Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення підсистеми автоматичного керування виробничим обладнанням з використанням розпізнавання жестів оператора

(тема)

Виконав:
Здобувач 4 року навчання,
групи АКТАКІТ-21-3
Едуард НІКОНЕНКО
(власне ім'я, прізвище)

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

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

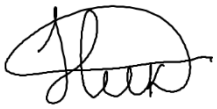
Керівник доцент Артем БРОННІКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Ніконенко Едуард Романович, як здобувач(ка) вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав(ла) і не одержував(ла) недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував(ла) штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

02 червня 2025 р.  Едуард НІКОНЕНКО

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

Кафедра КІТАР

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

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

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«30» квітня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ніконенко Едуарду Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення підсистеми автоматичного керування виробничим обладнанням з використанням розпізнавання жестів оператора

затверджена наказом університету від 19.05.2025 р. № 390 Ст.

2. Термін подання здобувачем роботи до екзаменаційної комісії 11.06.2025 р.

3. Вихідні дані до роботи Бібліотеки для комп'ютерного зору (MediaPipe, OpenCV); Алгоритм класифікації жестів (KNN); Програмне середовище розробки (Python); Web-камера.

4. Перелік питань, що потрібно опрацювати в роботі: аналіз систем розпізнавання жестів; вибір платформи; побудова моделі об'єкта; розробка алгоритму та програми керування; моделювання роботи системи; оформлення роботи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) слайди у форматі Power Point у кількості 12 слайдів з розширенням .pptx

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	20.04.2025	Виконано
2	Аналіз літератури за темою	22.04.2025	Виконано
3	Вибір та обґрунтування технічних засобів	10.05.2025	Виконано
4	Розробка програмного забезпечення	18.05.2025	Виконано
5	Оформлення пояснювальної записки	02.06.2025	Виконано
6	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism	03.06.2025	Виконано
7	Подання роботи на рецензію	09.06.2025	Виконано
8	Подання роботи на підпис зав.кафедри	15.06.2025	Виконано
9	Подання кваліфікаційної роботи в ЕК	16.06.2025	Виконано

Дата видачі завдання 28.04.2025 р.

Здобувач _____ Едуард НІКОНЕНКО
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ доц. Артем БРОННІКОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 60 с., 3 табл., 21 рис., 3 дод., 27 джерел.

РОЗПІЗНАВАННЯ ЖЕСТІВ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОМП'ЮТЕРНИЙ ЗІР, PYTHON, MEDIAPIPE, OPENCV, KNN.

Об'єкт розробки – процес автоматичного керування виробничим обладнанням на виробництві.

Предмет розробки – система розпізнавання жестів оператора виробничого обладнання.

Мета роботи – підвищення ефективності автоматизованого керування виробництвом на основі жестової взаємодії оператора з системою.

Розробка відповідає сучасним викликам сталого розвитку, зокрема Цілям ООН №8 "Гідна праця та економічне зростання" і №9 "Промисловість, інновації та інфраструктура", адже сприяє підвищенню ефективності та безпеки виробництва завдяки впровадженню інноваційних технологій керування.

Дана кваліфікаційна робота пройшла апробацію на III Міжнародній науково-практичній конференції "Research in Science, Technology and Economics" [1], де було представлено основні положення дослідження, зокрема використання жестових систем у виробничих процесах та повсякденному житті.

ABSTRACT

Explanatory note: 60 p., 21 figures, 3 tables, 27 sources according to the list of references, 3 appendices

GESTURE RECOGNITION, SOFTWARE, COMPUTER VISION, PYTHON, MEDIAPIPE, OPENCV, KNN

The object of development is the process of automatic control of production equipment in an instrument-making enterprise.

The subject of development is a gesture recognition system for the operator of production equipment.

Development goal is to improve the efficiency of production processes through the use of visual gesture recognition of the operator in automatic control systems.

The development aligns with the UN Sustainable Development Goals, particularly Goal 8 "Decent Work and Economic Growth" and Goal 9 "Industry, Innovation and Infrastructure", as it enhances production efficiency and safety through the implementation of innovative control technologies.

This explanatory note was approved at the 3rd International Scientific and Practical Conference "Research in Science, Technology and Economics" [1], where the main ideas of the study were presented, including the application of gesture recognition systems in industrial automation and everyday use.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПК – Персональний комп’ютер;

ОС – Операційна система;

САК – Система автоматичного керування;

AI (англ. Artificial Intelligence) – штучний інтелект;

API (англ. Application Programming Interface) – інтерфейс прикладного програмування;

GUI (англ. Graphical User Interface) – графічний інтерфейс користувача;

HCI (англ. Human–Computer Interaction) – взаємодія «людина-комп’ютер»;

KNN (англ. K-Nearest Neighbors) – метод k-найближчих сусідів;

RGB – модель представлення кольору червоний–зелений–синій;

SDK (англ. Software Development Kit) – набір інструментів розробника.

ВСТУП

Сучасне виробництво постійно змінюється під впливом технологічного прогресу. Зростання вимог до точності, швидкості та адаптивності процесів в підприємства шукати нові підходи до організації взаємодії між людиною та технікою. У цьому контексті особливої актуальності набуває впровадження інноваційних систем керування, що дозволяють підвищити ефективність без фізичного контакту з обладнанням.

Одним із таких рішень є використання технологій розпізнавання жестів. Це напрям, який об’єднує комп’ютерний зір, штучний інтелект і елементи кіберфізичних систем. Застосування жестового керування у виробничому середовищі дозволяє створити інтуїтивно зрозумілий спосіб комунікації з обладнанням, що знижує ймовірність помилок, підвищує безпеку праці та покращує виробничі процеси.

Поява доступних високочутливих сенсорів, камер глибини та відкритих бібліотек для обробки відеопотоку зробила можливим використання таких систем не лише у лабораторних умовах, а й у реальних виробничих середовищах. Переваги безконтактної взаємодії відкривають перспективи для впровадження подібних рішень на підприємствах різного масштабу – від невеликих автоматизованих дільниць до великих промислових комплексів.

Такі системи мають також економічний сенс для зменшення витрат на навчання персоналу, скорочення простоїв обладнання та можливість швидкої адаптації до змін у виробництві.

Розроблення підсистеми керування, здатної розпізнавати жести оператора та відповідно реагувати на них, є актуальним та перспективним напрямом у сфері автоматизації.

Метою цієї роботи є підвищення ефективності автоматизованого керування виробництвом на основі жестової взаємодії оператора з системою.

Об’єкт розробки – процес автоматичного керування виробничим обладнанням на виробництві.

Предмет розробки – система розпізнавання жестів оператора виробничого обладнання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– аналіз існуючих систем розпізнавання жестів;

– вибір компонентів та здійснення розробки структурної схеми програми;

– розробка програми розпізнавання жестів;

Кваліфікаційна робота виконана відповідно до вимог ДСТУ 3008:2015 [2] щодо структури та правил оформлення звітів у сфері науки і техніки, а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» [3].

1 АНАЛІЗ ЛІТЕРАТУРИ ЗА ТЕМОЮ

1.1 Технологічні тенденції сучасного виробництва та розвиток систем розпізнавання

Сучасне виробництво активно трансформується під впливом цифрових технологій, автоматизації та штучного інтелекту. Одним із ключових напрямів цієї трансформації є впровадження інтелектуальних систем керування, що забезпечують ефективну, безпечну та зручну взаємодію між людиною і машиною. У межах концепції «Індустрія 4.0» зростає інтерес до технологій безконтактного керування – зокрема голосового, мімічного та жестового.

Інтелектуальні системи розпізнавання дозволяють адаптивно реагувати на дії оператора, зменшують кількість помилок, знижують фізичне навантаження на персонал і підвищують загальну продуктивність. Подібні технології активно впроваджуються у виробничих процесах, медицині, транспорті, сфері освіти та побуті .

Серед основних напрямів розпізнавання можна виділити:

– розпізнавання обличчя – використовується для ідентифікації особи, контролю доступу, платіжних операцій тощо;

– оптичне розпізнавання символів (OCR) – дозволяє переводити текст із друкованих або рукописних носіїв у цифровий формат ;

– розпізнавання мови та голосу – забезпечує керування пристроями голосовими командами, використовується в побутових і професійних пристроях;

– розпізнавання жестів – дозволяє інтерпретувати рухи рук або тіла як керуючі команди, забезпечуючи природну форму взаємодії.

Вагомим стимулом до розвитку подібних технологій стало широке впровадження методів глибокого навчання, зокрема згорткових нейронних мереж. Завдяки цим методам стало можливим розпізнавання складних образів

у реальному часі з високою точністю, навіть у складних умовах з варіативним освітленням, перешкодами або шумом.

1.2 Сучасні підходи до реалізації систем розпізнавання жестів

У сучасних системах взаємодії "людина–комп'ютер" (HCI) розпізнавання поле досліджень, що включає інформатику, психологію, ергономіку, когнітивні науки, соціологію, мовознавство та інші галузі. Такий підхід дозволяє не лише оптимізувати інтерфейси з точки зору зручності та ефективності, але й створювати інтелектуальні та адаптивні системи, орієнтовані на користувача (рис. 1.1).

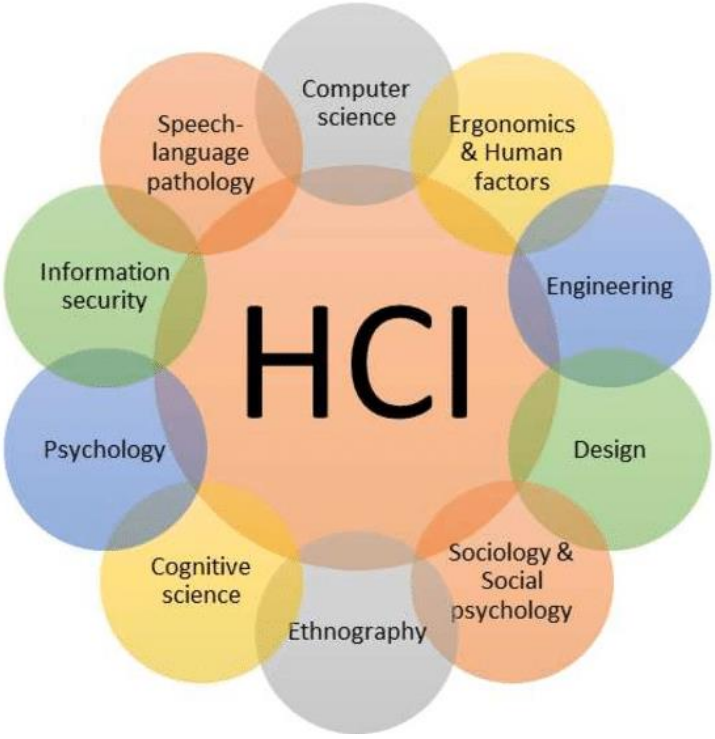


Рисунок 1.1 – Основні галузі в HCI

Системи розпізнавання жестів класифікують за кількістю використовуваних каналів на унімодальні та мультимодальні. Найбільш поширеними є візуальні унімодальні системи, що використовують камери для аналізу зображення рук, обличчя або міміки. Мультимодальні системи

поєднують візуальну інформацію з аудіо або сенсорними даними, наприклад поєднання жесту та голосу, що забезпечує вищу точність та природність у взаємодії [4].

Візуальне розпізнавання жестів (VGR) активно використовується в медицині, віртуальній реальності, «розумних» будинках, системах керування транспортом, комп'ютерних іграх, дистанційному навчанні тощо. Значним проривом стало застосування Microsoft Kinect (рис. 1.2) як безконтактного інтерфейсу, однак його продуктивність і точність залишаються обмеженими.

