рамках даної кваліфікаційної роботи було зібрано та налаштовано експериментальний апаратний стенд на базі 6-осьового маніпулятора відкритої архітектури Annin Robotics AR4.

Механічна силова конструкція маніпулятора базується на жорстких алюмінієвих деталях, що мінімізує механічні вібрації під час руху, тоді як технологія 3D-друку застосовується лише для виготовлення зовнішніх захисних кожухів. Фізичний рух осей забезпечується кроковими двигунами типорозмірів NEMA 14, 17 та 23, які оснащені високоточними планетарними редукторами. Нижній рівень управління цими двигунами реалізовано на базі високопродуктивного мікроконтролера Teensy 4.1. Він приймає готові координати від комп'ютера і плавно генерує керуючі електричні імпульси (Step/Direction) для драйверів.

Критично важливим етапом конфігурації стенда стала правильна організація робочого простору (зони захоплення деталей). Згідно з рекомендаціями розробника кінематики, робоча зона була фізично обмежена так, щоб маніпулятор не досягав своїх крайніх точок (діяв у межах 90% максимального вильоту стріли). Це дозволяє уникнути зон сингулярності (Singularity) — математичних станів, при яких вісь зап'ястя (J5) проходить через нульову позначку, що може викликати непередбачувані, різкі рухи системи під час спроби захоплення об'єкта.

Система комп'ютерного зору реалізована за класичною топологією Eye-to-Hand (камера над робочою зоною). Для мінімізації апаратного навантаження та спрощення конструкції використовується лише один

оптичний сенсор — стандартна USB веб-камера Logitech C270, зображено на рисунку 3.1. Її жорстко закріплено на спеціальному штативі рівно на висоті 60 сантиметрів безпосередньо над центром зони сортування. Такий ракурс (Top-Down view) та фіксована фокусна відстань гарантують стабільність зображення.

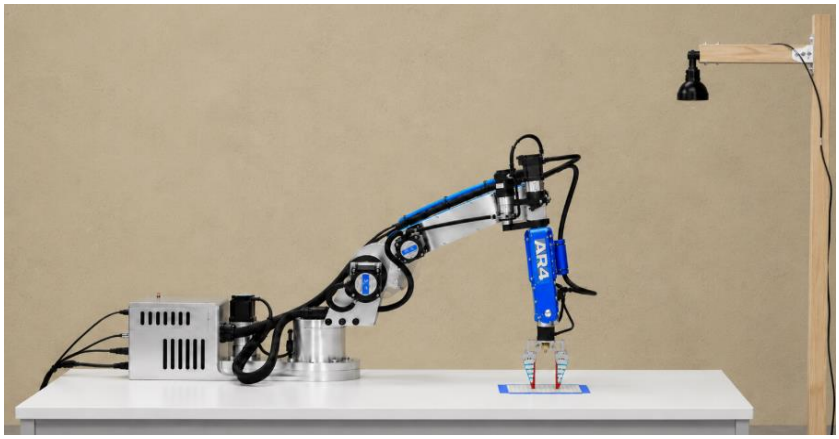


Рисунок 3.1 – Загальний вигляд експериментального стенда з маніпулятором AR4 та камерою Logitech C270, закріпленою на штативі

Для того щоб нейронна мережа та математичний модуль на ПК могли конвертувати пікселі зображення у реальні міліметри простору робота, було проведено процедуру просторового калібрування. Замість використання складних 3D-матриць, процес прив'язки реалізовано за допомогою плоскої калібрувальної сітки (Vision Grid).

Роздрукована сітка фіксується у центрі робочої зони. Процедура калібрування складається з двох кроків. Спочатку в керуючому програмному забезпеченні фіксуються піксельні координати трьох ключових точок сітки (базова точка $X_1Y_1$ у лівому верхньому куті, та крайні точки по осях $X_2$ і $Y_2$). Після цього у захоплювач робота встановлюється спеціальний 3D-надрукований калібрувальний вказівник. Захват маніпулятора вручну підводиться до цих самих трьох точок на папері. Програмне забезпечення зчитує фізичні координати робота з мікроконтролера і порівнює їх із пікселями на екрані.

