МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему Комп'ютерна система керування роботом для збору тенісних м'ячів

Виконав: студент 2 курсу, групи 8.1210-2іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Д. В. Троян
(ініціали та прізвище)
Керівник проф., д.ф.-м.н., В. Г. Вербицький
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Альтер Віжен Груп»
В.С. Тряпічко
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ

Кафедра _____ програмного забезпечення автоматизованих систем
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри___Вербиц_____В.Г. Вербицький
"_01_" __вересня_____2021_ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

__Трояну Дмитру Віталійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи __Комп'ютерна система керування роботом для збору тенісних м'ячів_____
керівник роботи ___Вербицький Володимир Григорович, доктор фізико-математичних наук, професор_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від ___"30" червня 2021 року № 974-с_____

2. Строк подання студентом кваліфікаційної роботи_____30.11.2021_____
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення системи керування роботом для збору тенісних м'ячів;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
___слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-11.09.21	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	12.09-12.09.2021	виконано
3	Аналіз існуючих методів рішення	13.09-1.09.21	виконано
4	Дослідження існуючих засобів для розпізнавання предметів	19.09-24.09.21	виконано
5	Узгодження подальших дій з науковим керівником	25.09-26.09.21	виконано
6	Аналіз теоретичних відомостей	27.09-15.10.21	виконано
7	Проектування прототипу застосунку	15.10-23.10.21	виконано
8	Узгодження застосунку з науковим керівником	23.10-25.10.21	виконано
9	Реалізація функціоналу застосунку	26.10-14.11.21	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	15.11-16.11.21	виконано
11	Реалізація функціоналу застосунку для симуляції	16.11-24.11.21	виконано
12	Проведення аналізу можливостей розробленого застосунку	25.11-26.11.21	виконано
13	Оформлення звіту	27.11-30.11.21	виконано

Студент _____ Д. В. Троян
(підпис) (прізвище та ініціали)

Керівник роботи _____ В.Г. Вербицький
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 96

Рисунків: 60

Таблиць: 4

Джерел: 31

Троян Д.В. Комп’ютерна система керування роботом для збору тенісних м’ячів : кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В.Г. Вербицький. Запоріжжя : ЗНУ. 2021. 96 с.

Об’єкт дослідження – сукупність складових ІТ, що забезпечує роботоздатність системи - це навігація, фізика руху і нейронна мережа для розпізнавання об'єктів. Завданням науково-дослідницької роботи є дослідження можливостей використання технології для реалізації системи керування роботом для збору тенісних м’ячиків на корті. Задача також вимагає використання нейронної мережі для розпізнавання об’єктів на карті корту. Мета роботи – створення програмного застосунку, який дасть змогу налаштувати систему для керування колісним роботом у автоматичному режимі. Методи досліджень – синтез та аналіз. Результатом роботи є програмна система, що дасть змогу роботу збирати м’ячики на тенісному корті. Галузь застосування охоплює увесь тенісний спорт, де стає проблемою використання людських ресурсів на часу для збору тенісних м’ячів.

Ключові слова: *робот, навігація, OpenCV, Aruco-маркер, нейронна мережа, алгоритм A*, PID-контролер, YOLOv4, Darknet, CNN.*

SUMMARY

Pages: 96

Figures: 60

Tables: 4

Sources: 31

Troyan D.V. Computer-controlled robot control system for collecting tennis balls: master's thesis in the specialty 121 "Software Engineering" / Science head V.G. Verbytsky. Zaporizhzhia : ZNU, 2021. 96 p.

The object of study - a set of components of IT that ensure the operability of the system - a navigation, physics of motion and neural network for object recognition. The task of the research work is to study the possibilities of using the technology to implement a robot control system to collect tennis balls on the court. The task also requires the use of a neural network to detect objects on the map of the court. The purpose of the work is to create a software system that will allow to configure the system to control the robot in automatic mode. Research methods - synthesis and analysis. The result is a software system that will allow you to collect balls on the tennis court.The field of application covers the entire sport of tennis, where it becomes a problem to use human resources in time to collect tennis balls.

Keywords: *robot, navigation, OpenCV, Aruco-marker, neural network, A * algorithm, PID-controller, YOLOv4, Darknet, CNN.*

ЗМІСТ

ВСТУП

Актуальність теми. У світі спорту тільки починає розвиватися автоматизація. Вже використовуються різні камери на сенсори для фіксації гола в футболі і інших подібних спортивних іграх. Однак, роботів що їздять по ігровому полю побачиш ще не часто. Під час гри в теніс на корті зазвичай залишаються м'ячики, які підбирають болбои, але під час тренувань обов’язки по збиранню м’ячів перекладаються на гравців, що значно збільшує їх нераціональне використання часу та взагалі погіршує психологічний стан. Також можуть залучатися співробітники, що несе додаткові втрати власника на заробітну платню. Використовуючи колісного робота для такого завдання, як збір тенісних м'ячиків, можна спростити цей процес зовсім не залучаючи людей. Колісний робот, за допомогою спеціальної системи навігації та нейронних мереж для виявлення цілей і перешкод, зможе зібрати всі лежачі на корті м'ячики і відвести їх в задану точку біля тенісного корту, яка була заздалегідь визначена.

Мета і завдання дослідження. Метою є створення програмного застосунку, який дасть змогу налаштувати систему для керування колісним роботом у автоматичному режимі. Завданням науково-дослідницької роботи є дослідження можливостей використання технології для реалізації готового робота для збору тенісних м’ячиків на корті.

Об’єкт дослідження. Об’єкт дослідження – вся сукупність складових ІТ, що забезпечує роботоздатність системи - це навігація, фізика руху і нейронна мережа для розпізнавання об'єктів.

Предмет дослідження. Предмет дослідження – процес об'єднання різних складових ІТ в єдину структуру та розробки алгоритмів навігації для коректної роботи робота зі штучним інтелектом.

Наукова новизна одержаних результатів. Якщо розглянути існуючі рішення, то можна знайти стартапи, які займаються розробкою подібних роботів, однак навіть стартова орієнтовна ціна рішення досить висока. Знайдені рішення мають недоліки, зв’язані з масою робота, і через те, у процесі руху по корту, колеса можуть залишати сліди та псувати поверхню. Метою є також і збірка робота з невеликим бюджетом, що повинно значно підвищити доступність рішення на ринку, якщо робот піде у комерційну побудову.

Практичне значення одержаних результатів. Готове рішення системи керування роботом для збору тенісних м'ячів може допомогти гравцям економити всій час, проводячи тренування більш ефективно. Можливість зовсім не задіяти людей у процесі збирання м’ячиків допоможе компаніям або власникам тенісних майданчиків не витрачати зайві кошти на людську працю. Також наявність такого інноваційного інвентаря на тенісному корті, як автоматизований колісний робот, може зацікавити більше гравців-клієнтів, що вмотивує їх прийти та пограти.

Апробація результатів роботи. Результати дослідження були представлені на XIII науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького національного університету «Молода наука-2021» [1], а також на XXV науково-технічній конференції студентів, аспірантів, магістрантів і викладачів Інженерного навчально-наукового інституту Запорізького національного університету [2].

Глосарій

- *Kickstarter* – сайт фінансування творчих проєктів за схемою краудфандингу. Kickstarter фінансує різноманітні проєкти, у 13-ти категоріях: мистецтво, комікс, танець, дизайн, мода, фільми і відео, їжа, відеоігри, музика, фотографія, видавництво, технологія, театр. За 3 роки існування більш ніж 1 800 000 людей своїми пожертвами повністю успішно профінансували понад 20 000 проєктів, зібравши більше $200 000 000.
- *YOLO* – це сучасна система виявлення об'єктів в режимі реального часу. Вона був розроблений Джозефом Редмондом. Це система розпізнавання

об'єктів в реальному часі, яка може розпізнавати кілька об'єктів в одному кадрі. З часом YOLO перетворився на новіші версії, а саме: YOLOv2, YOLOv3 і YOLOv4.

- *MS COCO* – це набір даних для великомасштабного виявлення об'єктів та сегментації.
- *PID-контролер* – це інструмент, який використовується в промислових системах управління для регулювання температури, витрат, тиску, швидкості та інших технологічних параметрів. Під-регулятори (пропорційна інтегральна похідна) використовують механізм зворотного зв'язку з контуром управління для управління змінними процесу і є найбільш точним і стабільним контролером.
- *A** - це алгоритм пошуку за першим найкращим збігом на графі, який знаходить маршрут з найменшою вартістю від однієї вершини (початкової) до іншої (цільової, кінцевої). Порядок обходу вершин визначається евристичною функцією.
- *Маркери ArUco* – це двійкові квадратні фідуціарні маркери, які можна використовувати для позиціонування об’єктів за допомогою камери. Їх головна перевага полягає в тому, що їх виявлення є надійним, швидким і простим.
- *Unity3D* – це гнучка і потужна платформа розробки для створення мультиплатформенних 3D і 2D ігор та інтерактивних можливостей.
- *Arduino* – це платформа для створення прототипів електроніки з відкритим вихідним кодом, що дозволяє користувачам створювати інтерактивні електронні об'єкти.
- *Фреймворк* (англ. Framework) — це задана структура, у якій завдання виконуються чи завершуються. Як правило, фреймворк відноситься до часто багаторівневої структури, яка вказує, які програми можуть бути створені і як вони будуть з'єднуватися один з одним.

- *Енкодер* (перетворювач кутових переміщень) – це електронний пристрій, який дозволяє з необхідною точністю виміряти різні параметри обертання будь-якої деталі, як правило, валу електродвигуна або редуктора.
- *Стартап* – це компанія, створена для пошуку відтворюваної та масштабованої бізнес-моделі.
- *Slam* – метод, що використовується роботами та автономними транспортними засобами для побудови карти в невідомому просторі або для оновлення карти у заздалегідь відомому просторі з одночасним контролем поточного розташування та пройденого шляху.
- *Одометрія* – це використання даних від датчиків руху, щоб оцінити зміну положення з часом.
- *Numpy* – високо оптимізована бібліотека для числових операцій.
- *Darknet* – це фреймворк нейронної мережі з відкритим кодом.
- *Колайдер (Unity)* – об’єкт, що визначає форму об'єкта для цілей фізичних зіткнень.
- *Khepera робот* – це невеликий (5,5 см) диференціальний колісний мобільний робот, розроблений в лабораторії LAMI професора Жана-Даніеля Нікуди в EPFL (Лозанна, Швейцарія) в середині 90-х років.
- *Turtlebot* – це недорогий персональний комплект робота із програмним забезпеченням з відкритим вихідним кодом.
- *Сплайн* – функція, область визначення якої розбита на кінцеве число відрізків, на кожному з яких сплайн збігається з деяким багаточленом алгебри (поліномом).
- *IDE* – це інтегроване середовище розробки, яке використовується розробниками для створення різного програмного забезпечення.
- *Fortran* – перша мова програмування високого рівня, що має транслятор. Створена в період з 1954 по 1957 рік групою програмістів під керівництвом Джона Бекуса у корпорації IBM.
- *Nvidia* – компанія, що створює відеокарти.

- *GPU (скор. від Graphics Processing Unit, графічний процесор)* – це окремий процесор розташований на відеокарті, який виконує обробку 2D або 3D графіки.
- *CPU (скор. від Central Processing Unit, центральний процесор комп'ютера)* – електронний блок або інтегральна схема, що виконує машинні інструкції.
- *FPS (англ. frames per second)* – кількість кадрів, що змінюються, за одну секунду.
- *Колісний робот або диференціальний робот* – це мобільний робот, рух якого заснований на двох роздільно ведених колесах, розміщених по обидва боки від корпусу робота. Таким чином, він може змінювати свій напрямок, змінюючи відносну швидкість обертання своїх коліс, і, отже, не вимагає додаткового кермового руху.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРОБЛЕМАТКИКИ СТВОРЕННЯ СИСТЕМИ КЕРУВАННЯ РОБОТОМ ДЛЯ ЗБОРУ ТЕНІСНИХ М’ЯЧИКІВ

1.1 Аналіз нероботизованих рішень для збору тенісних м’ячиків

1.1.1 Сачок для ручного збору тенісних м’ячиків

Цей інструмент має досить великий кошик та ручку, яку можна регулювати по довжині (Рис. 1).

Рис. *1 Сачок для ручного збору тенісних м’ячиків*

1.1.2 Трубка для збору тенісних м’ячиків

Трубка має зручну форму та працює як клапан, що дозволяє зібрати м’ячик при натисканні на нього, але затримує всередині, не даючи впасти назовні (Рис. 2).

Рис. *2 Трубка для збору тенісних м’ячиків*

1.1.3 Конструкція для масового збирання тенісних м’ячиків

Це пристосування потребує прикладання сили, щоб воно проковзувало по поверхні. Має велику область збору, що буде актуально на кортах, які мають на собі велику кількість м’ячиків (Рис. 3).

Рис. *3 Конструкція для масового збирання тенісних м’ячиків*

1.1.4 Полумеханізований пристрій у формі рола-колеса с ручкою

Зручний полумеханізований пристрій у формі рола-колеса с ручкою. Тверді сталеві пружні спиці, які не гнуться і не випадають, дозволяють м’ячикам проскакувати всередину ролу і не випадати з нього. Простий у використанні, але має невеликий розмір об’єму кошика (Рис. 4).

Рис. *4 Полумеханізований пристрій у формі рола-колеса с ручкою*

1.1.5 Тенісний колекторний колісник

Ця система може одночасно збирати до 80 тенісних м’ячів. Підходить для будь-якої поверхні тенісного корту. Ефективний пристрій для збору тенісних м’ячиків, але потребує великої людської сили, щоб працювати з ним (Рис. 5).

Рис. *5 Тенісний колекторний колісник*

1.2 Аналіз роботизованних рішень для збору тенісних м’ячиків

1.2.1 Tennibot

Tennibot (Рис. 6) – це автономний робот, який економить гравцям та тренерам цінний час на практику, уникаючи нудьги переслідування тенісних м'ячів, перевищив мету своєї кампанії у 35 000 доларів менш ніж за 24 години на Kickstarter [3].

Tennibot, який був представлений в LA Times, Time Magazine та інших, вже отримав міжнародне визнання та престижні нагороди, включаючи Інноваційну нагороду Асоціації тенісної індустрії та Почесну нагороду 2018 року від Consumer Electronics Show (CES2018) [4].

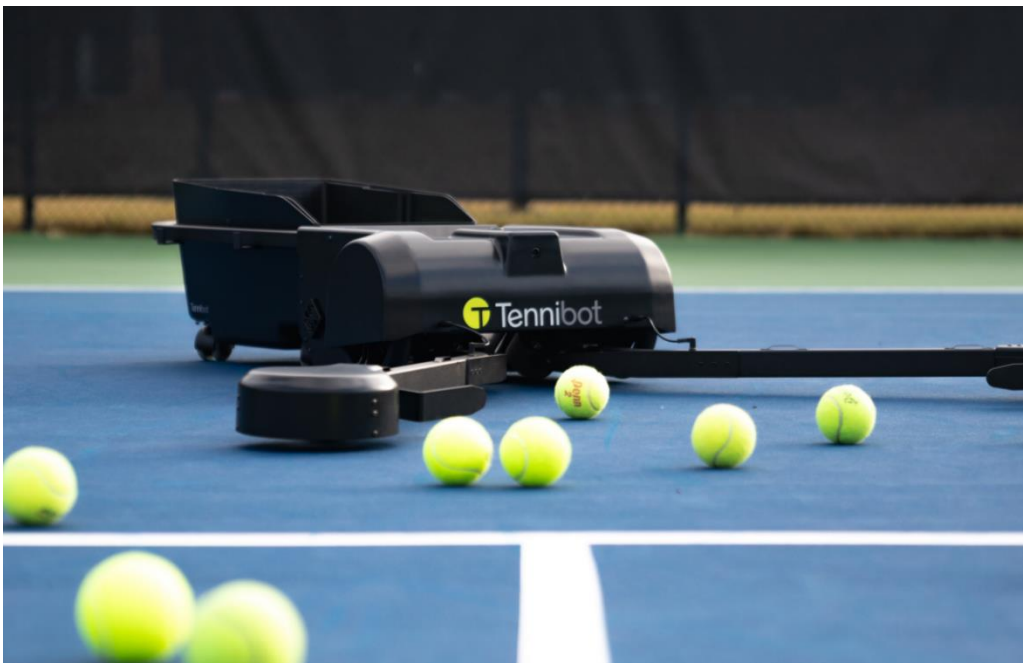

Рис. *6 Робот для збору тенісних м’ячів Tennibot*

Відмінно інтегруючи комп'ютерний зір та штучний інтелект, Tennibot використовує виявлення об'єктів та злиття датчиків для виявлення тенісних м'ячів, тоді як додаток дозволяє гравцеві налаштувати, де на майданчику Теннібот очищатиметься під час сесії. Він збирає до 80 м'ячів, працює як на твердих, так і на глиняних кортах і має акумулятор, який витримує до п’яти годин на одному заряді. Теннібот також автоматично відстежує кількість м'ячів, які гравець б'є, і те, як часто вони практикуються.

Разом з Tennibot поставляється програмний застосунок для смартфона, з якого можна налаштовувати поведінку робота, бачити статистику збору, а також вручну керувати ним (Рис. 7).

Рис. *7 Мобільний застосунок Tennibot*

Tennibot можна використовувати в багатьох ситуаціях на тенісному корті, зокрема, коли:

- один гравець б’є кульовим автоматом;
- один-два гравці практикують подачі, форхеди чи бекхенди, тоді як пристрій підбирає бродячі м’ячі вздовж огорожі та сітки;
- один або кілька гравців б'ють з великого кульового кошика, тоді як Теннібот збирає м'ячі по периметру майданчика;
- тренер проводить індивідуальний або груповий урок.

Елетрабі розробив ідею для Тенібота під час свого навчання в Обернському університеті, де він отримав ступінь магістра ділового адміністрування та доктора філософії в галузі цивільного будівництва.

1.2 Аналіз програмних рішень для розробки системи керування роботом

1.3.1 Нейронна мережа YOLOv4

Виявлення об’єктів - одна з класичних проблем комп’ютерного зору (Рис. 8), щоб розпізнати, що і де - зокрема, які об’єкти знаходяться всередині даного зображення, а також де вони на зображенні. Проблема виявлення об’єкта є більш складною, ніж класифікація, яка також може розпізнавати об’єкти, але не вказує, де об’єкт знаходиться на зображенні. Крім того, класифікація не працює на зображеннях, що містять більше одного об’єкта.

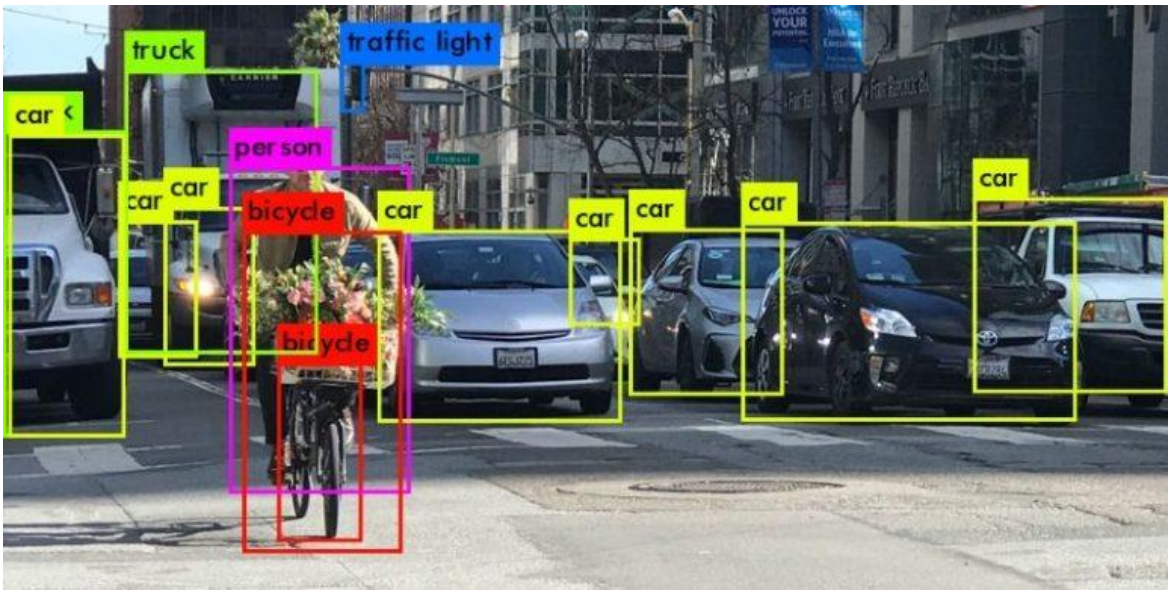

Рис. *8 Виявлення об’єктів нейронною мережою*

YOLO популярний, оскільки він досягає високої точності, а також здатний працювати в режимі реального часу [5]. Алгоритм «only looks once» – «дивиться лише один раз» на зображення в тому сенсі, що для прогнозування йому потрібно лише одне проходження вперед через нейронну мережу (Рис. 9). Після застосування алгоритму Кенні (Що гарантує, що алгоритм виявлення об’єктів виявляє кожен об’єкт лише один раз), він видає розпізнані об’єкти разом із обмежувальними полями.

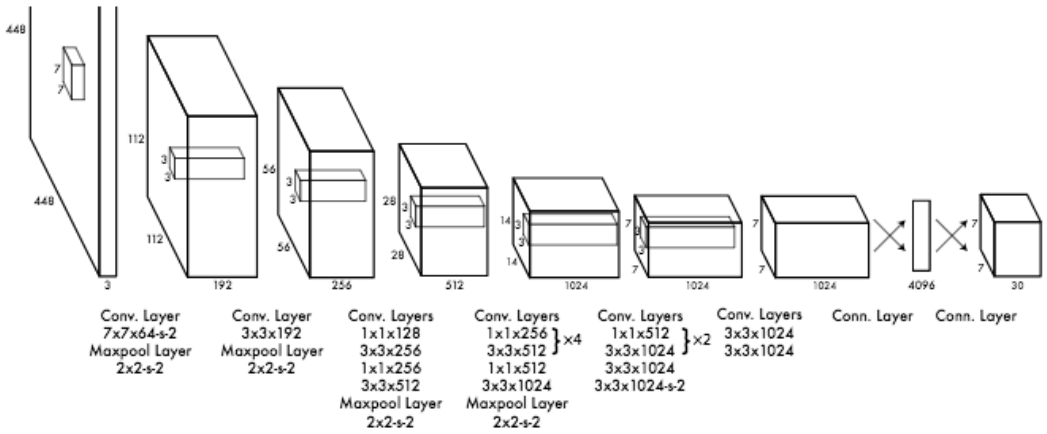

Рис. *9 Архітектура YOLOv1*

З YOLO, один згортковий шар одночасно передбачає кілька обмежуючих полів та ймовірності класів для цих полів. YOLO тренується на повноцінних зображеннях і безпосередньо оптимізує ефективність виявлення. Ця модель має ряд переваг перед іншими методами виявлення об’єктів:

- YOLO надзвичайно швидкий;
- YOLO бачить ціле зображення під час навчання та тестування, тому неявно кодує контекстну інформацію про класи, а також їх зовнішній вигляд;
- YOLO вивчає узагальнюючі подання об'єктів, щоб, навчаючись на природних зображеннях та тестуючи на ілюстраціях, алгоритм перевершував інші найкращі методи виявлення.

Подальші дослідження були проведені в результаті публікації у грудні 2016 року «YOLO9000: Краще, швидше, сильніше» Редмона та Фархаді, обидва з Університету Вашингтона [6]. Ця версія YOLO забезпечила ряд удосконалень методу виявлення, включаючи виявлення понад 9000 об'єктів категорії шляхом спільної оптимізації виявлення та класифікації. Архітектуру другої версії нейронної мережі можна побачити на рисунку 10.