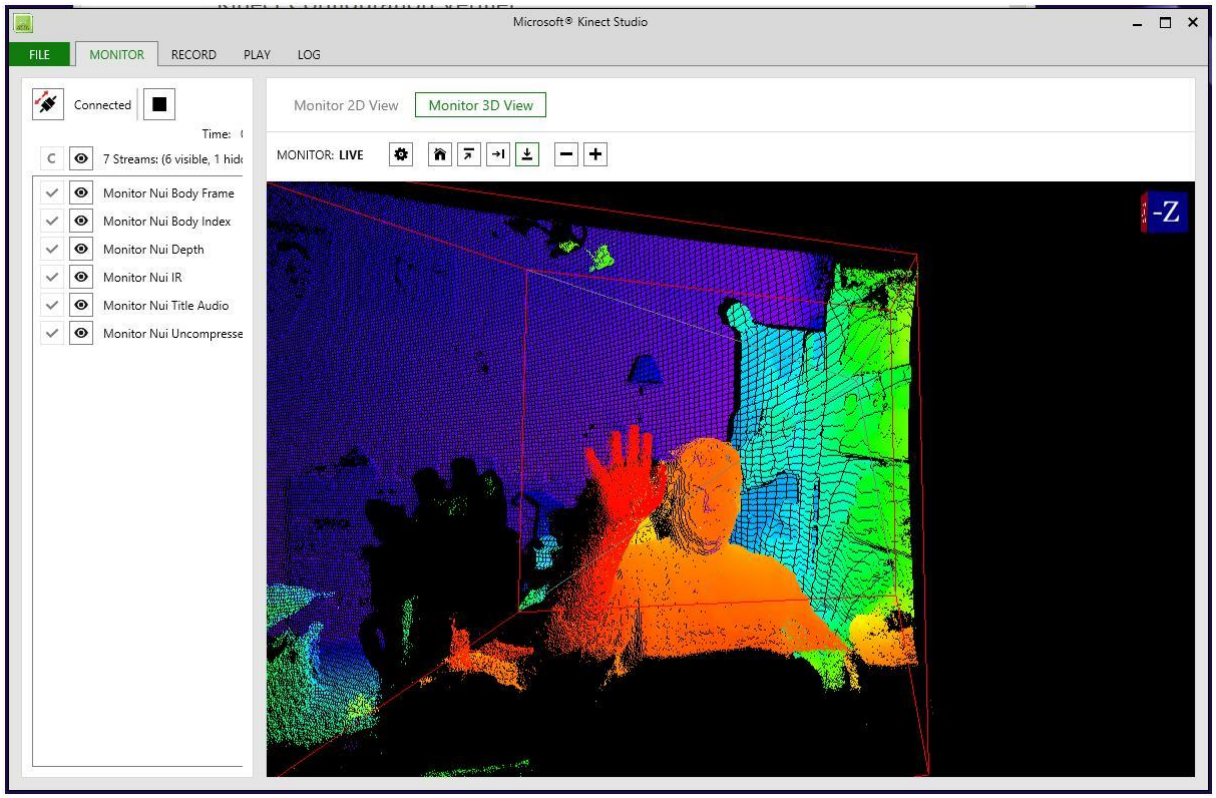


Рисунок 1.2 – Вигляд засобу з використанням Microsoft Kinect

Під час реалізації систем VGR виникає низка проблем, серед яких зміна освітлення, складність сегментації руки на складному фоні, неможливість точного відтворення тривимірної траєкторії з використанням лише однієї камери, високе навантаження на обчислювальні ресурси та потреба у точному апаратному синхронізуванні для забезпечення роботи в режимі реального

часу. Ці обмеження є критичними для застосування в таких сферах, як доповнена реальність та телемедицина.

Процес розпізнавання жестів умовно поділяється на три послідовні етапи: виявлення, відстеження та розпізнавання. На першому етапі пристрій фіксує рух руки або тіла за допомогою камери, а алгоритм машинного навчання виконує сегментацію зображення з метою визначення контурів та положення рук. Далі рухи фіксуються покадрово для формування точного потоку вхідних даних, що дозволяє здійснити якісний аналіз. На завершальному етапі система намагається знайти закономірності у зібраних даних, співставити їх із відомими шаблонами та виконати відповідну дію.

1.3 Технічні засоби для реалізації систем розпізнавання жестів

Для ефективного функціонування систем розпізнавання жестів необхідне використання спеціалізованого апаратного забезпечення, яке дозволяє точно й швидко реєструвати положення та рухи руки користувача. Вибір технічних засобів безпосередньо впливає на якість виявлення ознак, точність класифікації та загальну продуктивність системи. На сьогоднішній день переважають два основні підходи до технічної реалізації – на основі комп’ютерного зору та на основі інерційних або глибинних сенсорів.

Більшість рішень ґрунтуються на використанні зорових даних для відстеження рук, проте такі системи мають певні обмеження, зокрема потребу в обмеженій зоні взаємодії, втрату точності при частковому перекритті або виході руки з поля зору. У зв’язку з цим зростає інтерес до систем на основі датчиків руху та глибини, які здатні забезпечити розпізнавання як статичних, так і динамічних жестів у режимі реального часу.

У таких системах використовуються датчики глибини, які дозволяють поєднати комп’ютерне зображення з реальними координатами. Зокрема, вони забезпечують визначення тривимірного положення пальців, центру долоні та орієнтації руки. Оброблені дані містять інформацію про кути нахилу кінчиків

пальців, відстань до центру долоні, висоту розташування пальців, а також координати в тривимірному просторі. Система розпізнавання, що базується на комп’ютерному зорі, використовує шаблони, сформовані за допомогою алгоритмів машинного навчання, які тренуються на даних, отриманих із сенсорів глибини та руху.

Зображення руки поділяється на кілька областей: зап’ястя, долоня та пальці. Основна увага приділяється аналізу положення та форми пальців, тоді як зап’ястя зазвичай ігнорується через низьку інформативність. Отримані ознаки – відстані, висоти, форма долоні тощо об’єднуються у вектор ознак, який система використовує для розпізнавання конкретного жесту шляхом зіставлення з базою даних.

Завдяки використанню датчиків глибини стало можливим відмовитися від спеціалізованих носимих пристроїв, таких як рукавички, що значно підвищило природність взаємодії користувача з системою. Одним із провідних виробників таких технологій є компанія Intel, яка пропонує лінійку рішень RealSense. Зокрема, камера глибини Intel RealSense D455 (рис. 1.3) має функції лідара, стереозору, відстеження та кодованого світла, що забезпечує високий рівень точності розпізнавання жестів. Завдяки цьому її можна застосовувати у багатьох сферах – від робототехніки та дронів до 3D-сканування та систем відстеження людей [5].

Рисунок 1.3 – Камера глибини Intel RealSense D455

Варто зазначити, що висока функціональність таких камер супроводжується й відповідною високою вартістю. Так, на офіційному сайті компанії Intel вартість камери RealSense D455 становить від 419 доларів США, що може бути стримувальним фактором для їх масового застосування у споживчих пристроях.

Альтернативним рішенням є Leap Motion Controller 2 (рис. 1.4) – компактний пристрій, орієнтований на відстеження рухів рук без потреби в додаткових носимих елементах. На відміну від глибинних камер, він використовує інфрачервоні камери високої роздільної здатності для точного захоплення положення кисті, пальців та їхніх суглобів у реальному часі. Пристрій підтримує просторове відстеження з кутом огляду 160° та робочою відстанню до 110 см, що робить його зручним для застосування в умовах обмеженого простору [6].

Leap Motion 2 легко інтегрується з VR/AR системами, медичними симуляторами та навчальними середовищами, завдяки підтримці стандартів Windows, Android XR2, а також плагінів для Unity та Unreal Engine. Компактність і простота підключення через USB-C роблять його зручним у вбудованих рішеннях, включаючи інтерактивні термінали чи елементи управління без дотику.

Рисунок 1.4 – Leap Motion Controller 2

В окремих системах застосовуються навіть звичайні RGB-камери, які у поєднанні з потужними алгоритмами комп’ютерного зору (наприклад,

MediaPipe [7] від Google) дозволяють реалізувати розпізнавання жестів із мінімальними витратами, але цей варіант обмежені в точності при поганому освітленні та не здатні точно визначати глибину.

1.4 Аналіз використання систем розпізнавання

На сьогодні технології розпізнавання жестів активно інтегруються у різні сфери людської діяльності, забезпечуючи зручність, ефективність та безконтактність взаємодії з технікою. Їх використання охоплює широкий спектр галузей: від побутової електроніки й автомобілебудування до медицини та освіти. Основною метою впровадження таких технологій є створення природних, інтуїтивно зрозумілих інтерфейсів, які полегшують керування електронними пристроями та забезпечують високий рівень доступності.

1.4.1 Автомобільна сфера

У галузі автомобілебудування прикладом успішного впровадження систем розпізнавання жестів є інноваційна розробка компанії BMW. У нових моделях автомобілів реалізовано систему BMW Gesture Control, яка дозволяє керувати мультимедійними функціями за допомогою жестів без фізичного дотику до елементів керування. Наприклад, обертання пальця у повітрі змінює гучність звуку, а вказівний рух може використовуватись для прийому або відхилення вхідного дзвінка. Основу системи складають інфрачервоні сенсори та камери глибинного зору, які зазвичай розміщуються у центральній частині салону, зокрема в зоні дзеркала заднього виду або над панеллю приладів (рис. 1.5) [8].

Рисунок 1.5 – Система розпізнавання жестів BMW Gesture Control

1.4.2 Повсякденне життя

У повсякденному житті технології розпізнавання жестів поступово інтегруються у пристрої, що стали звичними для широкого кола користувачів. До них належать розумні телевізори, смартфони, пристрої інтернету речей , а також окуляри доповненої реальності. Одним з найяскравіших прикладів є пристрої Apple Vision Pro [9] та Meta Quest 3 [10]. Завдяки вбудованим камерам глибини та алгоритмам комп’ютерного зору вони дозволяють взаємодіяти з віртуальними об’єктами в реальному просторі за допомогою жестів рук, таких як натискання, перетягування або масштабування (рис. 1.6).

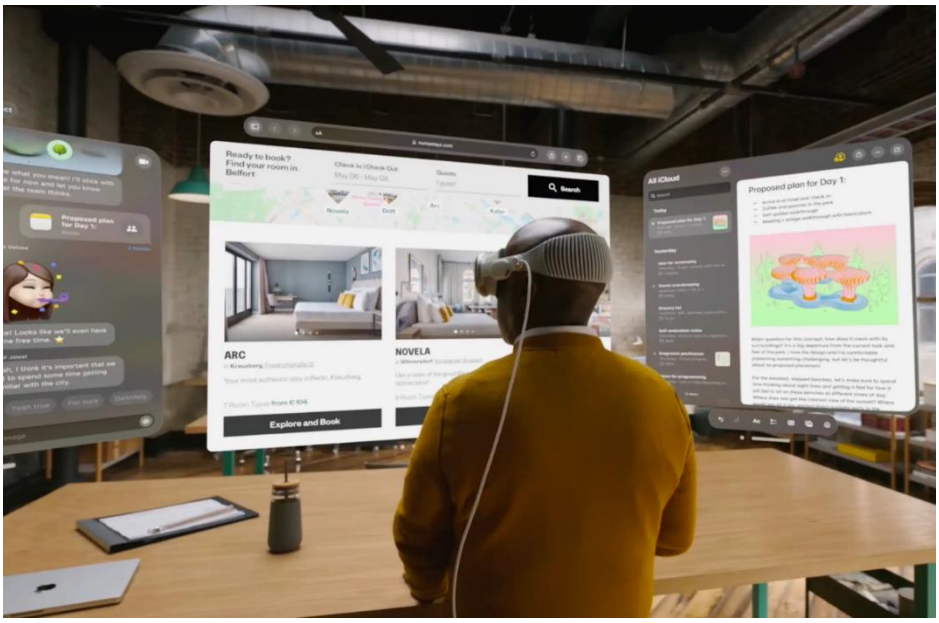


Рисунок 1.6 – Пристрій доповненої реальності Apple Vision Pro

Розпізнавання жестів також активно застосовується у смартфонах. Зокрема, у Google Pixel 4 реалізовано мікрорадар Soli, який дає змогу розпізнавати незначні рухи пальців без потреби доторку до екрана. Завдяки цій технології користувачі можуть перемикати треки, відхиляти дзвінки або вимикати будильник за допомогою простих жестів. Подібні рішення підвищують зручність користування пристроями та відкривають нові можливості для людей з обмеженими фізичними можливостями (рис. 1.7) [11].

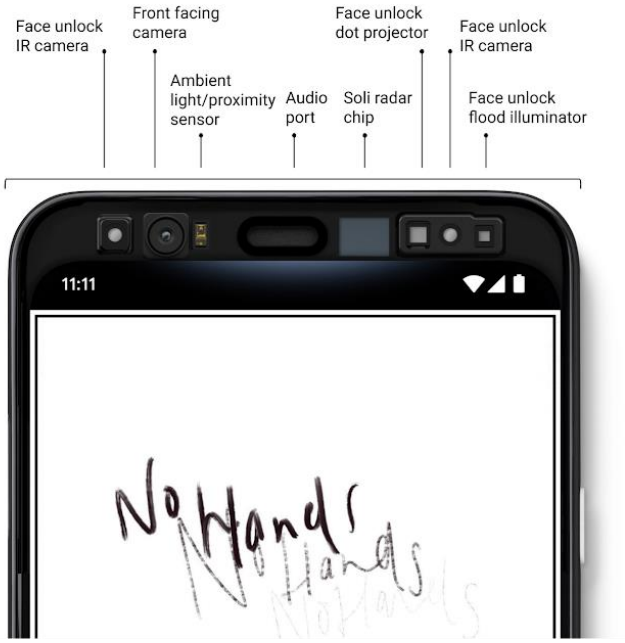


Рисунок 1.7 – Розташування сенсорів на передній панелі Google Pixel 4

1.4.3 Медицина

Медична сфера є однією з провідних у впровадженні безконтактних технологій, де розпізнавання жестів відіграє ключову роль у забезпеченні стерильності під час взаємодії з медичними інтерфейсами. Одним з прикладів є український стартап «eXtra Vision», розроблений лікарями з Харкова. Цей програмний продукт, призначений для окулярів доповненої реальності, дозволяє лікарям переглядати анатомічні моделі, історії хвороб та діагностичні зображення без фізичного контакту з пристроями, що суттєво знижує ризик забруднення стерильного поля під час хірургічних втручань (рисунок 1.8) [12].

