Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Перший (бакалаврський)
(рівень вищої освіти)

Розроблення підсистеми підтримки технічних рішень працівника з використанням елементів доповненої реальності
(тема)

Виконала:
здобувачка 4 року навчання,
групи АКТСІ - 21- 3
Діана ЗАЯЦЬ
(власне ім'я, прізвище)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва освітньої програми)
Керівник доц. Артем БРОННІКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

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

2025 р.

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

10 червня 2025 р.  Діана ЗАЯЦЬ

Харківський національний університет радіоелектроніки

Факультет АКТ

Кафедра КІТАР

Рівень вищої освіти перший (бакалаврський)

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

Тип програми Освітньо-професійна

Освітня програма Системна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«19» травня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Заяць Діані Євгенівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення підсистеми підтримки технічних рішень працівника з використанням елементів доповненої реальності
Затверджена наказом університету від 19.05.2025 р. №391 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 24.06.2025 р.
3. Вихідні дані до роботи
3.1 Середовище розробки PyCharm
3.2 Мова програмування Python
3.3 Використання бібліотеки OpenCV
3.4 Використання фреймворку MediaPipe
3.5 Використання бібліотеки CVZone
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз технічного завдання
4.2 Аналіз сучасних інтелектуальних систем
4.3 Вибір та обґрунтування технічних засобів для створення помічника
4.4 Розроблення та реалізація підсистеми підтримки технічних рішень працівника
4.5 Висновки та перелік джерел посилань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Демонстраційний матеріал, представлений у форматі презентації Power Point (*.ppt) 10 с. формату А4

6. Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	28.04-04.05.2025	Виконано
2	Опрацювання літератури за темою роботи	05.05-10.05.2025	Виконано
3	Виконання розділу 1 Аналіз сучасних інтелектуальних систем	11.05-17.05.2025	Виконано
4	Виконання розділу 2 Вибір та обґрунтування технічних засобів для	18.05-25.05.2025	Виконано
5	Виконання розділу 3 Розроблення та реалізація підсистеми підтримки технічних рішень працівника	26.05-02.06.2025	Виконано
6	Висновки та перелік джерел посилань	03.06-04.06.2025	Виконано
7	Оформлення пояснювальної записки	05.06-09.06.2025	Виконано

Дата видачі завдання 28.04.2025

Студент _____ Діана ЗАЯЦЬ
(підпис) (власне ім'я, прізвище)

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

РЕФЕРАТ

Пояснювальна записка: 96 с., 2 табл., 21 рис., 3 дод., 24 джерела.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КОМП`ЮТЕРНИЙ ЗІР, ІДЕНТИФІКАЦІЯ, РОЗПІЗНАВАННЯ, МАШИННЕ НАВЧАННЯ.

Об’єкт розробки – процес керування роботою працівника приладобудівного виробництва.

Предмет розробки – підсистема підтримки прийняття рішень працівника на основі розпізнавання жестів.

Мета кваліфікаційної роботи – розробити підсистему підтримки технічних рішень працівника з використанням елементів комп'ютерного зору та доповненої реальності.

В кваліфікаційній роботі розглянуто актуальні питання підтримки працівників на сучасному виробництві, запропоновано рішення для підвищення ефективності їхньої роботи за допомогою інтелектуальних систем. У результаті розроблена підсистема підтримки прийняття рішень працівника, яка функціонує на основі розпізнавання жестів та інтеграції зі штучним інтелектом.

За результатами роботи опубліковано статтю у збірнику студентських наукових статей та тези доповіді у збірнику міжнародної конференції.

ABSTRACT

Explanatory note: 96 pages, 2 tables, 21 figures, 3 appendices, 24 references.

SOFTWARE, COMPUTER VISION, IDENTIFICATION, RECOGNITION, MACHINE LEARNING.

The object of development is the process of managing an operator's workflow in instrumentation manufacturing.

The subject of development is an operator decision support subsystem based on gesture recognition.

The purpose of the qualification work is to develop a technical decision support subsystem for an employee, utilizing elements of computer vision and augmented reality.

This qualification thesis addresses topical issues in employee support within modern manufacturing and proposes a solution for enhancing operational efficiency through intelligent systems. As a result of this research, a decision support subsystem for employees has been developed.This system functions on the basis of gesture recognition and integration with artificial intelligence.

The findings of this work have been disseminated through two primary channels: an article published in a student research journal and a conference abstract in international proceedings.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІТ – інформаційні технології;

СППР – системи підтримки прийняття рішень;

ШІ – штучний інтелект;

AR – Augmented reality – доповнена реальність;

CPPS – Cyber-Physical Production System – кіберфізичні виробничі системи;

GPU – Graphics processing unit – графічний процесор;

IBM – International Business Machines – міжнародні бізнес-машини;

IDE – Integrated development environment – інтегроване середовище розробки;

IoT – Internet of Things - інтернет речей;

KNN – K-Nearest Neighbors – к-найближчі сусіди;

OpenCV – Open Source Computer Vision – бібліотека комп'ютерного зору з відкритим кодом;

PIL – Python Imaging Library – бібліотека зображень Python;

SVMs – Support Vector Machines – машина опорних векторів.

ВСТУП

Сучасна промисловість переходить до нових підходів, серед яких посідає концепція Індустрії5.0. Вона робить акцентна людиноцентричному підході, стійкості та гнучкості виробництва. На відміну від Індустрії 4.0, де основну увагу приділяли автоматизації та цифровим технологіям, нова парадигма орієнтована на ефективну взаємодію людини й машини. Технології використовуються як інструмент підтримки й підсилення можливостей працівника, а не як його заміна. Це змінює підхід до організації праці, зокрема, зростає потреба в наданні інтелектуальної допомоги та оперативної підтримки безпосередньо на робочому місці.

У відповідь на нові вимоги, спостерігається активне впровадження інтелектуальних систем, які допомагають працівникам у роботі. Світові розробки зосереджуються на створенні цифрових асистентів на основі штучного інтелекту, сучасних систем підтримки прийняття рішень, а також технологій, що дозволяють людині взаємодіяти з машиною максимально інтуїтивно. Такі рішення допомагають автоматизувати рутинні завдання, швидко обробляти великі масиви даних і сприяти прийняттю обґрунтованих рішень. Окрема увага приділяється розробці адаптивних систем, які можуть підлаштовуватися під динаміку виробничих умов та індивідуальні потреби користувача.

Серед найбільш перспективних технологій, що лягають в основу людиноорієнтованих систем, є комп'ютерний зір та доповнена реальність. Завдяки комп’ютерному зору машина може «бачити» навколишній простір і розпізнавати об’єкти, жести або дії, що надає можливість аналізувати робочі процеси в реальному часі. А доповнена реальність, у свою чергу, дозволяє накладати цифрові елементи на реальне середовище, що дозволяє працівнику отримувати підказки, інструкції або візуалізації просто під час виконання

завдань. Поєднання цих технологій відкриває великі можливості для створення сучасних інструментів підтримки рішень.

Актуальність теми визначається зростаючою потребою підприємств у підвищенні продуктивності персоналу та зниженні кількості помилок під час виконання складних технічних операцій. Створення підсистеми підтримки прийняття рішень працівника на основі розпізнавання жестів є важливим кроком до формування більш гнучких, безпечних і зручних умов праці. Завдяки інтеграції інтуїтивних способів взаємодії, таких як розпізнавання жестів із можливостями штучного інтелекту, вдається отримати інструмент, що допомагає не лише оптимізувати виробничі процеси, а й сприяє професійному зростанню працівників.

Метою бакалаврської роботи є розробка підсистеми підтримки технічних рішень працівника з використанням елементів комп'ютерного зору та доповненої реальності.

Об’єкт розробки – процес керування роботою працівника приладобудівного виробництва.

Предмет розробки – підсистема підтримки прийняття рішень працівника на основі розпізнавання жестів.

Задачі, які потрібно вирішити у бакалаврській роботі:

– провести аналіз сучасних інтелектуальних систем;

– провести вибір та обґрунтування технічних засобів для створення помічника;

– розробити інтелектуальної системи допомоги працівнику приладобудівного виробництва;

– розглянути питання охорони праці;

– оформити пояснювальну записку згідно [1] та [2].

Проведена робота відповідає цілям сталого розвитку (ЦСР): ЦСР 4, ЦСР 9 та ЦСР 12.

1 АНАЛІЗ СУЧАСНИХ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

1.1 Індустрія 5.0 та її вплив на сучасне виробництво

Індустрія 5.0 – це концепція, яка є доповненням Індустрії 4.0, але з більшою увагою до впливу промисловості на суспільство, а не на підвищення ефективності підприємства. Вона засновується на наукових дослідженнях та інноваційних підходах, щоб перейти до більш стійкої та людиноорієнтованої промисловості (рис. 1.1). Відмінність Індустрії 5.0 від Індустрії 4.0 виражається в тому, що промисловість тепер орієнтована не тільки на отриманні прибутку, але й на створення цінності для всіх, хто з нею пов'язаний, що збільшує її соціальну роль [3].

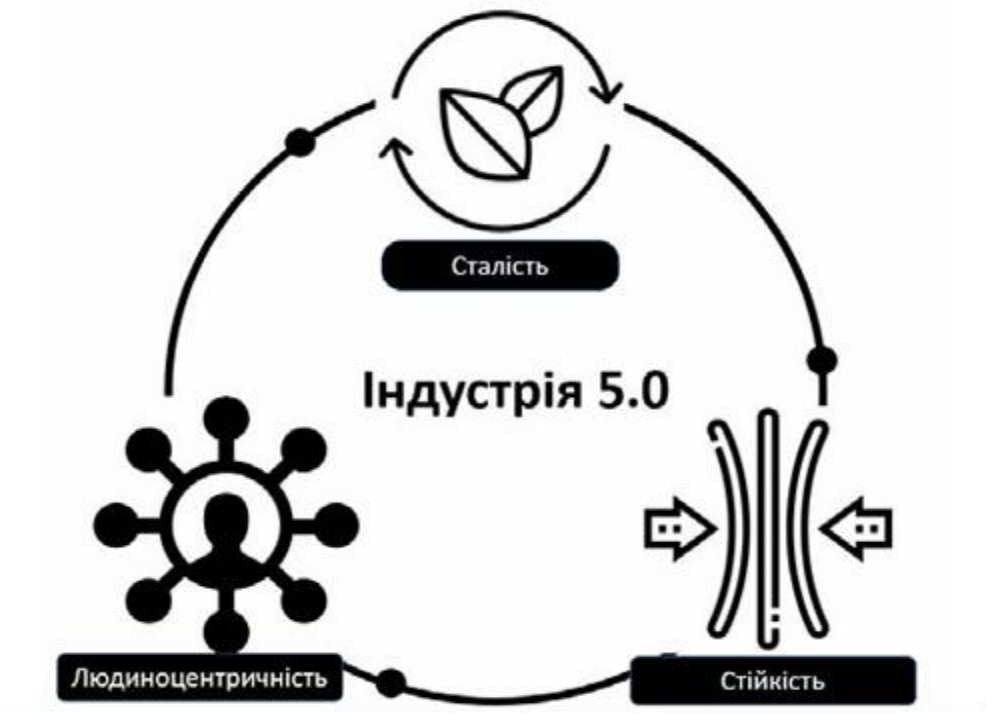

Рисунок 1.1 – Основні принципи Індустрії 5.0 [3]

Основною ідеєю Індустрії 5.0 вважається розвинення сучасного виробничого процесу таким чином, щоб людина та машини могли ефективно співпрацювати як партнери, об'єднуючи розумові здібності людини з технічною точністю роботів. Головна ціллю є краще задовольняти потреби клієнтів та відкриття нових ринків, створюючи продукти, які максимально підходять до вимог споживачів.

Якщо порівнювати Індустрію 4.0, де основний акцент є на автоматизації та оцифруванні, та Індустрію 5.0, то Індустрія 5.0 приділяє багато уваги тому, як людина інтегрується в технологічні процеси. Ключовим є співробітництво між людиною та машиною і розробка рішень, орієнтованих на потреби людини. Це дозволяє, як і раніше, зробити людську складову значущою у виробництві, роблячи можливим створювати продукти під конкретні потреби та роблячи виробництво більш гнучким [3].

На основі цієї ідеї, роботи-помічники, або як ще їх називають – коботи, будуть виконувати прості та одноманітні завдання, а з іншого боку люди зможуть приділити більше уваги на більш складні та творчі завданнях, а також на прийнятті рішень. Коботи мають різні вбудованні датчики та передові технології, що в свою чергу робить їх безпечними та ефективними при спільній роботі з людьми, а також підвищує точність, швидкість і загальну якість виробництва. На рис. 1.2 можна побачити приклад взаємодії людини та кобота.

Рисунок 1.2 – Приклад кобота на виробництві [4]

В Японії цю ідею називали "Суспільство 5.0", вважаючи її справжньою революцією, де головне – людина, яка в свою чергу допомагає знайти баланс між економічним зростанням та розв'язанням важливих соціальних проблем. Це означає, що ми поступово відходимо від стандартного робочого дня з 9:00 до 18:00, до більш гнучких і творчих видів діяльності, що показує зміни в тому, як люди хочуть працювати, та в тому, як працює бізнес [5].

Також, важливий момент що, Індустрія 5.0 не хоче замінити людей роботами, а хоче лише допомогти їм працювати краще. Вона сприяє тому, щоб між людьми та штучним інтелектом була співпраця, де штучний інтелект є тільки помічником, але не забирає у людей можливість творити та самостійно приймати рішення. Як результат, це призведе до появи нових робочих місць для людей, які вміють критично мислити, та для спеціалістів зі штучним інтелектом. Індустрія 5.0 – це про постійні зміни, які допомогають компаніям бути краще підготовленими до нових проблем, бути більш стабільними та мати змогу конкурувати.

1.2 Сучасний стан розвитку інтелектуальних систем підтримки працівників

Інформаційні технології (ІТ) постійно рухаються вперед, пройшовши через декілька значних етапів, як на практиці, так і в теорії. Цей процес показує, як змінювалось використання технологій, їхня роль та вплив на сучасне суспільство:

– виникнення обчислювальних машин (1960 – 1980);
– ера персональних комп’ютерів (1980 – 2000);
– розвиток інтернету й мереж (1990 – 2010);
– масове використання мобільних технологій та IoT (2010 – сьогодні);
– швидкий розвиток штучного інтелекту та машинного навчання (2020 – сьогодні) [6].

Штучний інтелект (ШІ) – це розділ комп'ютерних наук, де створюють програми і системи, які можуть «думати» та виконувати різноманітні завдання, які зазвичай підвладні лише людям. Основні компоненти ШІ це: машинне навчання, розуміння природної мови та автоматизовані механізми. Саме ці технології дозволяють штучному інтелекту виконувати задачі за заданими правилами, що раніше здавалося можливим лише для людини.