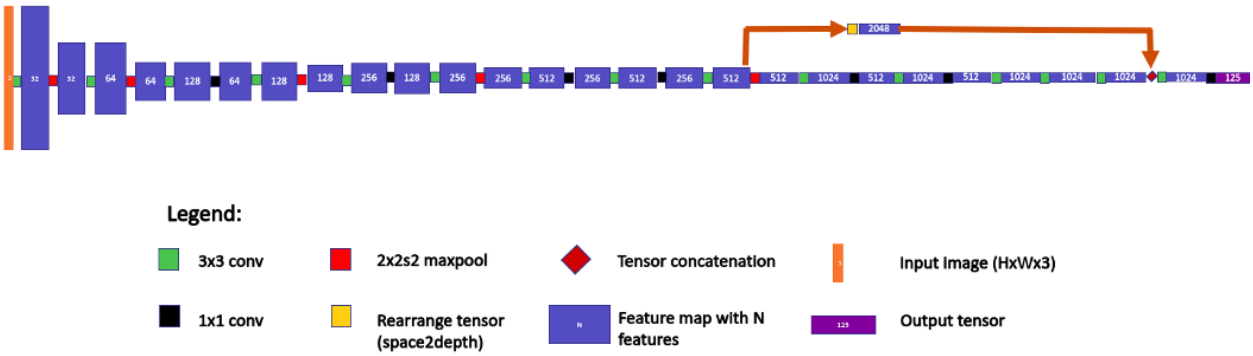

Рис. *10 Архітектура YOLOv2*

Далі ті самі дослідники у квітні 2018 року написали ще одну статтю про свій прогрес у розвитку YOLO – «YOLOv3: Покращення». Як і у випадку з великою кількістю дослідницьких робіт у галузі глибокого навчання, велика частина зусиль – це спроби та помилки. У переслідуванні YOLOv3 цей ефект був сильним, оскільки команда спробувала багато різних ідей, але багато з них так і не вийшло. Деякі речі, які застрягли, включають: нову мережу для вилучення функцій, що складається з 53 згорткових шарів, нову метрику виявлення, прогнозування оцінки «об'єктивності» для кожного обмежувального поля за допомогою логістичної регресії та використання двійкових втрат перехресної ентропії для передбачення класу під час навчання. Кінцевим результатом є те, що YOLOv3 працює значно швидше, ніж інші методи виявлення зі порівнянною продуктивністю [7]. Крім того, YOLO більше не бореться з дрібними предметами. Архітектуру показано на рисунку 11.

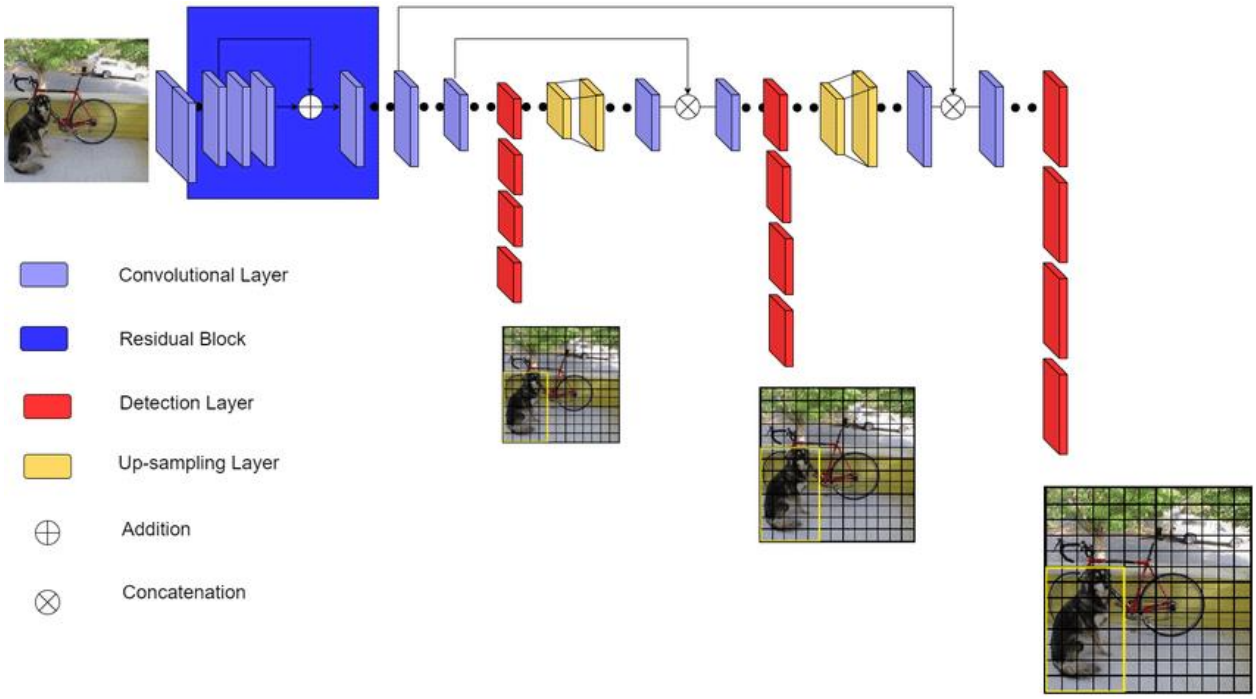

Рис. *11 Архітектура YOLOv3*

Всі моделі YOLO є моделями виявлення об'єктів. Моделі виявлення об'єктів навчаються перегляду зображення і пошуку підмножини класів об'єктів. При виявленні ці класи об'єктів полягають в обмежувальну рамку, і їх клас ідентифікується. Моделі виявлення об'єктів зазвичай навчаються і оцінюються на основі набору даних COCO, який містить широкий діапазон з 80 класів об'єктів. Звідси передбачається, що моделі виявлення об'єктів будуть узагальнені на нові завдання виявлення об'єктів, якщо вони будуть піддані новим навчальним даним. Ось приклад того, як можна використовувати YOLOv4 для виявлення клітин у крові (Рис. 12).

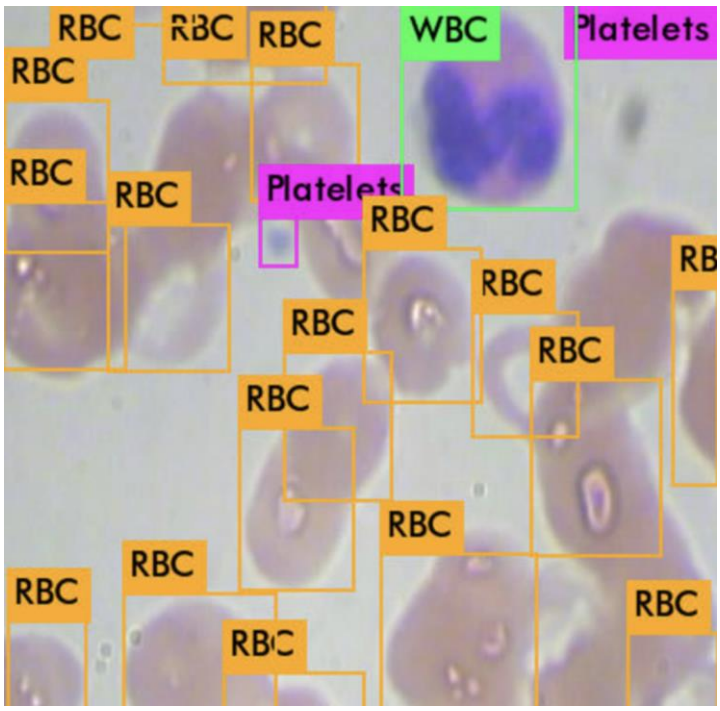

Рис. *12 YOLOv4 робить висновки про клітини в крові*

Реальний час роботи особливо важливий для моделей виявлення об’єктів, які працюють на відеопотоках. Інша перевага моделей виявлення об’єктів у реальному часі полягає в тому, що вони невеликі і ними можуть легко користуватися усі розробники. YOLOv4 робить виявлення в режимі реального часу пріоритетом і проводить навчання на одному графічному процесорі [8]. Намір авторів полягає в тому, щоб інженери та розробники легко використовували свою структуру YOLOv4 у спеціальних сферах. Архітектуру YOLOv4 можна бачити на рисунку 13.

Всі детектори об'єктів приймають зображення і стискають об'єкти за допомогою згорткової нейронної мережі. При класифікації зображень ці зворотні зв'язки є кінцем мережі, і на їх основі можна робити прогнози. При виявленні об'єктів навколо зображень разом з класифікацією необхідно намалювати кілька обмежуючих прямокутників, тому шари об'єктів згорткової основи необхідно змішувати і відображати в світлі один одного.

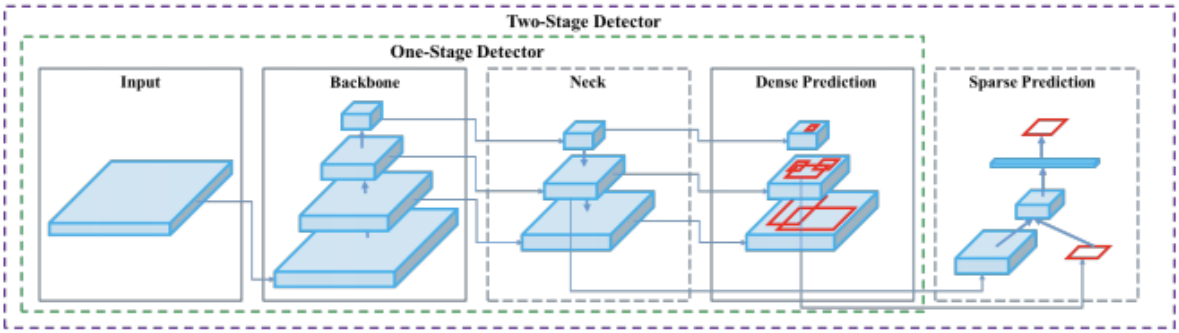

Рис. *13 Архітектура YOLOv4*

Автори розглянули такі основи для детектора об'єктів YOLOv4:

- CSPResNext50;
- CSPDarknet53;
- EfficientNet-B3.

CSPResNext50 і CSPDarknet53 засновані на DenseNet (Рис. 14). DenseNet був розроблений для з'єднання шарів у згорткових нейронних мережах із наступними спонуканнями: пом'якшити проблему зникаючого градієнта (важко повернути сигнали втрат через дуже глибоку мережу), посилити поширення функцій, спонукати мережу до повторного використання функцій та зменшити кількість параметрів мережі.

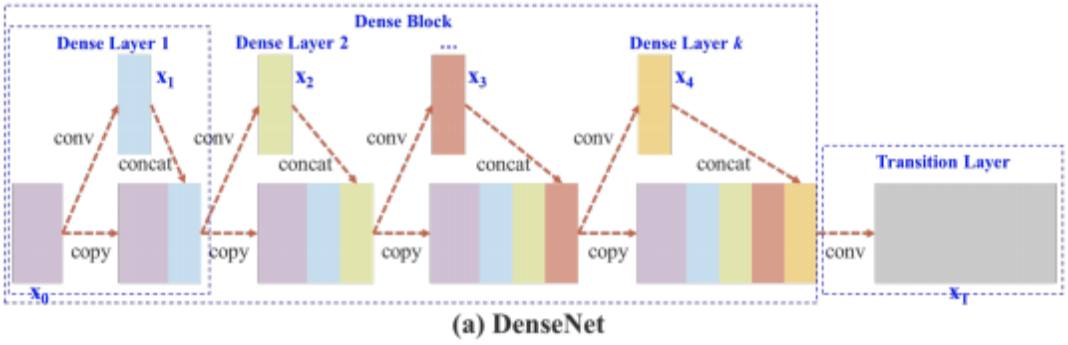

Рис. *14 Архітектура DenseNet*

У CSPResNext50 та CSPDarknet53, DenseNet відредаговано (Рис. 15), щоб відокремити карту функцій базового шару, скопіювавши її та відправи-

вши одну копію через щільний блок, а іншу направляючи на наступний етап. Ідея CSPResNext50 та CSPDarknet53 полягає у тому, щоб усунути обчислювальні вузькі місця в DenseNet та покращити навчання, передавши нередаговану версію карти функцій.

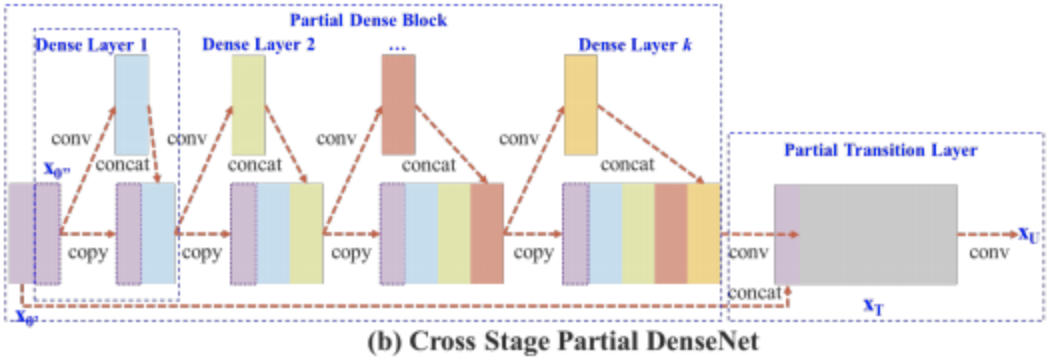

Рис. *15 Архітектура CSP DenseNet*

EfficientNet був розроблений Google Brain для вивчення в першу чергу проблеми масштабування згорткових нейронних мереж. Існує багато рішень, які можна прийняти при масштабуванні ConvNet, включаючи розмір вводу, масштабування по ширині, масштабування по глибині та масштабування всього вищезазначеного. EfficientNet перевершує інші мережі порівнянного розміру за класифікацією зображень. Однак автори YOLOv4 стверджують, що інші мережі можуть працювати краще в налаштуваннях виявлення об'єктів і вирішити експериментувати з усіма ними.

Виходячи з їх інтуїції та експериментальних результатів, остаточна мережа YOLOv4 реалізує CSPDarknet53 як основну мережу. Методи у YOLOv4 були реально перевірені експериментами на MS COCO. COCO містить 80 об'єктивних класів та призначених для представлення широкого кола сценаріїв виявлення об'єктів, з якими детектор може зіткнутися у реальному світі.

В остаточній конфігурації YOLOv4 досягає гарних статистичних характеристик для виявлення об’єктів. У роботі авторів досліджується час виве-

дення на багатьох різних графічних процесорах. На рисунку 16 показано загальне порівняння.

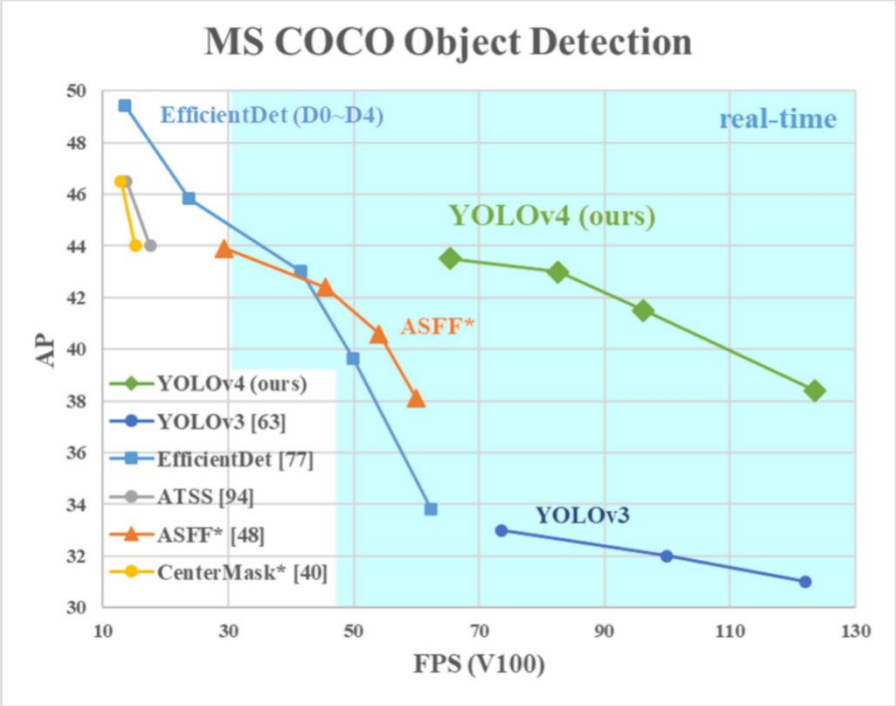

Рис. *16 Порівняння продуктивності різних версій нейронних мереж*

1.3.2 Aruco маркер

Для того, щоб отримувати позицію робота у реальному часі, потрібна відповідна система трекінгу у просторі. Для рішення цієї задачі можна розглянути наступні варіанти:

- Колісна одометрія за допомогою енкодерів, розташованих безпосередньо на провідних або додаткових вимірювальних колесах дозволяє вважати пройдену роботом відстань і визначати його положення щодо початкової точки. Це один з найбільш швидких джерел інформації про переміщення для робота. Однак на практиці, одометрія схильна до проблеми накопичується помилки, яку якимось чином необхідно коригувати. Можна вирівнюватися об поверхню з свідомо відомими координатами, а можна комплексувати з додатковою системою. Також якщо використовувати енкодери на провідних колесах, то

існує проблема прослизання колеса при різких маневрах і при зіткненнях робота. Вимірювальні колеса дозволяють позбутися від цієї проблеми, але їх важко розмістити в невеликому роботі.

- Лідари мають вагомі переваги: можна отримати майже круговий огляд і відстань до перешкод. З мінусів варто відзначити високу ціну, складності при спробі відрізнити невеликі далекі об'єкти від сміття, відносно складний алгоритм локалізації. Дешеві лідари мають досить низьку частоту оновлення і кутовий дозвіл, а так само схильні до засвітів від інших лідарів, далекомірів і т. д. У роботах складно знайти місце для лідара на рівні бортика для використання класичних Slam-алгоритмів, а на передбаченій правилами висоті для систем локалізації мало статичних об'єктів для прив'язки.
- Інерціальні системи теж можуть вікористовуватися для локалізації робота у просторі, але є значні недоліки. Акселерометр шумить, гіроскоп пливе, про використання компаса всередині приміщення і зовсім можна забути. Навіть застосування фільтра Калмана не дозволяє досягти бажаної точності. IMU виходить ефективно використовувати тільки для корекції на дуже короткому проміжку часу.
- GPS не працює в приміщеннях, а точність без RTK – метри.
- Ультразвукова локалізація. Нажаль, точність не дуже висока (+-2 см), сильно впливають перехресні перешкоди, або, якщо використовувати датчики із захистом від цього, виходить низька частота опитування.
- Стереокамери мають вузький кут, що не дозволяє ні накрити область трекінгу при розташуванні на центральному пристрої стеження, ні отримати кругової або хоча б прийнятний огляд при установці на робота.
- Фідуціарні маркери найбільш підходять для мети проекту, які можна розглянути більш детально. Існують кілька різних типів подібних маркерів (Рис. 17).

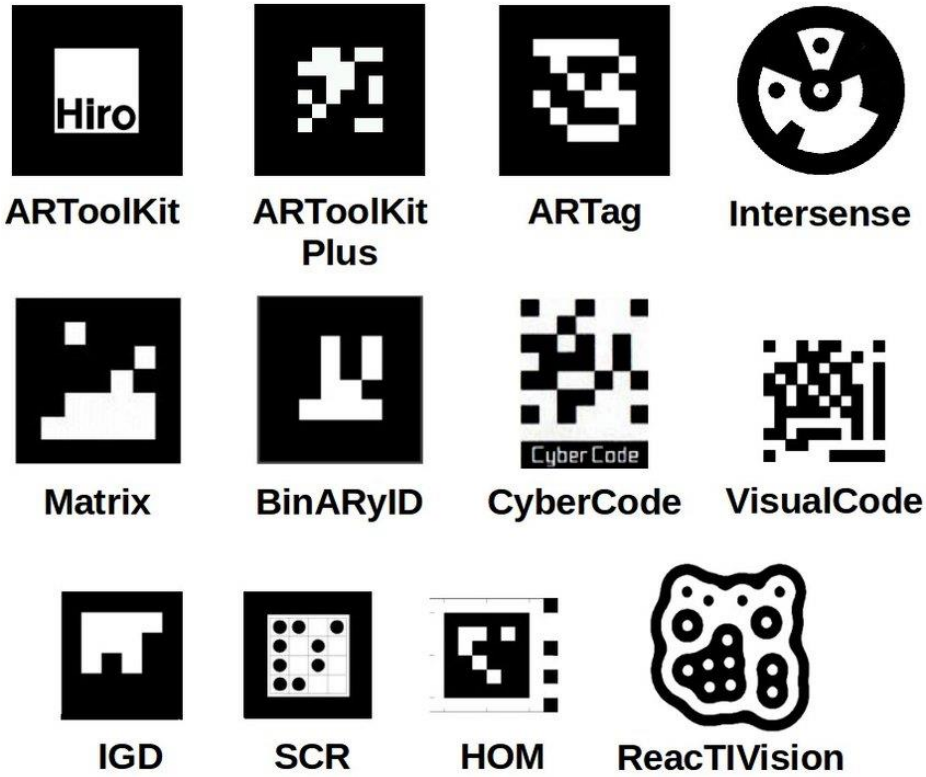

Рис. *17 Різні види фідуціарних маркерів*

Крім визначення положення маркера часто виникає потреба визначення його унікальності, наприклад, з метою розпізнавання відразу декількох об'єктів. Для цього можна змінювати кольори або форму, але тоді можна швидко зіткнутися з падінням надійності розпізнавання.

Дана проблема досить поширена і існує багато стандартів маркерів. Найвідоміший з них – QR код, але він рідко застосовується для завдань локалізації через надмірність і необхідність високого дозволу. Як правило, всі маркери двоколірні (найчастіше чорно-білі), простої форми (найчастіше квадрат) і якимось чином кодують ідентифікатор в собі. Найбільш відомими маркерами для локалізації є Aruco, AprilTag, ARToolKit.

ArUco – це бібліотека з відкритим кодом для оцінки пози камери з використанням квадратних маркерів (Рис. 18). ArUco написаний на C++ і працює надзвичайно швидко [9].

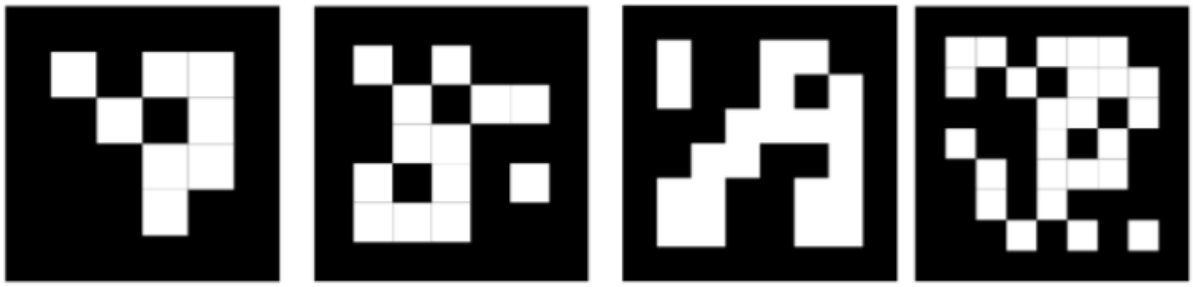

Рис. *18 Маркери ArUco з різними розмірами матриць (матриці 4x4, 5x5, 6x6, 7x7)*

Основні характеристики:

- Виявлення маркерів за допомогою одного рядка коду C++.
- Працює з різними словниками: ARUCO, AprilTag, ARToolKit+, ARTAG, CHILITAGS.
- Швидше, ніж будь-яка інша бібліотека для виявлення маркерів.
- Кілька залежностей OpenCV (>=2.4.9) і eigen3 (включені в бібліотеку).
- Тривіальна інтеграція з OpenGL і OGRE.
- Швидкий, надійний і кроссплатформенний (Windows, Linux, Mac OS, Android).

Цей варіант фідуціарного маркера ідеально підходить для використання у проекті з роботом через швидкість та функціонал (можна отримати як позицію у просторі так і напрям руху) [10]. На рисунку 19 можна побачити можливості Aruco-маркерів.