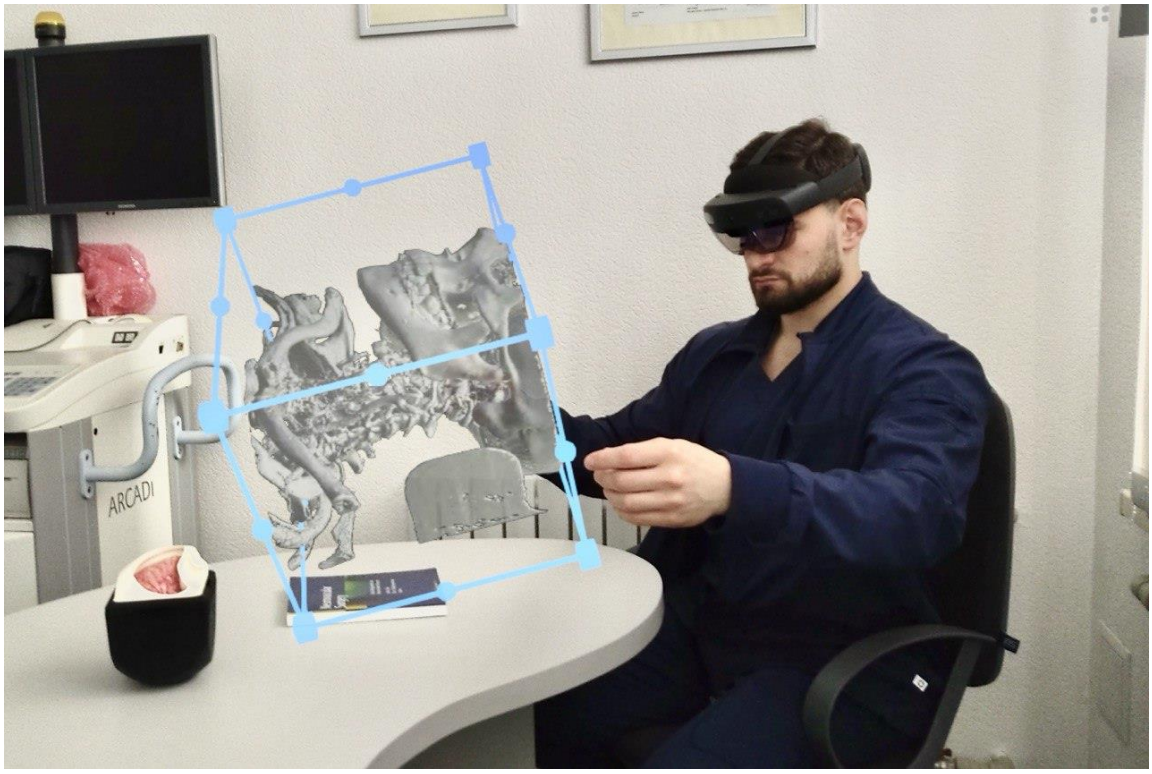


Рисунок 1.8 – Візуалізація анатомії за допомогою системи eXtra Vision

Крім застосування в операційних, технології розпізнавання жестів активно використовуються для моніторингу стану пацієнтів. Медичні працівники можуть здійснювати навігацію цифровими інтерфейсами, зокрема перегляд електронних медичних карток та результатів обстежень, за допомогою простих рухів рук. Це особливо актуально в умовах високої інфекційної загрози, де мінімізація фізичного контакту з поверхнями є вкрай важливою.

1.4.4 Освітні технології

Технології розпізнавання жестів знаходять активне застосування в освітній сфері, відкриваючи нові можливості для створення інтерактивного навчального середовища. Вони дозволяють студентам взаємодіяти з навчальними матеріалами за допомогою рухів, що особливо актуально у поєднанні з віртуальною та доповненою реальністю. Жести використовуються для навігації, вибору об’єктів, масштабування, запуску відеоінструкцій тощо,

що сприяє підвищенню рівня залученості студентів та кращому засвоєнню матеріалу.

Крім того, технології розпізнавання жестів можуть бути корисними для навчання дітей з особливими освітніми потребами. Безконтактна взаємодія з комп'ютером або інтерактивною дошкою дозволяє уникати фізичних бар'єрів, що є надзвичайно важливим для інклюзивної освіти.

Прикладом такого застосування є система SIGNIFY, яка використовує технологію машинного навчання для розпізнавання жестів рук з метою навчання італійської мови жестів у початковій школі. Ця система забезпечує зворотний зв'язок у реальному часі, що робить навчання більш захоплюючим та ефективним для дітей. SIGNIFY інтегрує елементи гри, включаючи навчальний режим та гру на зразок "вішалки", де учні відгадують слова, виконуючи відповідні жести (рис. 1.9). Система використовує камеру для виявлення 21 ключової точки руки, аналізуючи жести для підвищення точності. Вона сумісна як зі стандартними RGB-камерами, так і з камерами RGB-D, що забезпечує гнучкість при використанні різного обладнання [13].

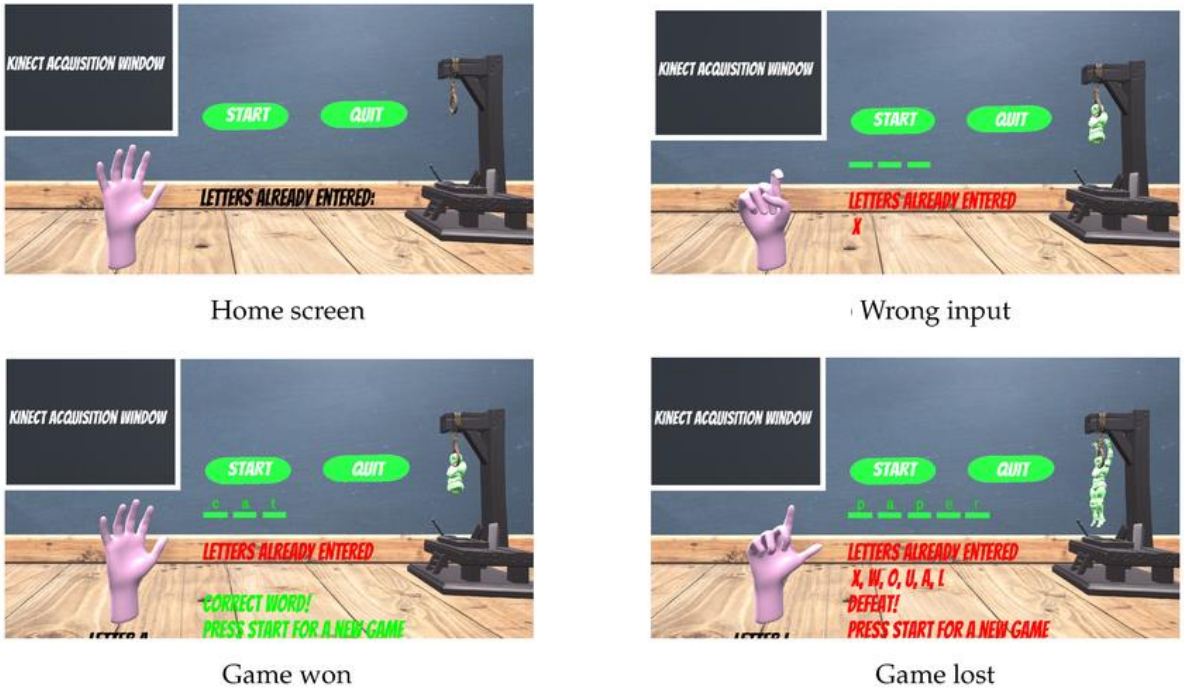


Рисунок 1.9 – Інтерфейс системи SIGNIFY

1.5 Перспективи розвитку

Упродовж останніх років ринок технологій розпізнавання жестів демонструє стабільне зростання. Згідно з даними Precedence Research, у 2024 році обсяг світового ринку становив $28,96 млрд, а в 2025 році очікується зростання до $34,48 млрд. Прогнозується, що до 2034 року ринок досягне $165,49 млрд, що відповідає середньорічному темпу зростання 19,04% у період з 2024 по 2034 роки (рис. 1.10) [14].

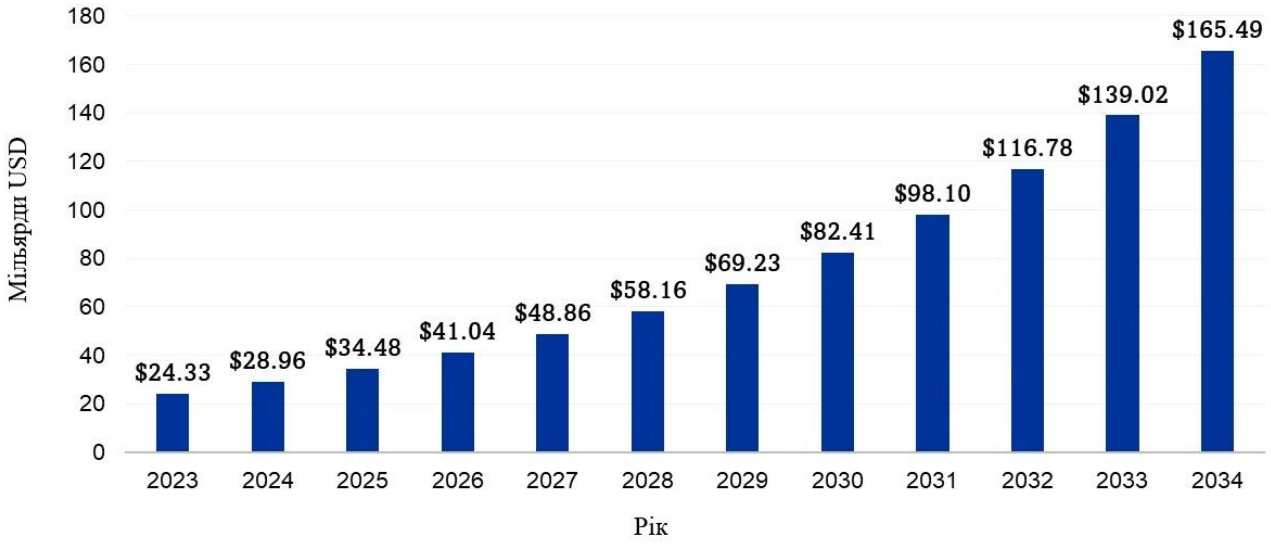


Рисунок 1.10 – Графік розвитку ринку систем розпізнавання жестів

Основною причиною зростання є потреба у безконтактних засобах взаємодії між людиною та електронними системами. Після пандемії COVID-19 інтерес до гігієнічних способів керування технікою значно зріс, що стимулювало впровадження технологій розпізнавання жестів у громадському транспорті, медичних установах, побуті та індустрії обслуговування. Значну роль відіграє також стрімкий розвиток алгоритмів машинного навчання та штучного інтелекту, які підвищують точність розпізнавання, адаптивність систем та дають змогу розширити функціональність без суттєвого ускладнення взаємодії користувача з пристроєм.

Водночас активне зростання сегментів споживчої електроніки, включно зі смартфонами, планшетами, телевізорами та карманними гаджетами, забезпечує стабільний попит на інтегровані рішення з розпізнаванням жестів. Такі функції поступово стають стандартом для пристроїв середнього та високого цінового сегмента. Не менш важливим є впровадження даної технології в автомобільній галузі: сучасні автомобілі преміум-класу все частіше оснащуються системами жестового управління мультимедійними функціями, що дозволяє знизити навантаження на водія та підвищити безпеку дорожнього руху.

З регіонального погляду, лідером ринку є Північна Америка, на яку у 2024 році припав обсяг у 8,69 мільярда доларів США. Цей регіон вирізняється високою концентрацією технологічних компаній, потужною інфраструктурою для науково-дослідних робіт та відкритістю до впровадження інновацій. Прогнозований середньорічний темп зростання ринку в Північній Америці становить 19,23% у період з 2024 по 2034 роки. Європейський ринок також демонструє позитивну динаміку, зумовлену впровадженням рішень для «розумних міст» і розвитком телемедицини. У свою чергу, країни Азіатсько-Тихоокеанського регіону, зокрема Китай, Японія та Південна Корея, активно застосовують технології розпізнавання жестів у побутовій електроніці, мобільних пристроях і автомобілебудуванні. На рисунку 1.11 можна побачити розподіл ринку за регіонами [14].