Об'єднання інформаційних технологій та штучного інтелекту повністю змінило підхід до управління персоналом, що значно підвищило його ефективність. Завдяки ним тепер можна автоматизувати рутинні процеси та аналізувати великі обсяги даних для прийняття важливих рішень. Це, в свою чергу, допомагає точніше керувати розвитком та утриманням працівників у компанії. Крім того відкриваються можливості для навчання кожного працівника окремо і гнучкі умови для їхнього професійного зростання.

Штучний інтелект може стати незамінним помічником для підбору персоналу. Він може допомогти на багатьох етапах: від відбору кандидатів до оцінки їхньої ефективності, від коригування завдань до рекомендацій курсів підвищення кваліфікації, судячи з їх досягнень та якість роботи. Використання ШІ зменшує вплив особистої думки при взаємодії з потенційними працівниками, оскільки оцінка базується виключно на об'єктивних даних. Але, варто розуміти, що штучний інтелект має й певні недоліки, наприклад система може іноді робити помилки в простих або очевидних речах чи функціонувати не так, як очікується. Тому керівникам необхідно постійно перевіряти її роботу [6].

Хоча цифрові помічники на базі штучного інтелекту відкривають великі та значні перспективи, але вони також можуть і створювати проблеми. Очікується, що вони візьмуть на себе рутинні, прості та одноманітні завдання, звільняючи при цьому час і ресурси для більш важливих завдань. Зокрема, за даними IBM (International Business Machines), чат-боти можуть зменшити витрати на клієнтський сервіс до 30%. Однак важливо пам’ятати, що такі технології це не

універсальне рішення. Вони можуть створювати нові проблеми, якщо їх використовувати бездумно.

Вважається, що зовсім скоро цифрові помічники, які працюють на ШІ, стануть невід'ємною частиною нашого робочого життя. Вже сьогодні стають популярні бізнес-платформи спілкування в компаніях, наприклад, Slack або Microsoft Teams. Вони активно використовують багато різних ботів – від простих помічників до складних, які спрощують повсякденні задачі. Як виглядає інтеграція таких інструментів, як чат-боти, у бізнес-платформи Microsoft Teams показано на рис. 1.3.

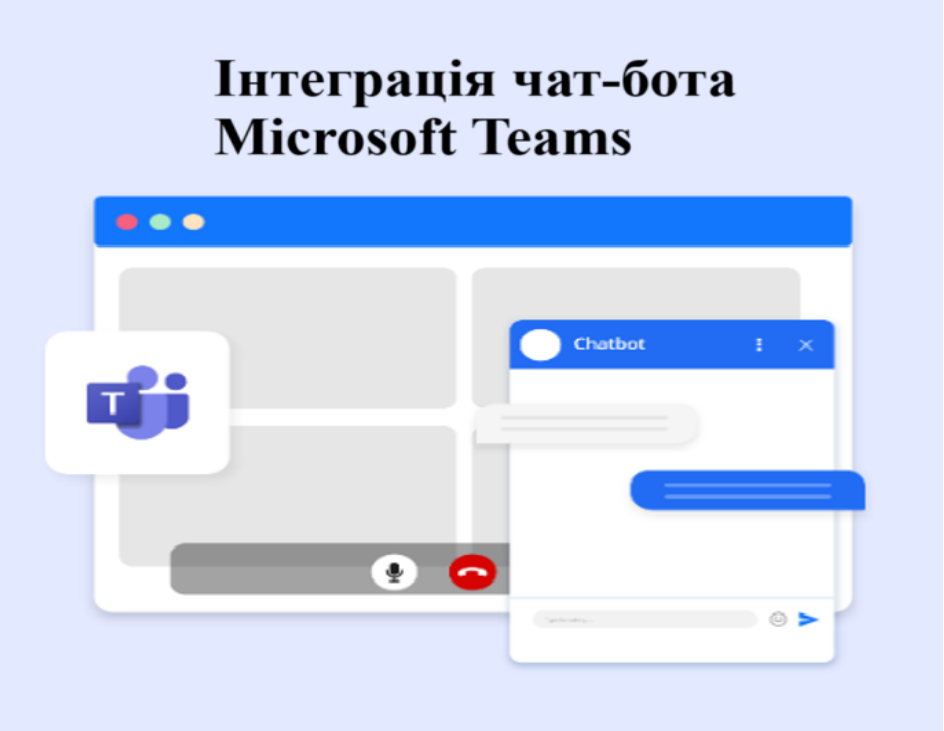

Рисунок 1.3 – Цифрові помічники в бізнес-середовищі [7]

Сучасні цифрові помічники зі ШІ сьогодні працюють набагато ефективніше та краще ніж їхні попередники або звичайні програми. Вони розумніші, швидше взаємодіють із людьми і допомагають виконувати завдання продуктивніше. У майбутньому, з розвитком технологій ШІ, коли технології

стануть ще досконалішими, вони зможуть вирішувати складніші проблеми. Це звісно принесе не лише користь, але й певні ризики для людей. Взаємодія між людиною та штучним інтелектом може відбуватися по-різному: іноді ШІ повністю замінює людей у певних справах, іноді лише доповнює їхні можливості, а буває, що людина й алгоритм працюють разом як єдине ціле, постійно підлаштовуючись один до одного.

Використання цифрових помічників є найбільш ефективним у форматі взаємовигідної співпраці. І люди, і машини мають свої переваги: машини чудово справляються з рутинними та чітко визначеними завданнями, тоді як люди краще справляються зі складними і абстрактними проблемами та вміють мислити творчо. Коли люди та машини працюють разом та ефективно доповнюють сильні сторони один одного, рішення на основі ШІ можуть принести найбільшу користь [8].

Завдяки кіберфізичним виробничим системам (CPPS), що застосовують ШІ – системи для допомоги працівникам, можна контролювати фізичні процеси виробництва. Це не лише оптимізує витрати та підвищує ефективність, а й робить виробничі лінії економічно вигіднішими. У майбутньому на виробництві буде багато різних систем допомоги працівникам, які будуть поєднувати такі технології, як колаборативна робототехніка, голосовий зв'язок, доповнена реальність та природні інтерфейси між людиною та машиною зі штучним інтелектом. Все це перетворить звичайні робочі місця на справжні командні центри, де люди і машини працюватимуть разом.

У лабораторії Smart Mini Factory в Unibz з 2019 року розробляють робочу станцію для спільного складання, яка є прикладом розумної кіберфізичної виробничої системи та активно використовує штучний інтелект. На рис 1.4 продемонстровано як виглядає ця робоча станція. Людина і робот разом збирають деталі, а штучний інтелект допомагає їм працювати безпечно. Камери відстежують рухи оператора, і якщо він підходить занадто близько, робот

сповільнюється або повністю зупиняється. Так технологія запобігає травмам, але при цьому не заважає спільній роботі [9].

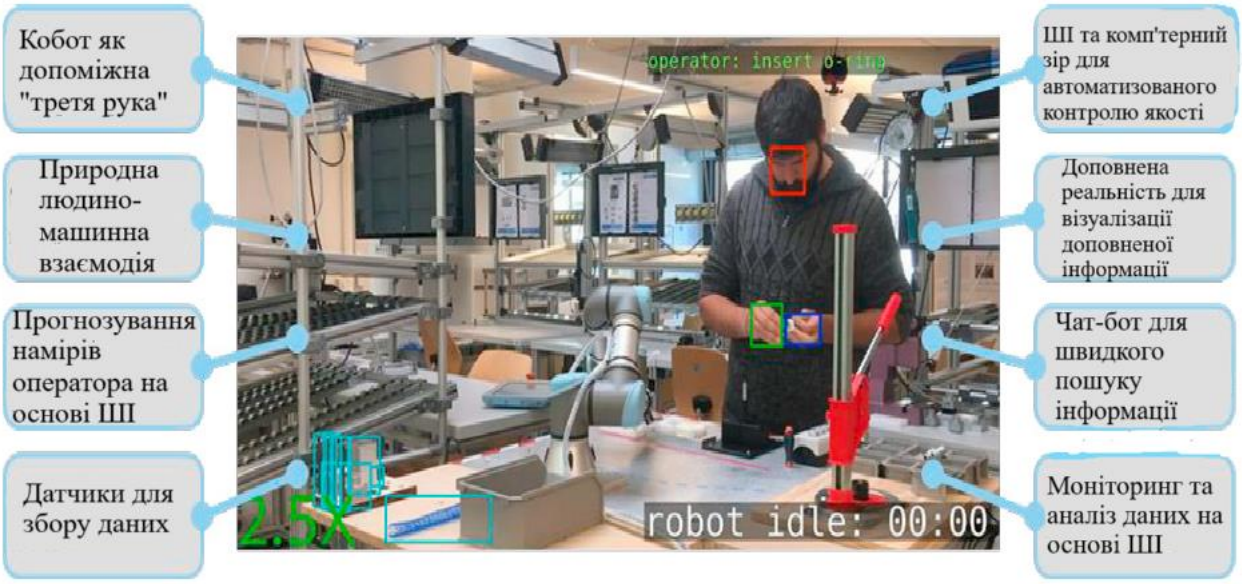

Рисунок 1.4 – Роботизована система допомоги працівникам на виробництві на основі ШІ [9]

1.3 Системи підтримки прийняття рішень

Системи підтримки прийняття рішень (СППР), які розвиваються з 60-70-х років минулого століття, стали невід’ємною частиною сучасних ІТ-рішень. Спочатку це були досить прості автоматизовані системи, але потім вони перетворилися на більш складні інструменти. Ці системи вже не просто зберігають інформацію, а й активно допомагають керівникам у процесі прийняття рішень Сьогодні СППР здатні оцінювати різні варіанти, використовувати досвід користувачів, застосовувати математичні моделі та бази даних, що суттєво покращує якість управління [10].

Прийняття рішень – це важливий процес, коли людина розглядає різні варіанти, оцінює ризики та можливі наслідки. Залежно від рівня визначеності,

рішення поділяють на три типи: структуровані (чіткі, з однозначним алгоритмом), неструктуровані (суб'єктивні, без чітких критеріїв), напівструктуровані (поширені в бізнес-середовищі, де поєднуються об'єктивні дані та експертна оцінка) [11].

Найчастіше в управлінні стикаються саме з напівструктурованими рішеннями, які вимагають як аналітики, так і професійного досвіду. Щоб краще зрозуміти природу напівструктурованих завдач, доцільно порівняти характеристики двох крайніх типів рішень. Основні особливості та відмінності між структурованими та неструктурованими рішеннями узагальнено в табл. 1.1.

Таблиця 1.1 – Особливості слабо структурованих рішень [10]

Структуровані рішення	Неструктуровані рішення
Регулярні, повторювані рішення	Ненадійні, рідко повторювані рішення
Засновані на стабільних умовах	Засновані на випадкових і умовах і даних
Альтернативи чіткі	Альтернативи неясні
Наслідки альтернатив визначені	Наслідки альтернатив невизначені
Критерії вибору чітко визначені	Критерії вибору неоднозначно визначені
Спеціальні знання відомі	Спеціальні знання необхідні для прийняття рішень невідомі
Необхідне знання легко доступні	Необхідне знання недоступно
Заздалегідь відомо повний набір кроків, для досягнення рішення	В процесі прийняття рішень застосовуються не стандартне мислення, мозковий штурм, синтез, аналогія.
Опора на традиції	Опора на розвідку, творчість, прозріння, винахід ливість

Одну з перших зрозумілих моделей процесу прийняття рішень – модель очікуваної корисності – представив Леонард Севідж. Суть її в тому, що рішення залежить від того, наскільки ймовірно станеться якась подія, і від того, наскільки

корисним буде результат. Ця модель добре працює, коли ми знаємо все про ситуацію, але її критикують за надмірну ідеалізацію.

Герберт Саймон запропонував більш практичну модель, яка називається bounded rationality (обмежена раціональність). Вона враховує, що ми не завжди маємо всю інформацію, час і можливості думати ідеально. Модель Саймона включає чотири ключові етапи:

– інтелект – виявлення проблеми;
– проектування – пошук варіантів;
– вибір – обрання найкращого варіанту;
– реалізація – втілення рішення [11].

Зазвичай система підтримки прийняття рішень складається з чотирьох ключових частин: лінгвістична система, система подання даних та інформації, система обробки задач, системи обробки задач конкретних предметних областей та системи знань.

Усі ці елементи пов'язані між собою та з користувачем, допомагаючи аналізувати проблеми, шукати рішення та оцінювати їх ефективність. Ця взаємопов'язана структура дозволяє розробляти інструменти різного рівня складності для широкого спектра завдань.

Основні концепції архітектури систем підтримки прийняття рішень (СППР) з'явилися ще у у 70-80-х роках XX століття. Пізніше ці ідеї стали більш конкретними та були розвинуті в кількох різних напрямках. Створенні інформаційні системи можна класифікувати наступним чином: текстові СППР, гіпертекстові СППР, орієнтовані на використання баз даних та сховищ даних СППР, табличні СППР, орієнтовані на моделі СППР, СППР, які використовують штучний інтелект, гібридні СППР, групові СППР [10].

У сучасному світі, де інформаційні технології розвиваються шаленими темпами, СППР отримують все більшого значення у економіці, технічних галузях і навіть соціальній сфері. Їхня необхідність виникла через те, що приймати

рішення в умовах сьогодення стало вкрай складно, неясно і залежить від багатьох факторів. Такі системи не просто допомагають опрацьовувати величезні масиви даних, а й дають змогу приймати чіткі, логічні та науково підтверджені рішення.

1.4 Штучний інтелект та інтелектуальні помічники

Штучний інтелект – це наука, що займається створенням автоматизованих інтелектуальних систем. Наразі найбільш перспективними напрямками у цій сфері є нейронні мережі, еволюційні обчислення та експертні системи. Наприклад, нейронні мережі вже зараз успішно використовуються для прогнозування ринків, аналізу кібератак, захисту даних та технічної діагностики. Розвиток інтелектуальних експертних систем і нейронних мереж – це тільки початок створення потужного ШІ.

ШІ використовується в найрізноманітніших сферах: розпізнавання мовлення, обробка зображень, інтелектуальний аналіз даних, медична діагностика, автоматизоване керування, робототехніка, системи підтримки прийняття рішень (рис. 1.5).

Рисунок 1.5 – Приклади застосування комп'ютерного зору [12]

Нейронні мережі – це основа штучного інтелекту. Їхня робота багато в чому нагадує людський мозок, адже вони вміють вчитися на прикладах і знаходити закономірності. Зараз їх використовують майже скрізь: від банків і лікарень до транспорту, зв'язку, енергетики та навіть оборони [13].