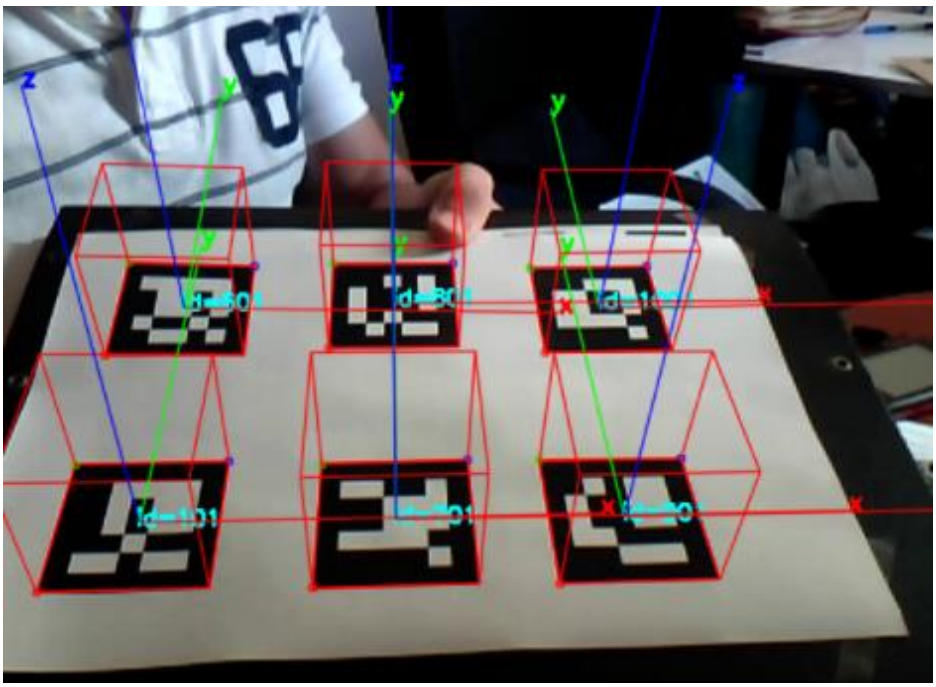

Рис. *19 Приклад розпізнвання Aruco маркерів у реальному часі*

1.3.3 OpenCV

OpenCV – це величезна бібліотека з відкритим кодом для комп'ютерного зору, машинного навчання та обробки зображень. OpenCV підтримує широкий спектр мов програмування, таких як Python, C++, Java і багато інших. Бібліотека може обробляти зображення і відео для ідентифікації об'єктів, осіб або навіть почерку людини. Також можлива інтеграція з різними бібліотеками, такими як Numpy, яка є високо оптимізованою бібліотекою для числових операцій. Тобто будь-які операції, які можна виконувати в Numpy, можна комбінувати з OpenCV [11].

Використовуючи бібліотеку OpenCV, можна отримати наступний функціонал:

- Читання та запис зображень.
- Захоплення та збереження відео.
- Обробка зображень (фільтрація, перетворення).
- Розпізнавання особливих рис.
- Виявлення певних об'єктів, такі як обличчя, очі, автомобілі, на відео або зображеннях.

- Аналіз відео, тобто оцінка руху в ньому, зміна фону і відстеження об'єктів в ньому.

OpenCV спочатку був розроблений на C++. На додаток до цього були передбачені прив'язки Python і Java. OpenCV працює в різних операційних системах, таких як Windows, Linux, OSx, FreeBSD, NetBSD, OpenBSD та інші.

Для дипломного проекту з функціоналу бібліотеки OpenCV будуть використовуватися перетворення для зображень, функція перспективної трансформації та розпізнавання Aruco маркерів.

1.3.4 Проективні і афінні перетворення

У математиці лінійне перетворення – це функція, яка відображає один векторний простір в інший і часто реалізується матрицею. Відображення вважається лінійним перетворенням, якщо воно зберігає векторне додавання і скалярне множення. Щоб застосувати лінійне перетворення до вектора (тобто координатам однієї точки, в даному випадку – значенням x і y пікселя), необхідно помножити цей вектор на матрицю, що представляє лінійне перетворення [12]. Як висновок отримаємо вектор з перетвореними координатами.

Існують два класи лінійних перетворень – проективні і афінні. Афінні перетворення є окремим випадком проективних перетворень. Обидва перетворення можуть бути представлені наступною матрицею:

$$\begin{pmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ \mathrm{c}_1 & \mathrm{c}_2 & 1 \end{pmatrix},$$

де

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix},$$

являє собою матрицю обертання. Ця матриця визначає вид перетворення, яке буде виконано: масштабування, поворот і так далі.

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

є вектором перекладу. Він просто переміщує точки.

$$(c_1\ c_2),$$

є вектором проекції. Для афінних перетворень всі елементи цього вектора завжди рівні 0.

Якщо x і y є координатами точки, перетворення може бути виконано простим множенням:

$$\begin{pmatrix} a_1\ a_2\ b_1 \\ a_3\ a_4\ b_2 \\ c_1\ c_2\ 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}.$$

Тут x' і y ' — координати перетвореної точки.

Проективне перетворення показує, як змінюються сприймані об'єкти, коли змінюється точка зору спостерігача [13]. Це перетворення дозволяє створювати спотворення перспективи. Афінне перетворення використовується для масштабування, перекосу і повороту.

Різниця між проективними і афінними перетвореннями. Єдина відмінність між цими двома перетвореннями полягає в останньому рядку матриці перетворень. Для афінних перетворень перші два елементи цього рядка повинні бути нулями. Але це призводить до різних властивостей двох операцій:

- Проективне перетворення не зберігає паралельність, довжину і кут. Але він все ще зберігає колінеарність і частоту.

- Оскільки афінне перетворення є окремим випадком проективного перетворення, воно володіє тими ж властивостями. Однак, на відміну від проективного перетворення, воно зберігає паралелізм.

Проективне перетворення може бути представлено як перетворення довільного чотирикутника (тобто системи з чотирьох точок) в інший. Афінне перетворення — це перетворення трикутника. Оскільки останній рядок матриці обнулена, достатньо трьох точок. Різницю можна бачити на рисунку 20.

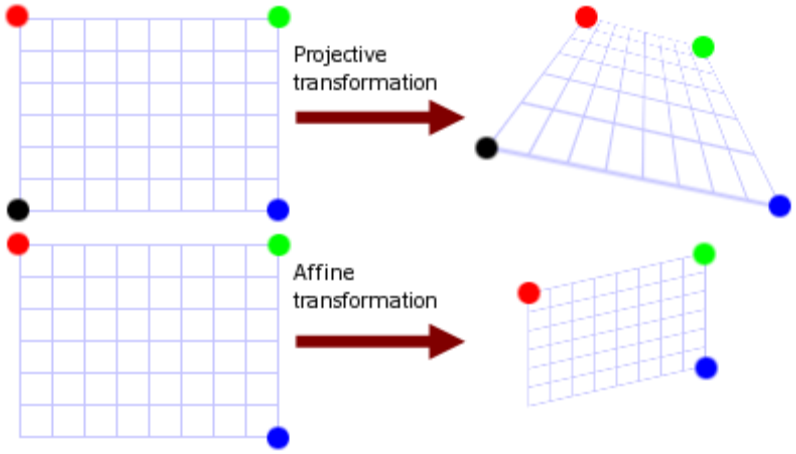

Рис. *20 Проективне і афінне перетворення*

Лінійні перетворення не завжди можуть бути розраховані за допомогою матричного множення. Якщо матриця перетворення сингулярна, це призводить до проблем. Матриця перетворення є сингулярною, коли вона являє собою невипуклий чотирикутник. Форма є опуклою, коли кожна точка, що лежить між двома точками, що належать цій формі, також належить тій же формі. Говорячи простіше, якщо чотирикутник є самоперетинаючим, або якась вершина лежить «всередині» чотирикутника, або деякі вершини розташовані в одній і тій же точці, цей чотирикутник неопуклий (Рис. 21). Тому при виконанні лінійного перетворення, треба переконатеся, що воно не буде сингулярним.

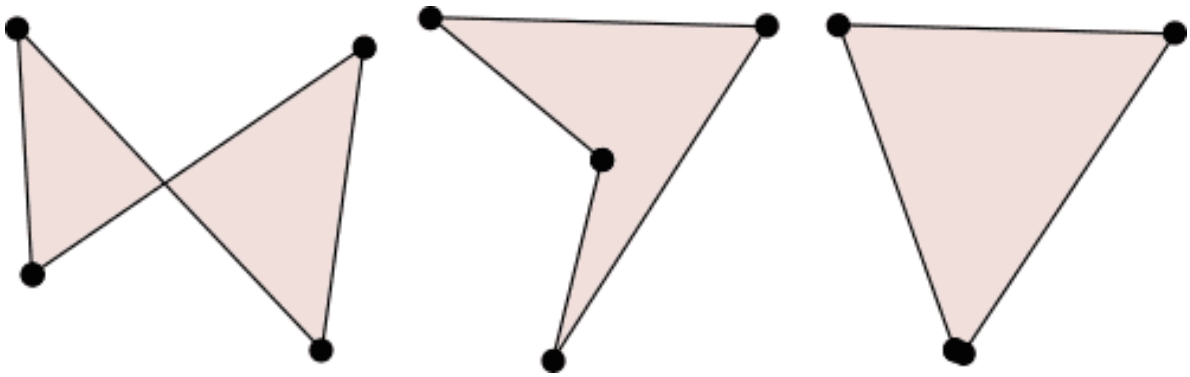

Рис. *21 Приклади невипуклих чотирикутників*

1.3.5 Unity

Unity – це движок для створення ігор на декількох платформах. Unity був випущений компанією Unity Technologies в 2005 році (Рис. 22). У центрі уваги Unity знаходиться розробка 3D і 2D ігор та інтерактивного контенту [14]. Unity тепер підтримує 27 різних цільових платформ для розгортання. Найбільш популярними платформами є системи Android, ПК і iOS.

Особливості ігрового движка Unity:

- Unity – це інтегрована платформа, яка використовується в якості ігрового движка і фреймворку.
- Unity дозволяє вам розробляти один раз і публікувати скрізь.
- Хоча Unity вважається більш підходящим для створення 3D-ігор, його також можна в рівній мірі використовувати для розробки 2D-ігор.
- В Unity можна розробляти ігри з великими ресурсами, не залежними від додаткових фреймворків або движків. Це дійсно покращує роботу користувачів.
- За допомогою Unity розробники ігор можуть безкоштовно отримати доступ до безлічі ресурсів, таких як інтуїтивно зрозумілі інструменти, готові ресурси, зрозуміла документація, онлайн-спільнота і тому подібне. для створення захоплюючого 3D-контенту в іграх.
- Відстеження активів і рендеринг, написання сценаріїв.

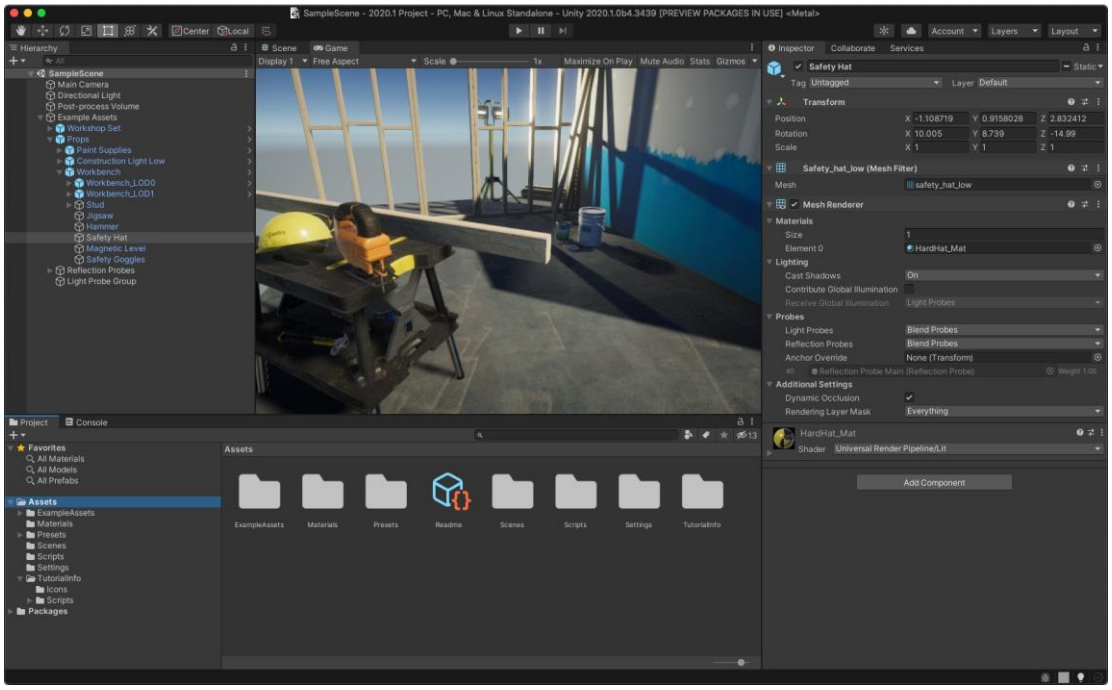

Рис. *22 Інтерфейс редактора Unity*

Ігровий движок Unity ідеально підходить для розробки симулятора середовища системи керування робота для збору тенісних м’ячиків. Вбудована в движок фізика в єдності гарантує, що об'єкти прискорюються і реагують на зіткнення, гравітацію і різні інші сили. Unity надає різні реалізації фізичного движка, які можна використовувати відповідно до потреб проекту [15]. У Unity є усе, щоб створити необхідне середовище для тестів у симуляції.

Для симуляції фізики коліс можна використовувати компонент Wheel Collider (Рис. 23). Це особливий вид колайдера, який використовується для транспортних засобів. Він має вбудовану техніку виявлення зіткнень і фізику реального колеса.

Його можна використовувати для об'єктів, відмінних від коліс (наприклад, човнів-бамперів, автомобілів-бамперів і тому подібних, які використовують силу підвіски), але він спеціально розроблений для транспортних засобів з колесами [16].

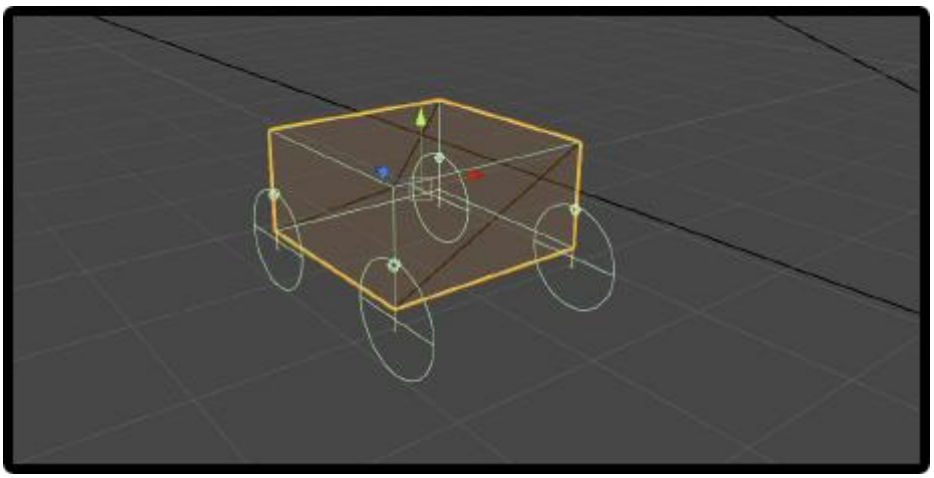

Рис. *23 Візуалізація компонента Wheel Collider у редакторі Unity*

1.3.6 Алгоритм A*

A* – найпопулярніший вибір для пошуку шляхів, оскільки він досить гнучкий і може використовуватися в широкому діапазоні контекстів. A* схожий на інші алгоритми пошуку графіків в тому, що він потенційно може виконувати пошук на величезній площі карти. Він схож на алгоритм Дейкстри (Рис. 24) в тому, що його можна використовувати для пошуку найкоротшого шляху. Також він схож на Greedy Best First Search (Рис. 25) в тому сенсі, що може використовувати евристику [17]. У простому випадку це так само швидко, як і у Best First Search.

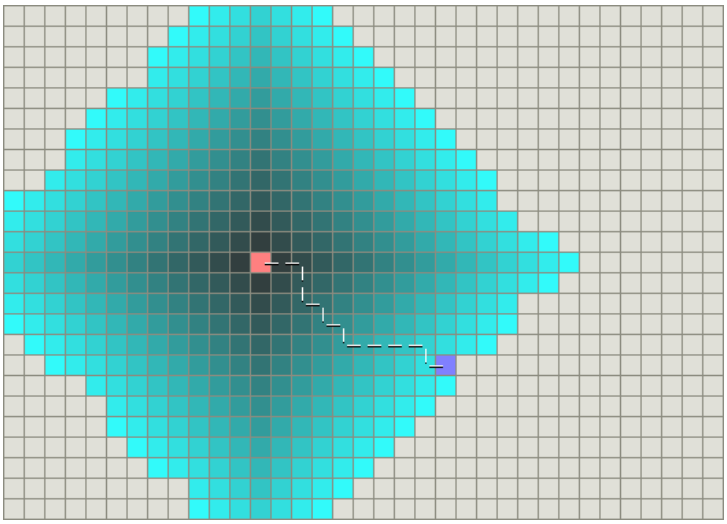

Рис. *24 Обробка алгоритмом Дейкстри. Синій колір показує кількість вузлів, відвіданих алгоритмом Дейкстри*

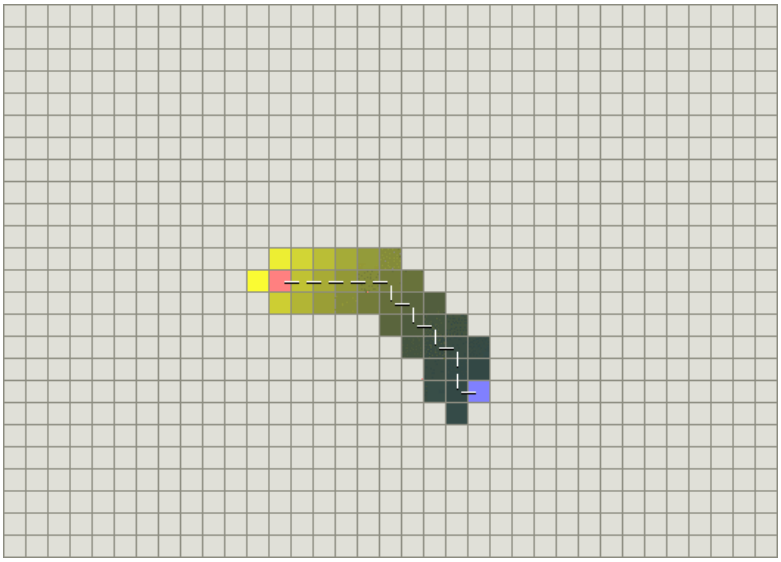

Рис. *25 Обробка алгоритмом Best First Search. Кольорові вузли показують кількість відвіданих вузлів. Жовтий колір означає вузли, розташовані ближче до вихідного вузла, а зелений – вузли, розташовані ближче до місця призначення*

Алгоритм Дейкстри гарантовано завжди дасть кращий шлях. Оскільки він використовує точну вартість, необхідну для переходу з одного вузла на інший. Але найкращий перший пошук не гарантує, що завжди дасть вам найкращий шлях. Це пов'язано з тим, що він використовує функцію евристика, яка являє собою оцінку відстані від одного вузла до місця призначення. Тому найкращий перший пошук не дасть правильного найкоротшого шляху, коли є перешкода.

Це призводить до важливості алгоритму пошуку, який являє собою комбінацію алгоритмів Дейкстри і Best First Search. Він використовує суму витрат (відстань від джерела до поточного вузла, що визначається g (x)), а також евристику (розрахункова відстань від поточного вузла до місця призначення, що визначається h (x)).

Таким чином, A* дасть правильні результати, навіть якщо буде передбачено перешкоду, оціночні значення, тобто евристика несуперечлива.

1.3.7 PID-контролер

PID-контролер означає пропорційно-інтегрально-похідне управління. Це механізм зворотного зв'язку, який використовується в системі управління. Цей тип управління також називається тричленним управлінням і реалізується PID-контролером. Обчислюючи і контролюючи три параметри – пропорційний, інтегральний і похідний, від того, наскільки змінна процесу відхиляється від бажаного заданого значення, – можна домогтися різних керуючих впливів для конкретної роботи (Рис. 26).

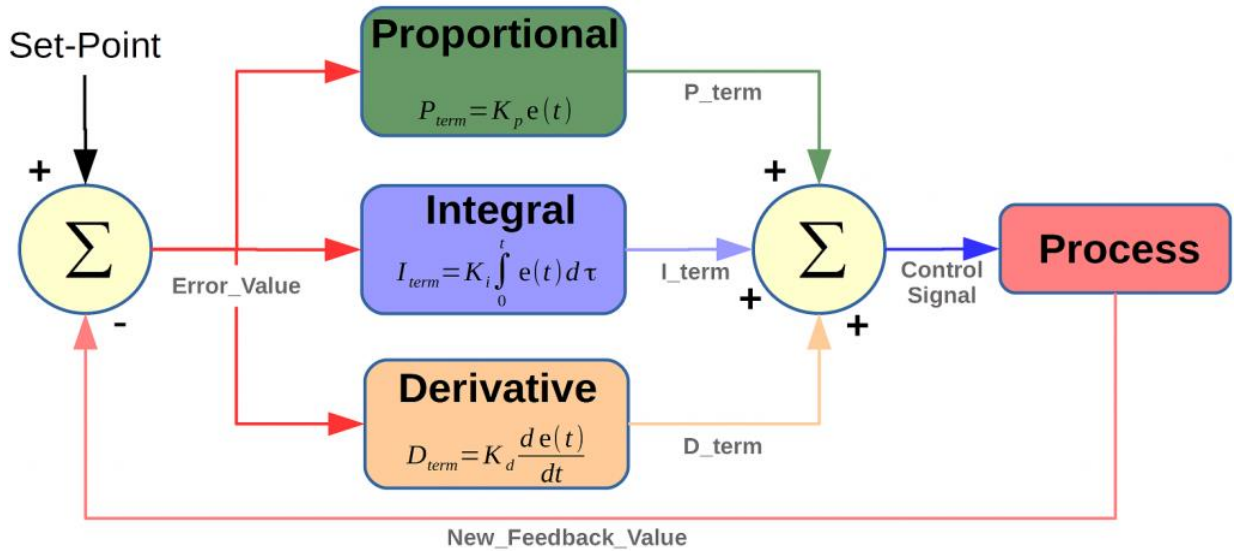

Рис. *26 Загальна схема PID-контролера*

Під-контролери вважаються кращими контролерами в сімействі систем управління. Ніколас Мінорський опублікував теоретичну аналітичну роботу з під-контролера [18]. Для PID-управління виконавчий сигнал складається з пропорційного сигналу помилки, доданого з похідною і інтегралом сигналу помилки. Отже, виконавчий сигнал для управління є:

$$e_a(t) = e(t) + T_d \frac{de(t)}{dt} + K_i \int e(t)dt.$$

Перетворенням Лапласа виконавчого сигналу, що включає PID-контроль, є:

$$E_a(s) = E(s) + sT_d E(s) + \frac{K_i}{s} E(s),$$

чи

$$E_a(s) = E(s)\left[1 + sT_d + \frac{K_i}{s}\right].$$

Існують деякі керуючі дії, які можуть бути досягнуті за допомогою будь-якого з двох параметрів PID-контролера. Два параметри можуть працювати, зберігаючи третій рівним нулю. Таким чином, PID-контролер іноді стає PI (пропорційно-інтегральним), PD (пропорційно-похідним) або навіть P або I. Похідний член D відповідає за вимірювання шуму, в той час як інтегральний член призначений для досягнення цільового значення системи. У перші дні PID-контролер використовувався в якості механічного пристрою. Це були пневматичні контролери, так як вони були стиснуті повітрям. Механічні регулятори включають пружину, важіль або масу. Багато складні електронні системи забезпечені контуром PID-управління. В наші дні PID-контролери використовуються в ПЛК (програмованих логічних контролерах) в промисловості. Пропорційні, похідні та інтегральні параметри можуть бути виражені як – Kp, Kd і Ki. Всі ці три параметри впливають на систему управління із замкнутим контуром. Це впливає на час наростання, час осідання і перевищення, а також на усталену помилку (Таблиця 1).

Таблиця *1*

Вплив компонент PID-контролера на вивідне значення

Реакція контролю	Час зростання	Час встановлення	Час перевиконання	Помилка стійкого стану
Kp	зменшення	мала зміна	збільшення	зменшення
Kd	мала зміна	зменшення	зменшення	без змін
Ki	зменшення	збільшення	збільшення	усунення

PID-контролер поєднує в собі переваги пропорційного, похідного та інтегрального керуючих впливів. Можна коротко описати ці контрольні дії:

Пропорційне управління. Тут виконавчий сигнал для керуючого впливу в системі управління пропорційний сигналу помилки. Сигнал помилки являє собою різницю між опорним вхідним сигналом і сигналом зворотного зв'язку, отриманим з вхідного сигналу.

Управління похідною. Виконавчий сигнал складається з пропорційного сигналу помилки, доданого до похідної сигналу помилки. Отже, виконавчий сигнал для похідного керуючого впливу задається:

$$e_a(t) = e(t) + T_d \frac{de(t)}{dt},$$

де T_d – це константа.

Інтегральне управління. Для інтегрального керуючого впливу виконавчий сигнал складається з пропорційного сигналу помилки, доданого до інтегралу сигналу помилки. Отже, виконавчий сигнал для інтегрального керуючого впливу задається:

$$e_a(t) = e(t) + K_i \int e(t)dt,$$

де K_i – це константа.