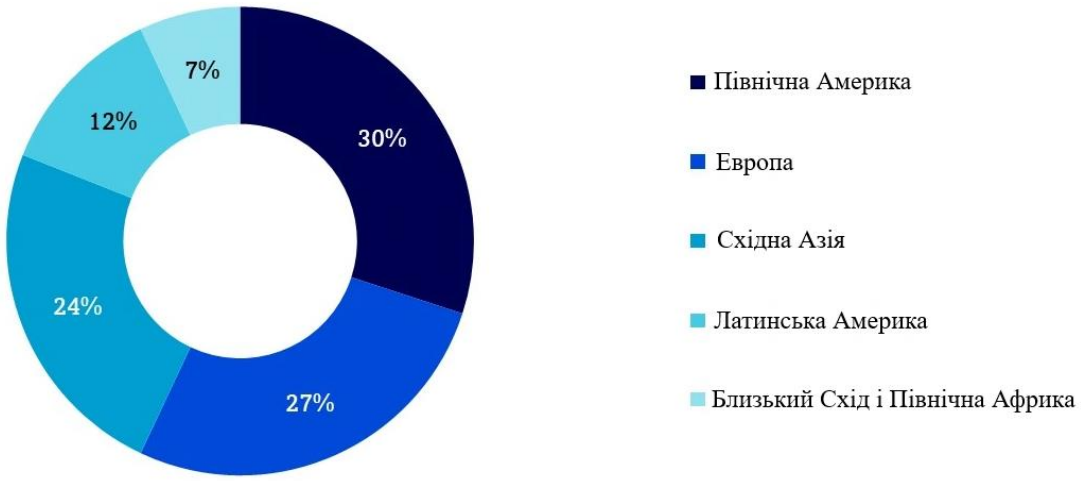


Рисунок 1.11 – Діаграма розподілу ринку розпізнавання жестів на 2023 р.

Серед основних галузей, які визначають подальші перспективи розвитку ринку, особливу увагу привертає охорона здоров’я. Зокрема, розпізнавання жестів набуває популярності у хірургічній практиці, де важливо зберігати стерильність та одночасно взаємодіяти з цифровими інтерфейсами. Також високий попит на такі рішення спостерігається в галузі роздрібної торгівлі, де інтерактивні екрани та інформаційні кіоски дозволяють покупцям взаємодіяти з системою жестами. В освітньому секторі технології розпізнавання жестів відкривають нові можливості для створення віртуального навчального середовища з підвищеною інтерактивністю, особливо у зв’язці з доповненою та віртуальною реальністю. Ще одним перспективним напрямком є системи безпеки, де жести можуть використовуватись як форма автентифікації та контролю доступу [15].

2 ВИБІР І ОБҐРУНТУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ

2.1 Обґрунтування вибору мови програмування

2.1.1 Вибір мови програмування

Для розробки системи розпізнавання жестів, серед мов програмування я обрав Python. Головним критерієм вибору можна означити його велику популярність в галузі машинного навчання та комп’ютерного зору. Також важливим чинником є велика кількість різноманітних бібліотек, які дозволяють облегшувати процес роботи з відео та їх обробкою.

Python має велику кількість бібліотек, спеціально оптимізованих для роботи з комп'ютерним зором, наприклад, OpenCV [16], MediaPipe [7], TensorFlow та PyTorch. Вони надають готові компоненти для детекції та відстеження руху, що суттєво облегшує процес розробки.

Одним із головних плюсів Python вважається низький поріг входу, тобто, простий та зрозумілий синтаксис навіть для новачків в програмуванні. Це дозволяє підвищити швидкість написання коду, та спрощує процес дебагінгу та пошук помилок у коді.

Python також вирізняється своєю гнучкістю та масштабованістю. Проєкт може починатися як невеликий прототип на локальному комп'ютері, а потім легко трансформуватися у повноцінну систему, яка працює на різних пристроях. Завдяки багаторічній спільноті та відкритому коду, більшість проблем, які виникають під час розробки, вже мають готові рішення та документацію.

Продуктивність Python, яка часто згадується як недолік, у контексті систем розпізнавання жестів не є критичною проблемою. Це пояснюється тим, що обчислювально складні операції у бібліотеках типу NumPy [17], TensorFlow чи OpenCV виконуються оптимізованим C/C++ кодом, в той час як Python виступає зручною оболонкою для керування цими процесами. Такий

підхід забезпечує баланс між швидкістю розробки та продуктивністю кінцевого продукту.

Мультиплатформеність Python дозволяє створювати системи, які працюють на різних операційних системах без значних змін у коді. Це особливо важливо для проєктів, що мають розгортатися на різноманітних пристроях – від потужних робочих станцій до вбудованих систем на базі Raspberry Pi або інших платформ, що для мого проекту має великий сенс.

Альтернативні мови програмування мають плюси та мінуси, але їх недоліки у розробці таких систем стають очевидними. Наприклад, C++ забезпечує високу продуктивність, але складний підхід до реалізації алгоритмів що значно збільшують час розробки. Крім того, робота з пам'яттю та складний процес налагодження в C++ можуть стати серйозними бар'єрами, особливо коли потрібно швидко перевірити різні ідеї та методики розпізнавання жестів.

Java є стабільною та кросплатформеною, але не має такої кількості спеціалізованих бібліотек для комп'ютерного зору й жестового трекінгу. Вертикальна інтеграція Java з корпоративними системами є перевагою для бізнес-застосунків, але для систем комп'ютерного зору це не дає суттєвих переваг. Водночас більш складний синтаксис та система типів Java ускладнюють швидке прототипування порівняно з Python.

C# та JavaScript не були розглянуті як альтенатива, бо їх застосування в основному використовується для розробки ігор чи роботи у веб-браузері. Вони не забезпечують тієї гнучкості, яка необхідна для інтеграції з апаратними пристроями або розробки систем, що працюють у реальному часі. Хоча C# має потужну екосистему .NET та якісні інструменти для розробки, його застосування для завдань комп'ютерного зору обмежене порівняно з Python. JavaScript, попри його велику перевагу у веб-розробці, не пропонує ефективних інструментів для обробки відеопотоку та складних математичних обчислень, необхідних для аналізу жестів.

Отже, вибір Python для розробки системи розпізнавання жестів є обґрунтованим та оптимальним рішенням, яке дозволяє максимально ефективно використовувати доступні ресурси та інструменти для створення якісного та надійного продукту.

2.1.2 Бібліотеки для Python

Для реалізації системи розпізнавання жестів лише мови програмування недостатньо. Важливою складовою ефективної розробки є використання спеціальних бібліотек. У проєкті було застосовано бібліотеки з відкритим кодом, які дозволяють працювати з комп'ютерним зором у режимі реального часу, швидко обробляти дані та візуалізувати результати. Ці бібліотеки стали основою для створення всієї програмної частини системи.

У реалізації системи розпізнавання жестів ключовими інструментами програмного забезпечення, зі сторони обробки відеопотоку, стали бібліотеки OpenCV [16] та MediaPipe Hands [7], вони забезпечують повноцінну роботу з відео, зображенням і виявленням жестів у реальному часі. Кожна з них виконує окрему, але тісно пов'язану функцію в системі, утворюючи зрозумілий та простий пайплайн обробки візуальних даних.

OpenCV – це потужна, багатоплатформна бібліотека з відкритим кодом, яка є стандартом де-факто в галузі комп'ютерного зору. У цьому проєкті вона виконує одразу кілька важливих функцій. Перш за все, саме за допомогою OpenCV здійснюється захоплення відео з камери в реальному часі, що є базою для подальшої обробки. Бібліотека дозволяє змінювати параметри кадру, зменшувати розмір зображення, переводити його в інші кольорові простори (наприклад, з RGB з HSV), а також застосовувати фільтри для зменшення шуму або покращення контрастності. Ці попередні кроки суттєво підвищують точність подальшого розпізнавання.

Також OpenCV широко використовується для побудови графічного відображення результатів: нанесення ключових точок на руку, відображення координат, побудова ліній між пальцями та інших візуальних елементів, що

допомагають зрозуміти, як система інтерпретує жести. Крім того, OpenCV підтримує модулі для виявлення об’єктів, трекінгу руху, аналізу контурів і навіть базове розпізнавання облич, що робить її універсальним інструментом у проєктах комп’ютерного зору.

MediaPipe Hands – це компонент бібліотеки MediaPipe, розроблений компанією Google, спеціально для відстеження й розпізнавання положення рук. На відміну від OpenCV, яка зосереджена на загальній обробці зображення, MediaPipe Hands відповідає за високорівневе розпізнавання конкретного об’єкта руки. Вона працює за двоетапною схемою: спочатку виявляє руку в кадрі за допомогою нейронної мережі, після чого застосовує регресійну модель, яка визначає координати 21 ключової точки на кожній руці. Ці точки відповідають суглобам пальців, долоні та кінчикам пальців, що дозволяє отримати повну картину положення кисті (рис. 2.1).

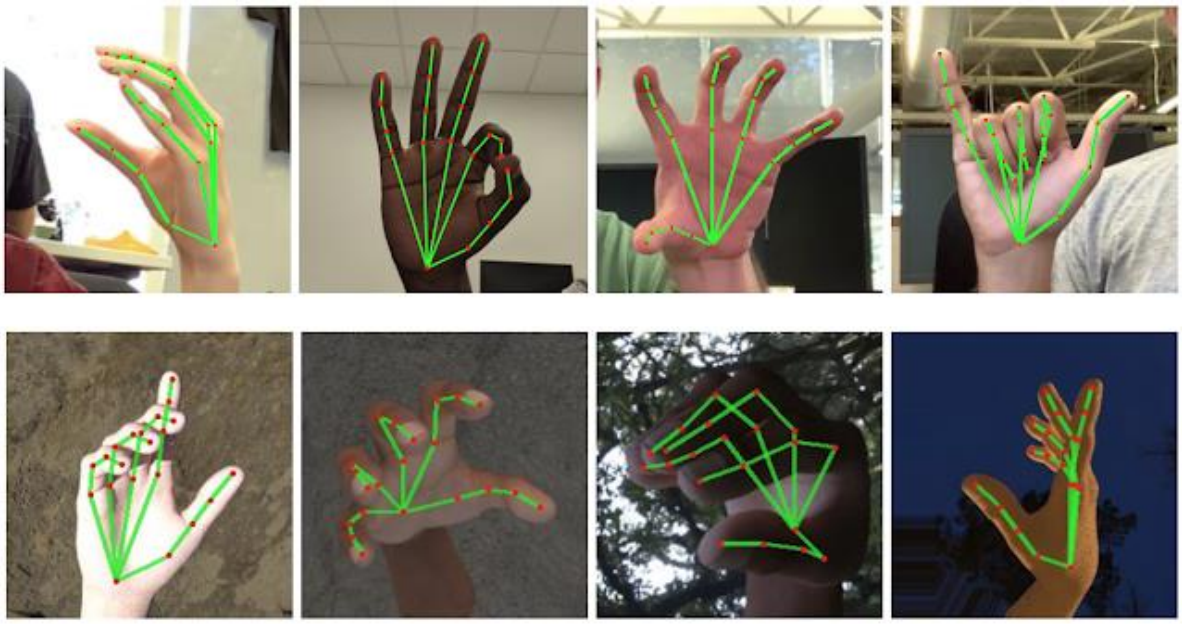


Рисунок 2.1 – Розпізнавання ключових точок долоні

Однією з ключових переваг MediaPipe є її здатність працювати в реальному часі навіть на пристроях без графічного прискорення. Завдяки оптимізованому пайплайну вона здатна обробляти відео з частотою 30 кадрів на секунду на звичайному CPU. Крім того, MediaPipe Hands автоматично

розрізняє ліву й праву руку, фільтрує координати для зменшення тремтіння та збоїв, а також підтримує одночасну обробку двох рук, що розширює функціональність системи.

В програмі MediaPipe Hands використовується для побудови геометричної моделі руки в кожному кадрі. Відповідні координати потім можуть бути використані для розпізнавання жестів – наприклад, шляхом порівняння відстаней між певними точками або аналізу кутів згину пальців.

Для ефективної роботи з великими об’ємами числових даних потрібні інструменти, які забезпечують швидкість, точність та зручність обчислень. У Python для цього є бібліотека NumPy, яка широко використовується в математичних розрахунках, аналізі даних та машинному навчанні.

NumPy дозволяє працювати з багатовимірними масивами, виконувати математичні операції будь-якої складності та обробляти великі обсяги інформації з мінімальними витратами ресурсів. Завдяки своєму оптимізованому ядру на C, бібліотека забезпечує значно кращу продуктивність у порівнянні зі звичайними структурами Python, такими як списки. Наприклад, якщо потрібно швидко виконати обчислення між масивами координат або визначити статистичні параметри великої вибірки даних, це можна реалізувати за допомогою однієї команди.

Для роботи з даними у Python часто виникає потреба зберігати об'єкти в файли або передавати їх між різними частинами програми. У таких випадках одним із найзручніших інструментів є бібліотека Pickle [18]. Вона дозволяє серіалізувати об'єкти Python у байтовий формат і, при необхідності, десеріалізувати їх назад, відновлюючи початковий стан об'єкта.