Інтелектуальні персональні помічники – це програми на базі штучного інтелекту, які розуміють людську мову і можуть виконувати безліч завдань. До них відносяться, наприклад, Siri, Google Асистент, Alexa та інші. Вони допомагають із пошуком інформації, керуванням розкладом, відправкою повідомлень і навіть керуванням пристроями, і це все за допомогою голосових команд [14].

Інтелектуальні помічники стають все кращими завдяки таким технологіям, як інтернет речей (IoT), хмарні сервіси та глибинне навчання. Це дозволяє їм отримувати дані з датчиків, ефективно зберігати та аналізувати дані в хмарі, а також взаємодіяти з іншими пристроями. Ці системи постійно вдосконалюються: краще розуміють контекст, імітують людську поведінку (антропоморфізм), розширюють способи комунікації та навіть можуть самостійно приймати рішення.

Вони також здатні пристосовуватися до ситуації завдяки контекстуальній обізнаності, наприклад: враховуючи ваше місцезнаходження, час доби та звички. Антропоморфізм – це коли пристрій може імітувати людські емоції, мову, вираз обличчя, що змушує нас більше йому довіряти, а ще робить спілкування природнішим. А мультимодальність дає можливість взаємодіяти з ними найзручнішим способом – голосом, текстом, жестами чи зображеннями [15].

Отже, поєднання ШІ та розумних помічників відкриває неймовірні перспективи в найрізноманітніших сферах – від медицини й навчання до повсякденного життя та бізнесу. Проте для повноцінного впровадження таких систем необхідно подолати певні технологічні, етичні та правові перешкоди.

1.5 Комп'ютерний зір та його використання

Комп'ютерний зір – це розділ ШІ, який дає машинам здатність "бачити" та розуміти світ навколо. Він може аналізувати зображення та відео в режимі реального часу, розпізнаючи не тільки об'єкти, а й їхні взаємозв'язки та контекст. Завдяки сучасним нейромережам і глибокому навчанню, ці системи вже успішно справляються зі складними задачами: від відстеження руху до детального аналізу зображень і виявлення важливих деталей.

На сьогоднішній день комп'ютерний зір застосовується в багатьох галузях. У наукових дослідженнях це обробка зображень з високоточних приладів, у медицині – виявлення проблем на основі МРТ, КТ, УЗД. У військовій сфері вони допомагають аналізувати зображення з безпілотників у складних ситуаціях. Навіть в екології та сільському господарстві вони застосовуються для супутникового контролю стану довкілля та ґрунтів. А ще його активно використовують у віртуальній та доповненій реальності (рис.1.6) [13].

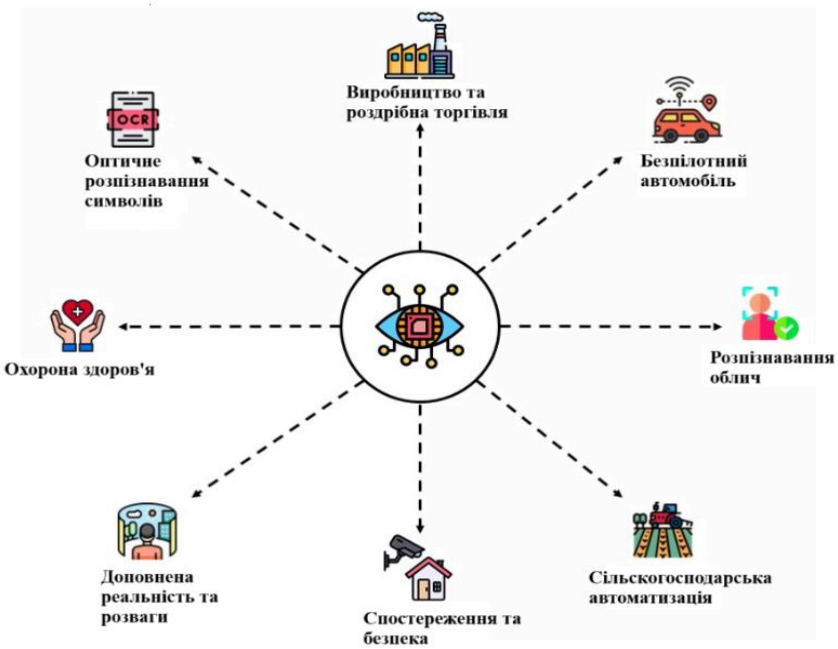

Рисунок 1.6 – Приклади застосування комп'ютерного зору [16]

Для навчання моделей у комп'ютерному зорі застосовують безліч різноманітних підходів: кероване та некероване навчання, напівкероване, з підкріпленням і трансферне навчання. Багато уваги приділяється тому, як краще будувати нейронні мережі, щоб вони були точнішими. Але дослідники стикаються з такими проблемами, як погана якість зображень, недостатньо розмічених даних, нерівномірність вибірок і велика потреба в обчислювальних ресурсах [17].

Раніше комп’ютери використовувались лише виключно для обчислень, але зараз все інакше. Сучасні пристрої з камерами та датчиками щомиті фіксують великі обсяги даних. Технології комп'ютерного зору дозволяють не просто записувати відео, а й автоматично ідентифікувати в ньому об'єкти чи людей, а також відстежувати їхні переміщення. Це суттєво спрощує роботу, наприклад, у сфері безпеки: тепер системи можуть самостійно виявляти щось незвичне або навіть розпізнавати конкретні обличчя.

Комп'ютерний зір – це також наукова дисципліна, яка розробляє моделі для отримання важливої інформації з зображень. Для цього вона об'єднує різноманітні підходи, такі як геометрія, статистика, машинне навчання та фізика.

Загалом, сьогодні комп'ютерний зір – це не просто технологія, а й частина того, як ми зараз бачимо та сприймаємо світ. Людина сприймає світ через "механічні очі" – цифрові датчики, які формують нову реальність. Ця технологія має величезні перспективи для подальшого вивчення та використання у всіх сферах життя [18].

2 ВИБІР ТА ОБҐРУНТУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ПІДСИСТЕМИ ПІДТРИМКИ ТЕХНІЧНИХ РІШЕНЬ

2.1 Вибір технічних засобів

Під час створення програмного забезпечення, яке поєднує розпізнавання жестів у режимі реального часу та генеративний штучний інтелект, основним моментом став вибір технічних засобів. Головними вимогами були швидкодія системи, точність обробки відео, а також зручність у розробці і тестування.

Для розробки та перевірки роботи програми використовувався ноутбук Acer Swift 3 зі стандартною вбудованою вебкамерою. Параметри пристрою:

– процесор: AMD Ryzen 5 – надає необхідні обчислювальні потужності для роботи з відео та бібліотеками машинного навчання;

– відеокарта: AMD Radeon (інтегрована) – використовується для обробки графіки у реальному часі;

– оперативна пам’ять: 8 ГБ – достатньо для стабільної роботи IDE та виконання основних обчислень;

– операційна система: Windows 11;

– камера: вбудована HD-камера (720p) – використовується для захоплення відео та виявлення руки користувача.

Завдяки обраній конфігурації ми можемо ефективно обробляти відео в реальному часі, гарантуючи високу продуктивність програми без потреби в додатковому обладнанні.

Для реалізації проєкту використовувалась мова програмування Python. Вона підходить для розробки систем комп'ютерного зору та інтеграції зі штучним інтелектом. Також середовище розробки (IDE) PyCharm Community Edition.

PyCharm зручне середовище для написання, тестування та налагодження коду. Та використовувались такі бібліотеки та модулі:

– OpenCV – для захоплення та обробки відеопотоку;

– CVZone – для спрощення роботи з виявленням жестів;

– MediaPipe (через CVZone) – для точного визначення ключових точок на руці;

– NumPy – для створення полотна та обробки зображень як масивів;

– Streamlit – для створення вебінтерфейсу;

– Pillow (PIL) – для конвертації полотна перед надсиланням до ШІ;

– Google generativeai – для надсилання запитів до моделі Gemini та отримання відповіді;

– requests – для надсилання HTTP-запитів. Вона потрібна для взаємодії з API Telegram, щоб відправляти повідомлення та фото у чат;

– csv та datetime – використовуються разом для ведення журналу подій (логування);

– winsound – для відтворення простих системних звуків у Windows.

Завдяки вибраним технічним засобам вдалося реалізувати роботу з розпізнаванням жестів у реальному часі, досягти високої точності та швидкості обробки відео, а також легко інтегрувати систему з генеративним ШІ від Google. Це суттєво полегшило роботу на всіх етапах розробки.

2.2 Функціональна схема роботи програми

Розроблена програма складається з кількох модулів, які між собою пов’язані. Кожен модуль відповідає за певну частину загального процесу роботи програми. Завдяки функціональній схемі можна побачити, як відбувається обробка даних, як працює система всередині та яким чином дані передаються між

її компонентами. На рис. 2.1 представлена функціональна схема роботи програми.

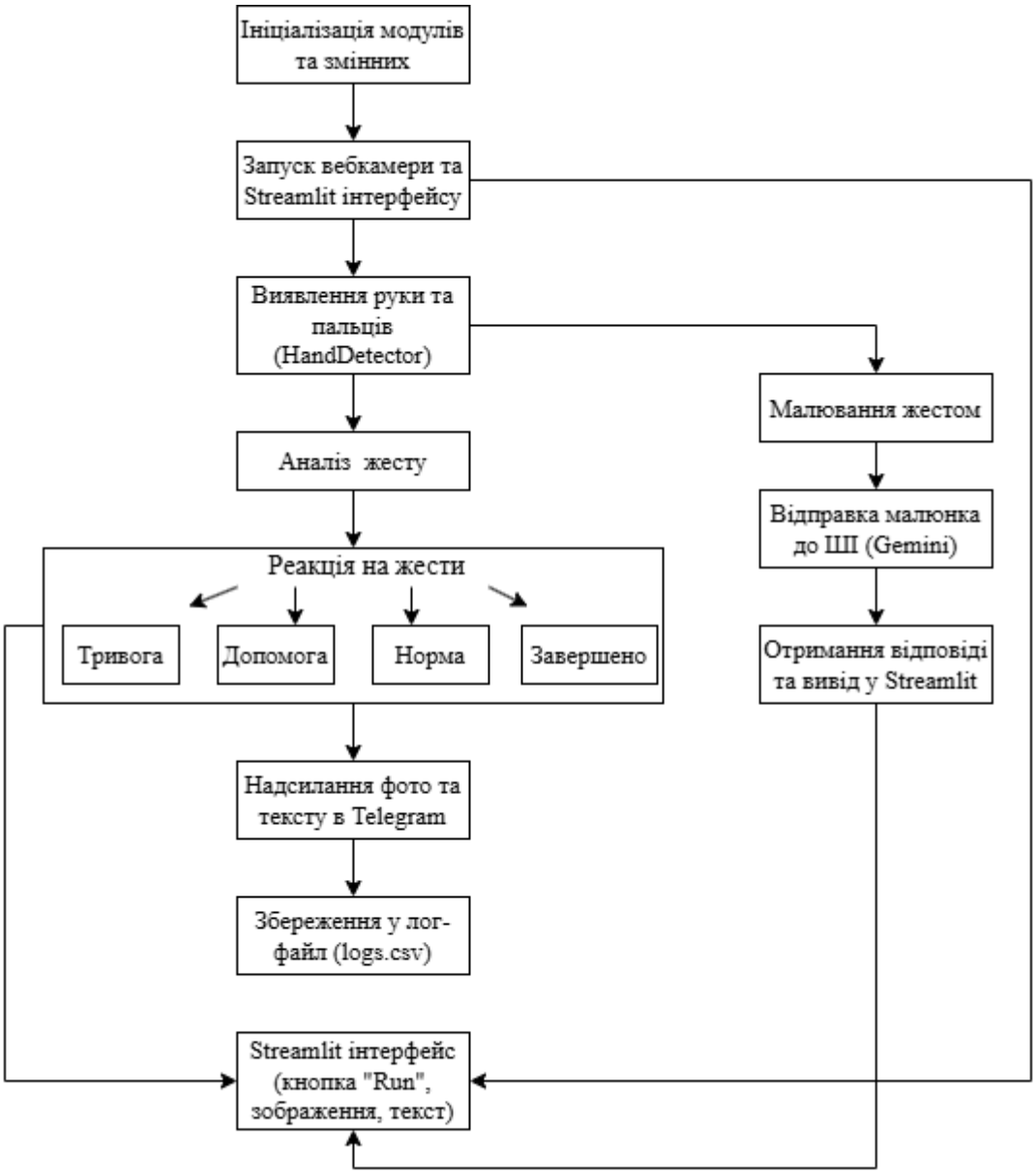

Рисунок 2.1 − Функціональна схема роботи програми

Функціональна схема показує, як працює програма, яка реагує на жести руки через вебкамеру з використанням комп’ютерного зору, штучного інтелекту та інтеграції з Telegram. Вона допомагає зрозуміти, які кроки виконує програма

– від початку роботи до того моменту, коли з’являється результат або відправляється повідомлення.

Програма починається з блоку ініціалізації, де відбувається підключення бібліотек, налаштування змінних, модулів виявлення руки та клієнта Gemini. Далі запускається вебкамера, створюється інтерфейс на Streamlit, і зображення з камери виводиться на екран. Наступний етап – це виявлення руки та визначення розташування пальців за допомогою модуля HandDetector.

Ключовим аспектом є аналіз жестів, для чого реалізовано перевірку стабільності положення пальців (тобто чи не змінюється протягом кількох секунд). Якщо жест є стабільним, подальша обробка залежить від його типу. У функціональній схемі це розділено на чотири окремі блоки, для кожного з чотирьох жестів: тривога, допомога, норма, завершення. Кожен із цих жестів викликає свою послідовність дій, а саме: підсвічування екрану, звукові сигнали, логування події та надсилання повідомлення в Telegram разом із фото.

Усі ці гілки об’єднуються в загальному блоці Telegram-повідомлення, після чого інформація записується у лог-файл. Окремо в схемі показано функцію малювання та розпізнавання математичних рівнянь. При відповідному жесті активується режим малювання, де відстежується рух вказівного пальця. Як тільки малюнок завершено, зображення надсилається до моделі ШІ Gemini, яка вирішує та надає розв’язок рівняння. Отриманий результат потім виводиться у Streamlit-інтерфейсі.

Ця функціональна схема дозволяє чітко побачити, як обробляється кожен сценарій взаємодії користувача з програмою: від виявлення жесту до реакції системи. У результаті схема демонструє як технічну, так і логічну структуру всієї програми, що робить її легкою для аналізу та подальших покращень.

2.3 Розпізнавання та ідентифікація долоні

2.3.1 OpenCV

Зараз у сфері комп'ютерного зору відбуваються кардинальні зміни. Причина цьому легкий доступ до даних, зростання обчислювальних потужностей та новітні алгоритми, які відкривають небачені досі можливості. Основну роль у цих змінах відіграють глибинне навчання та сучасні фреймворки, що полегшують його застосування.