PID-контролер має деякі обмеження, крім того, що він є одним з кращих контролерів в системі управління. PID-контролер застосовується до багатьох керуючих впливах, але він погано працює в разі оптимального управління [19]. Основним недоліком є шлях зворотного зв'язку. PID не забезпечений будь-якою моделлю процесу. Інші недоліки полягають у тому, що PID є лінійною системою, а похідна частина чутлива до шуму. Невелика кількість шуму може призвести до значних змін на виході.

1.3 Аналіз апаратних рішень для розробки робота для збору тенісних м’ячиків

1.3.1 Arduino Uno

Arduino Uno контролер побудований на ATmega328 [20]. Платформа має 14 цифрових вхід / виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, кварцовий генератор 16 МГц, роз'єм USB, силовий роз'єм, роз'єм ICSP і кнопку перезавантаження (Рис. 27). Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC/DC або батареї.

Рис. *27 Контролер Arduino Uno*

«Uno» перекладається як «один» з італійського і розробники тим самим натякають на прийдешній вихід Arduino 1.0 [21]. Нова плата стала флагманом лінійки плат Arduino. Для порівняння з попередніми версіями можна звернутися до повного списку плат Arduino.

Платформа програмується за допомогою ПЗ Arduino. Мікроконтролер ATmega328 поставляється з записаним завантажувачем, що полегшує запис нових програм без використання зовнішніх програматорів. Зв'язок здійснюється оригінальним протоколом STK500 [22].

Контролер Arduino Uno може бути із вбудованим wifi-адаптером, що дозволить передавати необхідні команди керування на робота.

1.3.2 Модуль драйверів двох колекторних моторів TA6586

Також для забезпечення функції керування моторами колес, потрібен відповідний модуль драйверів двох колекторних моторів, наприклад TA6586 [23] (Рис. 28).

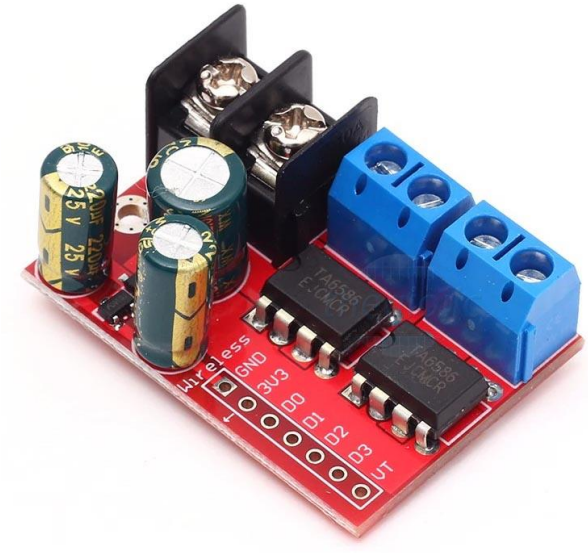

Рис. *28 Модуль драйверів двох колекторних моторів TA6586*

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ ДЛЯ СТВОРЕННЯ СИСТЕМИ КЕРУВАННЯ РОБОТОМ ДЛЯ ЗБОРУ ТЕНІСНИХ М’ЯЧІВ

2.1 Згорткові нейронні мережі

2.1.1 Загальні поняття

Останнім часом у світі виникає величезний інтерес до глибокого навчання [24]. Найбільш усталеним алгоритмом серед різноманітних глибоких моделей навчання є згорткова нейронна мережа (CNN) – це клас штучних нейронних мереж, який став домінуючим в задачах комп'ютерного зору з того моменту, як стали відомі приголомшливі результати на конкурсі розпізнавання об’єктів, відомому як ImageNet Large Scale Visual Recognition Competition (ILSVRC) у 2012 р. [25, 26]. Досить велика кількість досліджень вже опубліковано у таких областях, як виявлення уражень у медицині, класифікація, сегментація, реконструкція зображення, а також обробка природної мови.

CNN – це тип глибокої нейронної мережі для обробки даних, яка має сітчасту структуру, як у зображеннях. Модель взяла свій початок з організації будови візуального кортексу у тварин і спроектована автоматично та адаптивно навчатися на просторових ієрархіях унікальних особливостей. Згорткова нейронна мережа спроектована математично і складається з трьох типів шарів: згортковий (convolution), що об'єднує (pooling) і з'єднує (fully connected layer). Перші два шари, згортковий і об'єднуючий, відповідають виявлення деяких унікальних особливостей на зображенні. Третій, що з'єднує, служить як вихідний шар і бере на себе роботу з класифікації. Згортковий шар грає ключову роль у CNN, він заснований на цілому наборі математичних операцій, таких як згортка – особливий тип лінійних операцій. У цифрових зображеннях, пікселі зберігаються в двомірній (2D) сітці, тобто в масиві чисел (Рис. 29). Також існує маленька сітка параметрів, яка називається ядром (kernel). Ядро займається виявленням деяких особливостей і застосо-

вується на кожній ділянці зображення, що робить згорткову нейронну мережу дуже оптимізованої в роботі з обробкою зображень, де певна унікальна особливість може бути де завгодно на зображенні. Так як вихід одного шару є входом для іншого, знайдені унікальні особливості можуть ієрархічно і прогресивно ставати складнішими. Процес оптимізації параметрів, таких як ядро (kernel), називається навчанням, і служить для мінімізації відмінностей між тим, що видає вихідний шар згорткової нейронної мережі і даними, що описують те, що дійсно знаходиться на зображенні. Даний алгоритм оптимізації включає в себе зворотне поширення (backpropagation) і градієнтний спуск (gradient descent).

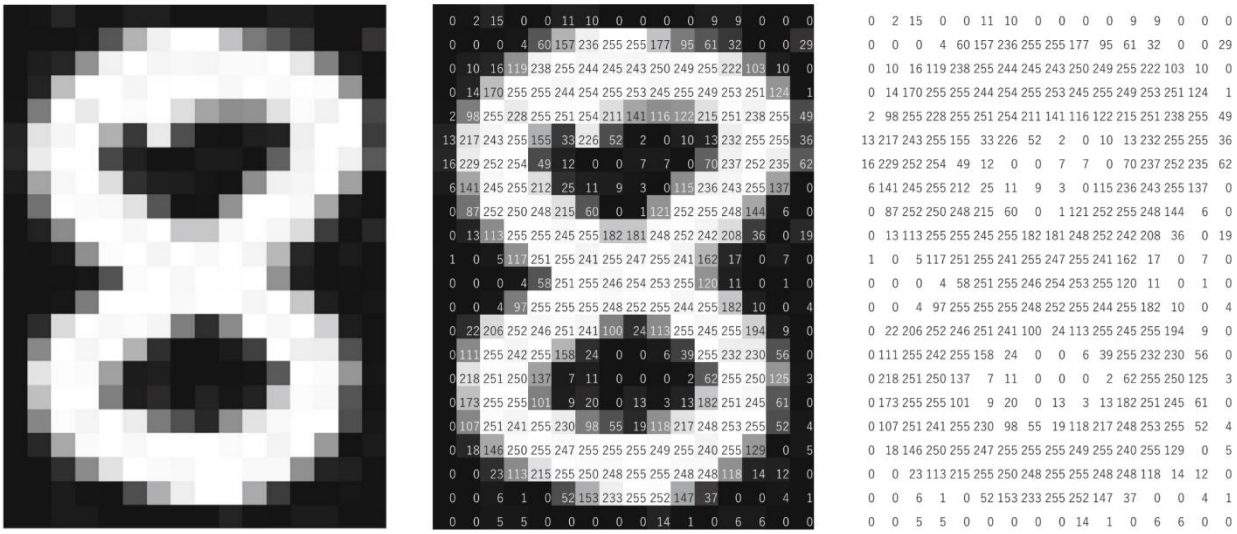

Рис. *29 Комп'ютер бачить зображення як масив чисел. Матриця містить числа від 0 до 255, кожне з яких відповідає відтінку пікселів*

2.1.2 Склад CNN

Архітектура згорткової нейронної мережі складається з таких частин як згорткові шари (convolution layers), шари підвибірки (pooling layers) і пов’язані шари (fully connected layers). Типова архітектура складається з повторень набору декількох згорткових шарів разом з об'єднуючим шаром, за якими слідує один або кілька шарів, що з'єднують. Крок, де вхідні дані перетворюються на вихідні через ці шари, називається зворотним поширенням

(Рис. 30). Операції з об'єднанням (pooling) може бути як і двомірному (2D), і у тривимірному (3D) просторі.

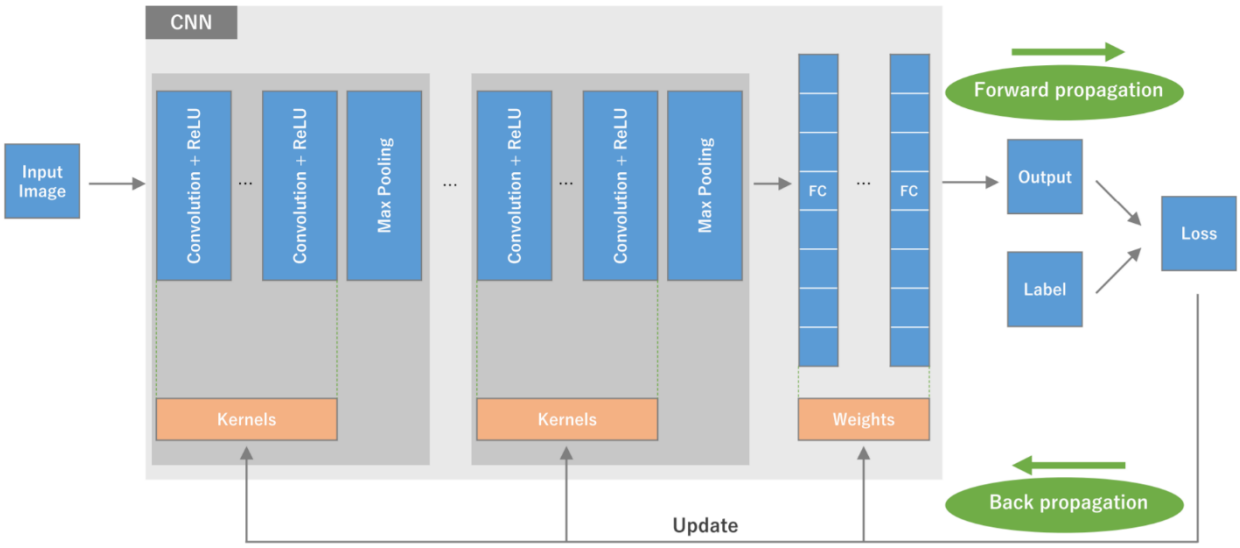

Рис. *30 Архітектура згорткової нейронної мережі (CNN) та процес навчання*

2.1.3 Згортковий шар

Згортковий шар є ключовим компонентом в архітектурі CNN. Він відповідальний за знаходження унікальних особливостей на зображенні і містить у собі комбінацію лінійних і нелінійних операцій, тобто операцій згортки та функцію активації. Згортка – це спеціалізований тип лінійної операції, який використовується для знаходження унікальних особливостей на зображенні, де невеликий масив чисел, званий ядром (kernel) застосовується до вхідних даних, званих тензором (tensor). Множення елементів з кожним елементом ядра та частиною вхідного тензора обчислюється як частина вихідного тензора, що називається картою ознак (feature map) (Рис. 31). Ця процедура повторюється багаторазово за допомогою ядер, щоб сформувати деяку кількість карт ознак, кожна з яких описує певні характеристики вхідного тензора. Окремі ядра можуть виконувати роль знаходження абсолютно різних унікальних особливостей на зображенні (Рис. 32). Існують два ключові гіперпараметри, які визначають операцію згортки: розмір та число ядер. Найчас-

тіше застосовується розмір ядра 3x3, але можуть використовуватись розміри 5x5 або 7x7. Число самих ядер визначає глибину виведення карти ознак.

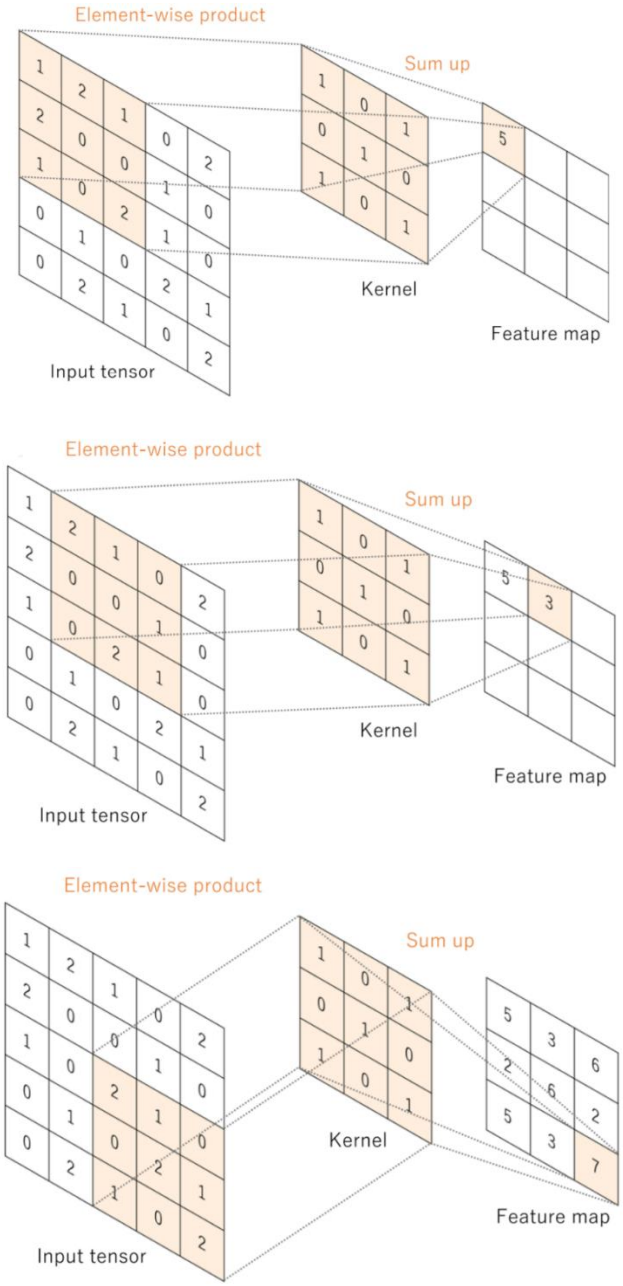

Рис. *31 Приклад операції згортки з розміром ядра 3×3.*

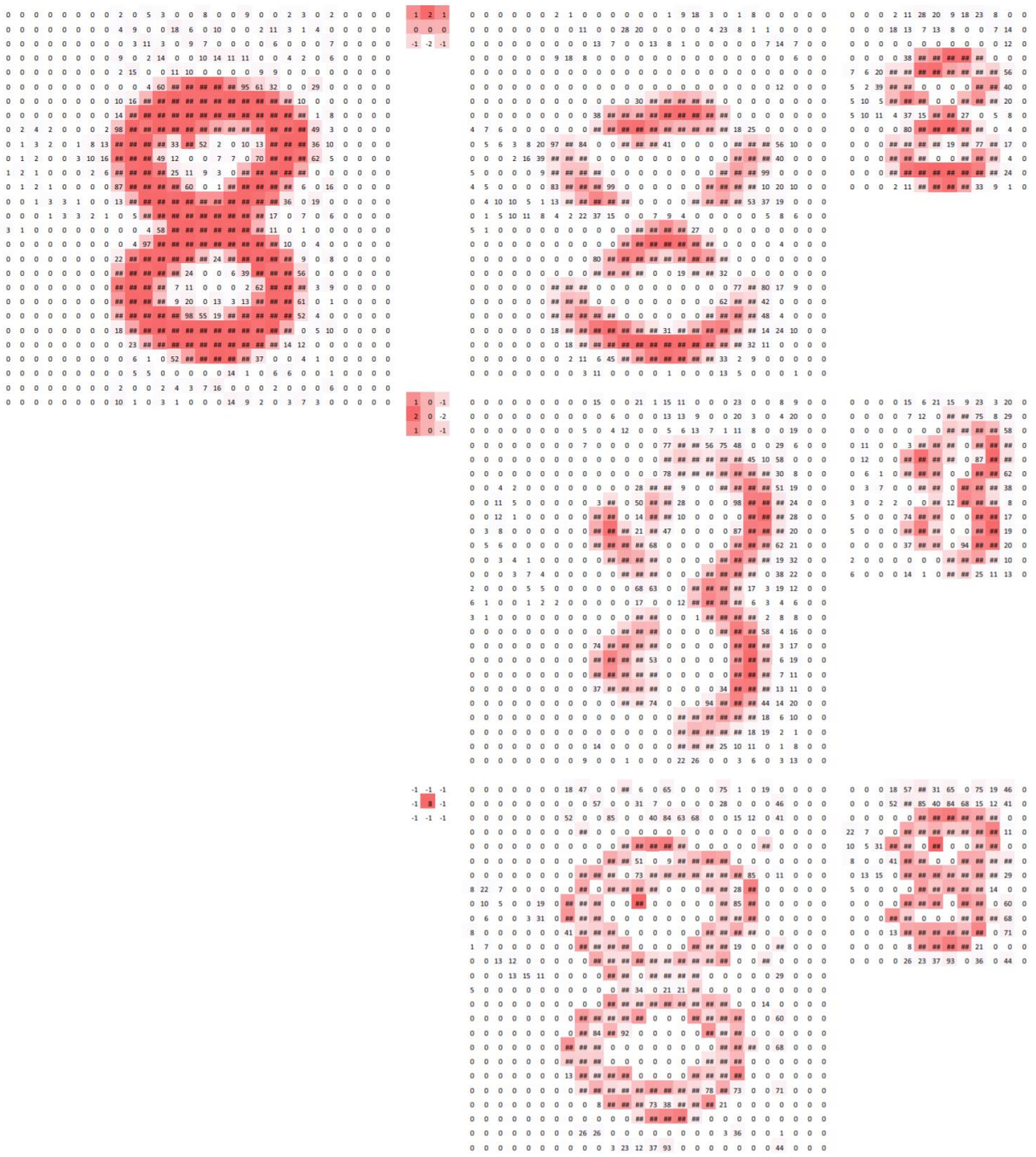

Рис. *32 Приклад вилучення ядром унікальних особливостей на зображенні в шарі згортки.*

Операція згортки, що описана вище, не дозволяє центру кожного ядра перекривати граничні елементи вхідного тензора та зменшує розмір картки ознак на виході. Заповнення (padding), зазвичай нульове – це спосіб обчислення, коли нульові рядки додаються з кожного боку сітки вхідного тензора (Рис. 33). Це дозволяє зберігати розміри сітки під час операції згортки. Сучасні архітектури CNN зазвичай використовують такий підхід,

щоб використовувати більше шарів. Без нульового заповнення кожен наступний шар ставатиме менше за розміром щодо попереднього.

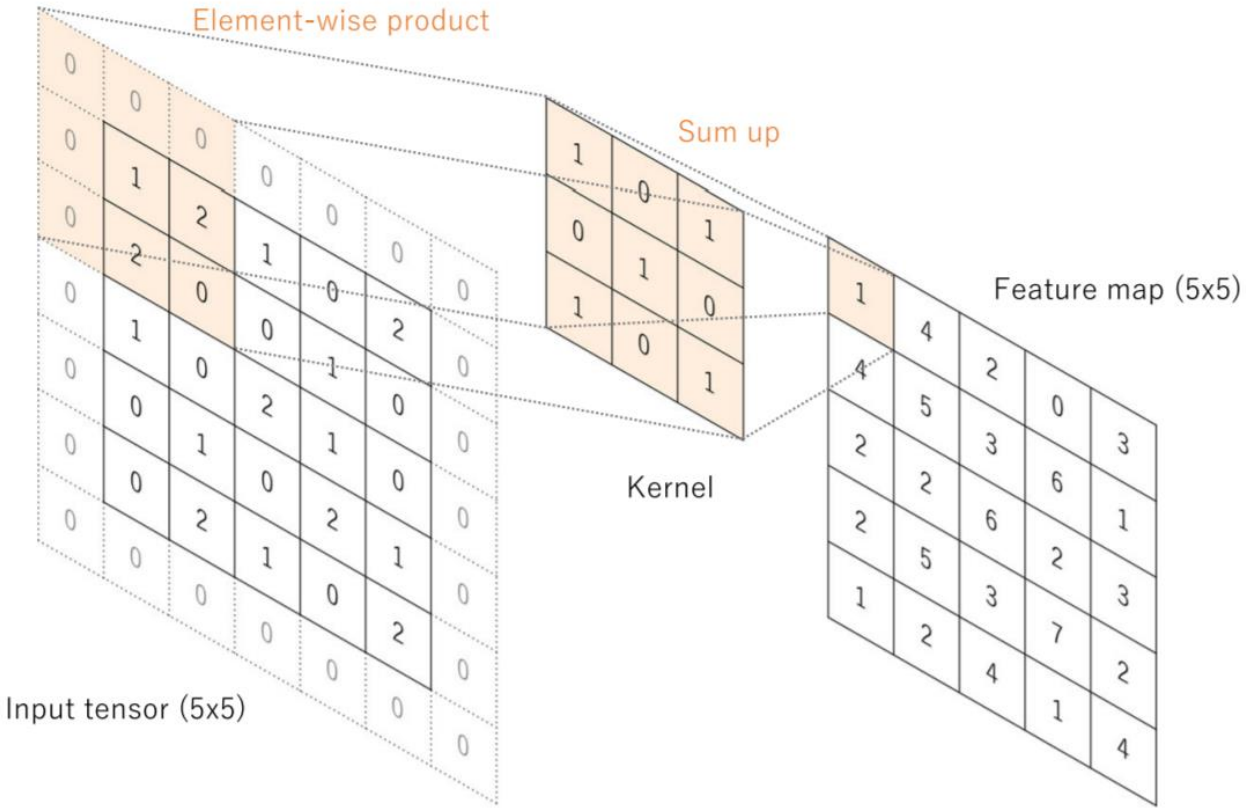

Рис 33. *Операція згортки з нульовим заповненням для збереження розмірів у плоскості.*

Дистанція між двома послідовними позиціями ядра називається кроком (stride), який також визначає операцію згортки. Зазвичай вибирають розмір кроку рівному одиниці, проте, якщо потрібно домогтися знижувальної дискретизації, потрібно встановити значення кроку більше одного. Процес навчання моделі CNN щодо згорткового шару полягає в тому, щоб ідентифікувати ядра, які найкраще працюють для цього завдання, на основі заданого набору вхідних даних. Ядра – це єдині параметри, які автоматично навчаються в процесі роботи нейронної мережі в згортковому шарі. З іншого боку, існують такі гіперпараметри як: розмір та кількість ядер, зміщення та крок, які необхідно налаштовувати вручну перед процесом навчання (Таблиця 2).

Таблиця *2*

Список параметрів та гіперпараметрів у згортковій нейронній мережі CNN

Шар	Параметри	Гіперпараметри
Convolution layer	Kernels	Kernel size, number of kernels, stride, padding, activation function
Pooling layer	None	Pooling method, filter size, stride, padding
Fully connected layer	Weights	Number of weights, activation function
Others		Model architecture, optimizer, learning rate, loss function, mini-batch size, epochs, regularization, weight initialization, dataset splitting

2.1.4 Нелінійна функція активації

Вихідні дані лінійної операції згортки потім обчислюються за допомогою нелінійної функції активації. Нелінійні функції, що згладжують, такі як сигмоїд (sigmoid) (Рис. 34), або гіперболічний тангенс (tanh) (Рис. 35), використовувалися раніше, тому що їх поведінка математично схожа з поведінкою нейронів у біології. Однією з найвідоміших нелінійних функцій активації є ReLU (Rectified Linear Unit), яка легко обчислюється за допомогою формули: $f(x) = \max(0, x)$ (Рис. 36).