Pickle підтримує серіалізацію багатьох вбудованих типів даних Python: списків, словників, кортежів, а також об'єктів користувацьких класів. Наприклад, якщо потрібно тимчасово зберегти вміст складної структури даних для повторного використання, Pickle легко конвертує цю структуру у файл, а потім дозволить відновити її в точному вигляді. Це особливо зручно для

кешування, збереження результатів обчислень або передачі даних між різними етапами програми.

Головною перевагою Pickle є зручність його використання: процес серіалізації та десеріалізації зводиться до виклику кількох простих функцій. Варто відзначити, що Pickle підходить для внутрішніх потреб програми, коли дані не потребують додаткової безпеки, оскільки цей метод не забезпечує шифрування. У випадках, коли потрібен максимально швидкий та легкий спосіб роботи з даними у Python, Pickle стає надійним вибором.

OS [19] – стандартна бібліотека для взаємодії з операційною системою. Вона дозволяє працювати з файловою системою: отримувати шляхи, створювати папки, перевіряти наявність файлів тощо. Її застосовують для керування файлами та каталогами, необхідними для зберігання даних або результатів обробки.

Для створення графічних інтерфейсів у Python часто використовують бібліотеку Tkinter [20]. Це стандартний інструмент, який поставляється разом із Python і дозволяє створювати прості та функціональні GUI-додатки без необхідності встановлювати сторонні модулі.

Tkinter забезпечує всі необхідні елементи для розробки інтерфейсів: кнопки, текстові поля, поля введення, випадаючі списки, панелі меню тощо. Наприклад, якщо потрібно створити просту форму для введення даних чи інтерактивну програму на основі кнопок, Tkinter дозволяє реалізувати це швидко та зручно. Крім того, він дозволяє легко налаштовувати зовнішній вигляд елементів і організовувати їхнє розташування за допомогою менеджерів геометрії, таких як pack, grid або place.

Основна перевага Tkinter – проста структура та низький поріг входу. Для створення базового вікна та інтерактивних кнопок не потрібно великих знань чи складного коду. Також, оскільки Tkinter входить до стандартної бібліотеки Python, додаткового встановлення не потрібно, що робить його доступним і універсальним інструментом для будь-якого проєкту.

Цей модуль підходить і для новачків, які тільки починають вивчати графічні інтерфейси, і для досвідчених розробників, яким потрібні швидкі рішення для простих додатків. Завдяки своєму широкому набору функцій, Tkinter є відмінним вибором для створення кросплатформених GUI-програм.

2.2 Програмне середовище розробки

У процесі реалізації програмної частини проєкту було здійснено аналіз декількох програмних рішень для розробки на мові Python. Основну увагу було зосереджено на інтегрованих середовищах розробки (IDE) та текстових редакторах, які широко використовуються в сучасній практиці програмування. Під час вибору враховувалися такі чинники, як зручність інтерфейсу, підтримка основних функцій автодоповнення та перевірки коду, наявність інструментів для налагодження, а також можливості інтеграції з системами контролю версій.

Одним із найпопулярніших інструментів є PyCharm [21] – середовище розробки, створене спеціально для Python. Воно має розвинену інфраструктуру підтримки коду, що включає автоматичне завершення, підсвічування синтаксису, а також зручні засоби для роботи з віртуальними середовищами та системами керування версіями. PyCharm доступне у двох версіях: безкоштовній (Community) та платній (Professional), що дає змогу обрати варіант відповідно до потреб проєкту.

Інша популярна платформа – Visual Studio Code [22]. Це легкий редактор з широкими можливостями налаштування та великою кількістю розширень, які дозволяють значно розширити його функціональність. Завдяки відповідному розширенню для Python, середовище забезпечує підтримку автодоповнення, налагодження, інтеграцію з Git.

Серед спеціалізованих інструментів для аналізу даних варто згадати Jupyter Notebook [23]. Це інтерактивне середовище, яке широко використовується в задачах машинного навчання та візуалізації даних. Воно

дозволяє працювати з кодом блоками, що робить його зручним для дослідницької діяльності, але менш придатним для побудови структурованих програмних систем.

Також було розглянуто середовище Spyder [24], орієнтоване на наукові обчислення. Воно містить вбудовану консоль та засоби для візуалізації, що робить його зручним для інженерних та математичних завдань. Проте для більш комплексних програмних рішень його функціональності може бути недостатньо.

Після порівняння функціональних можливостей вирішив використовувати середовище PyCharm Community Edition. Вибір зумовлений високою стабільністю, гнучкістю налаштувань, глибокою інтеграцією з інструментами Python-екосистеми, а також вбудованим помічником, що допомагає швидко вирішувати проблеми з кодом.

2.3 Вибір компонентів для роботи системи

Важливим етапом розроблення системи розпізнавання жестів було визначення, яке саме апаратне забезпечення потрібне для реалізації всіх необхідних функцій.

Для захоплення відео вирішено використовувати звичайну веб-камеру з роздільною здатністю не менше 720p. Такий тип камер є доступним, не вимагає складного підключення і повністю відповідає вимогам для базової роботи систем комп’ютерного зору. Частота кадрів у межах 30 fps є непоганим варіантом для забезпечення відносно плавного розпізнавання жестів без ривків або втрати кадрів. Але для більш якісної роботи програми, потрібна більш якісна веб-камера з високою роздільною здатністю, та більшою частотою передачі кадрів.

Для обробки даних системи можна використовувати персональні комп’ютери (ПК) або ноутбуки. Сучасні бюджетні ПК та ноутбуки мають достатньо високу обчислювальну спроможність і дозволяють запускати

ресурсоємні алгоритми машинного навчання, що забезпечує високу точність та швидкість обробки.

У процесі розробки системи розпізнавання жестів використовувався мій ноутбук ASUS ROG Strix G513IH із процесором AMD Ryzen 7 4800H та 16 ГБ оперативної пам’яті. Така конфігурація забезпечує достатню обчислювальну потужність для запуску алгоритмів комп’ютерного зору, роботи з відеопотоком у реальному часі та використання моделей машинного навчання без помітних затримок. Продуктивність системи дозволяє ефективно тестувати та розгортати програмне забезпечення у середовищах, наближених до реального використання.

Для захоплення відео застосовувалась веб-камера Logitech C310 (рис. 2.2), яка підтримує роздільну здатність 720p при 30 кадрах на секунду. Цих параметрів достатньо для базового розпізнавання жестів, оскільки камера забезпечує стабільне зображення з прийнятною чіткістю. Просте підключення через USB та сумісність із більшістю операційних систем роблять її зручною для інтеграції у систему.

Рисунок 2.2 – Веб-камера Logitech C310

За потреби, систему можливо встановити на компактний пристрій, наприклад одноплатний комп’ютер, гарним та бюджетним варіантом є Raspberry Pi 4 [25]. Він має достатню кількість ресурсів для обробки відео та

запуску бібліотек OpenCV чи MediaPipe, також можливість підключити як звичайну USB-камеру, так і спеціалізований модуль камери через CSI-інтерфейс, що є обов’язковим для цієї системи. Якщо потрібна більш потужна альтернатива. можна застосовувати більш продуктивне рішення (табл. 2.1) – наприклад, NVIDIA Jetson Nano [26]. За її допомогою можна обробляти дані швидше або реалізувати більш складну логіку, але і ціна значно вища ніж згаданий Raspberry Pi 4.

Серед переваг Raspberry Pi 4 (рис. 2.3) можна виділити що він доступний за ціною, популярний і має величезну спільноту користувачів. Має вбудовані WiFi 802.11ac, Bluetooth 5.0 та Gigabit Ethernet. Пропонується кілька варіантів конфігурації оперативної пам’яті до 8 ГБ LPDDR4. Має два micro-HDMI порти з підтримкою 4K, стандартний 40-піновий GPIO, повністю сумісний із попередніми моделями. Простий у використанні, добре підтримується різними операційними системами (ОС).

Щодо недоліків, немає апаратної підтримки AI/ML, GPU значно слабший. Відео кодування обмежене 1080p, а зберігання здійснюється через повільніші microSD карти. Обмежена продуктивність при великих навантаженнях або задачах машинного навчання.

Рисунок 2.3 – Raspberry Pi 4

Таблиця 2.1 – Порівняння характеристик одноплатних комп’ютерів

Характеристика	Raspberry Pi 4	NVIDIA Jetson Nano
CPU	Quad-core ARM Cortex A57	Quad-core ARM Cortex A72
GPU	Broadcom GPU	NVIDIA Maxwell
ОЗУ	4 ГБ LPDDR4, 1600 МГц	4 ГБ LPDDR4-3200 МГц
Накопичувач	MicroSD до 256 ГБ	16 ГБ eMMC 5.1
Відео кодування	H.264 (1080p@30)	HEVC: 1× 4K@30, 2× 1080p@60, 4× 1080p@30, 4× 720p@60, 9× 720p@30
Відео декодування	H.265 (4K@60), H.264 (1080p@60)	HEVC: 1× 4K@60, 2× 4K@30, 4× 1080p@60, 8× 1080p@30, 9× 720p@60
Відеовиходи	2× micro-HDMI (до 4K@60)	HDMI 2.0, eDP 1.4
Камери	2-лінії MIPI CSI camera port	12 ліній (MIPI CSI-2 D-PHY 1.1)
USB порти	2× USB 3.0, 2× USB 2.0	4× USB 3.0, 1× USB 2.0 Micro-B
Інтерфейси	GPIO (40-pin), I²C, SPI, UART, стерео та композитний відеовихі	GPIO, I²C, I²S, SPI, UART
Мережа	Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0, BLE	Gigabit Ethernet, M.2 Key E
Ціна	≈ 35$	≈ 400$

З іншої сторони – NVIDIA Jetson Nano (рис. 2.4), має потужний GPU з 128 CUDA ядрами, що ідеально підходить для задач штучного інтелекту, комп’ютерного зору та відеоаналітики. Підтримує апаратне кодування та декодування HEVC до 4K@60, що забезпечує відмінну роботу з відео. Оснащений вбудованою eMMC пам’яттю, яка значно швидша та надійніша за microSD. Є підтримка сучасних інтерфейсів, таких як HDMI 2.0, eDP, USB 3.0 та M.2 для Wi-Fi/SSD.

Серед важливих недоліків, немає вбудованого Wi-Fi або Bluetooth, але це можливо виправити через додавання спеціальних модулів. Трохи складніший для новачків у налаштуванні та розгортанні. Також ціна значно вища.

Рисунок 2.4 – NVIDIA Jetson Nano

2.4 Створення IDEF0 діаграми

Для повного розуміння принципу роботи розробленої системи управління жестами доцільно виконати функціональне моделювання за методологією IDEF0. Такий підхід дозволяє структурувати основні процеси та взаємозв’язки між компонентами системи, включно з вхідними даними, механізмами, керувальними елементами та очікуваними результатами.

Контекстна діаграма (рис. 1.4) ілюструє загальну функцію системи, в той час як наступна діаграма (рис. 1.5) подає декомпозицію цієї функції на окремі блоки. Кожен з блоків A1–A4 виконує окрему підфункцію, яка разом забезпечує повноцінне розпізнавання жестів та керування обладнанням. Розглянемо їх детальніше нижче.

Рисунок 2.5 – Діаграма IDEF0

Декомпозиція дозволить нам визначити ключові блоки, такі як розпізнавання жестів, обробка сигналів, передача команд на обладнання та зворотний зв'язок. Кожен з цих блоків буде поділений на менші підфункції, що допоможе краще зрозуміти механізми роботи застосунку (рис 2.6).