Поєднання ШІ та комп'ютерного зору дало машинам можливість машинам сприймати зображення та відео майже як люди. Ці технології вже застосовуються в безпілотниках, робототехніці та нових технологій обробки фото. Серед інструментів особливо виділяється OpenCV (Open Source Computer Vision) – провідна бібліотека з відкритим вихідним кодом для додатків комп'ютерного зору. На рис. 2.2 наведено приклад використання OpenCV для виявлення та ідентифікації кількох об'єктів на одному зображенні в реальному часі.

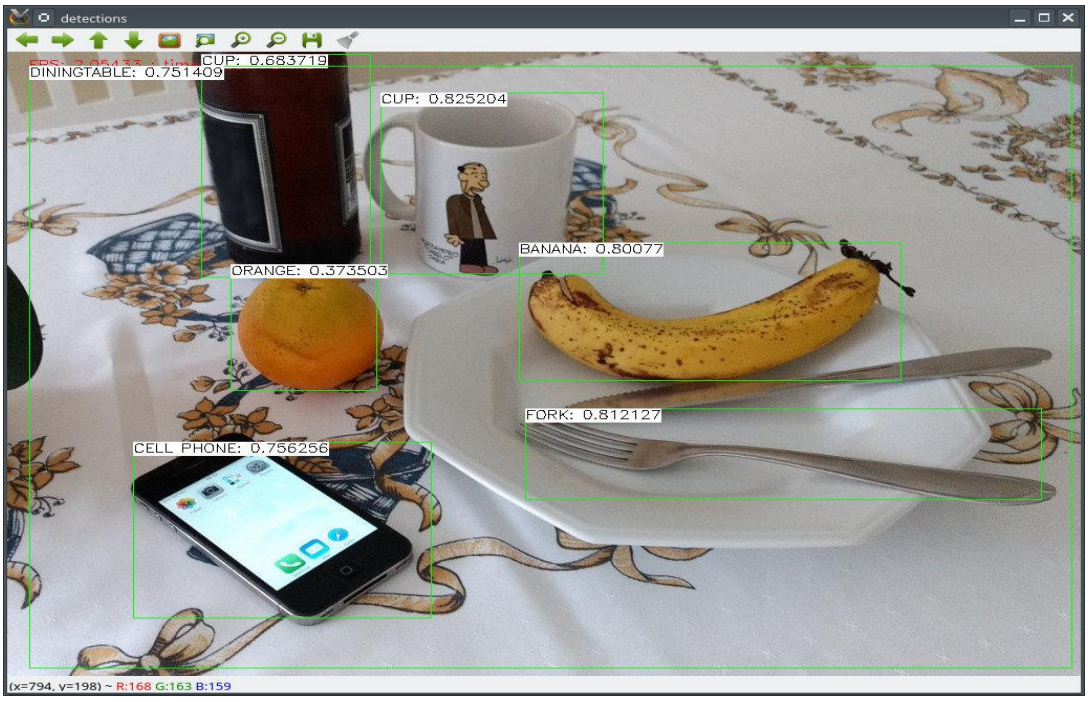

Рисунок 2.2 – Приклад використання OpenCV для виявлення об'єктів [19]

OpenCV – це потужний інструмент для роботи з відео та зображеннями. Він дозволяє вирішувати завдання різного рівня складності: від простих, як розпізнавання об'єктів, до складних, як відстеження руху та 3D-моделювання. OpenCV підтримує C++, Python, Java і MATLAB і працює на Windows, Linux, Android і MacOS. Завдяки оптимізованому коду та підтримці CUDA й OpenCL, OpenCV забезпечує роботу в реальному часі і є основним інструментом для розробки інновацій у сферах відеоспостереження, робототехніки та цифрового мистецтва а її функціональність охоплює понад 2500 алгоритмів для найрізноманітніших завдань:

– обробка зображень: надає широкі можливості та потужні інструменти для роботи зі зображеннями. Дозволяє виконувати такі завдання, як фільтрація, перетворення колірного простору і обчислення гістограми;

– виявлення і зіставлення ознак: має можливість знаходити і зіставляти характерні точки між різними зображеннями. Ця функціональність є основою для створення панорамних зображень і систем розпізнавання об'єктів;

– виявлення об'єктів: Бібліотека підтримує як класичні каскади Хаара, так і сучасні підходи, засновані на нейромережах. Такі можливості активно використовуються для завдань, наприклад, для розпізнавання облич;

– геометрія: є інструменти для роботи з геометрією зображень, зокрема для калібрування камери та 3D-реконструкції;

– машинне навчання: попри те, що OpenCV не розроблялася як бібліотека для машинного навчання, вона містить ключові алгоритми машинного навчання, які особливо корисні для комп'ютерного зору – такі як Support Vector Machines (svms) и K-Nearest Neighbors (KNN);

– аналіз відео: підтримує широкий спектр методів обробки відео, від аналізу руху до віднімання фону;

– інтеграція: Одна з головних переваг OpenCV є його сумісність. Він легко інтегрується з такими популярними бібліотеками та інструментами, як NumPy, TensorFlow та іншими (рис. 2.3) [19].

Рисунок 2.3 − Основні функції та можливості OpenCV [19]

Бібліотека OpenCV активно використовується в найрізноманітніших сферах – від побутових пристроїв до високотехнологічних рішень. Одним із найбільш популярних напрямів є розпізнавання жестів і взаємодія людини та комп'ютера. OpenCV значно спрощує розробку систем розпізнавання жестів. Такі технології дозволяють керувати пристроями або комп'ютерами просто за допомогою рухів, не використовуючи при цьому клавіатуру чи мишу. Це можна застосовувати в ігровій індустрії, додатках віртуальної реальності та інтерактивних вітринах. Наприклад, звичайна веб-камера може відстежувати

положення та рух рук, дозволяючи перемикати меню або грати без джойстиків, а лише рухати пальцями.

Ще одним напрямом є розуміння руху та виявлення об'єктів. У системах відеоспостереження, безпілотних авто та робототехніці широко застосовуються алгоритми OpenCV для розуміння руху та виявлення об'єктів. Вони можуть слугувати базою для програм, які аналізують відео з камер і миттєво сигналізують про підозрілі рухи в режимі реального часу – це допомагає запобігати крадіжкам чи аваріям [20].

Крім того, OpenCV широко застосовується у сфері доповненої реальності (AR). Він дає змогу розробникам накладати цифровий контент на реальний світ у режимі реального часу. Ця технологія вже активно використовується в іграх, освітніх програмах та маркетингових кампаніях. Наприклад, за допомогою OpenCV можна розпізнавати об'єкти чи спеціальні маркери навколо себе та накладати на них віртуальні елементи, такі як 3D-моделі або текстову інформацію.

2.3.2 MediaPipe

З сучасними технологіями ШІ, як MediaPipe від Google, з'являються нові можливості для поєднання реального та цифрового світів. Розроблений Google фреймворк MediaPipe – це фреймворк з відкритим вихідним кодом, розроблений для створення конвеєрів машинного навчання.

Його основна особливість це можливість обробки мультимедіа, а саме відео, аудіо, текст у реальному часі. Щоб створювати пайплайни комп'ютерного зору, також можна комбінувати готові компоненти, відомі як «калькулятори». Це стає можливим завдяки його модульній архітектурі та зручному графовому інтерфейсу.

В основі роботи MediaPipe лежить концепція обробки потокових даних. Інформація проходить через послідовність взаємозв'язаних "калькуляторів", де

кожен виконує певну функцію, а потім передає оброблений результат наступному елементу. Кожен такий потік це серія пакетів даних. Така структура графа не тільки спрощує поетапну обробку інформації, але й гарантує високу ефективність усього конвеєра машинного навчання [21].

Пайплайни, розроблені за допомогою MediaPipe, можуть стабільно працювати як у веб-середовищі, так і на мобільних пристроях, наприклад Android та iOS, а також на вбудованих системах з обмеженими ресурсами. Така кросплатформаність дає можливість розробникам створювати гнучкі та швидкі додатки для різноманітних пристроїв.

MediaPipe пропонує багато цікавих можливостей. Наприклад, одна з них – це використання потужності відеокарт (GPU) для швидшої обробки даних. Оскільки GPU беруть на себе найважчу обчислювальну роботу, MediaPipe здатний миттєво обробляти навіть дуже складні мультимедійні завдання. А ще, завдяки паралельній обробці, він може робити кілька речей одночасно, наприклад, обробляти багато відео або запускати одразу кілька систем комп'ютерного зору.

MediaPipe також використовує OpenCV. Завдяки цьому MediaPipe без зусиль вбудовує у свої конвеєри функції, як захоплення, обробка та показ відео. Крім того, MediaPipe працює разом з TensorFlow, інструментом Google для машинного навчання, що робить дуже простим додавання вже готових або своїх власних моделей. Це спрощує такі завдання, як розпізнавання облич або розуміння того, що говорять люди. А ще MediaPipe підтримує популярні мови програмування, такі як C++, Java і Python, роблячи його зручним для використання в будь-яких проєктах [21].

2.3.3 CVZone

Сфера розпізнавання рук є досить складною, вимагаючи глибоких досліджень, інноваційних розробок та впровадження непростих алгоритмів і

передових технік. CVZone являє собою рішення для розпізнавання рук, яке було розроблене на основі фреймворку MediaPipe.

CVZone визначається своєю високою продуктивністю, легкістю у використанні. Його практичність підкреслюється завдяки вбудованим функціям для попередньої обробки вхідних даних, отриманих з зображень та відео. Основа cvzone на платформі Google MediaPipe гарантує стабільну роботу та високу продуктивність. Цей інноваційний підхід є значним прогресом у галузі розпізнавання рук, підвищуючи точність та ефективність процедур розпізнавання завдяки розумному поєднанню сучасних технологій і алгоритмів. Спеціально для розпізнавання рук, CVZone створив рішення на основі фреймворку MediaPipe Hand Landmarker, що є простим підхідом до виявлення та ідентифікації ключових точок на руках на зображеннях [22]. На рис. 2.4. показаний приклад такої візуалізації, де на долоні визначаються ключові точки та з'єднуючі їх лінії.

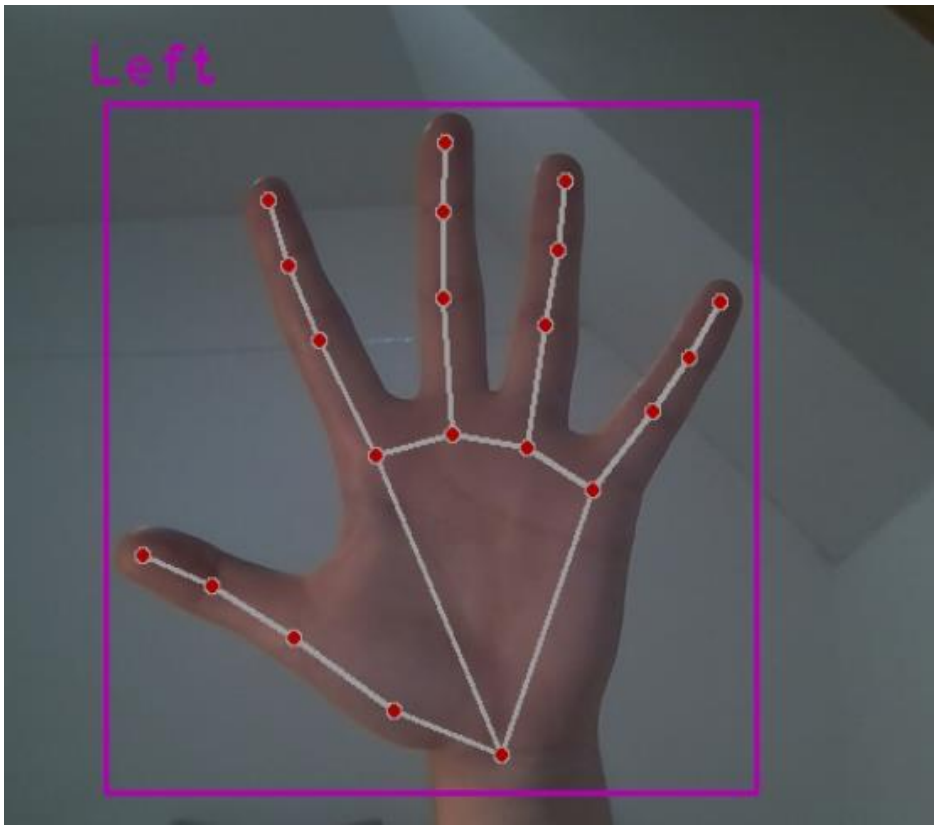

Рисунок 2.4 − Візуалізація орієнтира руки Mediapipe

Цей метод обробляє дані зображень за допомогою машинного навчання, ефективно працюючи як з окремими зображеннями, так і з безперервним відеопотоком. В результаті він надає таку інформацію, як ключові орієнтири на руці, визначені на координатах зображення, а також інформацію про те, ліва це рука чи права, для кількох виявлених рук (рис. 2.5) [23].

MediaPipe визначає загалом 21 таку ключову точку для кожної знайденої руки. Ось коротка інформація про ці ключові точки:

– зап'ястя: основа руки;

– 2 − 5. Великий палець: чотири точки: кінчик, суглоб і дві точки вздовж великого пальця;

– 6 − 9. Вказівний палець: чотири точки: кінчик, суглоб і дві точки вздовж вказівного пальця;

– 10 − 13. Середній палець: чотири точки: кінчик, суглоб і дві точки вздовж середнього пальця;

– 14 − 17. Безіменний палець: чотири точки: кінчик, суглоб і дві точки вздовж безіменного пальця;

– 18 − 20. Мізинець: чотири точки: кінчик, суглоб і дві точки вздовж мізинця [24].

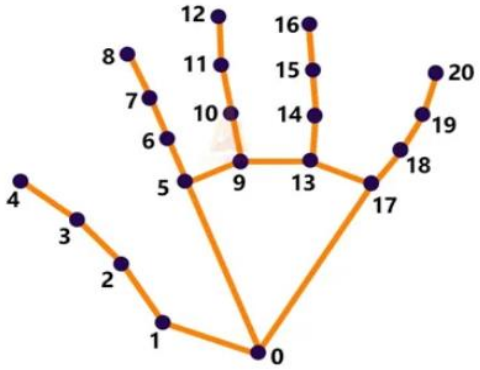

Рисунок 2.5 − Ключові точки долоні за бібліотекою MediaPipe [24]

Між цими ключовими точками проводяться з'єднувальні лінії, а кути між ними ретельно обчислюються. Саме цей складний процес дозволяє точно розпізнавати жести рук у реальному часі. Щоб зменшити обчислювальну складність, метод використовує область, визначену опорними точками в першому кадрі, що допомагає знайти положення руки в наступних кадрах [23].