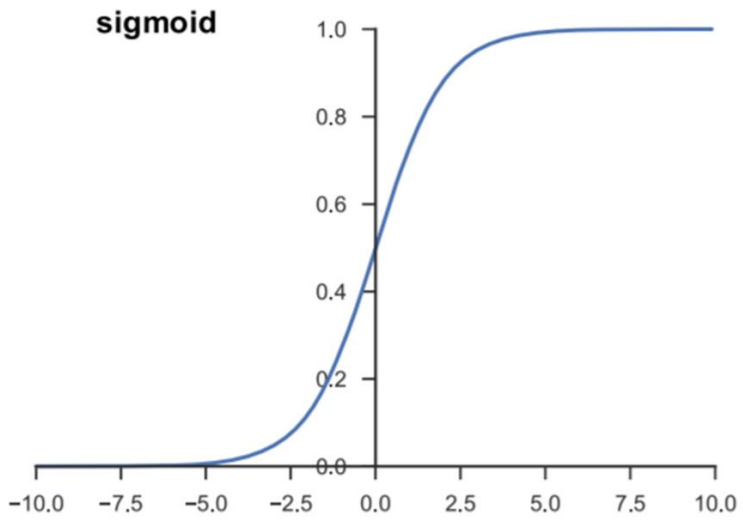

Рис. *34 Функція активації сигмоїд*

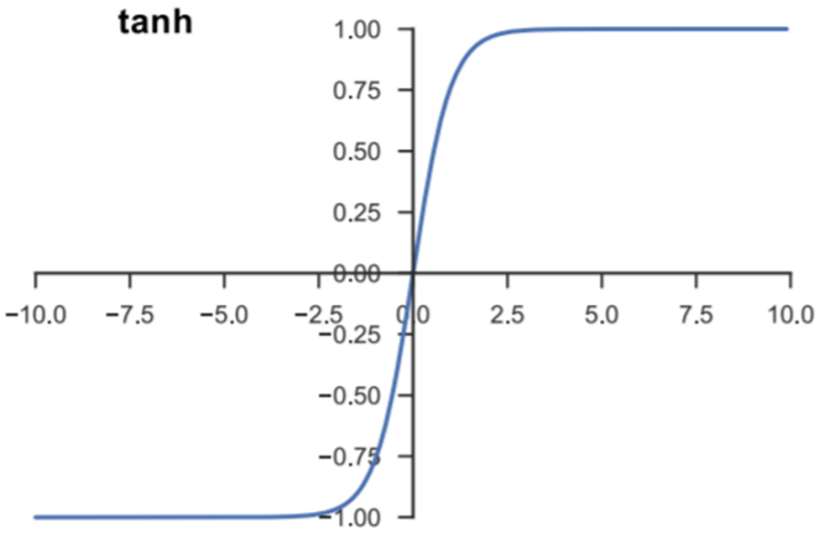

Рис. *35 Функція активації гіперболічний тангенс*

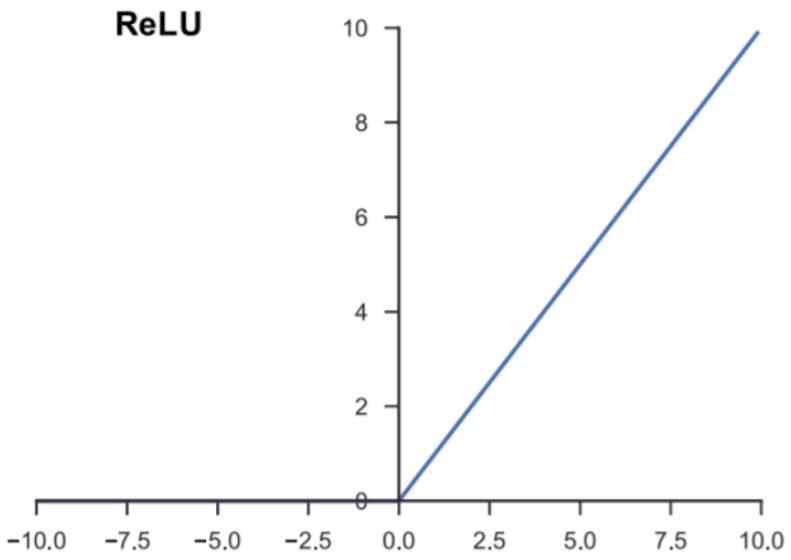

Рис. *36 Функції активації Rectified Linear Unit (ReLU)*

2.1.5 Шар підвибірки

Шар підвибірки (pooling layer) забезпечує типову операцію пониження дискретизації, яка зменшує розмірність карти ознак, тим самим знижуючи кількість параметрів для навчання.

2.1.6 Максимальне об’єднання

Найвідоміший тип операції об'єднання – це максимальне об'єднання (max pooling), який витягує набори чисел з вхідних даних карти ознак, посилаючи на вихід шару максимальне значення з кожного набору цих чисел, ігноруючи інші (Рис. 37). Насправді зазвичай використовується розмір сітки 2x2 з кроком два. Це зменшує розмір сітки вихідних даних у шарі вдвічі. Глибина карти ознак за такого підходу залишається незмінною.

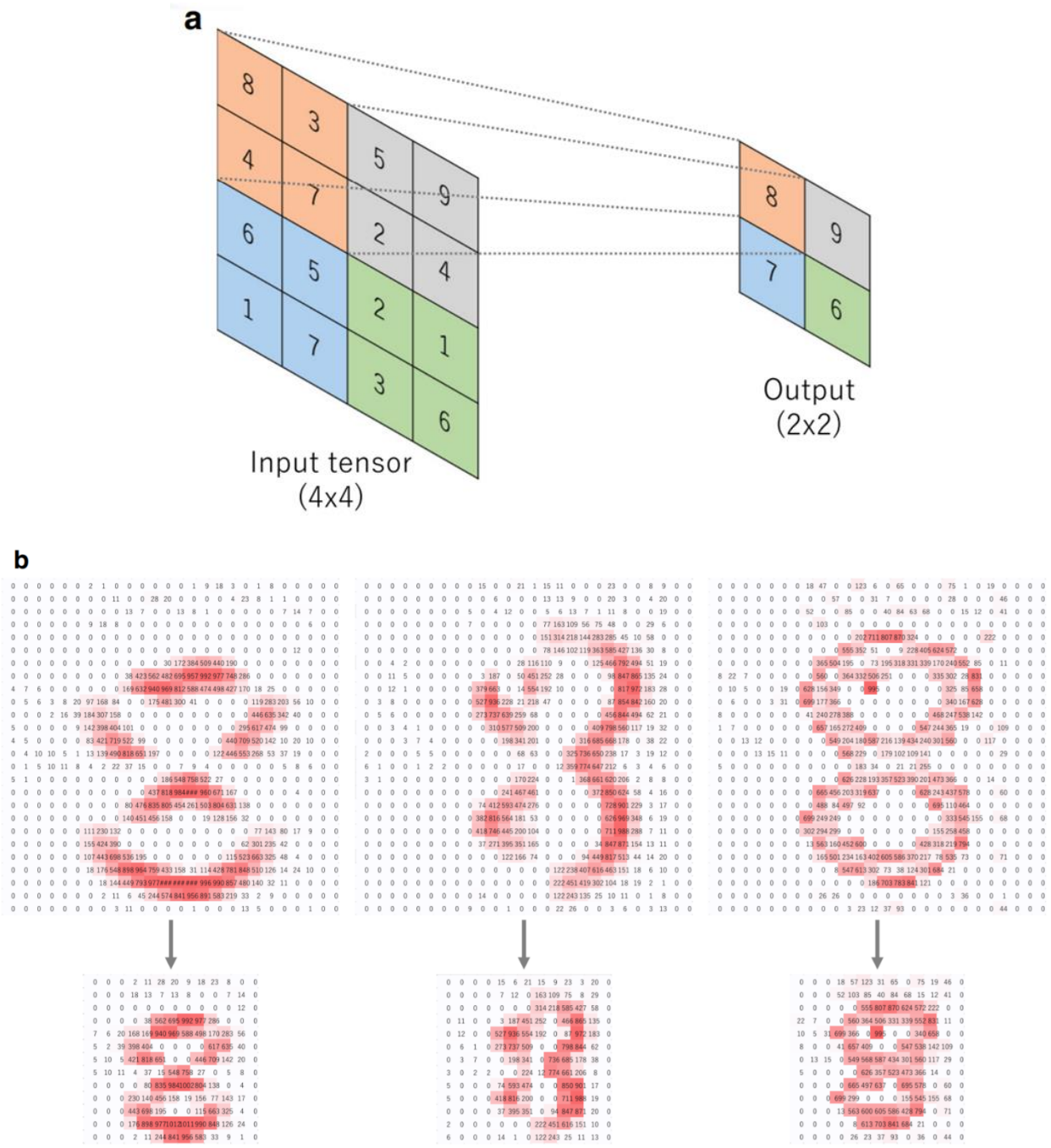

Рис. *37 Приклади максимальної операції об'єднання (max pooling) з розміром фільтра 2×2*

2.1.7 Середнє об’єднання

Ще одна операція по об'єднанню, яку варто відзначити – це глобальне середнє об'єднання (global average pooling). Ця операція виконує екстремальний тип знижувальної дискретизації, коли карта ознак з деяким вхідним розміром перетворюється на одиничний масив розміром 1x1, взявши середнє значення всіх її елементів у кожній карті ознак. При цьому глибина карти не

змінюється. Глобальне середнє об'єднання зазвичай застосовується лише один раз перед шаром, що з'єднує. Перевагами даного підходу є зменшення кількості параметрів, ща навчаються, а також можливість нейронної мережі приймати на вхід шару довільні розміри тензора.

2.1.8 Пов’язаний шар

Вихідні карти ознак в останньому згортковому або об'єднуючому шарі зазвичай згладжуються, тобто перетворюються на одномірний масив чисел або вектор. Далі йде процес з'єднання з одним або декількома повнозв'язними шарами, так само відомі як щільні шари (dense layers), де кожен вхід шару пов'язаний з виходом за допомогою ваг, що навчаються (learnable weight). Після того, як унікальні особливості на зображенні були вилучені згортковими шарами і перетворені шляхом об'єднань, вихід останнього шару зіставляється з ймовірностями кожного класу, для якого відбувається навчання нейронної мережі. Зазвичай кількість вихідних елементів на останньому шарі дорівнює кількості класів задачі і завершується обчисленням нелінійною функцією, такою як ReLU (Rectified Linear Unit).

2.1.9 Останній шар із функцією активації

Функція активації, що застосовується до останнього шару, зазвичай, відрізняється від інших. Для кожного завдання необхідно підібрати відповідну функцію активації. Функція активації, що застосовується до завдання мультикласової класифікації, являє собою функцію softmax, яка нормалізує вихідні значення останнього шару до ймовірностей певних класів, де кожне значення варіюється в діапазоні від 0 до 1 і всі значення шару в сумі рівні одиниці. Типовий вибір функції останнього шару для різних завдань представлений у таблиці (Таблиця 3).

Таблиця *3*

Список функцій активації останнього шару для різних завдань

Задача	Останній шар активації
Binary classification	Sigmoid
Mutliclass single-class classification	Softmax
Mutliclass mutliclass classification	Sigmoid
Regression to continuous values	Identity

2.1.10 Навчання нейронної мережі

Навчання нейронної мережі – це процес знаходження ядер у згорткових шарах і терезів у повністю з'єднаних шарах, що мінімізує різницю між виведенням ймовірностей та реальними даними класів конкретного набору тренувальних даних. Алгоритм зворотного поширення помилки (backpropagation) - це метод, який зазвичай використовується в навчанні нейронних мереж, де функція втрат та алгоритм оптимізації градієнтного спуску відіграють важливу роль. Продуктивність моделі при певних ядрах і вагах обчислюється за допомогою функції втрат за допомогою алгоритму прямого поширення (gradient descent) у наборі даних. Параметри, що навчаються, такі як ядра і ваги, оновлюються відповідно до значення помилки, отриманого після роботи алгоритму оптимізації, в більшості випадків це алгоритм зворотного поширення помилки.

2.1.11 Функція втрат

Функція втрат (loss function), також відома як функція вартості (cost function), визначає різницю між виведенням ймовірностей класів останнього шару і мітками істинності класів вхідного набору даних. Зазвичай функцією втрат багатокласової класифікації є перехресна ентропія (cross entropy), тоді як функція середньоквадратичної помилки (mean squared error) застосовується для регресії безперервних значень. Тип функції втрат є одним з гіперпараметрів і повинен бути визначений виходячи з поставленого завдання.

2.1.12 Градієнтний спуск

Градієнтний спуск зазвичай використовується як алгоритм оптимізації, який у циклі оновлює навчальні параметри, тобто ядра і ваги в нейронній мережі, щоб мінімізувати помилку. Градієнт функції втрат дає напрямок, в якому функція має найбільшу швидкість збільшення, і кожен параметр, що навчається, оновлюється в негативному напрямку градіэнта з довільним розміром кроку, який є гіперпараметром. Такий гіперпараметр для кроку має на увазі швидкістю навчання нейронної мережі (Рис. 38). З математичної точки зору, градієнт представляє собою приватну похідну втрат по кожному навчальному параметру. Оновлення параметрів можна представити у вигляді формули:

$$w := w - a * \frac{dL}{dw},$$

де w – це навчальний параметр, альфа – параметр швидкості навчання нейронної мережі, L – функція втрат. На практиці швидкість навчання є одним з найбільш значущих гіперпараметрів, які необхідно визначити перед початком навчання. У поточних реаліях, при великих обсягах вхідних даних і при недостатній кількості оперативної пам'яті проводяться розбиття набору даних на міні-набори (mini-batches), у кожному з яких відбувається оновлення параметрів. Такий метод називається градієнтним спуском для міні-наборів (mini-batch gradient descent), також він може називатися стохастичним градієнтним спуском (stochastic gradient descent), де розмір міні-набору також є гіперпараметром. Крім того, були запропоновані та активно використовуються багато вдосконалень алгоритму градієнтного спуску, такі як стохастичний градієнтний спуск з імпульсом (SGD with momentum), RMSprop і Adam [27].

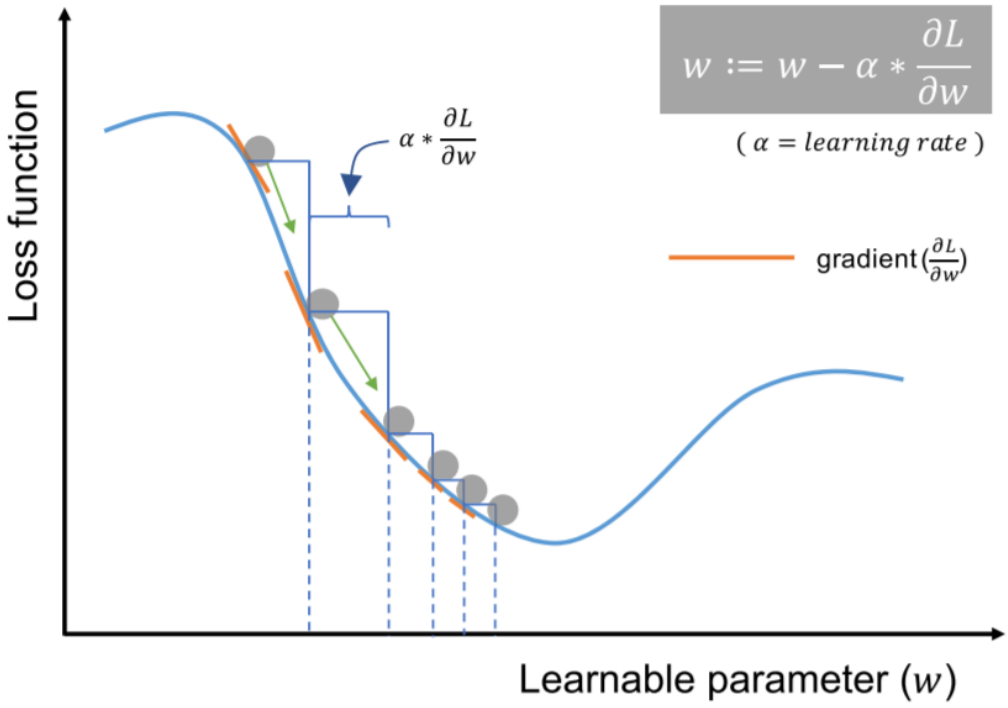

Рис. *38 Градієнтний спуск – це алгоритм оптимізації, який ітеративно оновлює параметри навчання, щоб мінімізувати втрати*

2.1.13 Набір даних та їх класифікація

Набір даних та їх класифікація – найбільш значущі компоненти в дослідженнях із застосуванням глибокого навчання та інших методів машинного навчання. Ретельний збір даних та достовірних міток класифікації для навчання та тестування моделі даних є обов'язковими для успішного проекту глибокого навчання. Однак сам процес отримання високоякісних даних з мітками класів може бути дуже дорогим, трудомістким і витратним за часом. Дані зазвичай поділяються на три типи: для навчання (training), перевірочний (validation) та тестовий (test) (Рис. 39). Існує ще кілька додаткових типів, як, наприклад, перехресна перевірка (cross validation). Набір даних для навчання використовується для навчання нейронної мережі, де значення помилки обчислюються за допомогою прямого поширення, а параметри, що навчаються, оновлюються за допомогою зворотного поширення. Набір даних для перевірки використовується для оцінки моделі в процесі навчання та точного настроювання гіперпараметрів. В ідеалі тестовий набір даних використовується

лише один раз на самому кінці процесу навчання нейронної мережі, щоб оцінити продуктивність остаточних результатів. Необхідні окремі набори для перевірки і тестування, оскільки навчання моделі даних завжди включає точне налаштування її гіперпараметрів і виконання вибору самої моделі. Оскільки процес навчання відбувається на основі продуктивності набору даних для тестів, деяка інформація з цього набору все-таки просочується в навчальні параметри моделі, тобто відбувається переоснащення набору даних для тестів, навіть незважаючи на те, що сама модель ніколи не навчалася безпосередньо з цих даних. Тому гарантується, що модель з гранично точно налаштованими гіперпараметрами при навчанні в тестовому наборі даних буде показувати хороші результати в тому ж наборі даних. Отже, обов'язково необхідний набір схожих даних для тестування, який не брав участь у самому процесі навчання.

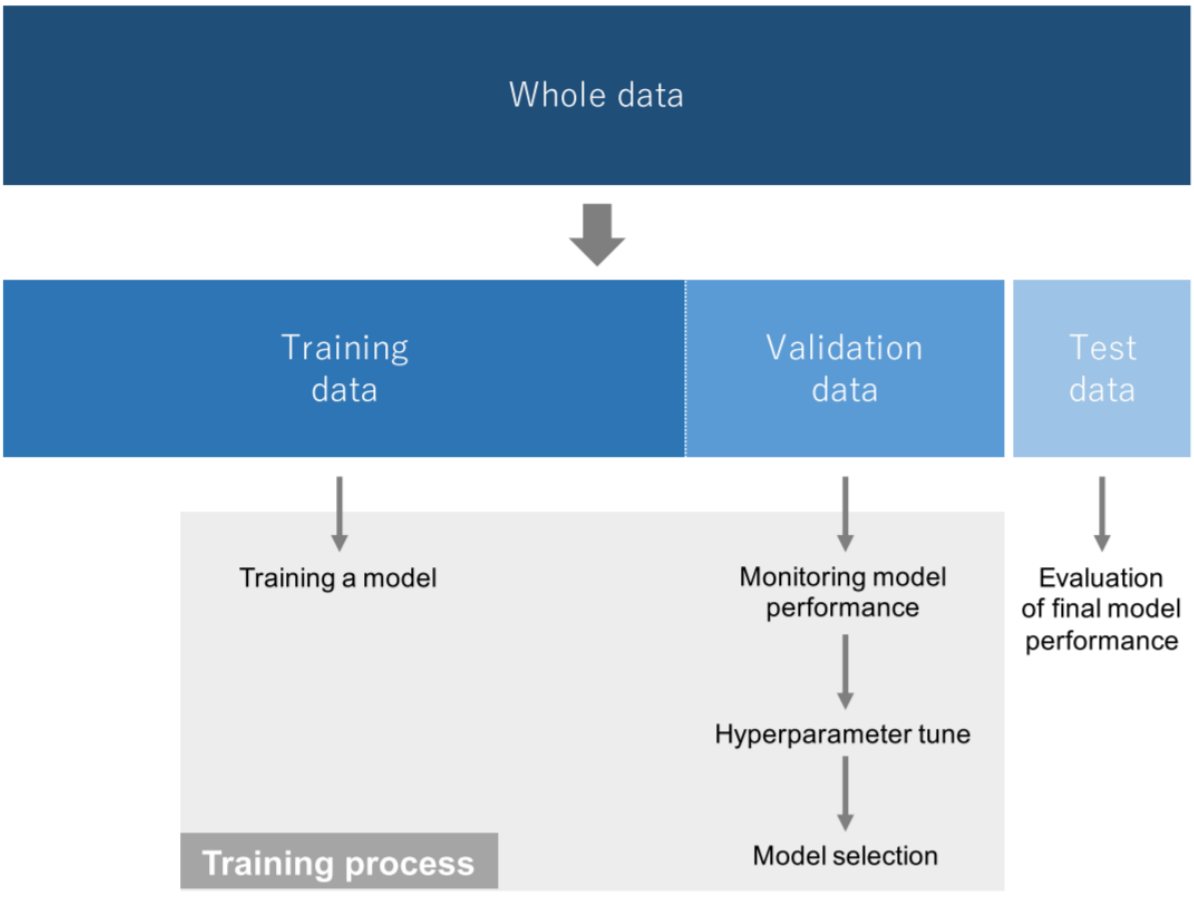

Рис. *39 Дані поділяються на три набори: для навчання, перевірки та тесту*

2.1.14 Перенавчання

Перенавчання відбувається тоді, коли модель вивчає статистичні закономірності, характерні для набору даних для навчання, тобто в кінцевому підсумку запам'ятовує тільки певні особливості, притаманні конкретному набору даних, і, отже, працює гірше з іншими, наступними наборами даних. Це одна з проблем машинного навчання, оскільки перенавчена модель нездатна задовільно застосовуватися на нових наборах даних. У цьому сенсі набір тестових даних грає ключову роль правильної оцінці продуктивності моделей машинного навчання. Перевірка для розпізнавання перенавчених моделей даних полягає у відстеженні помилок і точності на даних для навчання та тестування (Рис. 40).

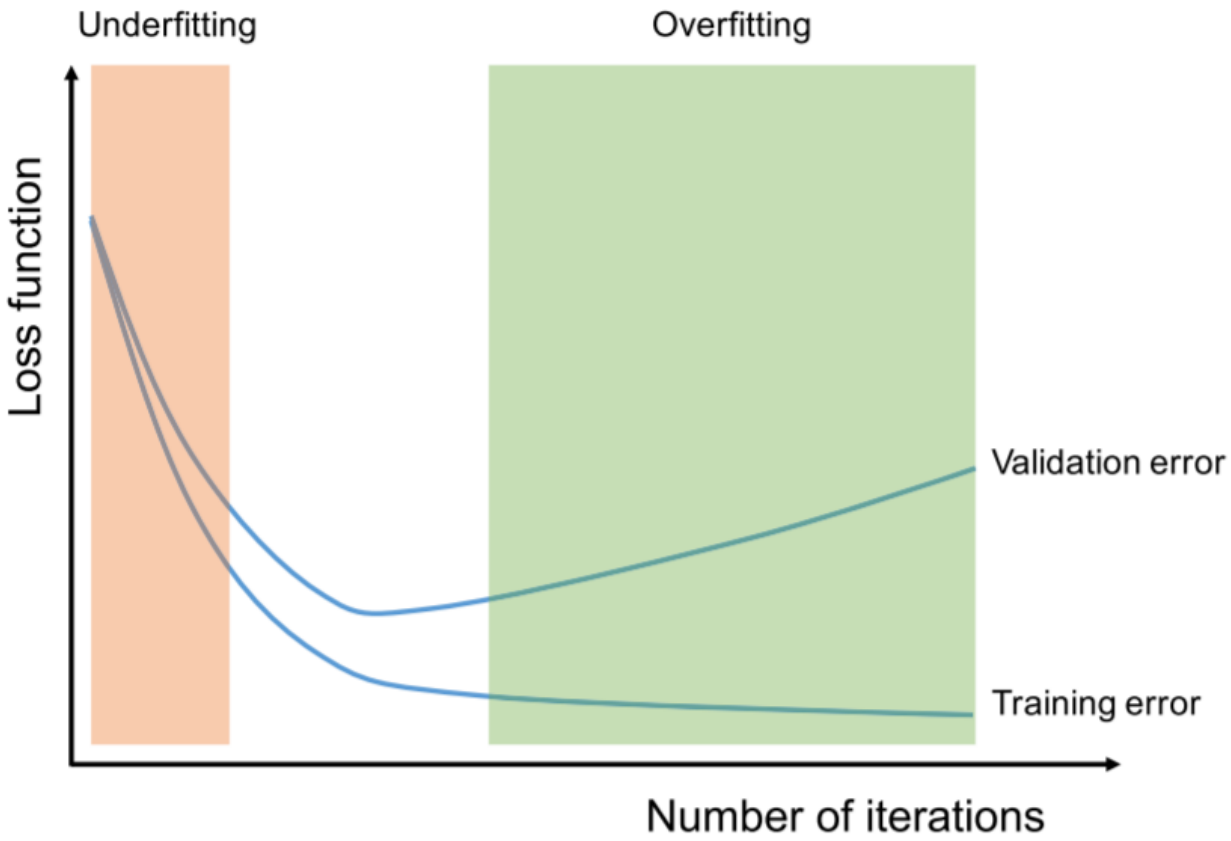

Рис. *40 Розпізнавання стадії навчання моделі нейронної мережі*

Якщо модель добре працює на навчальному наборі даних у порівнянні з тестовим набором, значить навчання було зайвим. Існує кілька способів мі-

німізації процесу перенавчання (Таблиця 4). Найкраще рішення для зменшення ефекту перенавчання – отримати більше даних. Модель, навчена більшому наборі даних, зазвичай краще узагальнює. Інші рішення проблеми перенавчання – це регуляризація (regularization) з видаленням (dropout) або зменшення ваг (weight decay), пакетна нормалізація (batch normalization), розширення даних (data augmentation), а також загалом зниження архітектурної складності моделі. Видалення – це нещодавно представлений спосіб регуляризації, у якому випадково обрані ваги під час навчання встановлюються рівними нулю. За такого підходу модель стає менш чутливою до певних параметрів нейронної мережі. Зниження ваг, також відоме як L2-регуляризація, зменшує ефект перенавчання за рахунок зменшення великих значень ваг, так, що на виході всі ваги приймають тільки невеликі значення. Ідея пакетної нормалізація полягає в тому, що замість того, щоб просто нормалізувати вхідні дані в мережі, відбувається нормалізація вхідних даних по шарах всередині мережі [28]. Під час навчання нормалізуються виходи попереднього шару для кожного пакета, тобто застосовується перетворення, яке підтримує середнє значення активації, близьке до 0, і стандартне відхилення активації, близьке до 1. Розширення даних також ефективно для зменшення ефекту перенавчання, яке є процес зміни вхідного набору даних за допомогою випадкових перетворень, таких як поворот, зсув, обрізання, стирання частини зображення. За такої модифікації дані стають вже новими для нейронної мережі і це позитивно позначається під час циклів навчання [29].