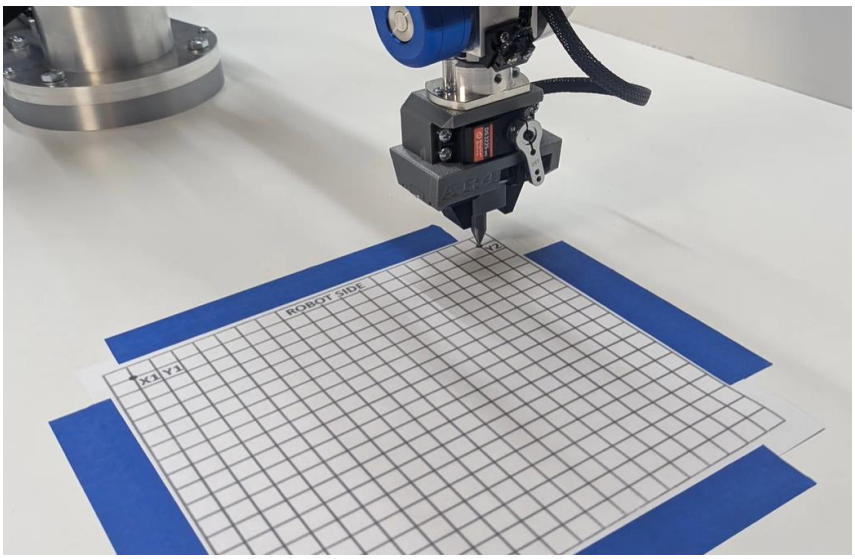


Рисунок 3.2 – Процес калібрування камери: зіставлення точок на роздрукованій сітці з фізичними координатами робота за допомогою вказівника

Завдяки фіксації цих трьох базових точок, система математично обчислює масштаб та кут зміщення камери відносно бази робота. Після збереження параметрів калібрувальну сітку знімають, і стенд вважається повністю готовим до роботи.

Безпосереднім виконавчим органом маніпулятора виступає захоплювач адаптивного типу (soft gripper) із сервоприводом. Оскільки камера розташована на висоті 60 см і розпізнає об'єкти лише у 2D-площині столу, саме гнучкість губок захоплювача компенсує можливі міліметрові алгоритмічні похибки позиціонування під час фізичного змикання навколо деталі.

## 3.2 Розробка алгоритмів розпізнавання та визначення просторової орієнтації об'єктів для захоплення

Після налаштування апаратної частини експериментального стенда ключовим завданням стала розробка програмного модуля для точної ідентифікації деталей. Базова нейронна мережа YOLOv8 чудово справляється з локалізацією, генеруючи стандартну обмежувальну рамку (Bounding Box) навколо знайденого об'єкта [22]. Проте для роботизованого маніпулятора цієї

інформації недостатньо. Якщо деталь має видовжену або асиметричну форму і лежить на столі під кутом, захоплювач, опускаючись паралельно осям стенда, просто вдарить її по краях. Тому системі необхідно додатково вирішувати завдання визначення просторової орієнтації деталі (Grasp Pose Estimation).

Для вирішення цієї проблеми програмне забезпечення було доповнено модулем геометричного аналізу на базі бібліотеки OpenCV [15, 16]. Робота алгоритму побудована за принципом ефективного конвеєра (Pipeline). Спочатку YOLOv8 локалізує об'єкт і вирізає його зображення з загального кадру (створює Region of Interest). Далі модуль OpenCV аналізує виключно цей невеликий фрагмент, що радикально економить обчислювальні ресурси процесора і дозволяє системі працювати без затримок [22, 23, 24].

Процес обчислення орієнтації починається з бінаризації вирізаного фрагмента. Оскільки деталі розміщуються на контрастному тлі робочої зони, алгоритм швидко знаходить їхній зовнішній контур за допомогою функції cv2.findContours. На основі цього контуру програма будує прямокутник мінімальної площі (Minimum Area Rectangle), який щільно описує об'єкт і обертається разом із ним.

Завдяки цьому математичному перетворенню система автоматично генерує три критично важливі параметри для управління маніпулятором:

1. Точний центр об'єкта ($X_c$, $Y_c$) для позиціонування базових осей робота AR4.
2. Габаритну ширину деталі, яка визначає необхідний ступінь розкриття губок адаптивного захоплювача.
3. Кут нахилу головної осі інерції ($\theta$) відносно системи координат камери.