Розпізнавання руки активується лише в тому випадку, якщо рука не була виявлена або її відстеження зникло. Це дозволяє зменшити частоту повторного запуску моделі розпізнавання з самого початку, що підвищує таким чином її ефективність.

Метод CVZone − це значне досягнення у сфері розпізнавання рук, що базується на технічних можливостях системи MediaPipe. Його простота та надійність роблять його безцінним інструментом для різноманітних застосувань. Подальші дослідження та розробки цього методу дозволять подолати існуючи обмеження та значно розширити його функціонал.

2.3.4 Приклад реалізаціїї розпізнавання та ідентифікації долоні

Цей фрагмент коду демонструє розпізнавання руки користувача за допомогою комп’ютерного зору.

Для забезпечення функціональності, пов'язаної з комп'ютерним зором та графікою, на початку реалізації підключаються всі необхідні бібліотеки:

– HandDetector – модуль, який містить готову логіку для виявлення руки, аналізу пальців та їх положення;

– cv2 – використовується для доступу до вебкамери та обробки зображення;

– import numpy as np – використовується для створення «порожнього полотна».

```
from cvzone.HandTrackingModule import HandDetector
import cv2
import numpy as np
```

Далі ініціалізуємо та налаштовуємо детектор долоні (HandDetector):

```
DETECTION_CONFIDENCE = 0.8
MIN_TRACKING_CONFIDENCE = 0.7

detector = HandDetector(
    staticMode=False,
    maxHands=1,
    modelComplexity=1,
    detectionCon=DETECTION_CONFIDENCE,
    minTrackCon=MIN_TRACKING_CONFIDENCE
)
```

Спочатку відбувається імпорт необхідного класу HandDetector. зі спеціалізованої бібліотеки cvzone. Далі створюється об’єкт detector, який відповідає за виявлення долоні у кадрі. Щоб система працювала надійно було встановлено параметри впевненості: пошук руки активується лише тоді, коли ймовірність її присутності перевищує 80% (detectionCon=0.8). А коли долоню вже знайдено, для подальшого відстеження цей поріг трохи знижується до 70% (minTrackCon=0.7) — це допомагає тримати руку «в полі зору», навіть якщо вона активно рухається. Крім того, програма налаштована на роботу лише з однією долонею (maxHands=1), що дозволяє уникнути плутанини у керуванні.

Головна функція для розпізнавання руки має наступний вигляд:

```
def getHandInfo(img):
    hands, img = detector.findHands(img, draw=True, flipType=True)
    if hands:
        hand = hands[0]
        lmList = hand["lmList"]
        fingers = detector.fingersUp(hand)
        return fingers, lmList, hand
    else:
        return None
```

Основна логіка розпізнавання виконується у функції getHandInfo(img). Вона приймає на вхід один кадр з вебкамери і передає його в метод findHands, який аналізує зображення, щоб знайти руки. У результаті цей метод повертає повноцінну цифрову модель знайденої долоні, що складається з 21 ключової точки (lmList). Ці точки включають кінчики пальців, суглоби та зап’ястя, і дозволяють точно визначити положення руки в кадрі. Після цього викликається метод fingersUp, який перевіряє, які саме пальці підняті, а які стиснуті. Він аналізує розташування певних точок і повертає список із п’яти значень: кожне з них − це 0 або 1, тобто палець опущений або піднятий.

Відеопотік не є ідеальним, і долоня людини може тремтіти. Це призводить до того, що розпізнаний жест може "мерехтіти" між двома схожими станами. Щоб уникнути хибних спрацьовувань, було впроваджено механізм стабілізації:

```
GESTURE_HISTORY_LENGTH = 5
gesture_history = []
current_gesture = None
gesture_stable_time = 0
```

```
def get_stable_gesture(new_gesture):
    global gesture_history, current_gesture, gesture_stable_time

    gesture_history.append(new_gesture)
    if len(gesture_history) > GESTURE_HISTORY_LENGTH:
        gesture_history.pop(0)

    if len(gesture_history) == GESTURE_HISTORY_LENGTH and all(g == new_gesture for g in gesture_history):
        if current_gesture == new_gesture:
            gesture_stable_time += 1
        else:
            current_gesture = new_gesture
            gesture_stable_time = 1

        if gesture_stable_time >= 3:
            return current_gesture
    return None
```

Як це працює: було створено список gesture_history, який зберігає кілька останніх розпізнаних жестів. Кожен новий жест, отриманий від fingersUp(), додається до цього списку. Жест вважається "стабільним" лише тоді, коли всі елементи в gesture_history однакові. Внаслідок цього, короткочасна випадкова зміна положення пальця не призведе до миттєвої зміни жесту і не викличе небажаної реакції системи. Це дозволяє реагувати тільки на усвідомлені та утримувані протягом певного часу жести.

В основному циклі програми відбувається виклик функції стабілізації, і якщо вона повертає підтверджений жест, система реагує (код наведено у додатку

Б):

```
while run:
    # ...
    info = getHandInfo(img)
    if info:
        fingers, lmList, hand = info

        stable_gesture = get_stable_gesture(tuple(fingers))

        if stable_gesture:
            if stable_gesture == (0, 0, 0, 0, 0):
                # ... дії для жесту "кулак"

            elif stable_gesture == (0, 1, 1, 1, 0):
                # ... дії для іншого жесту
```

У головному циклі програма постійно відстежує, чи з'явився чіткий, стійкий жест. Щойно такий жест зафіксовано, система порівнює його з підготовленим списком шаблонів. Наприклад, комбінація (0, 0, 0, 0, 0), яка означає стиснутий кулак, одразу активує режим "Тривога". Якщо ж виявлено жест із піднятим великим пальцем – (1, 0, 0, 0, 0), це вважається сигналом, що все гаразд. Тобто, програма переводить візуальний жест у зрозумілу команду, яка далі запускає потрібну частину логіки.

2.4 Розрахунок площі об'єкта за допомогою камери

Важливою задачею є оцінка площі об’єкта – кисті руки, що потрапляє в кадр. Ця інформація може використовуватись для покращення точності аналізу жестів, фільтрації шумів, а також подальшої класифікації поз рук.

Розрахунок площі об'єкта за допомогою камери є досить складною задачею і вимагає певних припущень та калібрування. Просто взяти фотографію об'єкта і визначити його площу неможливо, адже зображення є лише двовимірним відображенням тривимірного простору, і при цьому втрачається важлива інформація про глибину та справжні розміри. Одним з ефективних рішень є метод опорного об'єкта, тобто використання на зображенні об’єкта з відомими реальними розмірами.

Якщо на зображенні із рукою є об'єкт, чиї реальні розміри точно відомі, наприклад: картка, аркуш А4 або монета, то можна розрахувати масштаб пікселів на зображенні відносно сантиметрів, а далі площу кисті в реальних одиницях.

Визначення масштабу зображення. Щоб визначити справжню площу кисті, необхідно спочатку визначити масштаб зображення, тобто співвідношення пікселів до реальних одиниць довжини. Для цього, як опорний об’єкт, було використано лист картону (довжиною 13 см), який був розміщений в одній площині з рукою.

Довжина опорного об’єкта в пікселях була визначена за допомогою бібліотеки OpenCV. Зображення було попередньо оброблено, після чого координати кінцевих точок аркуша були вирахувані автоматично. Для обчислення піксельної довжини між двома точками та використовується формула Евклідової відстані:

$$L_{(\text{піксельний})} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \qquad (2.1)$$

Тому піксельна довжина опорного об’єкта дорівнює:

$$L_{(\text{піксельний})} = \sqrt{(878-894)^2 + (886-226)^2} = \sqrt{256 + 660^2} = \\ = 660{,}19 \text{ піксель.}$$

Нехай:

– $L_{(\text{реальний})}$ – реальна довжина опорного об'єкта (см);

– $L_{(\text{піксельний})}$ – його довжина на зображенні (пікселі).

Масштаб зображення обчислюється за формулою:

$$S = \frac{L_{(\text{реальний})}}{L_{(\text{піксельний})}} \frac{\text{см}}{\text{пікс}}. \qquad (2.2)$$

Тому масштаб зображення дорівнює:

$$S = \frac{13}{660{,}19} = 0{,}0196 \frac{\text{см}}{\text{пікс}}.$$

Вимірювання площі кисті на зображенні. Площа A_((піксельна)) кисті руки визначається вручну, наприклад, за допомогою графічного редактора. Після обробки зображення в графічному редакторі Photoshop було отримано розміри області, що відповідає кисті руки: 530 × 1013 пікселів.

Це дає площу:

$$A_{(\text{піксельна})} = 530 \cdot 1013 = 536{,}890 \text{ пікселів}^2.$$

Тепер можна розрахувати реальну площу кисті руки.

Реальна площа обчислюється за формулою:

$$A_{(\text{реальна})} = S^2 \cdot A_{(\text{піксельна})} \text{ см}^2. \quad (2.3)$$

Тому реальна площа кисті руки дорівнює:

$$A_{(\text{реальна})} = 0{,}01969^{\,2} \cdot 536{,}890 = 208{,}25 \text{ см}^2.$$

Отже, за допомогою простих математичних розрахунків на основі масштабу зображення, можна з достатньою точністю визначити реальну площу кисті руки. Ці дані є корисними не лише для оцінки положення руки, а й для покращення точності класифікації жестів, фільтрації фонових об'єктів і реалізації адаптивної системи розпізнавання.

2.5 Розрахунки з теорії автоматичного управління

Розглянемо випадок, коли ми маємо послідовність зображень (наприклад, відео), і ми хочемо відстежувати та розпізнавати об'єкт у часі. У цьому контексті ми могли б спробувати змоделювати деякі етапи обробки як лінійні фільтри.

Припустимо, у нас є часова послідовність ознак, витягнутих з кожного кадру відео, що описують потенційний об'єкт. Ми можемо розглядати кожен етап обробки цих ознак як лінійну систему.

Нехай f(t) є часовим рядом деякої ознаки об'єкта (наприклад, розмір, орієнтація, середній колір в області інтересу) на кадрі t.

Ми можемо застосувати лінійний фільтр для згладжування цього часового ряду ознак. У частотній області дія лінійного фільтра описується множенням спектра вхідного сигналу на передавальну функцію фільтра H(s) (у неперервному часі) або H(z) (у дискретному часі).

Наприклад, простий фільтр низьких частот може бути використаний для усунення високочастотних шумів у часовому ряді ознак. У неперервному часі передавальна функція фільтра першого порядку може мати вигляд:

$$H_{(s)} = \frac{1}{\tau s + 1}, \tag{2.4}$$

де τ – часова константа фільтра.

У дискретному часі відповідний фільтр першого порядку матиме передавальну функцію:

$$H_{(z)} = \frac{b_0}{1 - a_1 z^{-1}}. \tag{2.5}$$

Припустимо, ми маємо часовий ряд ознаки f(k) і ми застосовуємо простий згладжуючий фільтр з різницевим рівнянням:

$$y(k) = (1 - a)y(k - 1) + af(k), \tag{2.6}$$

де y(k) – відфільтроване значення на кроці;

0<a<1 – коефіцієнт згладжування.

Передавальна функція цього фільтра в z-області буде:

$$H_{(z)} = \frac{a}{1 - (1 - a)z^{-1}} = \frac{az}{z - (1 - a)}. \tag{2.7}$$

Нехай $a = 0{,}2$ і вхідна послідовність ознак буде $f(k) = \{10{,}12{,}15{,}13{,}16{,}\ldots\}$. Припустимо, початкове значення $y(-1) = 10$.

Ітерація 1 ($k = 0$):

$$y(0) = (1 - 0{,}2)y(-1) + 0{,}2f(0) = 0{,}8 \cdot 10 + 0{,}2 \cdot 10 = 8 + 2 = 10.$$

Ітерація 2 ($k = 1$):

$$y(1) = 0{,}8y(0) + 0{,}2f(1) = 0{,}8 \cdot 10 + 0{,}2 \cdot 12 = 8 + 2{,}4 = 10{,}4.$$

Ітерація 3 ($k = 2$):

$$y(2) = 0{,}8y(1) + 0{,}2f(2) = 0{,}8 \cdot 10{,}4 + 0{,}2 \cdot 15 = 8{,}32 + 3 = 11{,}32.$$

Так само можна зайти й інші $y(k)$. Відфільтрована послідовність $y(k)$ буде більш гладкою, ніж вихідна $f(k)$.

3 РОЗРОБЛЕННЯ ТА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ПІДТРИМКИ ТЕХНІЧНИХ РІШЕНЬ ПРАЦІВНИКА ПРИЛАДОБУДІВНОГО ВИРОБНИЦТВА

3.1 Блок-схема роботи програми

Блок-схема ілюструє логіку роботи програми. Вона працює з розпізнаванням жестів руки в реальному часі, реагує на окремі жести відповідними діями та, за потреби, передає зображення у штучний інтелект для генерації відповіді. Схема детально показує весь цикл роботи: від ініціалізації інтерфейсу, камери, модулів розпізнавання та сервісів, до обробки кадру, аналізу жестів, взаємодії з користувачем і завершення роботи. На рис. 3.1 наведена блок-схема роботи програми.