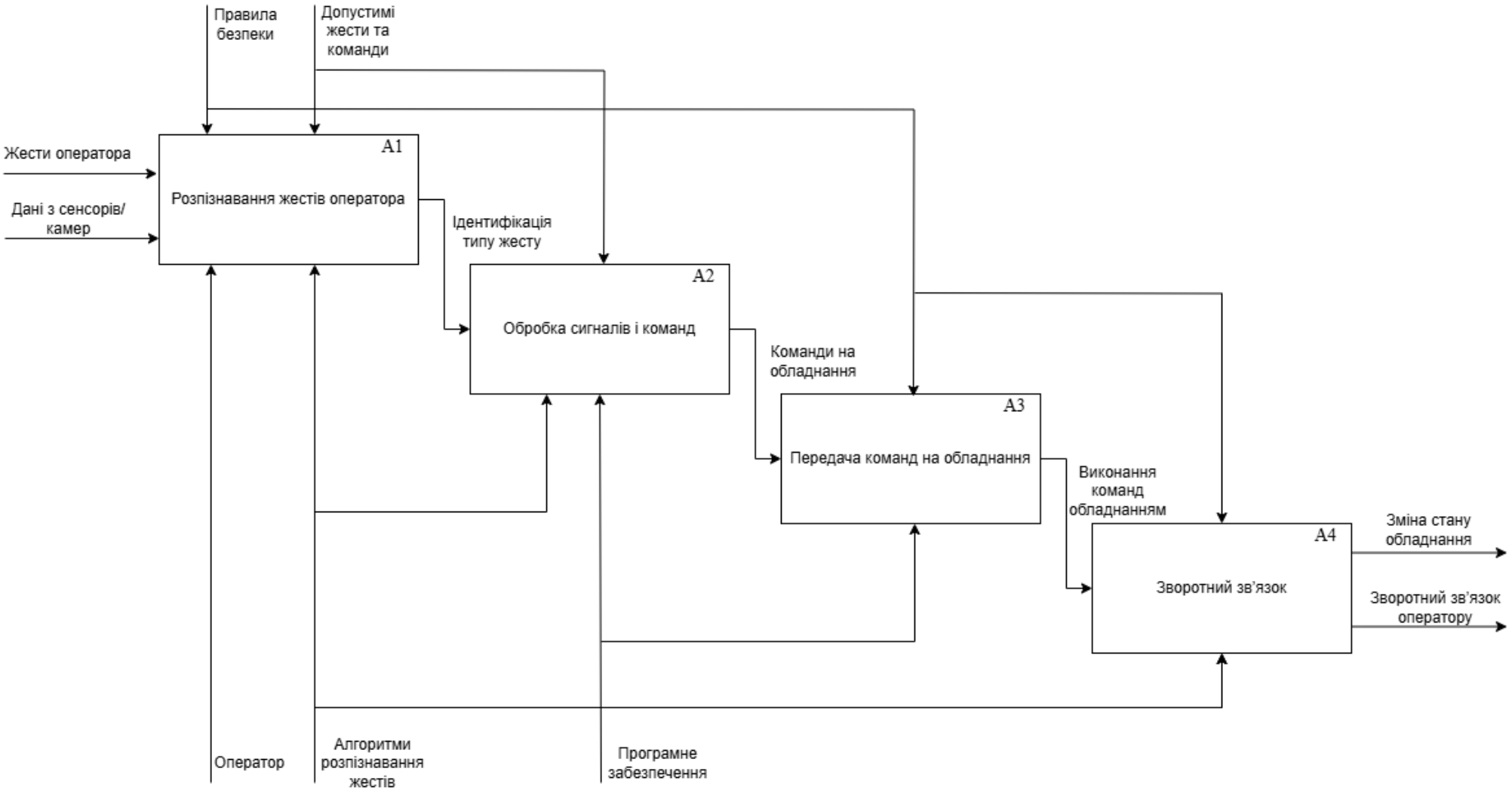


Рисунок 2.6 – Декомпозиція IDEF0

2.4.1 Функція A1 – Розпізнавання жестів оператора

Функція A1 відповідає за розпізнавання жестів оператора. У цьому блоці відбувається обробка вхідних даних, які надходять від сенсорів або камер, а також враховуються жести оператора. Основне завдання цього етапу зафіксувати рухи оператора та розпізнати, який саме жест було виконано.

У процесі розпізнавання також враховуються правила безпеки і допустимі жести та команди, які обмежують інтерпретацію лише до дозволених дій. Використання алгоритмів розпізнавання жестів дозволяє точно визначити тип жесту, який потім передається до наступного етапу для обробки.

2.4.2 Функція A2 – Обробка сигналів і команд

На другому етапі відбувається обробка сигналів і команд, які надходять із функції A1 після розпізнавання жесту. Цей блок відповідає за ідентифікацію типу жесту та перетворення його у формалізовану команду, яка надалі буде використана для керування обладнанням.

У цьому процесі враховуються також правила безпеки, які можуть впливати на дозволеність виконання певної команди. У блоці A2 використовуються програмне забезпечення і відповідні алгоритми обробки даних, що дозволяють з жесту створити конкретну інструкцію для системи.

2.4.3 Функція A3 – Передача команд на обладнання

Функція A3 реалізує передачу сформованої команди на виробниче обладнання. Після обробки в A2 команда надходить у цей блок, де здійснюється її направлення до технічного засобу, що має виконати відповідну дію.

Передача здійснюється автоматично через відповідні канали управління, а за її коректне виконання відповідає програмне забезпечення, яке забезпечує

стабільну комунікацію між системою розпізнавання жестів і керованим обладнанням.

2.4.4 Функція А4 – Зворотний зв’язок

Фінальний блок А4 відповідає за отримання зворотного зв’язку після виконання команди обладнанням. Цей зворотний сигнал надходить до системи для контролю виконання та, у разі потреби, інформує оператора про результат.

На цьому етапі фіксується зміна стану обладнання, яка може бути використана як частина подальшого процесу керування. Зворотний зв’язок дозволяє оператору розуміти, що система працює коректно або що потребується втручання.

2.5 Розрахунки ТАУ

2.5.1 Математичне моделювання САК

Розглянемо математичну модель виробничого об'єкта, наприклад, механізму транспортування — конвеєра, швидкість якого необхідно регулювати. Припустимо, що об'єкт керування має інерційні властивості та описується передавальною функцією інерційної ланки першого порядку (2.1):

$$W_p(s) = \frac{2}{0.5s+1} \tag{2.1}$$

де $K = 2$ — коефіцієнт підсилення;

$\tau = 0.5\ c$— постійна часу.

Для реалізації системи автоматичного регулювання використаємо ПІ-регулятор (пропорційно-інтегральний), що дозволяє забезпечити нульову статичну похибку при регулюванні постійного сигналу та підвищити якість перехідного процесу. Приймемо такі параметри регулятора:

$$K_p = 1, K_i = 0.5$$

Тоді передавальна функція ПІ-регулятора матиме вигляд (2.2):

$$W_c(s) = K_p + \frac{K_i}{s} = 1 + \frac{0.5}{s} = \frac{s+0.5}{s} \quad (2.2)$$

Передавальна функція розімкнутої системи (добуток регулятора й об'єкта керування) становить (2.3):

$$W_{open}(s) = W_c(s)W_p(s) = \left(\frac{s+0.5}{s}\right)\left(\frac{2}{0.5s+1}\right) = \frac{2(s+0.5)}{s(0.5s+1)} = \frac{2s+1}{0.5s^2+s} \quad (2.3)$$

Передавальна функція замкнутої системи з одиничним зворотним зв’язком обчислюється за формулою (2.4):

$$W_{closed}(s) = \frac{W_{open}(s)}{1+W_{open}(s)} = \frac{\frac{2s+1}{0.5s^2+s}}{1+\frac{2s+1}{0.5s^2+s}} = \frac{2s+1}{0.5s^2+3s+1} \quad (2.4)$$

Характеристичне рівняння замкнутої системи визначається знаменником передавальної функції замкнутої системи:

$$0.5s^2 + 3s + 1 = 0$$

Для зручності множимо обидві частини рівняння на 2, щоб позбутися дробових коефіцієнтів:

$$s^2 + 6s + 2 = 0$$

Отримане характеристичне рівняння можна використати для подальшого аналізу стійкості та якості регулювання системи.

2.5.2 Оцінка стійкості та якості перехідного процесу

Стійкість лінійної системи автоматичного керування визначається коренями її характеристичного рівняння. Система вважається стійкою, якщо всі корені характеристичного рівняння мають від'ємні дійсні частини (тобто знаходяться у лівій півплощині комплексної площини).

Характеристичне рівняння, отримане для розробленої системи:

$$s^2 + 6s + 2 = 0$$

Розв'яжемо його за допомогою формули коренів квадратного рівняння (2.5):

$$s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.5}$$

де $a = 1$;

$b = 6$;

$c = 2$.

Підставимо значення:

$$s = \frac{-6 \pm \sqrt{6^2 - 4 * 1 * 2}}{2} = s = \frac{-6 \pm \sqrt{28}}{2} = -3 \pm \sqrt{7}$$

Отже, корені:

$$s_1 = -3 + \sqrt{7} \approx -0{,}354;\ s_1 = -3 - \sqrt{7} \approx -5{,}646;$$

Оскільки обидва корені є дійсними та від’ємними, перехідний процес є аперіодичним (перенавантажень або коливань не буде). Домінуючим полюсом є значення з меншою за модулем дійсною частиною:

$$s_1 \approx -0{,}354$$

Яке визначає приблизну тривалість перехідного процесу за емпіричною формулою (2.6):

$$t_{\text{п.п.}} \approx \frac{3}{|s_1|} \approx \frac{3}{0.354} \approx 8{,}57\ c. \qquad (2.6)$$

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Програма для навчання розпізнавання жестів

Програма використовує бібліотеку OpenCV для захоплення відео з вебкамери та бібліотеку MediaPipe для детекції рук і ключових точок, повертаючи їх для кожної знайденої руки. Зібрані координати цих точок використовуються як ознаки для класифікації жестів за допомогою методу k-найближчих сусідів. Результати класифікації виводяться на екран, а програма також надає можливість додавати нові жести до навчальної бази, зберігати та завантажувати дані, а також взаємодіяти з користувачем через клавіші та графічні елементи інтерфейсу.

На початку створюється об’єкт cv2.VideoCapture, через який постійно зчитуються кадри з вебкамери. Отримані кадри конвертуються з простору BGR у RGB перед передачею в MediaPipe. Кожен кадр аналізується у циклі. Вхідні параметри MediaPipe (наприклад, min_detection_confidence) налаштовуються для забезпечення точності.

Клас KNN [27] реалізує простий, пасивний алгоритм класифікації, який не створює явної моделі, а працює безпосередньо з навчальними даними. Дані для навчання, що складаються з координат і міток класів, завантажуються за допомогою методу fit(). Для передбачення класу нового жесту використовується метод predict(), що обирає k найближчих прикладів із навчальної бази на основі евклідової відстані. На основі більшості збігів серед найближчих сусідів визначається клас введеного жесту. Такий підхід дозволяє легко додавати нові приклади до бази даних без потреби у повторній побудові моделі. На рисунку 3.1 можна побачити реалізацію класу KNN.

```python
class KNN:  1 usage
    def __init__(self, k):
        self.k = k
        self.coords = np.array([])
        self.labels = []
        self.label_to_int = {}  # Словник для перетворення текстових міток у числа
        self.int_to_label = {}  # Словник для перетворення чисел у текстові мітки

    def fit(self, coords, labels):  2 usages
        self.coords = np.array(coords)
        unique_labels = sorted(set(labels))
        self.label_to_int = {label: i for i, label in enumerate(unique_labels)}
        self.int_to_label = {i: label for label, i in self.label_to_int.items()}
        self.labels = np.array([self.label_to_int[label] for label in labels])

    def predict(self, samples):  1 usage
        predictions = []
        confidences = []
        for sample in samples:
            distances = np.linalg.norm(self.coords - sample, axis=1)
            nearest_indices = distances.argsort()[:self.k]
            nearest_labels = self.labels[nearest_indices]
            predicted_label = np.bincount(nearest_labels).argmax()

            # Розрахування впевненості як зворотної залежності від відстаней
            nearest_distances = distances[nearest_indices]
            confidence = 1 - (nearest_distances[0] / (np.sum(nearest_distances) + 1e-6))  # Уникнення ділення на 0

            predictions.append(self.int_to_label[predicted_label])
            confidences.append(confidence)
        return predictions, confidences
```

Рисунок 3.1 – Реалізація класу KNN

MediaPipe Hands визначає по 21 ключовій точці для кожної знайденої руки. Точки містять координати x та y в межах кадру. Обробка включає спершу детекцію долоні, а потім – пальців. За допомогою mp.solutions.hands.Hands() та методу hands.process(frame) отримується об’єкт hand_landmarks.

Отримані точки формуються у вектор: [x0, y0, x1, y1, ..., x20, y20]. У разі потреби координати нормалізуються відносно зап’ястя або центра маси руки для підвищення стійкості до положення руки.

Вектор координат передається у створений клас KNN. Алгоритм обирає k найближчих прикладів із навчальної бази за евклідовою відстанню та класифікує вхідний вектор за більшістю збігів. KNN є пасивним алгоритмом: не створює явної моделі, а працює напряму з даними. Дані перед класифікацією завантажуються через метод fit(), а передбачення відбувається через метод predict(). На рисунку 3.2 представлено код, який реалізує ці етапи

```
cap = cv2.VideoCapture(0)
active_label = None  # Активний жест для запису
captured_coords = []  # Тимчасові координати для жесту користувача
recording = False  # Статус запису

while True:
    success, img = cap.read()
    if not success:
        break

    # Віддзеркалення зображення по горизонталі
    img = cv2.flip(img, flipCode: 1)

    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(img_rgb)

    # Обробка виявлення рук та запис координат
    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_draw.draw_landmarks(img, hand_landmarks, mp_hands.HAND_CONNECTIONS)

            # Збираємо 21 ключову точку
            coords = []
            for lm in hand_landmarks.landmark:
                coords.append(lm.x)
                coords.append(lm.y)

            coords = np.array(coords)

            # Якщо активний запис жесту, додаємо координати
            if recording:
                captured_coords.append(coords)

            # Якщо є навчена база, передбачаємо жест
            elif len(data['coords']) > 0:
                labels, confidences = knn.predict([coords])
                label, confidence = labels[0], confidences[0]
                cv2.putText(img, text: f"Gesture: {label} ({confidence * 100:.2f}%)",
                            org: (50, 50), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1, color: (255, 0, 0), thickness: 2)
```

Рисунок 3.2 – Реалізація розпізнавання жестів за допомогою MediaPipe та KNN

Користувач може додавати нові жести: при натисканні клавіші вмикається режим запису, в якому користувач показує новий жест, зчитуються координати його координати, а потім користувач підтверджує запис жесту. Дані додаються до бази та зберігаються у файл .pkl за допомогою бібліотеки Pickle.