Таблиця *4*

Список поширених методів зменшення перенавчання

Більше даних
Розширення даних
Реруляризація
Пакетна нормалізація
Зниження складності архітектури

2.2 Фреймворк Darknet

Darknet – це фреймворк нейронної мережі з відкритим вихідним кодом, написаний на C та CUDA. Він швидкий, простий в установці та підтримує обчислення на CPU та GPU. Користувачі можуть знайти вихідний код GitHub. Darknet встановлюється тільки з двома необов'язковими залежностями: OpenCV, якщо користувач хоче ширший спектр типів зображень, що підтримуються, або CUDA, якщо їм потрібні обчислення на GPU. Ні те, ні інше не обов'язкове, але користувачі можуть почати з установки базової системи, яка була протестована на комп'ютерах Windows, Linux і Mac.

Фреймворк включає You Only Look Once (YOLO), сучасну систему виявлення об'єктів в реальному часі. На Titan X він обробляє зображення зі швидкістю 40-90 FPS і має MAP на VOC 2007 78,6% та MAP 44,0% на COCO tets-dev. Користувачі можуть використовувати Darknet для класифікації зображень завдання ImageNet класу 1000. Darknet відображає інформацію у міру завантаження файлу конфігурації та зважує, потім класифікує зображення та роздруковує топ-10 класів для зображення.

2.3 Кінематична модель мобільної платформи з диференціальним приводом

Багато мобільних роботів використовують приводний механізм, відомий як диференціальний привід. Він складається з двох провідних коліс, встановлених на спільній осі, і кожне колесо може незалежно рухатися вперед або назад. Можна змінювати швидкість кожного колеса, щоб робот міг крутитися. Він повинен обертатися навколо точки, що лежить вздовж загальної осі лівого та правого коліс. Позиція, навколо якої обертається робот, відома як ICC – миттєвий центр кривизни (Рис. 41).

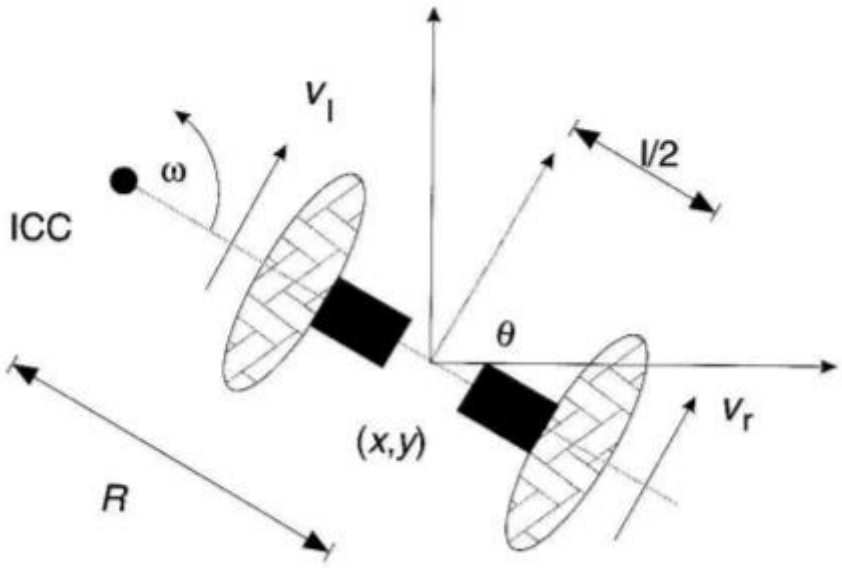

Рис. *41 Кінематика диференціального приводу*

Змінюючи швидкості двох коліс, можна змінювати траєкторії, якими може їхати робот. Оскільки швидкість обертання навколо ICC повинна бути однаковою для обох коліс, можна записати наступні рівняння:

$$\omega\,(R + l/2) = V_r\,,$$
$$\omega\,(R - l/2) = V_l\,,$$

де l — відстань між центрами двох коліс, V_r, V_l – швидкості правого і лівого коліс, а R – відстань зі знаком від ICC до середньої точки між колесами. У будь-який момент часу можна вирішити для R і ω:

$$R = \frac{l}{2}\frac{V_l + V_r}{V_r - V_l};\ \omega = \frac{V_r\ -\ V_l}{l}. \qquad (1)$$

Є три цікаві випадки з такими приводами:

1. Якщо $V_l = V_r$, то маємо поступальний поступальний рух прямою. R стає нескінченним, і обертання фактично немає – ω дорівнює нулю.

2. Якщо $V_l = V_r$, то R = 0, тоді відбувається обертання навколо середини осі колеса на місці.

3. Якщо $V_l = 0$, то маємо обертання навколо лівого колеса. І тут R = – l/2. Те ж саме, якщо $V_r = 0$, то R = l/2.

Важливо те, що робот з диференціальним приводом неспроможен рухатися у бік осі — це його особливість. Транспортні засоби з диференціальним приводом дуже чутливі до невеликих змін швидкості кожного колеса [30]. Невеликі помилки у відносних швидкостях між колесами можуть вплинути на траєкторію робота.

2.3.1 Пряма кінематика для роботів із диференціальним приводом

Припустимо, що робот знаходиться в певній позиції (x, y) і рухається у напрямку, що становить кут θ з віссю x. Вважається, що робот центрований у точці посередині осі колеса. Маніпулюючи параметрами управління Vl, Vr, можна змусити робота переміщатися різні положення і орієнтації. Знаючи швидкості Vl, Vr та використовуючи рівняння (1), можна знайти значення ICC:

$$ICC = [x - Rsin(\theta), y + Rcos(\theta)],$$

та в момент часу t + dt позиція робота буде:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega dt) & -sin(\omega dt) & 0 \\ sin(\omega dt) & cos(\omega dt) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega dt \end{bmatrix}.$$

Це рівняння просто описує рух робота, що обертається з відстанню R навколо свого ICC з кутовою швидкістю ω.

Інший спосіб зрозуміти це – те, що рух робота еквівалентний (Рис. 42):

1) переведення ICC на початок системи координат;

2) поворот навколо початку координат на кутову величину;

3) переведення назад до ICC.

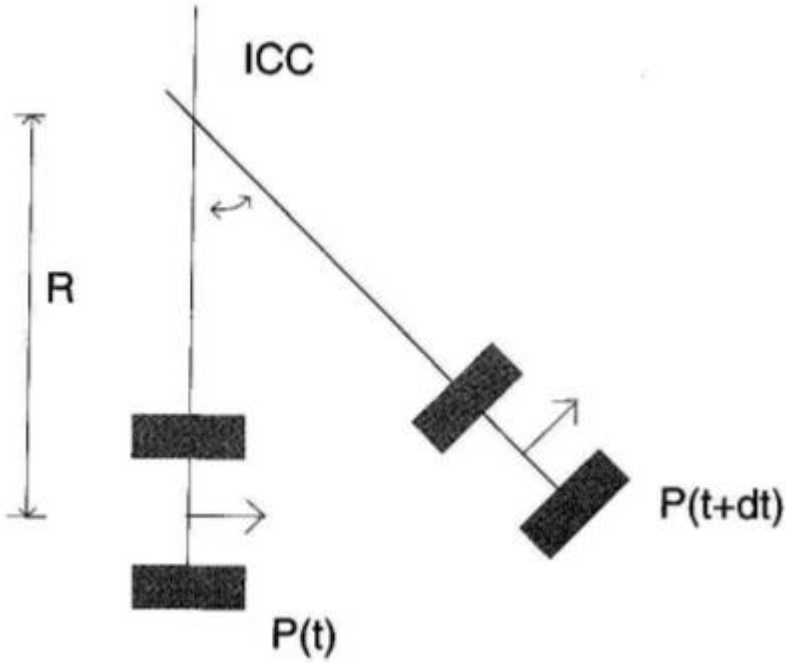

Рис. *42 Пряма кінематика для диференціального робота*

2.3.2 Зворотня кінематика мобільного робота

Ми можемо описати положення робота, здатного рухатися у певному напрямку Θt із заданою швидкістю V(t):

$$x(t) = \int_0^t V(t)cos[\theta(t)]dt,$$

$$y(t) = \int_0^t V(t)sin[\theta(t)]dt,$$

$$\Theta(t) = \int_0^t \omega(t)dt.$$

Для особливого випадку робота з диференціальним приводом, такого як черепаха, рівняння стають:

$$x(t) = \frac{1}{2}\int_0^t [v_r(t) + v_l(t)]cos[\theta(t)]dt,$$

$$y(t) = \frac{1}{2}\int_0^t [v_r(t) + v_l(t)]sin[\theta(t)]dt,$$

$$\Theta(t) = \frac{1}{l}\int_0^t [v_r(t) - v_l(t)]dt.$$

Пов'язане з цим питання: як можна керувати роботом, щоб досягти заданої конфігурації (x, y, θ) – це головна проблема зворотної кінематики. На жаль, робот із диференціальним приводом накладає так звані неголономні обмеження на визначення свого становища. Наприклад, робот не може рухатися вбік вздовж своєї осі. Аналогічне неголономне обмеження – це автомобіль, який може обертати тільки передні колеса. Він не може рухатися вбік, таким чином паралельне паркування автомобіля вимагає складнішого набору маневрів кермового управління. Таким чином, не можна просто вказати довільну позу робота (x, y, θ) та знайти швидкості, що приведуть нас до неї. Для окремих випадків Vl = Vr = V (робот рухається по прямій) рівняння руху виглядає так:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + vcos(\theta)dt \\ y + vsin(\theta)dt \\ \theta \end{bmatrix}.$$

Якщо Vr = −Vl = V, то робот обертається на місці, і рівняння приймають такий вид:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2vdt/l \end{bmatrix}.$$

Це визначає траєкторію руху робота з диференціальним приводом по прямій лінії з подальшим поворотом на місці, а потім знову рух прямо.

2.3.3 Зіставлення кутової швидкості коліс з лінійною швидкістю

Використовувані вище швидкості лівого та правого коліс Vl, Vr є лінійними швидкостями. Фактично відбувається керування колесами та можна задати кутову швидкість Vwheel для колеса, вказану в радіанах за секунду. Враховуючи Vwheel, потрібно з'ясувати, якою є результуюча лінійна швидкість руху цього колеса.

Можна використовувати як приклад робота Khepera (Рис. 43), який є невеликим роботом, схожим на Turtlebot, що використовує диференціальний привід.

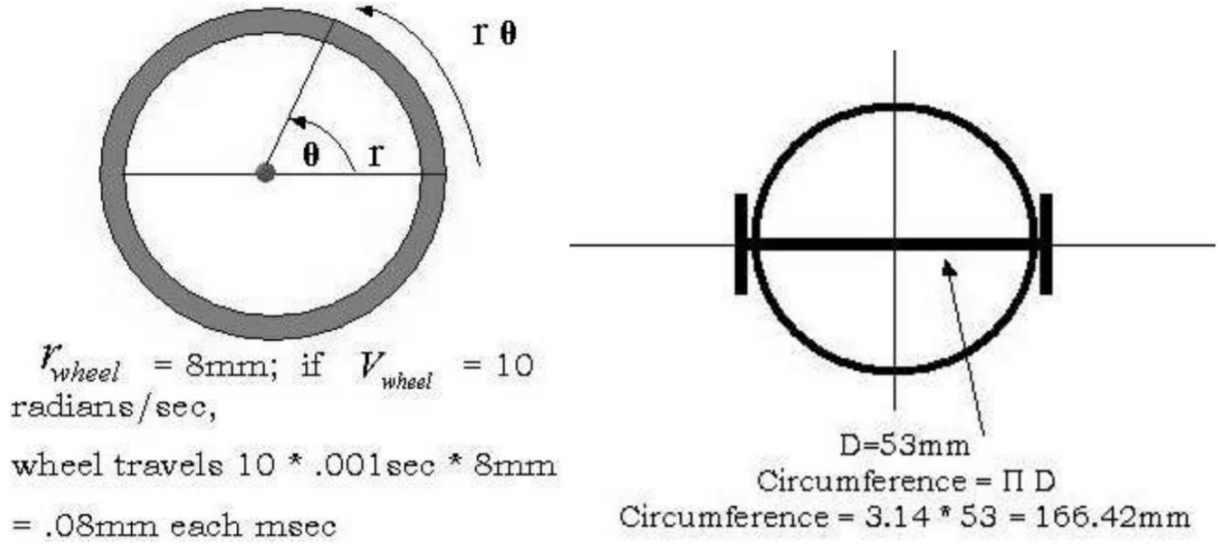

Рис. *43 Параметри робота Khepera та його колеса*

Визначаємо наступні терміни: rwheel – радіус колеса, Drobot – довжина осі ведучого колеса диференціала, Vwheel – величина швидкості колеса, що вимірюється в радіанах за секунду. Припустимо, що колеса обертаються у протилежному напрямку з однаковою швидкістю (робот обертається навколо). Щоб база робота оберталася на φ градусів, потрібно знайти рівняння для кількості часу t, яке необхідно, щоб запустити колісний двигун зі швидкістю Vwheel та повернути робот на кут φ градусів. Колесо повертається на лінійну відстань rwheel * θ по своїй дузі, де θ – це просто Vwheel * t. Колесо пройде відстань, що дорівнює rθ, за своєю дугою. Якщо припустити, що швидкість

колеса Vwheel = 10 радіан/сек, то колесо пройде 10 * 8 = 80 мм за 1 секунду, що також еквівалентно 0,08 мм за 1 мс. Через час t колесо обернеться:

$$Dist_{wheel} = r_{wheel} * V_{wheel} * t.$$

Щоб визначити час повороту робота на заданий кут на місці, треба выдзначити, що все коло C робота, коли він повертається на 360°, дорівнює π Drobot. Можна повернути кут φ за час t, використовуючи рівняння:

$$\frac{Dist_{wheel}}{C} = \frac{\varphi}{2\pi},$$

$$\frac{V_{wheel} * r_{wheel} * t}{C} = \frac{\varphi}{2\pi},$$

$$t = \frac{\varphi C}{2\pi * V_{wheel} * r_{wheel} * t}.$$

Приклад: припустимо, треба повернути робота Хепера на 90°. Для Khepera, rwheel становить 8 мм, Drobot – 53 мм, і можна встановити швидкість колеса на 10 радіан / сек:

Отже, щоб повернути на 90◦ (π/2 радіани):

$$T = \frac{\varphi C}{2\pi * V_{wheel} * r_{wheel} * t} = \frac{\frac{\pi}{2}\ 166.42}{2\pi\ 8\ *\ 10} = 0.52 \text{ сек.}$$

Таким чином, контрольний параметр t за заданої швидкості може бути знайдений.

2.4 Кубічна сплайн-інтерполяція

Кубічна сплайн-інтерполяція – це математичний метод, який зазвичай використовується для побудови нових точок у межах набору відомих точок.

Ці нові точки є значеннями функції інтерполяційної функції (званої сплайном), яка сама складається з кількох кубічних поліномів. З математичної точки зору – це процес побудови сплайну $f: [x_1, x_{n+1}] \longrightarrow \mathbb{R}$, який складається з n поліномів третього ступеня в межах від f_1 до f_n. Сплайн – це функція, що визначається частковими поліномами. На відміну від регресії, функція інтерполяції проходить по всіх заздалегідь заданих точках набору даних. Результуюча функція має таку структуру:

$$f(x) = \begin{cases} a_1x^3 + b_1x^2 + c_1x + d_1, & \text{якщо } x \in [x_1, x_2] \\ a_2x^3 + b_2x^2 + c_2x + d_2, & \text{якщо } x \in (x_2, x_3] \\ \dots & \\ a_nx^3 + b_nx^2 + c_nx + d_n, & \text{якщо } x \in (x_n, x_{n+1}]. \end{cases}$$

Потрібно звернути увагу, що всі поліноми дійсні тільки в межах інтервалу, тобто вони становлять функцію інтерполяції. У той час як екстраполяція передбачає поведінку кривої за межами діапазону даних, інтерполяція працює тільки в межах даних $[x_n, x_{n+1}]$. При правильно вибраних коефіцієнтах a_i, b_i, c_i та d_i, і для багаточленів, результуюча функція плавно проходить по точках. Для визначення коефіцієнтів сформулюється кілька рівнянь, які разом становлять однозначно розв'язувану систему рівнянь.

2.4.1 Граничні умови

Щоб мати можливість вирішити систему рівнянь, потрібно більше вхідних даних. Можуть використовуватися довільні обмеження, такі як зазначення, наприклад, третьої похідної четвертій точці дорівнює нулю. Однак зазвичай використовується вибір граничної умови, що складається з кількох рівнянь. Чотири умови природний сплайн (natural spline), сплайн без вузлів (not-a-knot spline), періодичний сплайн (periodic spline) та квадратичний сплайн (quadratic spline).

2.4.2 Природний сплайн

Природний сплайн визначається як зазначення другої похідної першого та останнього полінома рівної нулю в граничних точках інтерполяційної функції:

$$6a_1x_1 + 2b_1 = 0,$$
$$6a_nx_{n+1} + 2b_n = 0.$$

Візуальна інтерпретація полягає в тому, що зміна крутості функції наближається до нуля у першій та останній точці (Рис. 44).

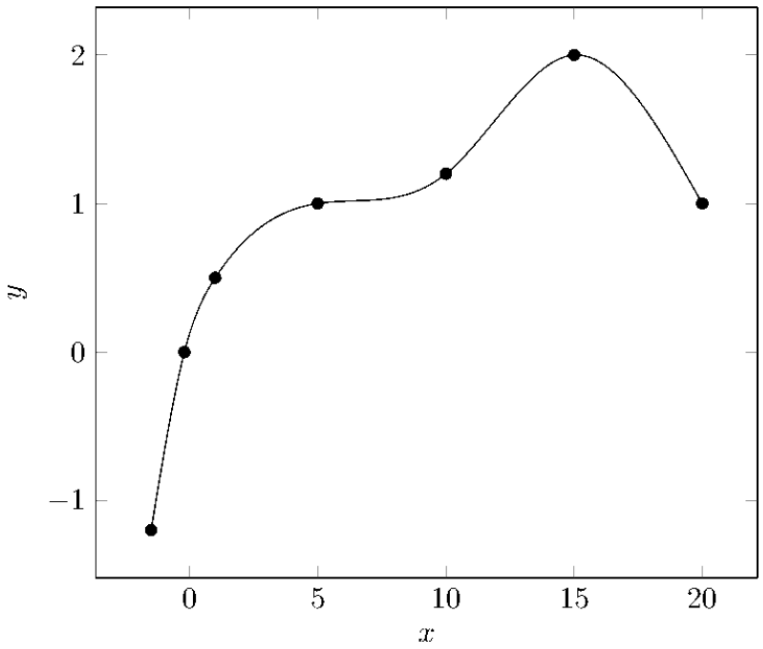

Рис. *44 Візуалізація природного сплайну*

2.4.3 Сплайн без вузлів

Для граничної умови даного виду сплайну треті похідні перших двох багаточленів дорівнюють точці дотику x_2 один з одним. Те ж саме відноситься до двох останніх багаточленів, які стосуються точки x_n, де:

$$\frac{d^3}{dx^3}\mathrm{f}_1(x) = \frac{d^3}{dx^3}\mathrm{f}_2(x), \text{при } x = x_2,$$
$$\frac{d^3}{dx^3}\mathrm{f}_{n-1}(x) = \frac{d^3}{dx^3}\mathrm{f}_n(x), \text{при } x = x_n.$$

Після розрахунку похідних це можна розбити перетворити на:

$$a_1 = a_2,$$
$$a_{n-1} = a_n.$$

що робить коефіцієнти альфа рівними один одному. Отриманий графік можна побачити на (Рис. 45).

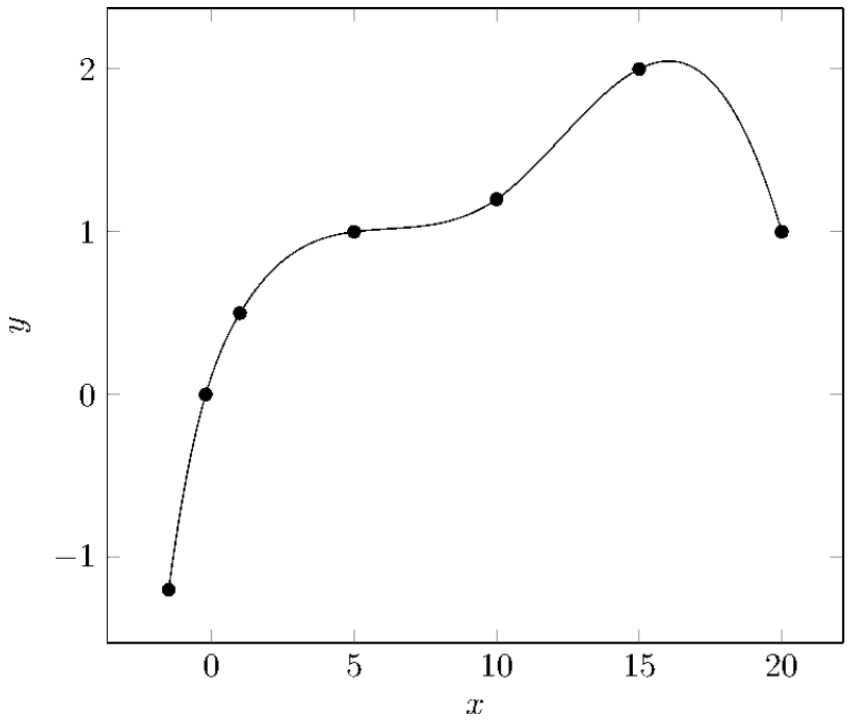

Рис. *45 Візуалізація сплайну без вузлів*

2.4.4 Періодичний сплайн

Для періодичної інтерполяції перша та друга похідна останнього багаточлена встановлюються рівними першої та другої похідної першого багаточлена. Графічно це означає, що це мозаїчна функція. Це може бути особливо корисно, якщо $\mathrm{f}(x_1) = \mathrm{f}(x_{n+1})$, але працює так само добре із функцією, де

значення у першої та останньої точки різняться. На (Рис. 46) показано, що можна скопіювати f_1 та прикріпити до останньої точки, а останній багаточлен f_n – до першої точки, не сильно змінюючи вигляд функції:

$$3a_1x_1^2 + 2b_1x_1 + c_1 = 3a_nx_{n+1}^2 + 2b_nx_{n+1} + c_n,$$
$$6a_1x_1 + 2b_1 = 6a_nx_{n+1} + 2b_n.$$

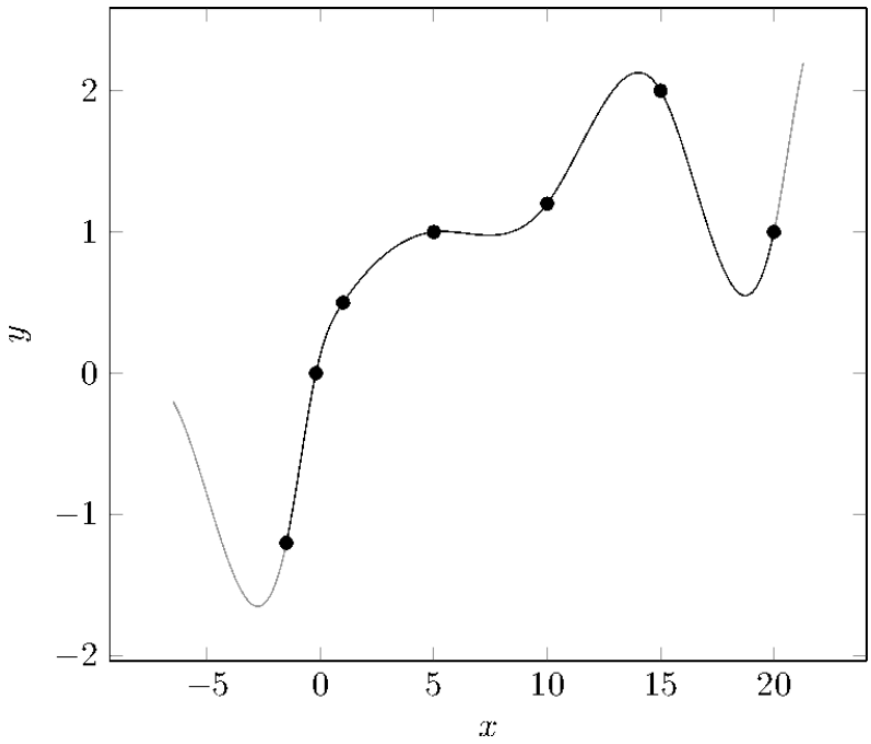

Рис. *46 Візуалізація періодичного сплайну*

2.4.5 Квадратичний сплайн

Квадратична гранична умова визначає квадратичність першого і останнього багаточленів, що робить цей метод найпростішим. Дві параболи виділені червоним (Рис. 47). Говорячи математично, перший коефіцієнт обох багаточленів дорівнює нулю: $a_1 = a_n = 0$.