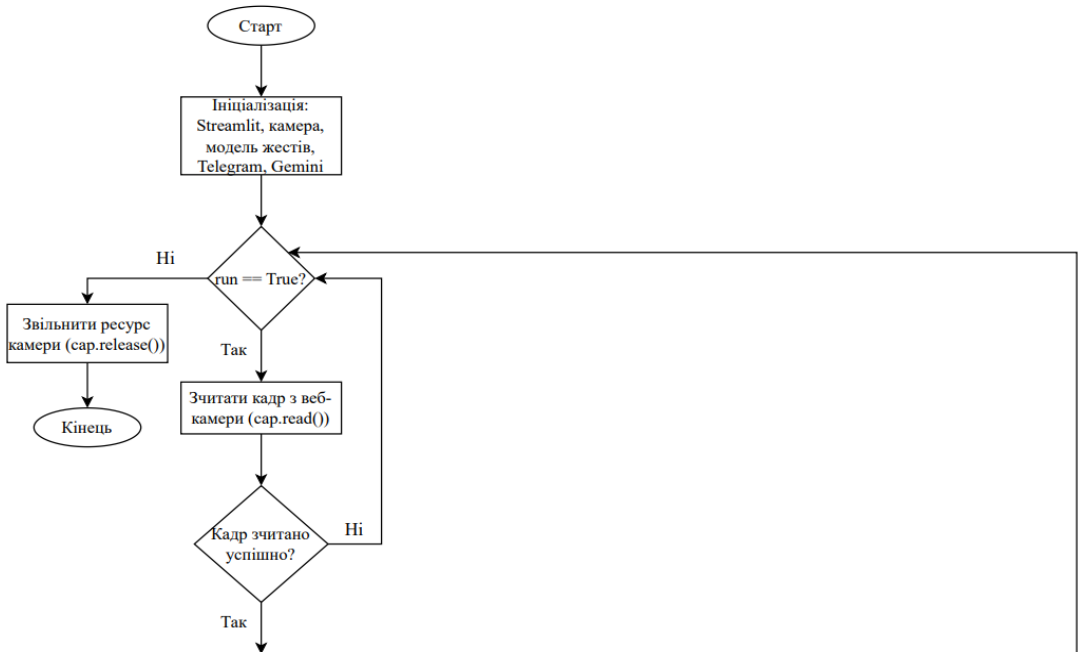

Рисунок 3.1 – Блок-схема роботи програми

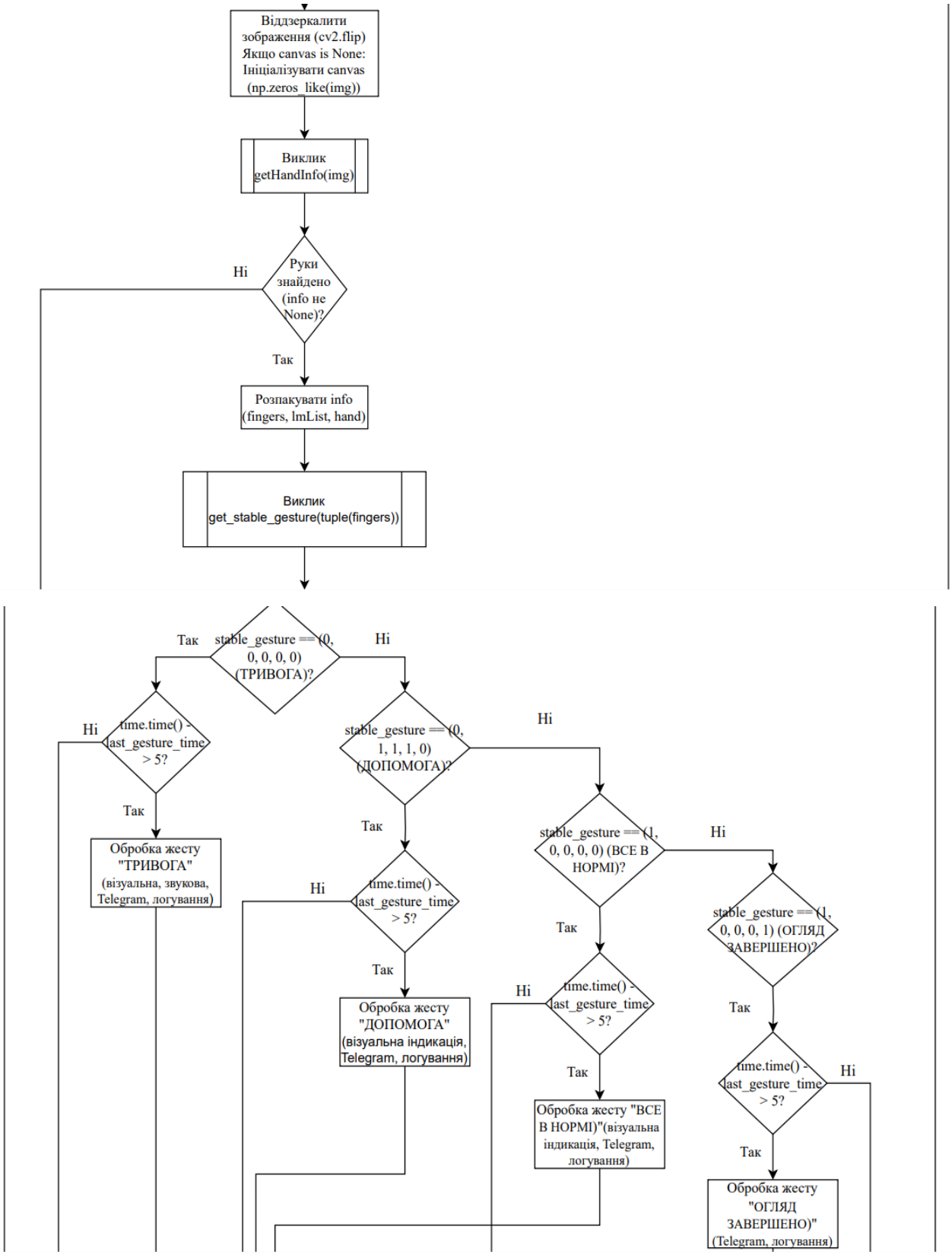

Рисунок 3.1, аркуш 2

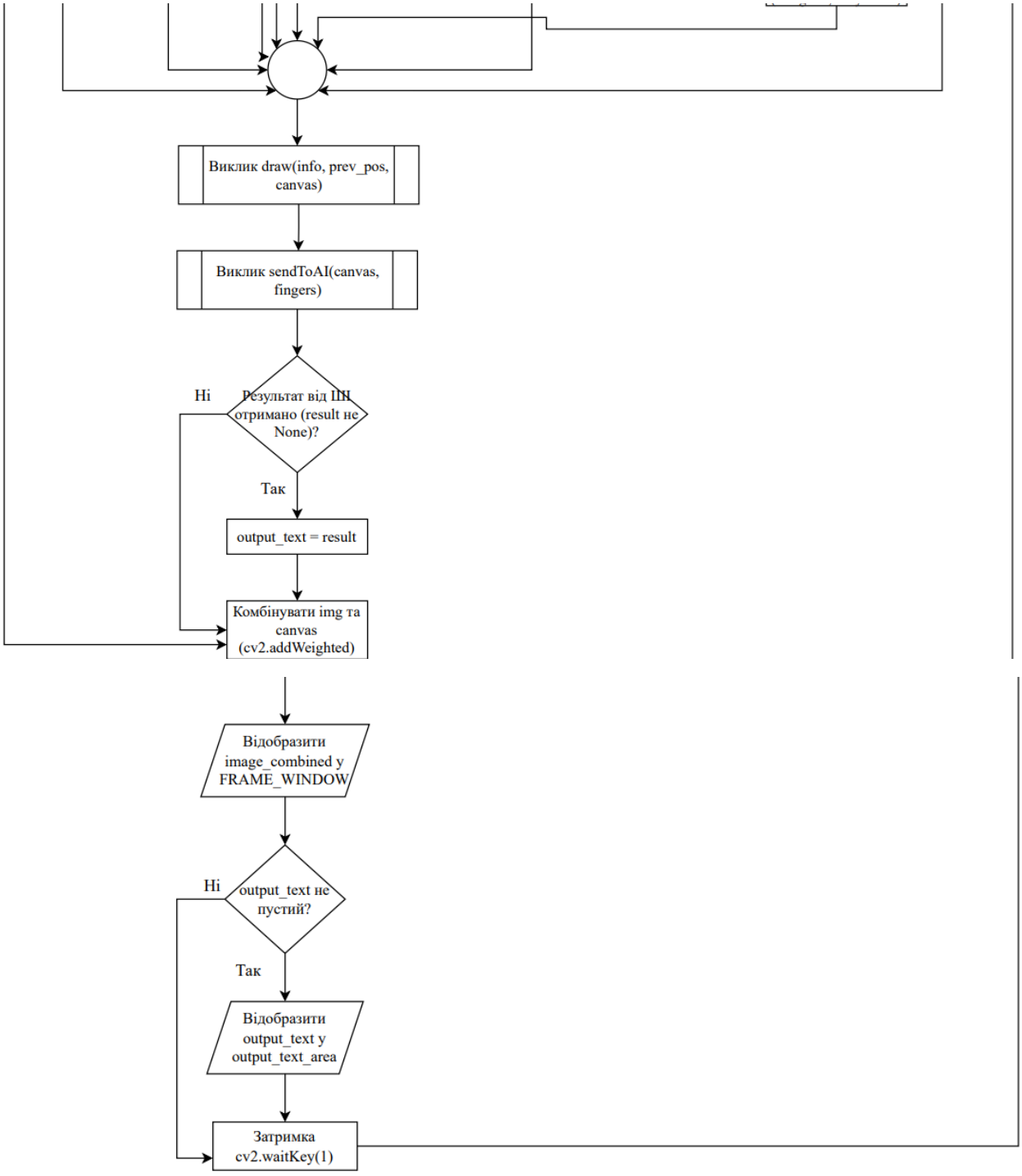

Рисунок 3.1, аркуш 3

Після запуску програми відбувається ініціалізація всіх необхідних компонентів: інтерфейсу Streamlit, відеопотоку з камери, модуля розпізнавання жестів, API Telegram та моделі Gemini від Google. Обробка даних починається,

коли прапорець run стає активним, запускаючи таким чином основний робочий цикл.

У кожній ітерації виконується зчитування кадру з камери. Якщо кадр не вдалося отримати, то цикл переходить до наступної ітерації. Успішно зчитане зображення віддзеркалюється, а потім перевіряється наявність полотна для малювання (canvas). Якщо полотно ще не створене, воно ініціалізується як порожнє зображення такого ж розміру, що й вхідний кадр.

Наступним кроком викликається функція getHandInfo, яка на основі кадру визначає, чи є рука в полі зору. Якщо рука не виявлена, то виконується наступна ітерація. Якщо рука знайдена, з об’єкта руки витягується список пальців, координати та інша інформація. Визначається стабільність показаного жесту, тобто чи залишається незмінним той самий жест кілька кадрів поспіль. Це дозволяє уникати випадкових спрацювань.

Якщо жест стабільний і відповідає певному шаблону, то далі запускається відповідна обробка:

– (0,0,0,0,0) − тривога. Виводиться візуальний ефект (миготіння червоного фону), звучить звуковий сигнал, надсилається повідомлення з фото у Telegram і фіксується запис у лог;

– (0,1,1,1,0) − допомога. Активується помаранчеве підсвічування, надсилається повідомлення у Telegram і записується лог;

– (1,0,0,0,0) – все в нормі. Відображається зелений фон, надсилається повідомлення у Telegram і додається запис у журнал подій;

– (1,0,0,0,1) – завершення огляду. Надсилається відповідне повідомлення в Telegram і лог-файл оновлюється.

Паралельно з цим, якщо розпізнаний жест це вказівний палець, активується режим малювання на полотні. Лінії проводяться між поточним і попереднім положенням пальця. При повному розкритті кисті (1,1,1,1,1) полотно очищується.

Якщо показано жест [1,1,1,1,0] (жест надсилання), і на полотні щось намальовано, то зображення передається у модель Gemini для обробки, наприклад, для розв’язання математичного рівняння. Якщо модель повертає результат, він відображається в окремій області інтерфейсу.

На фінальному етапі кадр із камери та полотно комбінуються в один кадр і показуються у вікні Streamlit. Якщо відповідь від ШІ була отримана, то вона також виводиться на екран. Цикл повторюється до завершення роботи.

Після зупинки програми відбувається звільнення ресурсу камери.

3.2 Розроблення програмного забезпечення

У рамках дипломного проєкту було створено програмне забезпечення, що дозволяє користувачеві взаємодіяти з комп’ютером за допомогою жестів руки. Основна мета програми − надати можливість малювати на екрані, розпізнавати певні жести пальців та надсилати зображення до штучного інтелекту (моделі Gemini від Google) для обробки, зокрема розв’язання математичних задач.

На початку програми підключаються такі основні модулі:

– cv2, cvzone − робота з відеопотоком, зображеннями та розпізнавання рук (HandDetector);

– numpy − для числових операцій, зокрема з масивами зображень;

– google.genai − доступ до генеративного ШІ для обробки запитів (наприклад, розв'язання рівнянь);

– PIL (Pillow) − маніпуляції із зображеннями, конвертація форматів;

– streamlit − створення інтерактивного веб-інтерфейсу;

– time − керування часовими інтервалами;

– csv, datetime − ведення журналу подій (логування у CSV-файл);

– threading − виконання задач у фоновому режимі (сповіщення, звуки);

– winsound − відтворення системних звуків Windows;
– requests − надсилання HTTP-запитів (взаємодія з Telegram API);
– io − робота з потоками байтів (кодування зображень).

```
import cvzone
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
from google import genai
from PIL import Image
import streamlit as st
import time
import csv
from datetime import datetime
import threading
import winsound
import requests
import io
```

Далі визначено константи для Telegram API: BOT_TOKEN (токен бота) та CHAT_ID (ID чату для сповіщень):

```
BOT_TOKEN = "7840194256:AAGCh2qDRpdChvTzWcz2PdYnDwpGMIl4gz8"
CHAT_ID = "-1002212573067"
```

Далі йде налаштування детектора рук:

```
DETECTION_CONFIDENCE = 0.8
```

```
MIN_TRACKING_CONFIDENCE = 0.7
```

Встановлено параметри впевненості для детектора рук: DETECTION_CONFIDENCE (початкове виявлення) та MIN_TRACKING_CONFIDENCE (подальше відстеження).

Потім йде створення лог-файлу:

```
def log_event(event):
    with open("logs.csv", "a", newline="", encoding="utf-8-sig") as file:
        writer = csv.writer(file)
        now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        writer.writerow([now, f"Подія: {event}"])
```

Функція log_event записує час, подію та розпізнаний жест у файл logs.csv.

Потім налаштовується надсилання повідомлення та фото у Telegram:

```
def send_telegram_alert(img, text, gesture_type):
    url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendPhoto"

    img_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    draw = ImageDraw.Draw(img_pil)

    font_path = "RobotoMono.ttf"  # Наприклад: "arial.ttf" або "times.ttf"
    font_size = 30
    try:
        font = ImageFont.truetype(font_path, font_size)
```

```
        except:
            font = ImageFont.load_default()
            print("Увага: використовується стандартний шрифт замість українського")

        if gesture_type == "alert":
            draw.text((50, 50), " ТРИВОГА", font=font, fill=(255, 0, 0))
        elif gesture_type == "help":
            draw.text((50, 50), " ПОТРІБНА ДОПОМОГА", font=font, fill=(0, 165, 255))

            alert_img = cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)

        _, img_encoded = cv2.imencode('.jpg', alert_img)
        img_bytes = img_encoded.tobytes()

        files = {'photo': ('alert.jpg', img_bytes)}
        data = {'chat_id': CHAT_ID, 'caption': text}

        requests.post(url, files=files, data=data)
```

Функція send_telegram_alert надсилає зображення з відповідним текстовим накладанням ("ТРИВОГА" або "ПОТРІБНА ДОПОМОГА") з підтримкою українських літер та підписом у вказаний Telegram-чат.