Клавіші керування:

– n – додати новий жест до бази;

– a – додати поточний жест до бази;

– s – підтвердити запис жесту;

– w – зберегти базу у файл,

– l – завантажити базу з файлу;

– q – вийти з програми.

Інтерфейс реалізовано з використанням бібліотеки Tkinter. Він дозволяє створювати вікна з кнопками, полями вводу та іншими елементами. У вікні програми може бути показано потік з камери, кнопки керування та поля для введення назв нових жестів. Приклад однієї з таких функцій зображено на рисунку 3.3.

```
if key == ord('n'):
    active_label = simpledialog.askstring( title: "Новий жест",  prompt: "Введіть назву нового жесту:")
    if active_label:
        captured_coords = []
        recording = True
        messagebox.showinfo( title: "Запис жесту",  message: f"Жест '{active_label}' записується. Натисніть 's' для завершення.")
    else:
        messagebox.showwarning( title: "Увага",  message: "Запис нового жесту скасовано.")
```

Рисунок 3.3 – Додавання нового жесту в базу

С загальною блок-схемою роботи програми можна ознайомитись на рисунку 3.4, на ній видно, як програма отримує зображення з камери, обробляє його для виявлення руки, створює вектор координат, визначає жест за допомогою алгоритму k-найближчих сусідів, а також взаємодіє з користувачем через клавіші та вікно програми.

3.2 Програма для розпізнавання жестів

Основа цієї програми майже ідентична минулій, але тут відбувається взаємодія користувача з існуючою базою навчених жестів, де користувач може сам налаштувати кожному жесту своє призначення. В даній програмі призначення жесту обмежено відкриттям файлів з розширенням «.exe», або ярликів «.lnk».

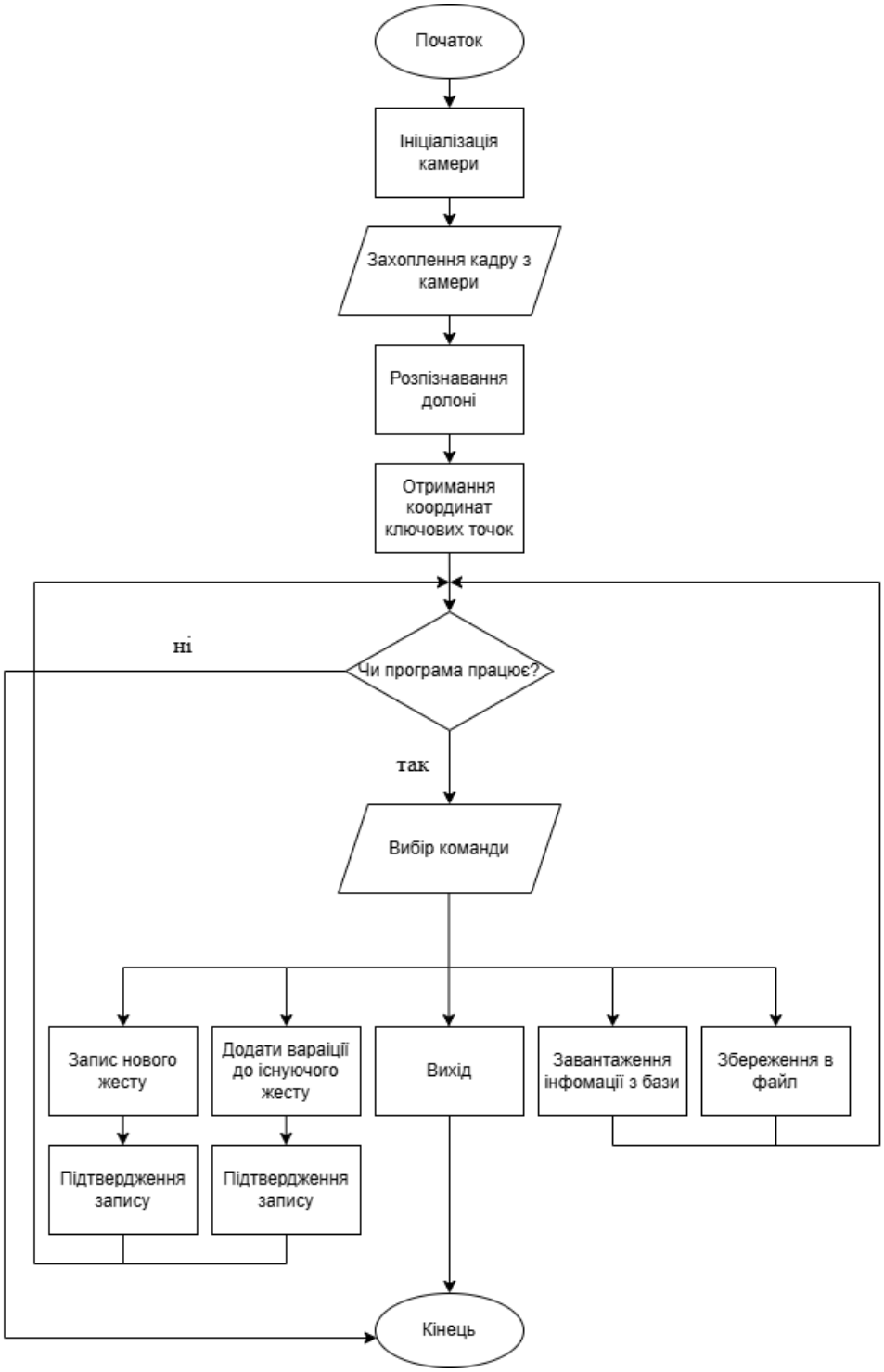


Рисунок 3.4 – Загальний алгоритм роботи програми для навчання розпізнавання жестів

На початку, користувач обирає навчену базу жестів, яку було створено в минулій програмі. Якщо ніякої бази не було вибрано, або виникне помилка з вибором файлу , програма закінчить свою роботу. У випадку коли не відбулось

помилок, далі програма шукає файл конфігурації жестів, якщо його не існує, програма створює в своїй папці пустий файл, в який потім програма збереже налаштування користувача. Якщо файл конфігурації існує, програма використовує його.

Для початку роботи, треба щоб користувач активував режим розпізнавання за допомогою жесту «ОК», його треба тримати протягом 3 секунд, який є в базовій версії бази жестів.

На рисунку 3.5 представлено реалізацію основного функціоналу, який дозволяє користувачу налаштовувати запуск програм за допомогою жестів.

```
# Завантаження конфігурації жестів
def load_gesture_config():  1 usage
    global gesture_actions
    if os.path.exists(CONFIG_FILE):
        with open(CONFIG_FILE, "rb") as config_file:
            gesture_actions = pickle.load(config_file)
            messagebox.showinfo( title: "Конфігурація",  message: "Файл конфігурації жестів завантажено.")
    else:
        gesture_actions = {}
        messagebox.showinfo( title: "Конфігурація",  message: "Файл конфігурації не знайдено. Використовуються порожні налаштування."

# Збереження конфігурації жестів
def save_gesture_config():  1 usage
    with open(CONFIG_FILE, "wb") as config_file:
        pickle.dump(gesture_actions, config_file)
    messagebox.showinfo( title: "Конфігурація",  message: "Конфігурація жестів успішно збережена.")

# Налаштування дії для конкретного жесту
def configure_gesture_action(gesture):  1 usage
    messagebox.showinfo( title: "Налаштування жесту",
                         message: f"Оберіть файл (ярлик або .exe), який буде запускатися для жесту '{gesture}'.")
    file_path = filedialog.askopenfilename(
        title="Оберіть виконуваний файл або ярлик",
        filetypes=(("Виконувані файли", "*.exe;*.lnk"), ("Усі файли", "*.*"))
    )
    if file_path:
        gesture_actions[gesture] = file_path
        save_gesture_config()
        messagebox.showinfo( title: "Налаштування жесту",  message: f"Жест '{gesture}' тепер запускає файл:\n{file_path}")

# Виконання прив'язаної дії
def execute_gesture_action(action):  1 usage
    try:
        subprocess.Popen(action, shell=True)
    except Exception as e:
        messagebox.showerror( title: "Помилка виконання",  message: f"Помилка під час виконання дії: {e}")
```

Рисунок 3.5 – Реалізація налаштування та запуску дій за допомогою розпізнаних жестів

Користувач може керувати програмою через клавіші: c – налаштування жесту;q – вихід з програми.

На рисунку 3.6 зображено алгоритм роботи програми для розпізнавання жестів. Вона демонструє, як завантажується база жестів, перевіряється конфігураційний файл, активується режим розпізнавання за допомогою жесту «ОК», а також як система виконує дію, якщо розпізнано відповідний жест із бази.

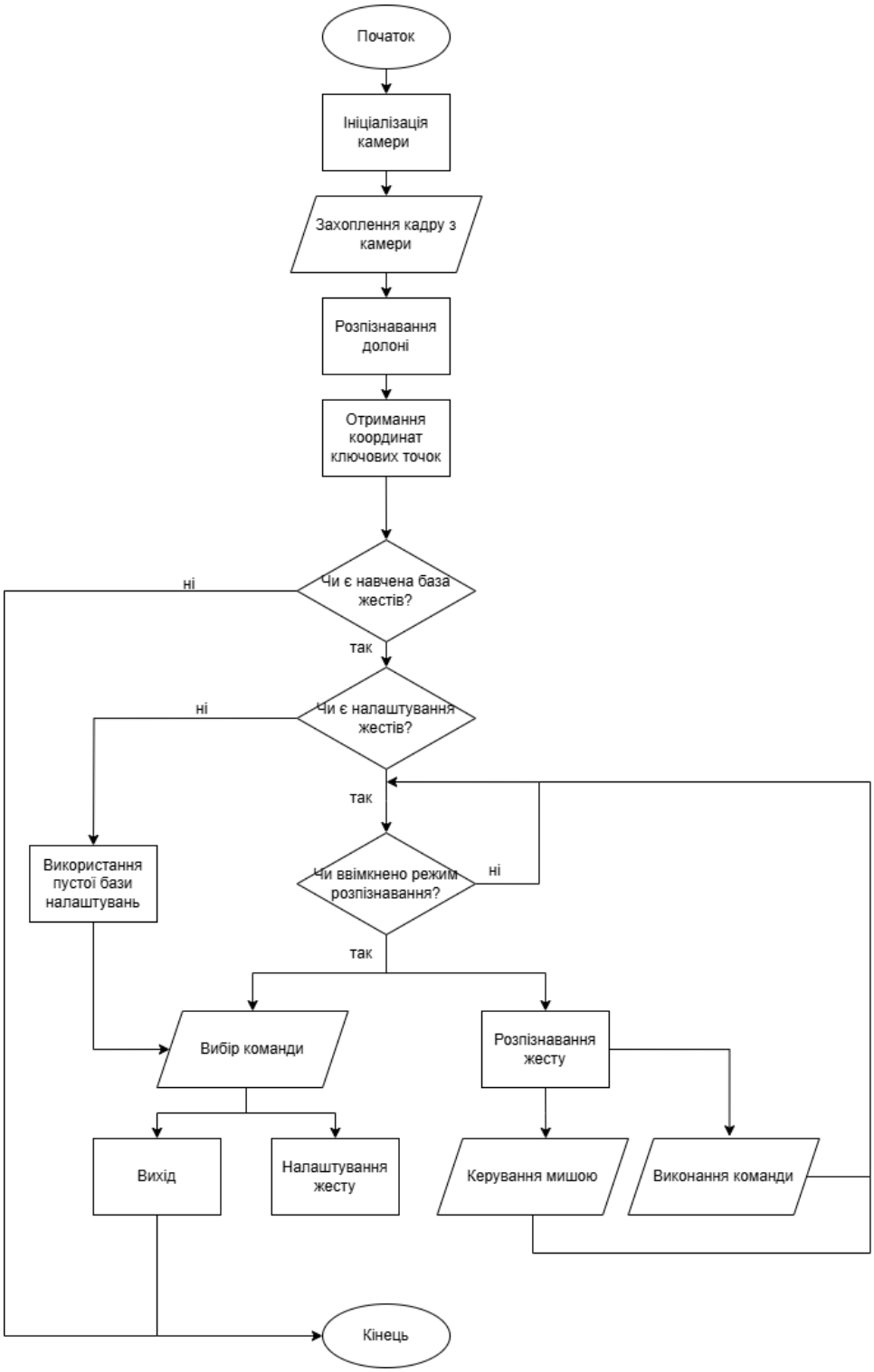


Рисунок 3.6 –Загальний алгоритм роботи програми для навчання розпізнавання жестів

3.3 Основні функції та методи програмного забезпечення

У таблиці 3.1 представлено ключові функції та методи, що реалізовані у програмному забезпеченні для навчання та розпізнавання жестів. Для кожної функції вказано її призначення та файл, де вона використовується.