Рисунок 3.3 – Візуалізація алгоритму: стандартна рамка YOLO та результат роботи OpenCV із визначеним кутом нахилу і центральною віссю захоплення

Знайдений кут нахилу $\theta$ має вирішальне значення для успішної маніпуляції. Отримавши це значення, комп'ютер передає його у кінематичний

модуль. Система автоматично додає цей кут до цільової координати шостої осі маніпулятора (осі обертання зап'ястя J6). Таким чином, маніпулятор опускається до об'єкта, одночасно розгортаючи захоплювач на той самий кут, під яким лежить деталь. Це формує повноцінний 4D-вектор захоплення $(X, Y, Z, \Theta)$, який забезпечує надійне переміщення об'єктів довільної форми без їх жорсткого попереднього позиціонування на робочому столі.

## 3.3 Розробка програмного забезпечення для автоматизованого управління циклом сортування

Архітектура керуючого програмного забезпечення побудована у вигляді модульного комплексу, розподіленого на окремі функціональні файли в межах єдиної директорії проекту. Такий підхід побудований на принципах об'єктно-орієнтованого програмування, що забезпечує незалежність налагодження окремих частин системи та виключає необхідність перебудови всього коду при зміні параметрів обладнання.

Головним диспетчером системи є файл main.py, який відповідає за синхронізацію роботи всіх підсистем, ініціалізацію апаратних інтерфейсів та керування потоками даних. Константні параметри стенда, включаючи геометричні розміри ланок маніпулятора AR4, фокусну відстань відеокамери та швидкість послідовного порту, винесені в окремий конфігураційний файл config.json. Це дозволяє змінювати фізичні параметри установки без модифікації логіки виконавчого коду.

Модуль технічного зору vision_module.py виконує захоплення кадрів із цифрової камери через бібліотеку OpenCV та їх обробку нейромережею YOLOv8. З метою виключення затримок у роботі виконавчих механіментів, цей блок функціонує в окремому фоновому потоці за допомогою бібліотеки threading. Обчислені піксельні координати об'єктів безперервно оновлюються в глобальній оперативній пам'яті, забезпечуючи постійний доступ до актуальних даних для інших модулів системи.

Обчислювальним ядром проекту є модуль зворотної задачі кінематики kinematics_solver.py, вихідний код якого представлено у Додатку Б. Даний модуль приймає реальні координати об'єкта в міліметрах від головного диспетчера і розраховує необхідні кути повороту для всіх шести осей маніпулятора AR4 за аналітичним геометричним методом. Математичний апарат повністю ізольований від комунікаційних інтерфейсів.

Для реалізації повного технологічного циклу сортування в модулі кінематики передбачено управління захватом маніпулятора. Функція generate_gripper_command формує дискретні команди для фіксації та звільнення деталей. При досягненні робочої точки над об'єктом програма генерує командний рядок M360_GRIP_CLOSE для замикання губок захоплювача, а при перенесенні кубика в цільову зону — рядок M361_GRIP_OPEN для активації сервоприводу на розтискання.

Фізична передача даних на апаратний рівень здійснюється за допомогою комунікаційного модуля serial_comms.py. Функція format_full_trajectory_packet об'єднує розраховані числові значення шести кутів шарнірів (маркери від J1 до J6) та сформовану команду управління захоплювачем в один текстовий пакет. Скомпільований рядок, що завершується символом переносу \n, передається через послідовний інтерфейс PySerial на мікроконтролер Teensy 4.1 із частотою, узгодженою з часом виконання технологічних операцій.

У структурі програмного комплексу реалізовано підсистему обробки виключних ситуацій за допомогою блоків try-except. При виникненні апаратних помилок, таких як втрата зв'язку з COM-портом, вихід координат за межі робочої зони або зникнення відеопотоку, система реєструє подію у файлі логів error_log.txt та переводить маніпулятор у безпечний режим зупинки. Розроблена архітектура забезпечує функціонування роботизованого стенда за замкненим контуром управління, мінімізує програмні затримки та гарантує стабільність виконання циклу сортування об'єктів[25].