Далі ініціалізація інтерфейсу:

```
col1, col2 = st.columns([2, 1])
```

```
with col1:
    run = st.checkbox("Run", value=True)
    FRAME_WINDOW = st.image([])

with col2:
    output_text_area = st.title("Answer")
    output_text_area = st.subheader("")
```

Створено інтерфейс Streamlit, а саме дві колонки. Перша містить прапорець "Run" для запуску/зупинки та область для відео (FRAME_WINDOW). Друга містить заголовок "Answer" та місце для виведення текстових результатів.

Далі ініціалізовано клієнт genai.Client з API-ключем для взаємодії з моделлю Google Gemini:

```
client = genai.Client(api_key="AIzaSyBbgm76F4zAxsftl2LpOpHfU0Tu6Wt6yM8")
```

Після цього відбувається ініціалізація камери та детектора руки:

```
cap = cv2.VideoCapture(0)
detector = HandDetector(
    staticMode=False,
    maxHands=1,
    modelComplexity=1,
    detectionCon=DETECTION_CONFIDENCE,
    minTrackCon=MIN_TRACKING_CONFIDENCE
)
```

Налаштовано захоплення відео з веб-камери (cv2.VideoCapture(0)) та ініціалізовано детектор рук HandDetector з параметрами для відстеження однієї руки в динамічному режимі.

Визначено змінні для стабілізації жестів:

```
GESTURE_HISTORY_LENGTH = 5
gesture_history = []
current_gesture = None
gesture_stable_time = 0
```

GESTURE_HISTORY_LENGTH (розмір історії), gesture_history (список останніх жестів), current_gesture (поточний стабільний жест), gesture_stable_time (лічильник стабільності).

```
def get_stable_gesture(new_gesture):
    global gesture_history, current_gesture, gesture_stable_time

    gesture_history.append(new_gesture)
    if len(gesture_history) > GESTURE_HISTORY_LENGTH:
        gesture_history.pop(0)

    if len(gesture_history) == GESTURE_HISTORY_LENGTH and all(g == new_gesture for g in gesture_history):
        if current_gesture == new_gesture:
            gesture_stable_time += 1
        else:
```

```
            current_gesture = new_gesture
            gesture_stable_time = 1

        if gesture_stable_time >= 3:
            return current_gesture
    return None
```

Функція get_stable_gesture аналізує історію жестів. Якщо жест залишається незмінним протягом GESTURE_HISTORY_LENGTH кадрів і gesture_stable_time досягає 3, він вважається стабільним і повертається.

Наступний крок обробка руки та розпізнавання жестів:

```
def getHandInfo(img):
    hands, img = detector.findHands(img, draw=True, flipType=True)
    if hands:
        hand = hands[0]
        lmList = hand["lmList"]
        fingers = detector.fingersUp(hand)
        return fingers, lmList, hand
    else:
        return None
```

Функція getHandInfo виявляє руку на зображенні, отримує список її ключових точок (lmList) та стан піднятих пальців (fingers).

Далі відбувається реалізація малювання:

```
def draw(info, prev_pos, canvas):
    fingers, lmList, hand = info
    current_pos = None

    if fingers == [0, 1, 0, 0, 0]:
        index_tip = lmList[8]
        thumb_tip = lmList[4]
        distance = ((index_tip[0] - thumb_tip[0]) ** 2 + (index_tip[1] - thumb_tip[1]) ** 2) ** 0.5

        if distance > 50:
            current_pos = index_tip[0:2]
            if prev_pos is not None:
                cv2.line(canvas, current_pos, prev_pos, color=(255, 0, 255), thickness=10)
            else:
                prev_pos = current_pos

    elif fingers == [1, 1, 1, 1, 1]:
        canvas = np.zeros_like(img)

    return current_pos, canvas
```

Функція draw реалізує малювання на полотні (canvas). Якщо піднято вказівний палець і він достатньо віддалений від великого, малюється лінія. Якщо піднято всі пальці, полотно очищується. Повертає поточну позицію малювання та оновлене полотно.

Потім відправляється зображення до ШІ:

```
def sendToAI(canvas, fingers):
    if fingers == [1, 1, 1, 1, 0]:
        if np.any(canvas > 0):
            pil_image = Image.fromarray(cv2.cvtColor(canvas, cv2.COLOR_BGR2RGB))
            response = client.models.generate_content(
                model="gemini-2.0-flash",
                contents=["Розв'яжи математичне рівняння", pil_image]
            )
            return response.text
    return None
```

Функція sendToAI(canvas, fingers) активується, коли користувач показує жест чотири пальці підняті ([1,1,1,1,0]). У цей момент зображення з полотна конвертується у формат PIL.Image і надсилається до моделі Gemini 2.0 Flash від Google.

Далі йде ініціалізація змінних для основного циклу:

```
prev_pos = None
canvas = None
image_combined = None
output_text = ""
last_gesture_time = time.time()
```

prev_pos (попередня позиція для малювання), canvas (полотно), image_combined (комбіноване зображення), output_text (текст для виводу), last_gesture_time (час останнього жесту для запобігання спаму).

Основний цикл програми:

```
while run:
    success, img = cap.read()
    if not success:
        continue

    img = cv2.flip(img, 1)

    if canvas is None:
        canvas = np.zeros_like(img)

    info = getHandInfo(img)
    if info:
        fingers, lmList, hand = info

        stable_gesture = get_stable_gesture(tuple(fingers))

        if stable_gesture:
            if stable_gesture == (0, 0, 0, 0, 0):
                if time.time() - last_gesture_time > 5:
                    blink_canvas = np.zeros_like(img)
                    blink_canvas[:] = (0, 0, 255)
                    for _ in range(3):
```

```
                    FRAME_WINDOW.image(blink_canvas, channels="BGR")
                    time.sleep(0.2)
                    FRAME_WINDOW.image(img, channels="BGR")
                    time.sleep(0.2)

                threading.Thread(target=lambda: winsound.Beep(1000,
500)).start()
                log_event("ТРИВОГА")

                threading.Thread(
                    target=send_telegram_alert,
                    args=(img, "ТРИВОГА! ", "alert")
                ).start()

                output_text = "ТРИВОГА!"
                last_gesture_time = time.time()

            elif stable_gesture == (0, 1, 1, 1, 0):
                if time.time() - last_gesture_time > 5:
                    help_canvas = np.zeros_like(img)
                    help_canvas[:] = (0, 165, 255)
                    FRAME_WINDOW.image(help_canvas, channels="BGR")
                    time.sleep(0.5)
                    FRAME_WINDOW.image(img, channels="BGR")
                    time.sleep(0.5)
                    FRAME_WINDOW.image(help_canvas, channels="BGR")
                    time.sleep(0.5)
                    FRAME_WINDOW.image(img, channels="BGR")
```

```
            log_event("Потрібна допомога")

            threading.Thread(
                target=send_telegram_alert,
                args=(img, "ПОТРІБНА ДОПОМОГА! ", "help")
            ).start()

            output_text = " Запитано допомогу"
            last_gesture_time = time.time()

        elif stable_gesture == (1, 0, 0, 0, 0):
            if time.time() - last_gesture_time > 5:
                green_canvas = np.zeros_like(img)
                green_canvas[:] = (0, 255, 0)
                FRAME_WINDOW.image(green_canvas, channels="BGR")
                time.sleep(1)
                log_event("Стан: НОРМА")
                threading.Thread(
                    target=send_telegram_alert,
                    args=(img, " Стан: НОРМА ", "normal")
                ).start()
                output_text = " Стан: НОРМА"
                last_gesture_time = time.time()

        elif stable_gesture == (1, 0, 0, 0, 1):
            if time.time() - last_gesture_time > 5:
                log_event("Огляд завершено")
```

```
                threading.Thread(
                    target=send_telegram_alert,
                    args=(img, " Огляд завершено ", "normal")
                ).start()
                output_text = " Огляд завершено"
                last_gesture_time = time.time()

            prev_pos, canvas = draw(info, prev_pos, canvas)
            result = sendToAI(canvas, fingers)
            if result:
                output_text = result

        image_combined = cv2.addWeighted(img, 0.7, canvas, 0.3, 0)
        FRAME_WINDOW.image(image_combined, channels="BGR")

        if output_text:
            output_text_area.text(output_text)

        cv2.waitKey(1)
```

Спочатку зчитується та дзеркально відображається кадр з камери. Потім ініціалізується полотно canvas, якщо воно ще не створене. Далі отримується інформація про руку (getHandInfo). Якщо рука виявлена, визначається стабільний жест (get_stable_gesture). Якщо стабільний жест розпізнано і минуло достатньо часу з попереднього жесту, то обробляються специфічні жести ("Тривога", "Потрібна допомога", "Все в нормі", "Огляд завершено"). Це включає візуальні/звукові ефекти, логування, надсилання сповіщень у Telegram у окремих потоках та оновлення тексту для виводу.

Далі викликається функція малювання (draw), оновлюється prev_pos. Якщо показано жест для надсилання рівняння, викликається sendToAI. У разі успішного розпізнання, результат виводиться, а полотно очищується.

Зображення з камери змішується з полотном (cv2.addWeighted) та відображається у Streamlit (FRAME_WINDOW.image). А вже наприкінці Виводиться текстова інформація (output_text_area.text).

cap.release()

Після завершення циклу звільняються ресурси камери (cap.release())

Отже, програмне забезпечення поєднує сучасні бібліотеки комп’ютерного зору, інтерфейс користувача у браузері та можливості генеративного штучного інтелекту. Воно демонструє комплексний підхід до взаємодії людини з комп'ютером за допомогою жестів. Створена підсистема здатна в реальному часі розпізнавати наперед визначені положення пальців руки, трансформуючи їх у конкретні команди або статусні повідомлення.

3.3 Результати роботи програми

Після завершення етапу розробки було проведено тестування роботи розробленого програмного забезпечення. Результати демонструють ефективність взаємодії користувача з системою за допомогою жестів, а також здатність штучного інтелекту обробляти зображення та надавати відповідь у текстовій формі. На рис. 3.2 – 3.14 наведені результати роботи програми.

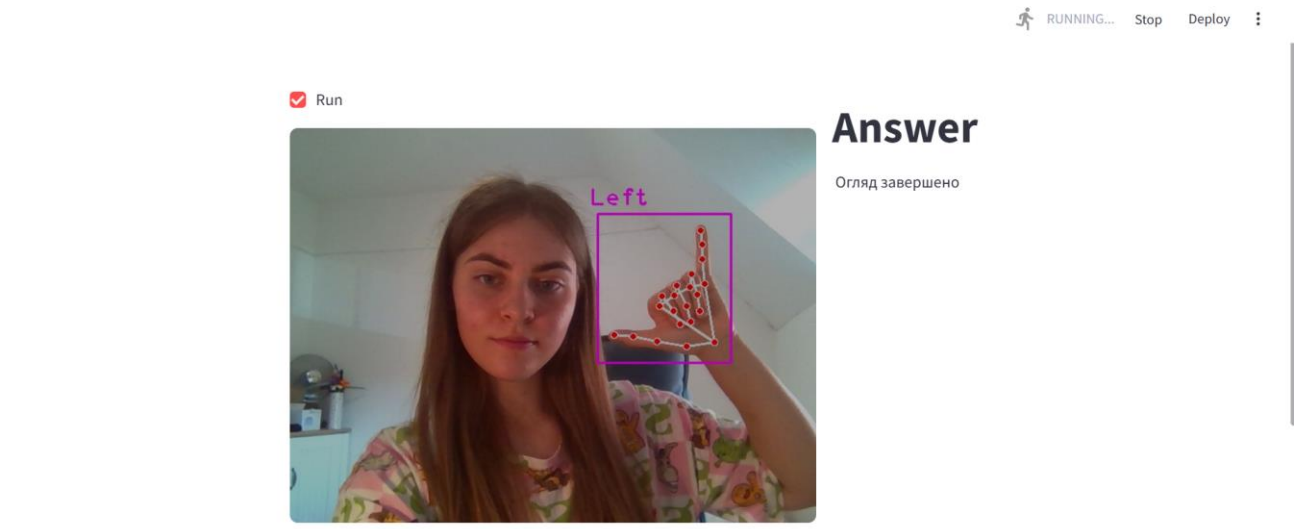

Рисунок 3.2 − Розпізнавання жесту «Огляд завершено»

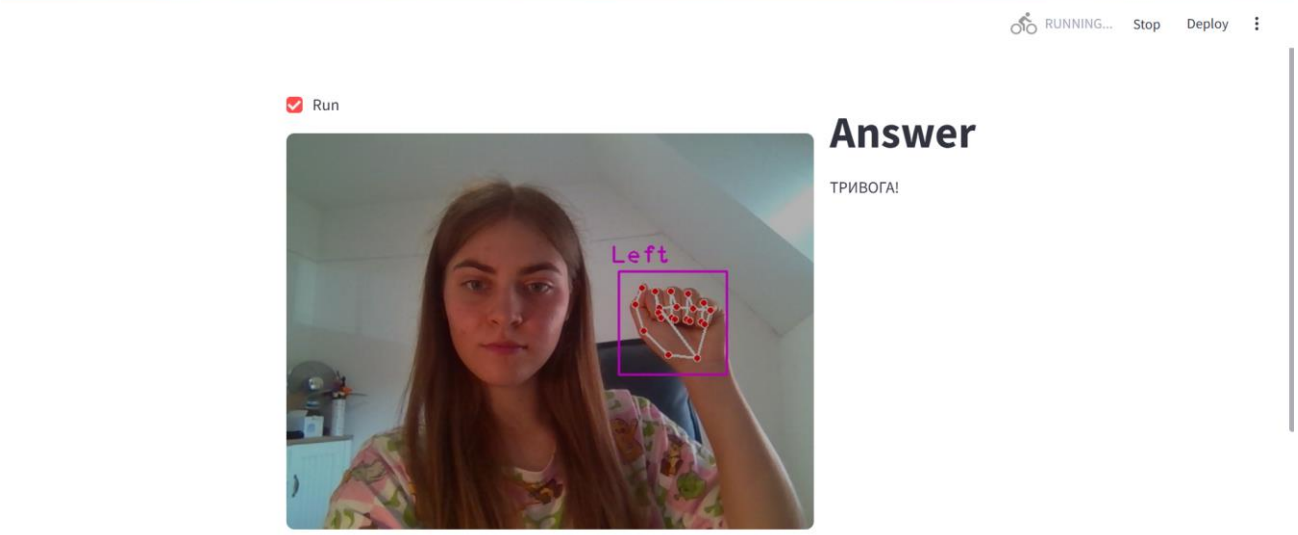

Рисунок 3.3 − Розпізнавання жесту «Тривога»