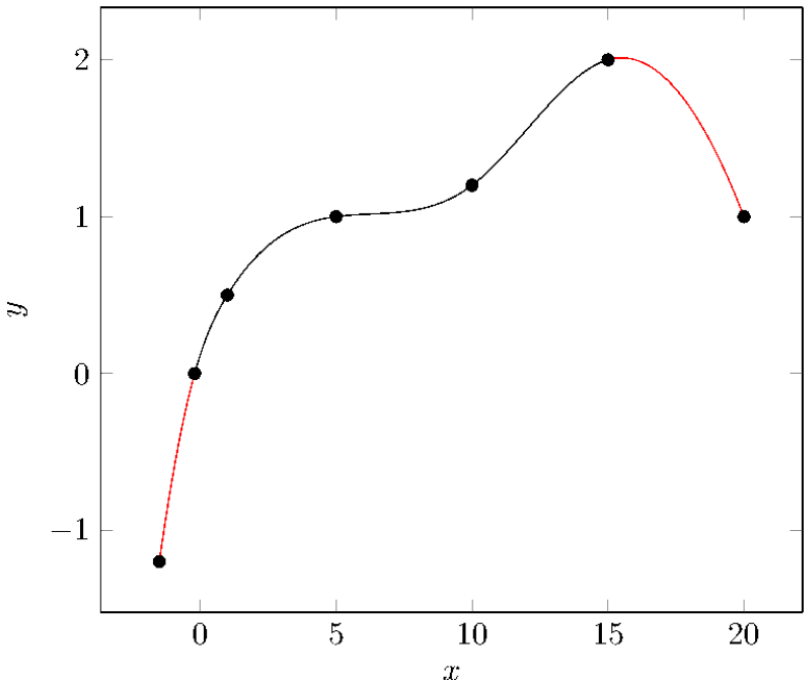

Рис. *47 Візуалізація квадратичного сплайну*

РОЗДІЛ 3 ПРОЕКТ ПРОГРАМНОЇ СИСТЕМИ КЕРУВАННЯ РОБОТОМ ДЛЯ ЗБОРУ ТЕНІСНИХ М'ЯЧІВ

3.1 Архітектура системи

Система керування роботом для збору тенісних м'ячиків складається з декількох модулів, які можна бачити на рисунку 48. Модуль «Darknet» – це модифікований застосунок існуючого фреймворку для нейронної мережі, який написаний на мові програмування C++. Всі інші модулі знаходяться у головному застосунку Unity.

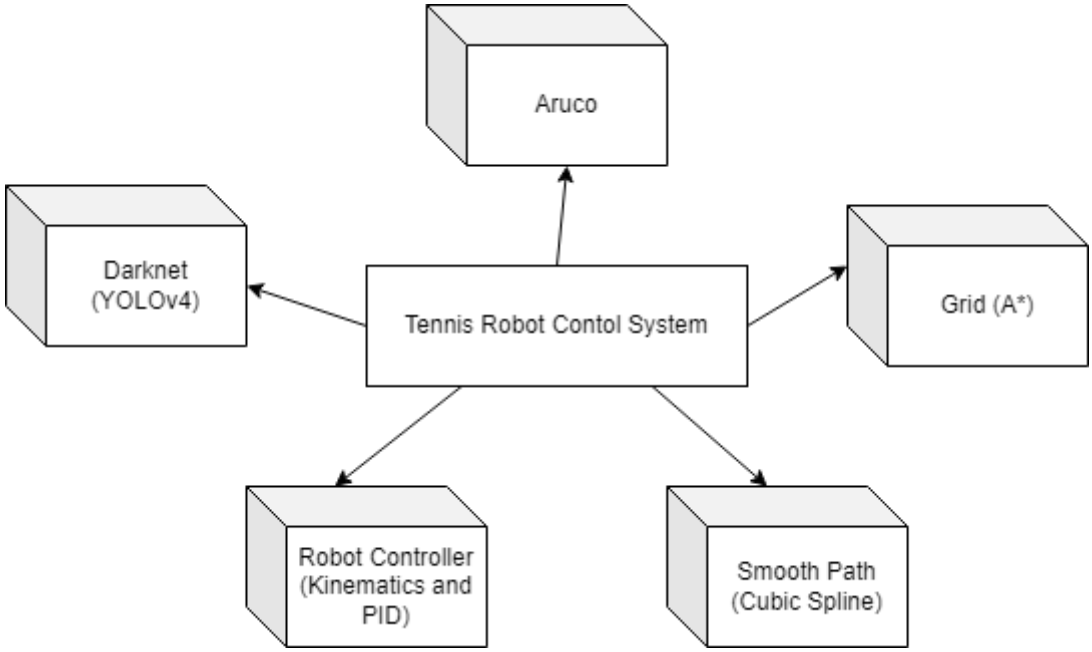

Рис. *48 Архітектура системи*

3.2 Засоби реалізації

Для розробки програмної системи, було вирішено використовувати наступні засоби:

1. середовище розробки Visual Studio 2019;
2. ігровий движок Unity 2019;
3. платформа для паралельних обчислень CUDA 10.2;

4. фреймворк .NET 4.x.

3.2.1 Середовище розробки Visual Studio

Visual Studi 2019 було обрано, як одно із найкращих рішень для написання коду на мовах програмування C++ та C#. Також це IDE має інтеграцію із ігровим движком Unity.

3.2.2 Ігровий движок Unity

Основний інструмент розробки – ігровий движок Unity 2019. Вміщує у себе реалістичну симуляцію фізики, що задовольняє потреби для розробки системи керування роботом для збору тенісних м’ячів.

3.2.3 Платформа для паралельних обчислень CUDA

CUDA – це платформа паралельних обчислень та модель програмування, яка робить використання графічного процесора для обчислень загального призначення простим та елегантним. У цієї платформі можна програмувати на C, C++, Fortran та інших. Список підтримуваних мов постійно розширюється.

Для проекту системи керування роботом для збору тенісних м’ячів CUDA використовується у фреймворку Darknet для обчислень нейронної мережі YOLOv4.

3.2.4 Фреймворк .NET

.NET – це платформа від Microsoft, яка дозволяє створювати програмні програми. Перший випуск .NET Framework відбувся у 2002 році. Вважається, що .NET Framework було створено як альтернативу платформі Java від компанії Sun. Головна відмінність у тому, що .NET Framework офіційно розрахована працювати саме з операційними системами сімейства Microsoft Windows. З того часу вона пройшла довгий шлях від версії 1.0 до 4.8 (18 квітня 2019), і на сьогоднішній день, незважаючи на появу платформи нового

покоління (.NET Core), як і раніше, досить популярна: існує безліч програмних продуктів, бібліотек і фреймворків, які написані та розвиваються під .NET Framework.

3.3 Модулі та алгоритми

Цілком систему керування роботом реалізовано через два незалежні застосунки. Нейронна мережа, яка працює на фреймворку Darknet знаходиться в окремому проекті, написаному на мові програмування C++. Це зроблено для того, щоб досягти максимальної продуктивності обчислень нейронної мережі. Модуль на мові C++ передає інформацію о знайдених м’ячах, або перешкодах, у застосунок на Unity, який у свою чергу вже виконує усі необхідні обчислення, щоб на виході відправляти коректні сигнали на контролер лівого та правого двигунів робота.

Як можна бачити, на рисунку 48 показана архітектура системи, головними компонентами якої є:

- модуль Darknet нейронної мережі YOLOv4;
- модуль Aruco для відстеження позиції на орієнтації робота;
- модуль Grid Map, де реалізовано алгоритм пошуку шляху A*;
- модуль Smooth Path для зглажування траекторії руху робота;
- модуль Robot Controller для обчислень кінематики та відправки сингалів на двигуни робота.

3.3.1 Модуль Darknet

Для комунікації головного застосунка із застосунком нейронної мережі YOLOv4 потрібно якось передавати дані між ними. Для передачі зображення з камери у симуляції було вирішено використовувати метод відображення файлу в пам'яті (Memory-mapped File), який можна реалізувати за допомогою WinAPI. WinAPI – це загальна назва для набору базових функцій інтерфейсів

програмування застосунків для операційних систем сімейства Windows корпорації Microsoft.

З боку застосунку на Unity реалізовано копіювання даних зображення у спосіб, як показано в лістингу 1. Перед початком оперції копіювання, зображення підгоняється за розміром за допомогою методу бібліотеки OpenCV, щоб у приймаючої сторони не було проблем із невідповідністю даних. Так як зображення у форматі RGB, потрібно лише три канала для колора.

Лістинг *1 Копіювання зображення у Memory-mapped File (C#)*

```
Imgproc.resize(image, imageSized, new Size(Settings.SIZE_W,
Settings.SIZE_H));
var size = Settings.SIZE_W * Settings.SIZE_H * 3;
Marshal.Copy((IntPtr)imageSized.dataAddr(),
mappedFileBytes, 0, size);
using (var accessor = mappedFile.CreateViewAccessor())
{
    accessor.WriteArray(0, mappedFileBytes, 0, size);
}
while (!darknetProc.Initialized)
{
    Thread.Sleep(Settings.THREAD_DELAY);
}
darknetProc.ProcessImage();
```

З боку приймаючої сторони потрібно зчитати дані зображення (Лістинг 2).

Лістинг *2 Копіювання зображення з Memory-mapped File (C++)*

```
HANDLE hMapFile;
LPCTSTR pBuf;
hMapFile = OpenFileMapping(
    FILE_MAP_ALL_ACCESS,
```

```
    FALSE,
    mappedFileName);
if (hMapFile == NULL)
{
    printf(TEXT("Could not open file mapping object
(%d).\n"), GetLastError());
    return;
}
pBuf = (LPTSTR)MapViewOfFile(hMapFile,
                                FILE_MAP_ALL_ACCESS,
                                0,
                                0,
                                608*608*3);
if (pBuf == NULL)
{
    printf(TEXT("Could not map view of file (%d).\n"),
GetLastError());
    CloseHandle(hMapFile);
    return;
}
```

Для отримання даних про знайдені за допомогою нейронної мережі YOLOv4 класи з відправленого зображення, було застосовано зчитування з вихідного потока. Тобто застосунок на Unity перехопляє вихідні консольні дані фреймворку Darknet та після аналізу цього тексту, отримує остаточні результати ймовірностей знайдених на зображенні класів. Окремо виділяється клас «sports ball», за яким можна фільтрувати м’ячики. Нейронна мережа на датасеті COCO вже навчена на великому наборі даних спортивних м’ячів. Під даним класом можуть розпізнаватися м’ячі с тенісу, гольфу, більярду, футболу та схожих видів спорту з м’ячом (Рис. 49-52). Але за таке універсальне рішення страждає значення ймовірності для єдиного потрібного для розпізнавання класу. М’ячі в різних видів спорту можуть мати дуже різни

особливості, такі як колір та структура. Тому для збільшення ймовірності розпізнавання саме тенісних м’ячиків необхідно довчити нейронну мережу на необхідному наборі нових даних.

Рис. *49 Розпізнавання м’яча у тенісі за допомогою YOLOv4*

Рис. *50 Розпізнавання м’яча у гольфі за допомогою YOLOv4*

Рис. *51 Розпізнавання м'яча у більярді за допомогою YOLOv4*

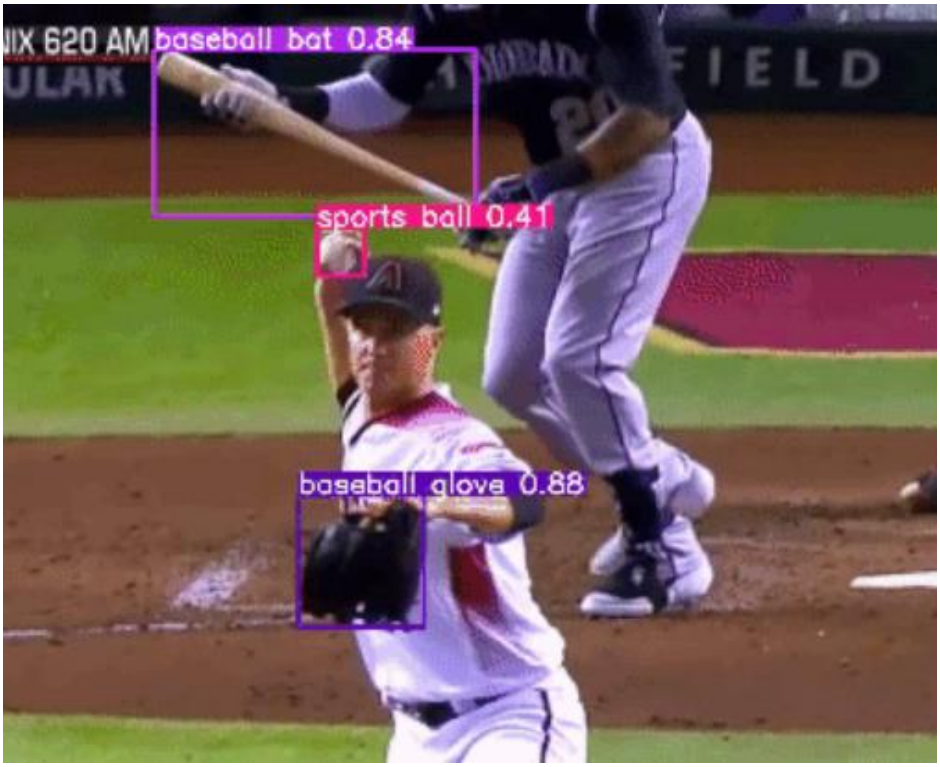

Рис. *52 Розпізнавання м'яча у бейсболі за допомогою YOLOv4*

У проекті, як перешкоди на шляху робота, розпізнаються лише люди. Звісно, це можна поправити та добавити щось з існуючого набора COCO, або донавчити додаткові класи на нових даних. Клас у нейронної мережі, який відповідає саме людям має назву «person». Цей клас також фільтрується з даних вихідного потоку фреймворка Darknet.

Зазвичай, нейронні мережі навчають розпізнавати людину у виді з переду, з заду або з боку (Рис. 53). YOLOv4 навчена на датасеті COCO, де положення людини у даних має різні орієнтації, в тому числі і вид з верху (Рис. 54). Тому це ще одна перевага використовувати саме цей варіант вже навчених ваг нейронної мережі, тому що камера для розпізнавання розташована саме зверху тенісного корта.

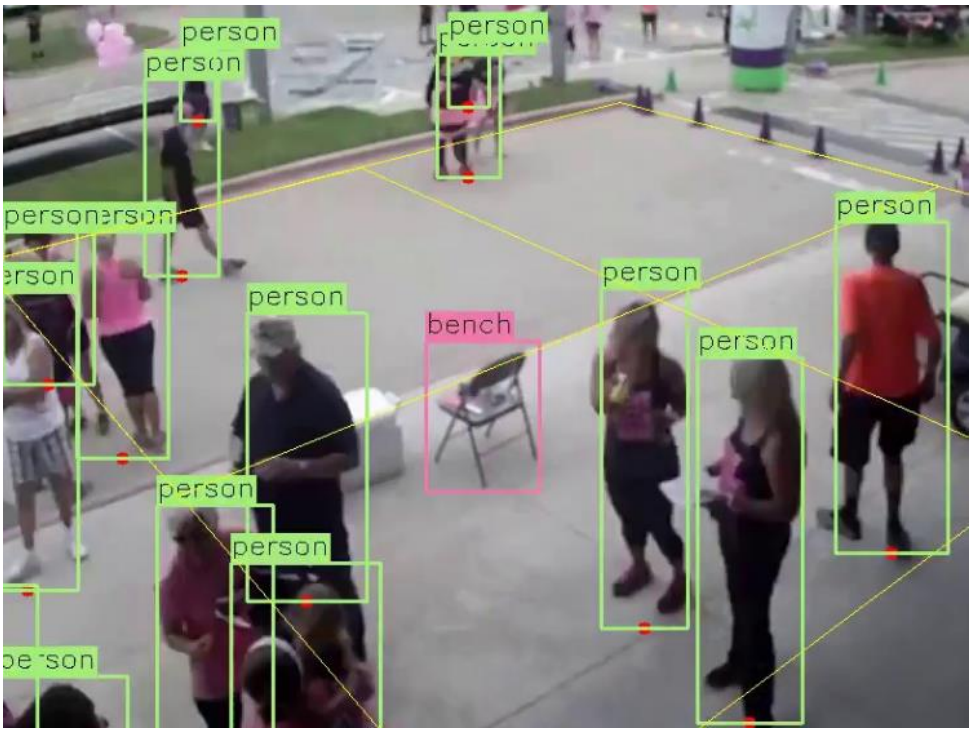

Рис. *53 Розпізнавання людини за допомогою YOLOv4*

Рис. *54 Розпізнавання людини з верху за допомогою YOLOv4*

3.3.2 Модуль Aruco

Aruco-маркери мають різний розміри сітки у бібліотеці OpenCV. Варіюються від 4x4 до 7x7. Існує також так званий APRILTAG, який уміщує у собі сітку із 16, 25 або 36 Aruco-маркерів. Занадто маленькі маркери можуть бути на щось схожі, це дасть помилкові розпізнавання, що зробить роботу застосунка по навігації робота некоректною. З іншої сторони досить велика сітка сповільнить час знаходження маркера, або може взагалі не знайтись на зображенні при певних умовах. Тому оптимальним варіантом було обрано розмір 5x5 з кодом «116» із зображенням, по якому можна однозначно візуально визначити, де у робота перед (Рис. 55), що дуже зручно при розробці та тестуванні.

Рис. *55 Aruco-маркер 5x5 з кодом «116» на прототипі робота у симуляції*

Обчислення по знаходженню Aruco-маркера на зображенні за допомогою бібліотеки комп'ютерного зору OpenCV виконується викликом методу `detectMarkers` класу Aruco модуля ArucoModule. Із даних, які повертає цей метод, можна ітеративно визначити потрібний знайдений маркер, якщо їх було декілька, а також провести додаткове обчислення для знаходження передньої сторони. Після усіх обчислень обов'язково потрібно очистити пам'ять. Код із застосунка показано на лістингу 3.

Лістинг *3 Знаходження Aruco-маркера з необхідним кодом на зображенні*

```
    Aruco.detectMarkers(map, predefinedDictionary, corners,
ids, detectorParams, rejectedCorners, camMatrix,
distCoeffs);
    var detectedCount = ids.total();
    if (detectedCount == 0)
    {
```

```
            RobotCenter = null;
            RobotHeading = null;
        }
        else
        {
            for (int i = 0; i < detectedCount; i++)
            {
                var id = (int)ids.get(i, 0)[0];
                if (id == Settings.ROBOT_ARUCO)
                {
                    var o1 = new Point(corners[i].get(0, 0)[0],
corners[i].get(0, 0)[1]);
                    var p1 = new Point(corners[i].get(0, 2)[0],
corners[i].get(0, 2)[1]);
                    var o2 = new Point(corners[i].get(0, 1)[0],
corners[i].get(0, 1)[1]);
                    var p2 = new Point(corners[i].get(0, 3)[0],
corners[i].get(0, 3)[1]);
                    RobotCenter = GetCenter(p1, o2);
                    var oPerpendickular =
GetNormalized(GetPerpecdickular(p1, o2));
                    RobotHeading = RobotCenter +
oPerpendickular * ROBOT_HEADING_DISTANCE;
                    break;
                }
            }
        }
        ids.Dispose();
        foreach (var corner in corners)
        {
            corner.Dispose();
        }
        corners.Clear();
```

```
foreach (var corner in rejectedCorners)
{
    corner.Dispose();
}
rejectedCorners.Clear();
```

3.3.3 Модуль Grid Map

Щоб застосувати алгоритм пошуку шляху A*, треба мати якесь середовище з декартовою системою координат. Було згенеровано сітку розміром 96x54, це 0.05 частина роздільної здатності відомого розміру екрана Full HD (Full High Definition) 1920x1080.

Основою модуля Grid Map є алгоритм пошуку шляху A*. Евристикою виступає оптимізоване під велику сітку обчислення дістанції [31] з певними кофіцієнтами (Лістинг 4).

Лістинг *4 Евристика, оптимізована під велику сітку*

```
static int GetDistance(Node a, Node b)
{
    var dstX = Mathf.Abs(a.gridX - b.gridX);
    var dstY = Mathf.Abs(a.gridY - b.gridY);
    if (dstX > dstY)
    {
        return 14 * dstY + 10 * (dstX - dstY);
    }
    else
    {
        return 14 * dstX + 10 * (dstY - dstX);
    }
}
```

Щоб алгоритм мав змогу генерувати маршрути від робота до цілі та для обходу перешкод, необідні дані додаються до параметрів сітки. Кожна клітка

у сітці містить у собі булеве значення, яке відповідає за проходимось для робота. Усі перешкоди, знайдені нейронною мережею, або внесені вручну (стовпи, інші статичні об’єкти на корті), помічаються як непроходимі. Також додаються позиції знайдених м’ячів як цілі, та позиція робота із модуля розпізнавання Aruco-маркера. Усі розміри масштабуються під розмір сітки, що дає змогу алгоритму A* коректно виконувати свою роботу (Рис. 56).