Таблиця 3.1 – Основні функції та методи програмного забезпечення

Функція/метод	Опис	Файл
`KNN.fit`	Завантажує координати навчальних прикладів та відповідні мітки жестів.	Base.py, Program.py
`KNN.predict`	Повертає передбачену мітку жесту на основі евклідової відстані до відомих прикладів.	Base.py, Program.py
`add_gesture`	Додає координати нового жесту до бази та оновлює модель KNN.	Base.py
`cv2.VideoCapture`	Ініціалізує захоплення відео з вебкамери.	Base.py, Program.py
`mp.solutions.hands`	Визначає ключові точки руки за допомогою бібліотеки MediaPipe.	Base.py, Program.py
`simpledialog.askstring`	Викликає вікно для введення назви нового жесту.	Base.py, Program.py
`messagebox.showinfo`	Виводить інформаційне повідомлення користувачу.	Base.py, Program.py
`pickle.dump`	Зберігає навчальну базу жестів у файл `.pkl`.	Base.py, Program.py

Продовження таблиці 3.1

Функція/метод	Опис	Файл
`pickle.load`	Завантажує базу жестів з файлу `.pkl`.	Base.py, Program.py
`subprocess.Popen`	Запускає виконуваний файл, прив’язаний до конкретного жесту.	Program.py
`configure_gesture_action`	Прив’язує обраний жест до конкретного файлу через діалогове вікно.	Program.py
`execute_gesture_action`	Виконує дію, відповідно до розпізнаного жесту.	Program.py
`hands.process`	Обробляє кадр та повертає координати ключових точок руки.	Base.py, Program.py
`cv2.putText`	Виводить текстову інформацію на відеопотік.	Base.py, Program.py
`cv2.imshow`	Відображає оброблене відео з камери.	Base.py, Program.py

3.4 Оцінка обсягу необхідних даних і ефективності моделі

Система розпізнавання жестів побудована на основі алгоритму K-ближчих сусідів (KNN), який класифікує нові жести, використовуючи схожість із уже збереженими зразками. Важлива особливість такого підходу полягає в його чутливості до обсягу навчальних даних. Тобто кількість записаних варіацій жестів безпосередньо впливає на точність і ефективність системи.

У процесі тестування було вивчено, скільки даних потрібно системі для досягнення прийнятної точності класифікації. Аналіз демонструє, що точність

моделі не зростає разом із кількістю інформації, а натомість демонструє нелінійну залежність, в результаті якої, точність після другого прикладу несуттєво зростає, а після третього падає до значень.

Як видно з таблиці 3.2, програма демонструє приріст точності при збільшенні кількості прикладів із 1 до 3 для кожного жесту. Це свідчить про те, що алгоритм KNN адаптується до базової вибірки даних. Проте подальше збільшення кількості прикладів приносить вже менший приріст точності. Таким чином, спостерігається ефект "насичення", коли внесення додаткових даних не призводить до пропорційного покращення результатів.

Таблиця 3.2 – Точність розпізнавання залежно від кількості прикладів

Кількість прикладів жесту	Точність визначення, %
1	66
2	68
3	71
4	72

Для практичного використання системи було визначено, що оптимальним є запис трьох прикладів кожного жесту. Такий обсяг даних забезпечує достатній рівень точності при порівняно невеликих витратах на навчання системи. Збільшення кількості варіацій (більше 3–5 для кожного жесту) не вважається доцільним, бо з більш ніж 5 варіаціями, точність не зростає.

3.5 Охорона праці

При проєктуванні та впровадженні систем автоматичного керування виробничим обладнанням із застосуванням технологій розпізнавання жестів необхідно особливу увагу приділяти питанням охорони праці. Сучасне

виробниче середовище характеризується наявністю потенційно небезпечних факторів, пов’язаних як із використанням електронного обладнання, так і з людським фактором. В умовах, коли оператор взаємодіє з машиною без фізичного контакту, зростають вимоги до безпечності, точності та надійності керувальних сигналів.

Під час експлуатації автоматизованих систем можуть виникати такі ризики: ураження електричним струмом, зорове перенапруження, накопичене фізичне та емоційне навантаження, а також аварійні ситуації, спричинені хибним розпізнаванням жестів. З метою зменшення або усунення цих ризиків в організації робочого місця оператора реалізовано комплекс технічних та організаційних заходів відповідно до законодавства України, зокрема Закону України «Про охорону праці» [28], ДСТУ. Заходи з охорони праці підготовлено згідно з методичними вказівками ХНУРЕ з охорони праці для бакалаврських робіт [29].

Електробезпека забезпечується шляхом використання сертифікованих пристроїв з подвійною ізоляцією, заземлення корпусів обладнання, застосування стабілізаторів напруги та блоків захисту від короткого замикання. Робоче місце обладнане за принципами ергономіки: монітор розташовується на відстані 60–70 см від очей, клавіатура – на рівні ліктів, а стілець має регульовану висоту та спинку з підтримкою поперекового відділу [29]. Усі кабелі прокладено приховано або за допомогою кабель-каналів, щоб уникнути ризику спотикання.

Усі програмні рішення включають механізми перевірки достовірності жесту та дублювання критичних команд за допомогою підтверджувального сигналу. У разі втрати відеопотоку або визначення невизначеного стану, передача команди на обладнання блокується, а оператор інформується про помилку. Введено спеціальний жест екстреної зупинки, який має вищий пріоритет над усіма іншими командами й реалізується незалежно від поточного стану системи.

З метою збереження здоров’я оператора рекомендовано дотримуватися режиму праці та відпочинку: після кожної години безперервної роботи – 10-15-хвилинна перерва, під час якої слід виконати вправи для очей, рук та спини. Робоче приміщення повинно бути добре освітленим, провітрюваним, обладнаним системою аварійного освітлення та забезпеченим первинними засобами пожежогасіння (вогнегасник типу ВВК-2 або ВП-5). Усі електропристрої мають проходити планову перевірку, технічне обслуговування та випробування згідно з регламентом експлуатації.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи розроблено підсистему автоматичного керування виробничим обладнанням з використанням технології розпізнавання жестів оператора. Такий підхід дозволяє забезпечити безконтактну, інтуїтивно зрозумілу взаємодію з машинами, що актуально в умовах сучасного виробництва.

Проведено аналіз сучасних систем розпізнавання жестів, обґрунтовано вибір технічних засобів та програмного забезпечення. Реалізація здійснена на мові Python з використанням бібліотек OpenCV та MediaPipe. Для класифікації жестів застосовано метод k-найближчих сусідів, що забезпечує просте навчання та достатню точність.

Розроблено програму, яка розпізнає жести та передає відповідні команди, а також дозволяє користувачеві додавати нові жести до бази даних. Проведено моделювання системи керування, визначено її передавальну функцію та характеристичне рівняння. Аналіз показав, що система є стійкою та має аперіодичний перехідний процес тривалістю до 9 секунд.

У роботі враховано вимоги охорони праці: передбачено заходи електробезпеки, ергономіку робочого місця, обробку аварійних ситуацій та психофізіологічні аспекти. Це дозволяє інтегрувати систему у виробниче середовище без ризику для оператора.

Таким чином, поставлену мету досягнуто: створено функціональний прототип системи керування на основі розпізнавання жестів, що може бути основою для подальшого розвитку в напрямку «інтелектуального» виробництва.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. III Goloven O., Mykhailov Ya. AI-Driven Cost Optimization of Office Procurement. Research in Science, Technology and Economics: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference. International Scientific Unity. May 28-30, 2025. Luxembourg, Luxembourg. 129-132 p. URL: https://isu-conference.com/en/archive/research-in-science-technology-and-economics-28-05-25/ (дата звернення: 29.05.2025).

2. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

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

4. Chakraborty B. K., Sarma D., Bhuyan M. K., MacDorman K. F. Review of constraints on vision-based gesture recognition for human–computer interaction // IET Computer Vision. – 2018. – Vol. 12. – P. 3–15. [Електронний ресурс]. – Режим доступу: https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-cvi.2017.0052

5. Intel® RealSense™ Computer Vision - Depth and Tracking cameras [Електронний ресурс]. – Режим доступу: https://www.intelrealsense.com (дата звернення: 29.04.2025).

6. Digital worlds that feel human | Ultraleap [Електронний ресурс]. – Режим доступу: https://www.ultraleap.com (дата звернення: 29.04.2025).

7. MediaPipe Hands [Електронний ресурс]. – Режим доступу: https://mediapipe.readthedocs.io/en/latest/solutions/hands.html (дата звернення: 29.04.2025).

8. BMW Gesture Control: Innovation in Human-Machine Interface [Електронний ресурс]. – Режим доступу:

https://www.bmw.com/en/innovation/gesture-control.html (дата звернення: 01.05.2025).

9. Apple Vision Pro - Apple [Електронний ресурс]. – Режим доступу: https://www.apple.com/apple-vision-pro/ (дата звернення: 01.05.2025).

10. Meta Quest 3: Mixed Reality VR Headset - Shop Now | Meta Store [Електронний ресурс]. – Режим доступу: https://www.meta.com/quest/quest-3/ (дата звернення: 01.05.2025).

11. Soli Radar-Based Perception and Interaction in Pixel 4 [Електронний ресурс]. – Режим доступу: https://atap.google.com/soli/ (дата звернення: 01.05.2025).

12. eXtra Vision [Електронний ресурс]. – Режим доступу: https://www.extrareality.io (дата звернення: 05.05.2025).

13. Ulrich L., Carmassi G., Garelli P., Lo Presti G., Ramondetti G., Marullo G., Innocente C., Vezzetti E. SIGNIFY: Використання машинного навчання та розпізнавання жестів для навчання жестової мови через серйозну гру // Future Internet. – 2024. – Т. 16, № 12. – С. 447. – DOI: 10.3390/fi16120447. – Режим доступу: https://www.mdpi.com/1999-5903/16/12/447 (дата звернення: 13.05.2025).

14. Gesture Recognition Market [Електронний ресурс] // Precedence Research. – Режим доступу: https://www.precedenceresearch.com/gesture-recognition-market (дата звернення: 13.05.2025).

15. Gesture Recognition and Hand Tracking using AI [Електронний ресурс]. – Режим доступу: https://onix-systems.com/blog/hand-tracking-and-gesture-recognition-using-ai (дата звернення: 13.05.2025).

16. OpenCV [Електронний ресурс]. – Режим доступу: https://opencv.org/ (дата звернення: 16.05.2025).

17. NumPy documentation – NumPy v2.2 Manual [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/ (дата звернення: 16.05.2025).

18. Бібліотека Pickle [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/pickle.html (дата звернення: 16.05.2025).

19. Бібліотека OS [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/os.html (дата звернення: 16.05.2025).

20. Бібліотека Tkinter [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/tkinter.html (дата звернення: 16.05.2025).

21. PyCharm: The only Python IDE you need [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/pycharm/ (дата звернення: 18.05.2025).

22. Visual Studio Code - Code Editing. Redefined [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com (дата звернення: 18.05.2025).

23. Project Jupyter | Home [Електронний ресурс]. – Режим доступу: https://jupyter.org (дата звернення: 18.05.2025).

24. Spyder IDE [Електронний ресурс]. – Режим доступу: https://www.spyder-ide.org/ (дата звернення: 18.05.2025).

25. Raspberry Pi 4 Model B [Електронний ресурс]. – Режим доступу: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (дата звернення: 18.05.2025).

26. Jetson Nano | NVIDIA Developer [Електронний ресурс]. – Режим доступу: https://developer.nvidia.com/embedded/jetson-nano (дата звернення: 18.05.2025).

27. Метод k-найближчих сусідів (K-Nearest Neighbor, KNN) [Електронний ресурс] // GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/k-nearest-neighbours/ (дата звернення: 19.05.2025).

28. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII. — Відомості Верховної Ради України. — 1992. — № 49. — Ст. 668.

29. Методичні вказівки до виконання розділу "Охорона праці" у випускних роботах ОКР "бакалавр" усіх форм навчання / упоряд.: В. А. Айвазов. Т. Є. Стиценко., Н. Л. Березуцька ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2018. – 28 с. – 1,81.