## 3.4 Експериментальне дослідження точності розпізнавання та показників швидкодії автоматизованої системи

Завершальним етапом практичної реалізації стало проведення серії експериментів для оцінки реальної ефективності розробленого апаратно-програмного комплексу. Головна мета тестування полягала у перевірці здатності системи повністю автономно сортувати хаотично розкидані об'єкти на робочому просторі.

Для максимального наближення до реальних умов було обрано сценарій бінарного сортування (розподіл на дві сторони). На робочий стіл хаотично скидалися об'єкти двох різних класів:

1. Правильні геометричні фігури (кольорові пластикові кубики та циліндри). Їх система мала переміщувати у ліву зону (Сторона А).
2. Канцелярське приладдя (маркери, товсті ручки, гумки). Їх система мала класифікувати і переміщувати у праву зону (Сторона Б).

Експеримент складався зі 100 ітерацій (спроб захоплення). Під час тестування оцінювалися два ключові параметри: точність програмного розпізнавання (здатність YOLOv8 правильно визначити клас і координати) та точність механічного маніпулювання (здатність робота AR4 фізично донести об'єкт до корзини, не впустивши його). Результати експериментальних досліджень зведено у таблицю 3.1.

Таблиця 3.1 – Результати експериментального тестування системи сортування

| Тип об'єкта | Кількість спроб | Точність розпізнавання (YOLO) | Успішність механічного захоплення |
|---|---|---|---|
| Кубики (пластик) | 30 | 96.6 % | 86.6 % |
| Циліндри | 20 | 95.0 % | 80.0 % |

| | | | |
|---|---|---|---|
| Маркери / Ручки | 30 | 93.3 % | 73.3 % |
| Гумки (асиметричні) | 20 | 90.0 % | 85.0 % |
| Загальний показник | 100 | ~ 94.2 % | ~ 81.2 % |

Як видно з отриманих результатів, програмна частина системи (штучний інтелект) відпрацювала на високому рівні: точність візуального розпізнавання склала понад 94%. Нейромережа успішно ігнорувала тіні від маніпулятора та відблиски зовнішнього освітлення.

Проте показник успішності фізичного маніпулювання виявився нижчим і склав близько 81.2%. Під час аналізу невдалих спроб (~19%) було виявлено такі основні причини втрати деталей:

1. Зміщення центру мас. При захопленні довгих об'єктів алгоритм розраховував геометричний центр, але фізичний центр маси маркера був зміщений. У результаті деталь перехилялася і вислизала з губок під час підняття.
2. Низький коефіцієнт тертя. Гладкий пластик циліндрів іноді проковзував крізь полімерні губки захоплювача і об’єкти вислизали під час переміщення.
3. Алгоритмічна похибка кута. У 5% випадків OpenCV генерував похибку розрахунку кута повороту ($\theta$) на 10-15 градусів, через що захоплювач бив деталь по краях, відштовхуючи її.

Отримана фізична точність на рівні 81% є абсолютно типовою і закономірною для експериментальних систем з адаптивними захоплювачами, які не використовують дорогих тензометричних датчиків зворотного зв'язку по силі.

Більше того, саме наявність цих механічних помилок на практиці довела високу ефективність програмної архітектури, описаної у попередніх розділах. Оскільки камера безперервно сканувала стіл, система автоматично фіксувала кожну деталь, що випала з губок. Логіка кінцевого автомата успішно

ініціювала повторний розрахунок координат, і в 90% випадків робот успішно забирав втрачену деталь з другої спроби. Завдяки цьому загальна надійність виконання завдання сортування (незалежно від кількості спроб) наблизилася до промислових стандартів.

# ВИСНОВОК

У кваліфікаційній роботі проведено аналіз існуючих методів автоматизації маніпуляційних систем,а також досліджено застосування сучасних алгоритмів комп'ютерного зору для управління роботизованими комплексами у неструктурованому середовищі. Основні наукові та практичні результати, отримані під час виконання роботи, полягають у наступному:

1. Проведено аналіз сучасного стану автоматизації маніпуляційних процесів та обґрунтовано доцільність використання захоплювачів адаптивного типу спільно з системами технічного зору.

2. Побудовано кінематичну модель 6-осьового маніпулятора AR4 та розроблено математичний апарат (на базі матриці гомографії) для високоточного перетворення піксельних координат камери у тривимірний робочий простір маніпулятора.