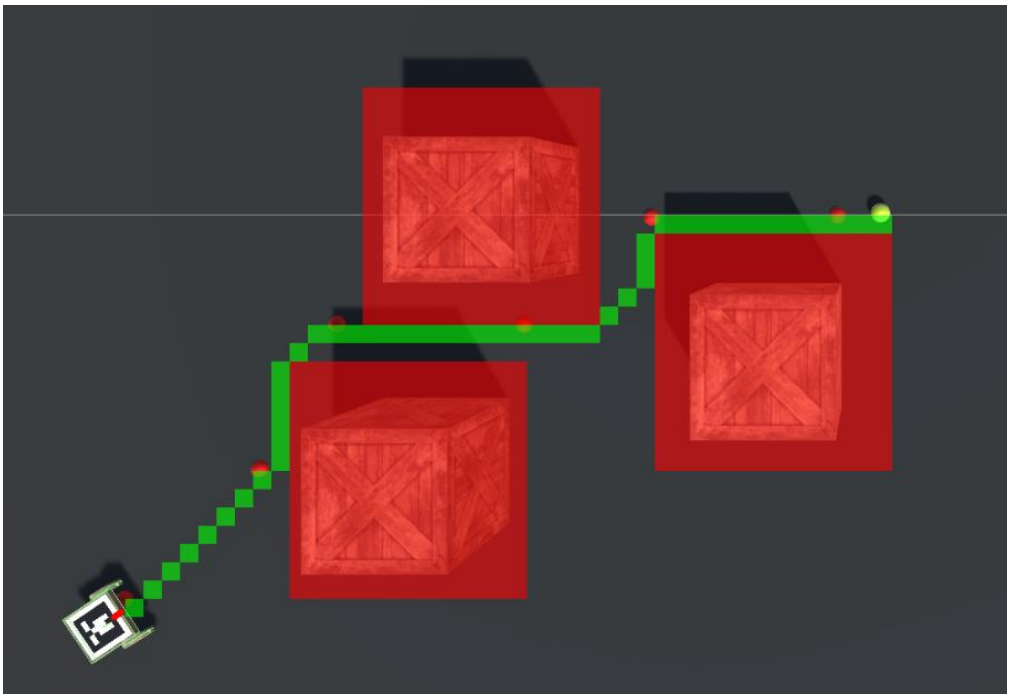

Рис. *56 Згенерований алгоритмом A* маршрут у симуляції. За перешкоди взяті статичні ящики*

3.3.4 Модуль Smooth Path

Якщо робот буде їхати по кліткової траєкторії, рух буде супроводжуватися зайвими поворотами вліво на вправо, що зменшить ефективність пересування та взагалі буде виглядати досить дивно. Тому обов’язково потрібно якось згладити маршрут. Для цієї мети застосовуються криві, у тому числі сплайни. За допомогою кубічного сплайна можна отримати необхідній результат. Усі послідовні позиції клітин згенерованого модулем застосунку Grid Map подаються на вхід алгоритму, що створює кубічний сплайн. Далі

цей сплайн використовується для обчислення необхідних параметрів руху робота у модулі керування.

3.3.5 Модуль Robot Controller

Для створення реалістичного руху робота у симуляції потрібен модуль, який буде правильно обчислювати актуальні кутові швидкості лівого на правого коліс у заданий час, виходячи з позиції робота відносно траєкторії до цілі. Модуль «Robot Controller» приймає на вхід згенерований сплайн необхідного маршруту та орієнтацію робота у просторі (позиція та кут повороту). На основі цих даних за допомогою формул векторної кінематики можна отримати необхідні значення кутових швидкостей (Лістинг 5). У даному лістингу змінна `dir_to_target` – це вектор у локальній системі координат робота відносно ближньої позиції на сплайні.

Лістинг *5 Обчислення кінематики руху робота*

```
var phi = Mathf.Atan2(dir_to_target.y, dir_to_target.x);
var om = ROBOT_OM_SPEED * phi;
var v_r = robot_speed + WHEEL_BETWEEN_DISTANCE * om;
var v_l = robot_speed - WHEEL_BETWEEN_DISTANCE * om;
var om_r = v_r / WHEEL_RADIUS;
var om_l = v_l / WHEEL_RADIUS;
```

Бажана загальна швидкість повороту робота вказується у константі `ROBOT_OM_SPEED` та дорівнює 10. Таке оптимальне значення було вибрано дослідним шляхом. Інші константи – це відстань між колесами (`WHEEL_BETWEEN_DISTANCE`) та радіус колеса (`WHEEL_RADIUS`). Лінійну загальну швидкість теж можна задати як константу, але через те, що фізика у симуляції максимально наближена до реального середовища, існують похибки при ковзанні та задіяні такі параметри як маса робота та його кінетична енергія, що робить неможливим постійний рівноприскорений рух. Тому для

вирішення цієї проблеми задіяно використання PID-контролера, який згладжує зміну лінійної швидкості у час руху (Лістинг 6).

Лістинг *6 Клас PID-контролера*

```
public class PIDController: MonoBehaviour
{
    public float p = 5.0f;
    public float i = 0.001f;
    public float d = 2.0f;
    float cte_old = 0f;
    float cte_sum = 0f;
    const float AVERAGE = 20.0f;
    public float Compute(float CTE)
    {
        var value = p * CTE;
        cte_sum += Time.fixedDeltaTime * CTE;
        cte_sum += (CTE - cte_sum) / AVERAGE;
        value += i * cte_sum;
        float d_dt_CTE = (CTE - cte_old) /
Time.fixedDeltaTime;
        value += d * d_dt_CTE;
        cte_old = CTE;
        return value;
    }
}
```

Параметри PID обрани дослідним шляхом, щоб отримати ефективний результат.

3.4 Вимоги до апаратного забезпечення

Більш за все у застосунку системи керування роботом для збору тенісних м’ячів навантажується відеокарта, так як усі обчислення нейронної мережі доцільно виконувати саме на GPU. CPU не підходить для цієї задачі, тому що потрібен високий FPS для реалізації навігації робота у реальному часі.

Рекомендовані вимоги до апаратного забезпечення:

- Чотирьохядерний процесор з базовою частотою від 2 GHz.
- Оперативна пам’ять ємністю не менше чим 4 Гб.
- Графічний пристрій від компанії Nvidia, починаючи з моделі GTX 1050ti.

3.5 Опис функціональних можливостей

Застосунок дозволяє налаштувати систему керування роботом для збору тенісних м’ячів у середовищі симуляції. Застосунок може розпізнавати робота, м’ячі та людей, як перешкоди. При доопрацюванні можна розширити список класів для розпізнавання та зробити необхідні покращення вірогідності результатів обчислення нейронною мережею.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ СИСТЕМИ КЕРУВАННЯ РОБОТОМ ДЛЯ ЗБОРУ ТЕНІСНИХ М'ЯЧИКІВ

4.1 Аналіз траєкторії руху робота під час збору тенісних м'ячів

Для аналізу параметрів руху було проведено кілька тестів. Вони показали рекомендовані значення загальної швидкості повороту робота (далі – θ). Заготовлене тестове середовище у симуляції можна побачити на рисунку 57.

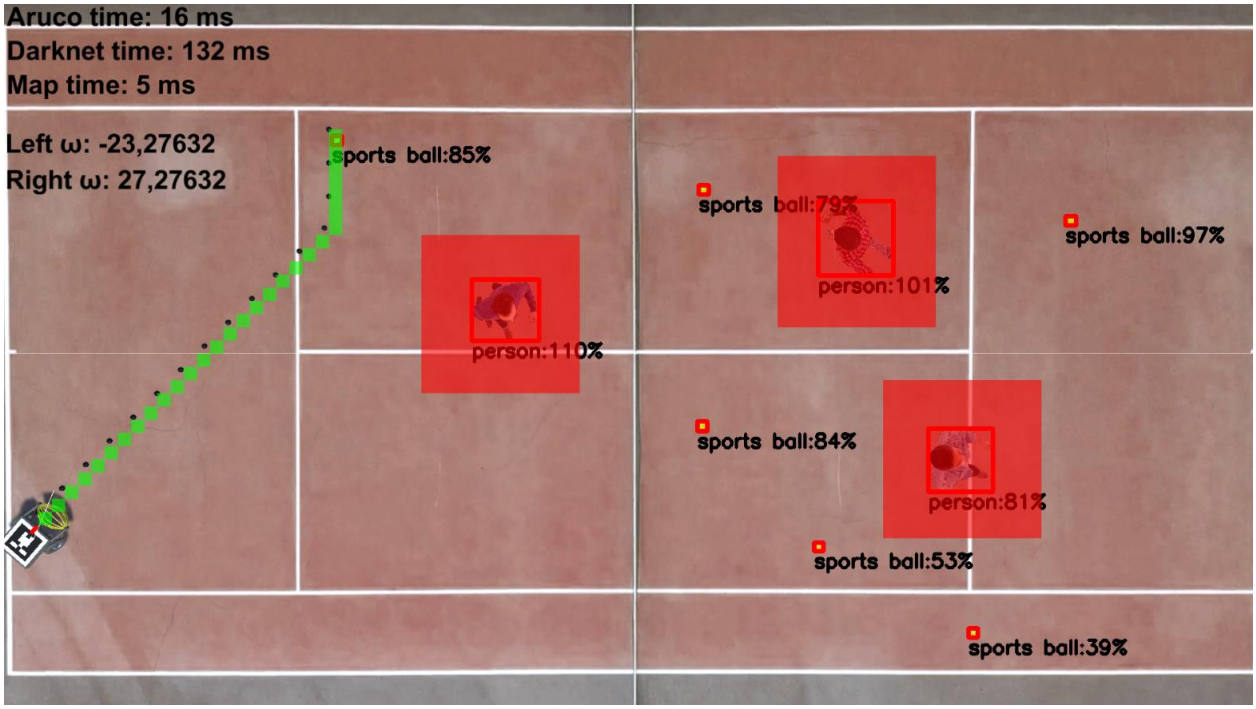

Рис. *57 Заготовлене середовище для робота із м'ячами та перешкодами*

У першому тесті було виставлено параметр θ = 5. Як можна бачити, траєкторія гладка, але є кілька зайвих маневрів робота (Рис. 58).

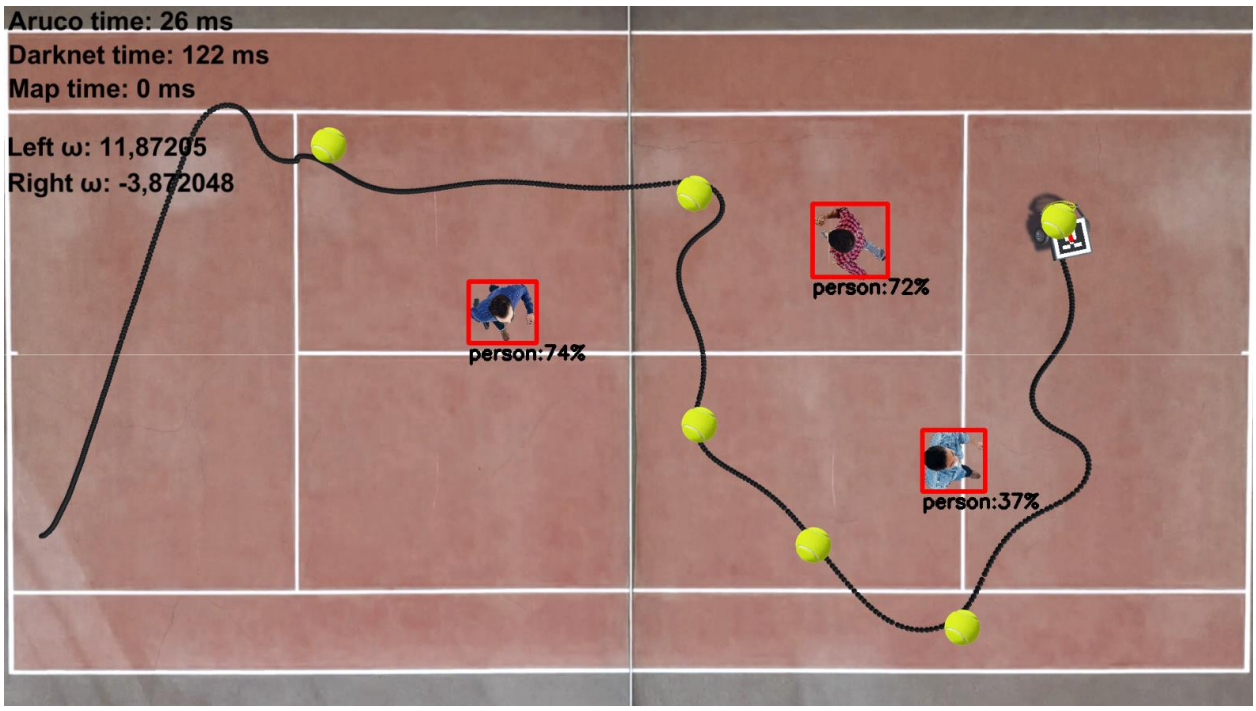

Рис. *58 Результати тестування з параметром θ = 5*

Наступний тест відбувався з параметром θ = 10. Траєкторія є гладкою з оптимальними маневрами робота (Рис. 59).

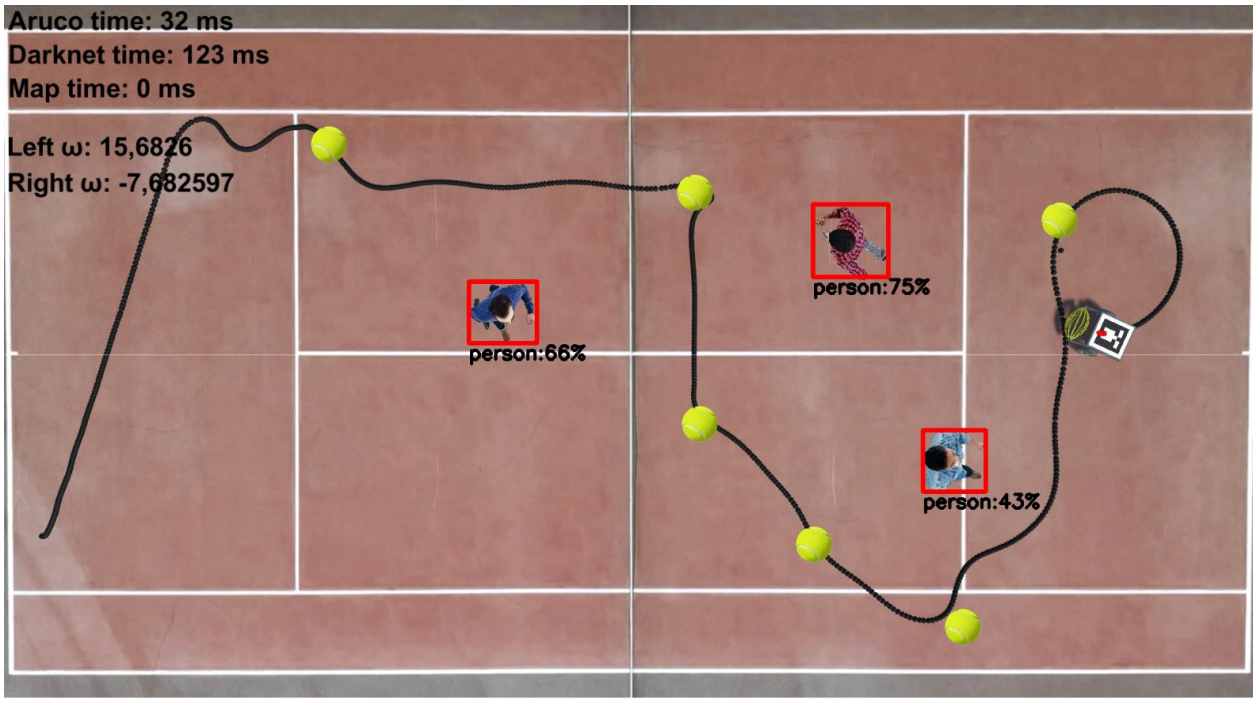

Рис. *59 Результати тестування з параметром θ = 10*

Останній тест відбувався з параметром θ = 20. Вже можна побачити наслідки зайвої швидкості повороту. У місці на траєкторії на рисунку 60 (показано червоною стрілкою), робот дуже різко звернув, що викликало зайвий поворот робота навколо своєї осі.

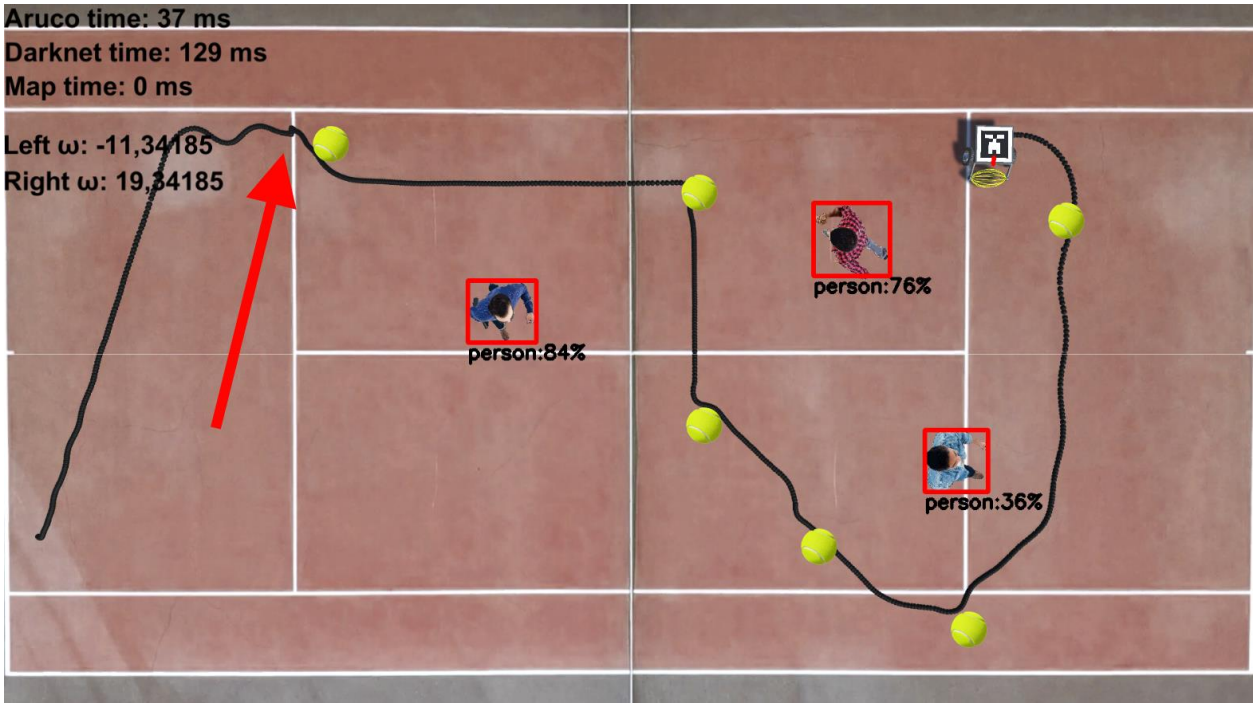

Рис. *60 Результати тестування з параметром θ = 20*

ВИСНОВКИ

1. Встановлено актуальність розробки системи керуваннями роботом для збору тенісних м’ячиків. Досліджено необхідні апаратні та програмні технології для реалізації повноцінної робочої системи. Проведено аналіз кожного програмного компоненту для реалізації навігації робота на тенісному корті.

2. Проблема залучення людського ресурсу у незначних для досягнення спортивних цілей справах залишається актуальною. Необхідно і далі працювати у цьому напрямку, отримуючи ефективні, допомагаючи людині у спорті рішення. Ця сфера тільки починає свій розвиток к зв’язці з такими технологіями як нейронні мережі та робототехніка загалом. Тому є велика область у якій можна покращити людський попит та задоволення від занять спортом.

3. Розроблена система робота для збору тенісних м’ячиків зібрала різні ефективні на даний час технології, такі як нейронна мережа YOLOv4, швидко працюючий трекінг маркерів Aruco та ін. Дана система допоможе використовувати час тренувань на тенісному корті більш ефективно.

4. Проведено аналіз траєкторії руху робота у процесі збора м’ячів для визначення оптимальної швидкості повороту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Троян Д.В., магістрант, Вербицький В.Г., доктор фізико-математичних наук, професор – науковий керівник. Комп'ютерна система керування роботом для збору тенісних м'ячів: Матеріали І Всеукраїнської науково-практичної конференції здобувачів вищої освіти, аспірантів та молодих вчених «Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України» 19-21 жовтня 2021 р. Запоріжжя : ЗНУ, 2021. С. 342-343.
2. Троян Д.В., магістрант, Вербицький В.Г., доктор фізико-математичних наук, професор – науковий керівник. Комп'ютерна система керування роботом для збору тенісних м'ячів. *Молода наука-2021* : зб. наук. праць студентів, аспірантів і молодих вчених. Запоріжжя : ЗНУ, 2021. Т. 5. С. 99–100.
3. Tennibot: The World's First Robotic Tennis Ball Collector. URL: https://www.kickstarter.com/projects/770435035/tennibot-the-worlds-first-robotic-tennis-ball-coll (дата звернення: 15.01.2021).
4. Tennibot is the world's first autonomous tennis assistant. URL: https://www.tennibot.com/ (дата звернення: 23.04.2021).
5. You Only Look Once: Unified, Real-Time Object Detection. URL: https://arxiv.org/pdf/1506.02640.pdf (дата звернення: 11.06.2021).
6. 1YOLO9000:Better, Faster, Stronger. URL: https://arxiv.org/pdf/1612.08242.pdf (дата звернення: 07.03.2021).
7. YOLOv3: An Incremental Improvement. URL: https://arxiv.org/pdf/1804.02767.pdf (дата звернення: 11.05.2021).
8. YOLOv4: Optimal Speed and Accuracy of Object Detection. URL: https://arxiv.org/pdf/2004.10934.pdf (дата звернення: 10.05.2021).

9. ArUco: a minimal library for Augmented Reality applications based on OpenCV. URL: https://www.uco.es/investiga/grupos/ava/node/26 (дата звернення: 23.05.2021).
10. Automatic Recognition of ArUco Codes in Land Surveying Tasks. URL: https://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/9_1_06_Siki.pdf (дата звернення: 16.01.2021).
11. OpenCV Python Tutorial. URL: https://www.geeksforgeeks.org/opencv-python-tutorial/ (дата звернення: 03.05.2021).
12. Affine and Projective Transformations. URL: https://www.graphicsmill.com/docs/gm5/Transformations.html (дата звернення: 22.05.2021).
13. Basic concepts of the homography explained with code. URL: https://docs.opencv.org/4.5.2/d9/dab/tutorial_homography.html (дата звернення: 18.05.2021).
14. Introduction to Unity. URL: https://www.javatpoint.com/introduction-to-unity (дата звернення: 17.05.2021).
15. Introduction to Physics in Unity. URL: https://medium.com/nerd-for-tech/introduction-to-physics-in-unity-8b3f9f467167 (дата звернен-ня: 15.02.2021).
16. Unity Wheel Collider for Motor vehicle Tutorial 2018. URL: https://www.gamasutra.com/blogs/VivekTank/20180706/321374/Unity_Wheel_Collider_for_Motor_vehicle_Tutorial_2018.php (дата звернен-ня: 23.05.2021).
17. A* Search Algorithm. URL: https://www.hackerearth.com/practice/notes/a-search-algorithm/ (дата звернення: 15.05.2021).
18. Advanced Fireworks Algorithm and Its Application Research in PID Parameters Tuning. URL: https://www.researchgate.net/publication/304187695_Advanced_Fireworks_Algorithm_and_Its_Application_Research_in_PID_Parameters_Tuning (дата звернення: 23.05.2021).

19. PID Controllers and PID Control in Control Systems. URL: https://www.electrical4u.com/pid-control/ (дата звернення: 13.05.2021).
20. Arduino Uno. URL: http://arduino.ru/Hardware/ArduinoBoardUno (дата звернення: 22.05.2021).
21. Аппаратная часть платформы Arduino. URL: http://arduino.ru/Hardware (дата звернення: 21.01.2021).
22. AVR061: STK500 Communication Protocol. URL: http://ww1.microchip.com/downloads/en/AppNotes/doc2525.pdf (дата звернення: 14.05.2021).
23. Motor Driver Circuit TA6586. URL: https://www.micros.com.pl/mediaserver/UITA6586_0001.pdf (дата звернення: 19.05.2021).
24. Lin M, Chen Q, Yan S. Network in network. URL: https://arxiv.org/pdf/1312.4400.pdf. (Дата звернення: 14.06.2021).
25. Russakovsky O, Deng J, Su H et al. ImageNet Large Scale Visual Recognition Challenge. Int J Computer Vision, 2015. 115 p. P. 211–252.
26. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neuralnetworks.pdf (Дата звернення: 14.05.2021).
27. Ruder S. An overview of gradient descent optimization algorithms, 2016. URL: https://arxiv.org/pdf/1609. 04747.pdf. (дата звернення: 14.05.2021).
28. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift, 2015. URL: https://arxiv.org/pdf/1502.03167.pdf. (дата звернення: 14.09.2021).
29. Zhong Z, Zheng L, Kang G, Li S, Yang Y. Random erasing data augmentation, 2017. URL: https://arxiv.org/pdf/1708.04896.pdf. (дата звернення: 17.10.2021).

30. Dudek and Jenkin, Computational Principles of Mobile Robotics. URL: https://www.researchgate.net/publication/263646055_Computational_Principles_of_Mobile_Robotics. (дата звернення: 24.09.2021).
31. Tilemap-based A* algorithm implementation in Unity game. URL: https://pavcreations.com/tilemap-based-a-star-algorithm-implementation-in-unity-game/ (дата звернення: 06.11.2021).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Троян Дмитро Віталійович, студент _2_ курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти se116-15@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Комп'ютерна система керування роботом для збору тенісних м'ячів**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2021 Підпис _____ Троян Дмитро Віталійович

(студент)

Дата 30.11.2021 Підпис _____ Вербицький Володимир Григорович

(науковий керівник)