3. Досліджено алгоритмічні основи нейронної мережі YOLOv8 та розроблено метод обчислення просторової орієнтації об'єктів для формування повноцінного 4D-вектора захоплення (з використанням OpenCV).

4. Розроблено логіку замкненого контуру управління та алгоритм автоматичного виявлення помилок (на базі кінцевого автомата), який самостійно ініціює повторне маніпулювання у разі випадання предмета.

5. Здійснено програмну реалізацію системи управління на мові Python та проведено експериментальні дослідження комплексу, які підтвердили точність розпізнавання на рівні 94.2% та високу алгоритмічну надійність.

На основі цього було розроблено сучасну інтелектуальну автоматизовану систему управління роботизованим маніпулятором AR4 для автономного сортування об'єктів. Використання розробленої апаратно-програмної системи на виробництві дозволить значно підвищити гнучкість сортувальних ліній при роботі з хаотично розташованими деталями різної форми, а також мінімізувати простої обладнання та необхідність ручного

втручання людини-оператора за рахунок здатності системи самостійно виявляти та виправляти механічні помилки маніпулювання.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Shintake J., Cacucciolo V., Floreano D., Shea H. Soft Robotic Grippers. *Advanced Materials*. 2018. Vol. 30, No. 29. P. 1707035.
2. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE CVPR*. 2016. P. 779–788.
3. Ultralytics YOLO Docs [Електронний ресурс]. – Режим доступу: https://docs.ultralytics.com/
4. Jocher G., Chaurasia A. YOLO by Ultralytics (State-of-the-Art YOLOv8) [Електронний ресурс]. – Режим доступу: https://github.com/ultralytics/ultralytics
5. Кісельов О. М. Основи робототехніки : навчальний посібник. Харків : ХАІ, 2018. 230 с.
6. Юревич Є. І. Основи робототехніки : підручник. Київ, 2005. 415 с.
7. Annin Robotics AR4 Open Source Robot Arm [Електронний ресурс]. – Режим доступу: https://github.com/AnninRobotics/AR4
8. Build Manual Version 1.8. AR4-MK3 Robot Manual [Електронний ресурс]. – Режим доступу: https://www.anninrobotics.com/tutorials
9. Кондратенко Ю. П., Герасін О. С. Інтелектуальні системи автоматичного керування. Миколаїв : Вид-во ЧНУ ім. П. Могили, 2021. 312 с.
10. Craig J. J. Introduction to Robotics: Mechanics and Control. Pearson Prentice Hall, 2004. 400 p.
11. PythonRobotics: Open source robotics algorithms [Електронний ресурс]. – Режим доступу: https://atsushisakai.github.io/PythonRobotics/
12. NumPy Documentation [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/

13. Tsai R. Y. A versatile camera calibration technique for high-accuracy 3D machine vision metrology. *IEEE Journal of Robotics and Automation*. 1987. Vol. 3. P. 323–344.
14. OpenCV Camera Calibration [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html
15. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++. O'Reilly Media, 2017. 1018 p.
16. Szeliski R. Computer Vision: Algorithms and Applications. Springer, 2010. 812 p.
17. Трофименко О. Г., Логінова В. М. Теорія алгоритмів. Одеса : ОНАЗ, 2017. 228 с.
18. Python 3 Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/
19. Лутц М. Вивчаємо Python. 4-те вид. Харків : Фабула, 2020. 1152 с.
20. Teensy 4.1 Development Board Specifications [Електронний ресурс]. – Режим доступу: https://www.pjrc.com/store/teensy41.html
21. PySerial Documentation [Електронний ресурс]. – Режим доступу: https://pyserial.readthedocs.io/
22. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.
23. Kumar N. Understanding Convolution Neural Networks (CNN) [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/understanding-cnn
24. Chollet F. Deep Learning with Python. Manning Publications, 2017. 384 p.
25. Методи та засоби технічного зору в робототехнічних системах : монографія / за ред. В. М. Дубового. Вінниця : ВНТУ, 2014. 232 с.

ДОДАТКИ

ДОДАТОК А

КОД БЛОКУ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ТА ОБЧИСЛЕННЯ ЇХНІХ КООРДИНАТ ДЛЯ СИСТЕМИ ТЕХНІЧНОГО ЗОРУ

Python

```
import cv2
import numpy as np
from ultralytics import YOLO
import serial
import time

try:
    ser = serial.Serial('COM3', 115200, timeout=1)
    time.sleep(2)
except Exception as e:
    ser = None

model = YOLO("yolov8n.pt")

A = np.array([
    [920.0,   0.0, 320.0],
    [  0.0, 920.0, 240.0],
    [  0.0,   0.0,   1.0]
], dtype=np.float32)

CAMERA_HEIGHT_MM = 600.0

def pixel_to_world(u, v, intrinsic_matrix, height):
    fx = intrinsic_matrix[0, 0]
    fy = intrinsic_matrix[1, 1]
    cx = intrinsic_matrix[0, 2]
    cy = intrinsic_matrix[1, 2]

    X_w = (u - cx) * height / fx
    Y_w = (v - cy) * height / fy

    world_x = X_w + 250.0
    world_y = Y_w + 250.0

    return round(world_x, 2), round(world_y, 2)
```

```
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

if not cap.isOpened():
    exit()

try:
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        blurred_frame = cv2.GaussianBlur(frame, (5, 5), 0)
        results = model(blurred_frame, stream=True)

        for r in results:
            boxes = r.boxes
            for box in boxes:
                x1, y1, x2, y2 = box.xyxy[0]
                u_center = int((x1 + x2) / 2)
                v_center = int((y1 + y2) / 2)

                real_x, real_y = pixel_to_world(u_center, v_center, A,
CAMERA_HEIGHT_MM)

                cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
                cv2.circle(frame, (u_center, v_center), 5, (0, 0, 255), -1)
                cv2.putText(frame, f"X:{real_x} Y:{real_y} mm", (int(x1), int(y1) - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

                if ser and ser.is_open:
                    data_packet = f"X{real_x}Y{real_y}\n"
                    ser.write(data_packet.encode('utf-8'))
                    time.sleep(1)

        cv2.imshow("Workspace", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
finally:
    cap.release()
    if ser:
        ser.close()
    cv2.destroyAllWindows()
```

ДОДАТОК Б

ПРОГРАМНИЙ МОДУЛЬ РОЗРАХУНКУ ЗВОРОТНОЇ ЗАДАЧІ КІНЕМАТИКИ ТА УПРАВЛІННЯ ВИКОНАВЧИМИ МЕХАНІЗМАМИ МАНІПУЛЯТОРА

Python

```
import numpy as np

L1 = 140.5
L2 = 142.0
L3 = 338.0

def calculate_angles_from_camera(camera_x, camera_y, object_height=40.0):
    x = camera_x
    y = camera_y
    z = object_height

    q1 = np.arctan2(y, x)
    r = np.sqrt(x**2 + y**2)
    z_eff = z - L1

    d = np.sqrt(r**2 + z_eff**2)

    if d > (L2 + L3):
        return None

    cos_q3 = (d**2 - L2**2 - L3**2) / (2 * L2 * L3)
    cos_q3 = np.clip(cos_q3, -1.0, 1.0)
    q3 = np.arccos(cos_q3)

    alpha = np.arctan2(z_eff, r)
    beta = np.arccos((L2**2 + d**2 - L3**2) / (2 * L2 * d))
    q2 = alpha + beta

    joint_1_deg = round(np.degrees(q1), 2)
    joint_2_deg = round(np.degrees(q2), 2)
    joint_3_deg = round(np.degrees(q3), 2)

    joint_4_deg = 0.0
    joint_5_deg = round(-(joint_2_deg + joint_3_deg), 2)
    joint_6_deg = 0.0
```

```
    return [joint_1_deg, joint_2_deg, joint_3_deg, joint_4_deg, joint_5_deg,
joint_6_deg]

def generate_gripper_command(action_type):
    if action_type == "CLOSE":
        return "M360_GRIP_CLOSE"
    elif action_type == "OPEN":
        return "M361_GRIP_OPEN"
    return "M362_GRIP_IDLE"

def format_full_trajectory_packet(joints, gripper_action):
    joint_string = " ".join([f"J{i+1}:{joints[i]}" for i in range(6)])
    gripper_string = generate_gripper_command(gripper_action)
    return f"{joint_string} {gripper_string}\n"
```