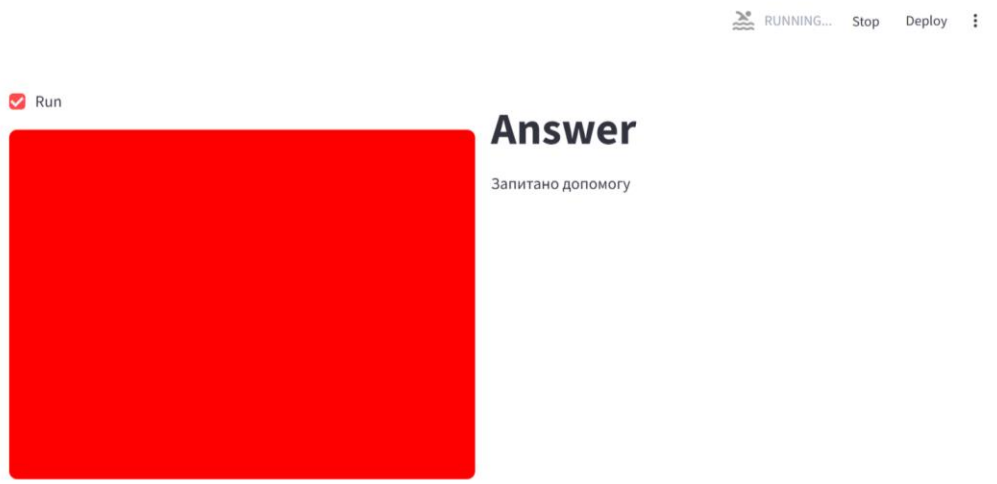

Рисунок 3.4 – Візуалізація та сигнал тривоги

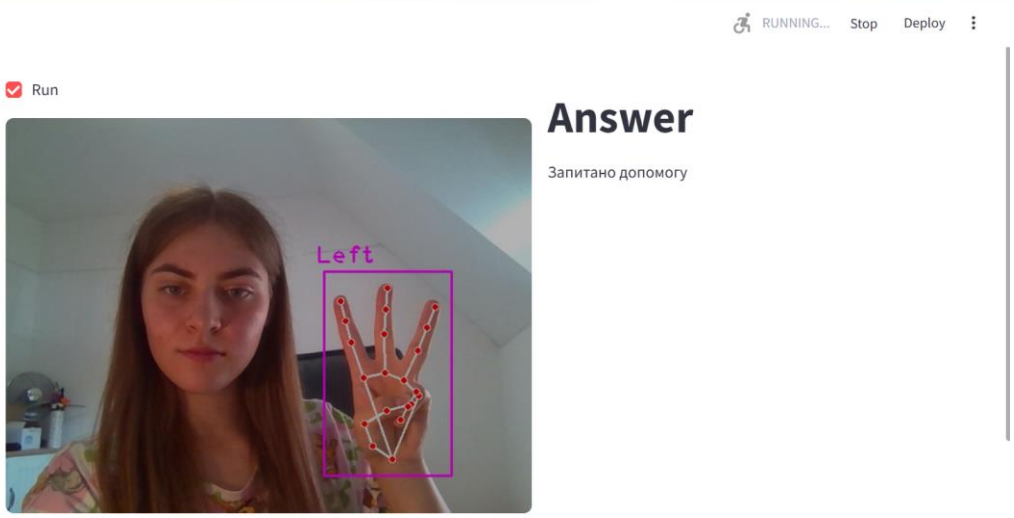

Рисунок 3.5 – Розпізнавання жесту «Запитано допомогу»

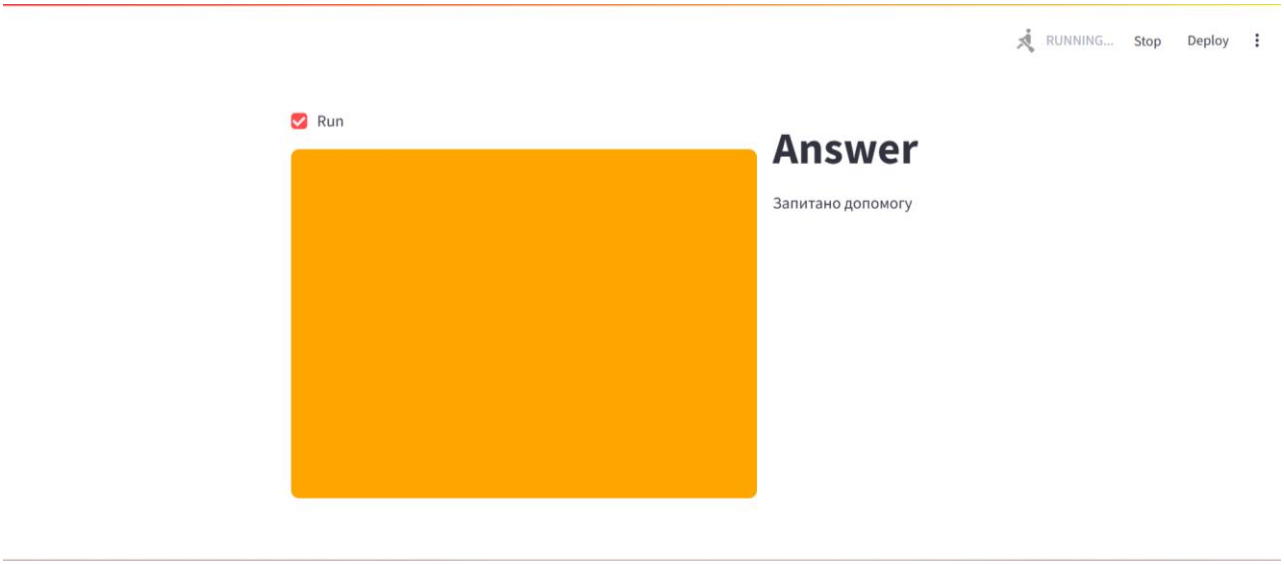

Рисунок 3.6 – Візуалізація

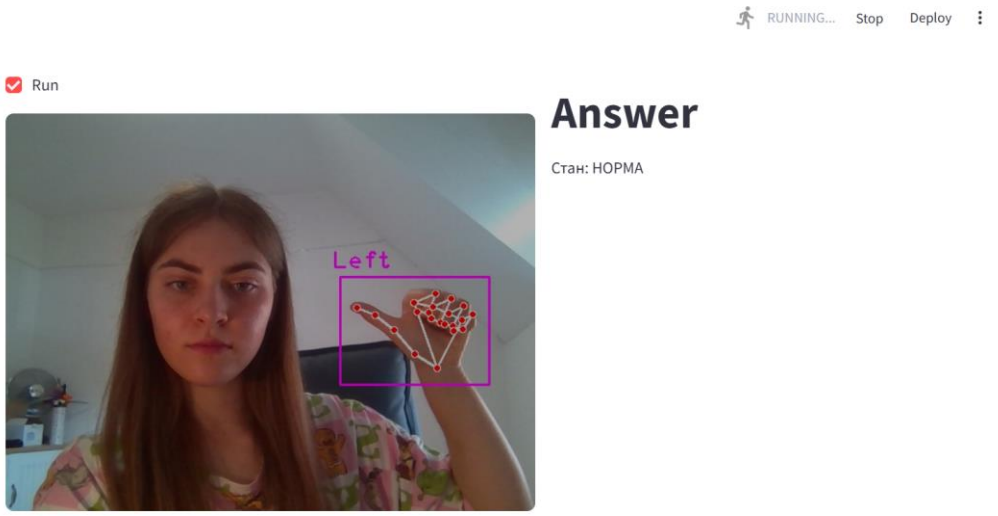

Рисунок 3.7 − Розпізнавання жесту «Норма»

Рисунок 3.8 – Розпізнавання жесту руки для малювання

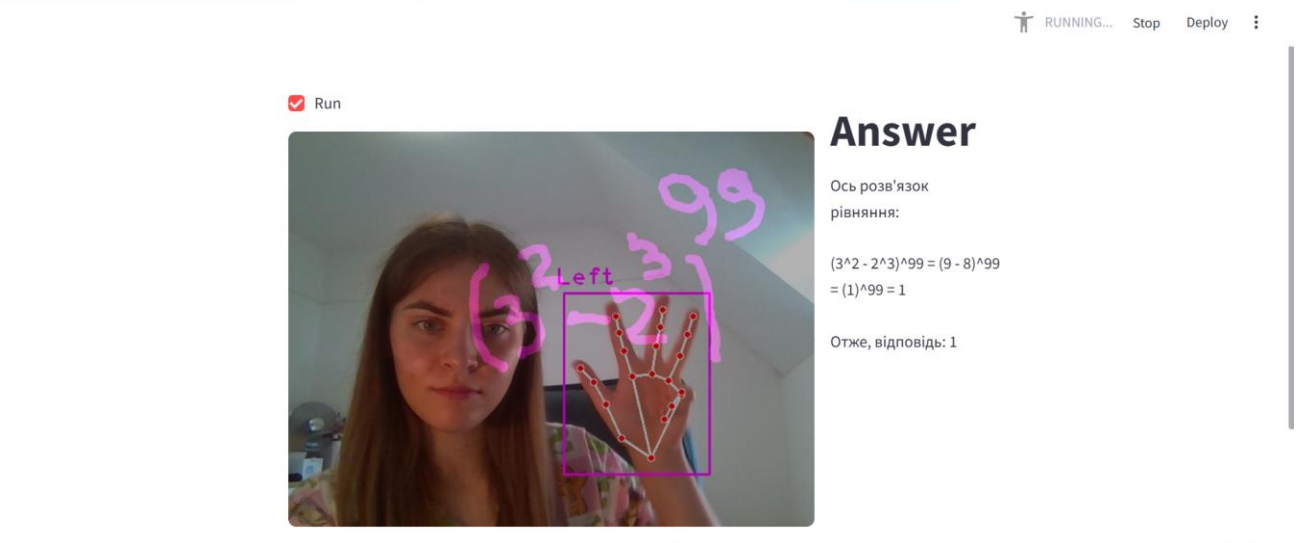

Рисунок 3.9 – Розпізнавання жесту руки для відправки запросу ШІ та відповідь ШІ на поставлену математичну задачу

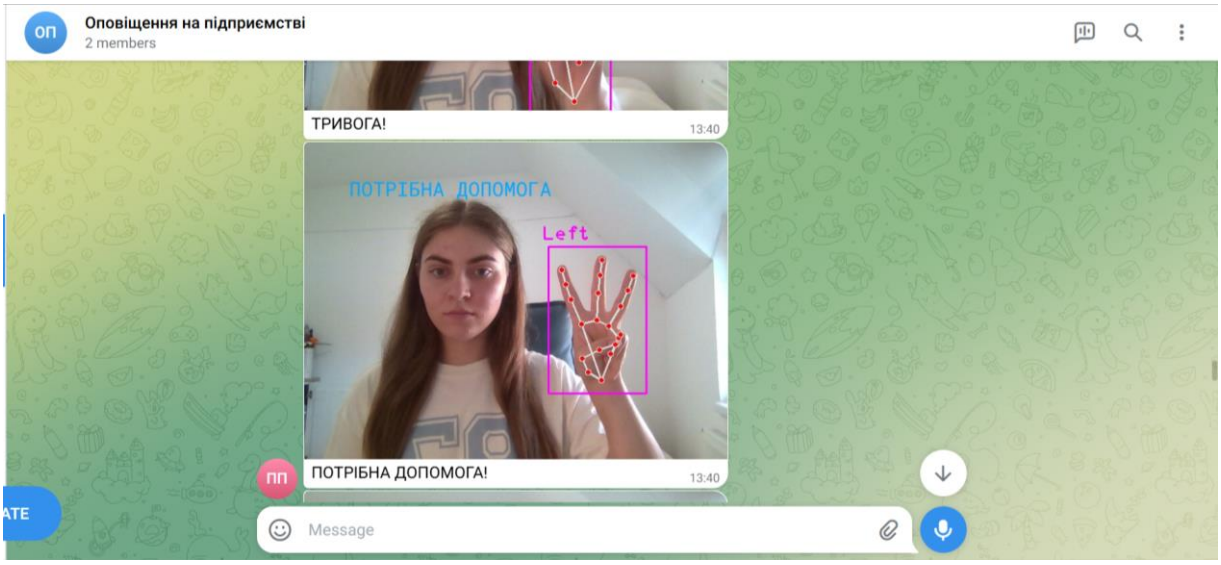

Рисунок 3.10 – Відправлення фото до Телеграм каналу з жестом «Запитано допомогу»

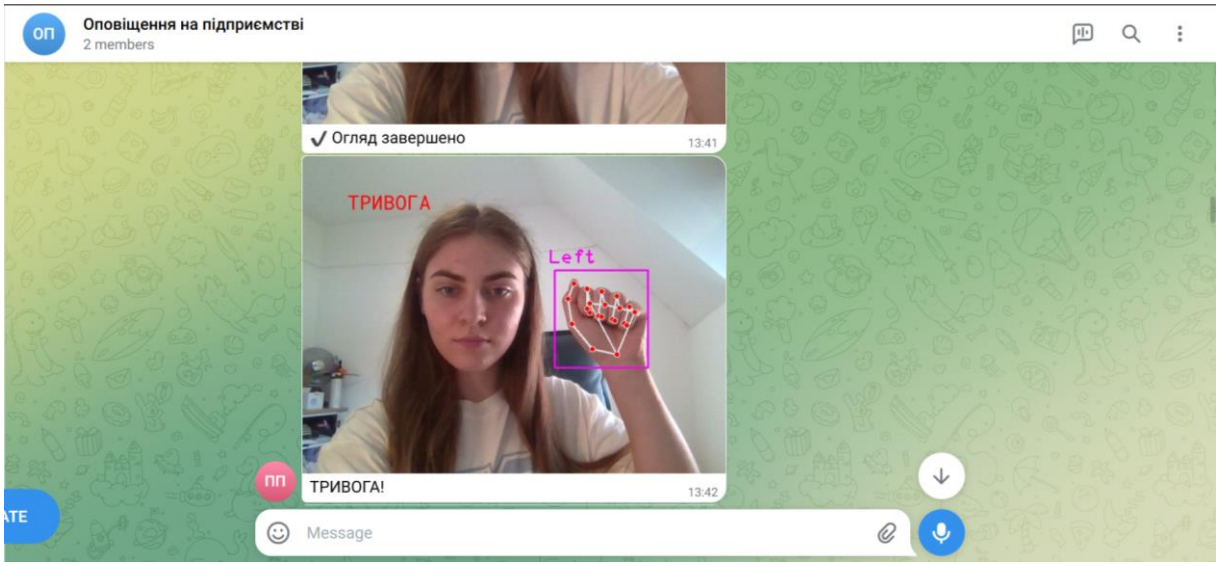

Рисунок 3.11 – Відправлення фото до Телеграм каналу з жестом «Тривога»

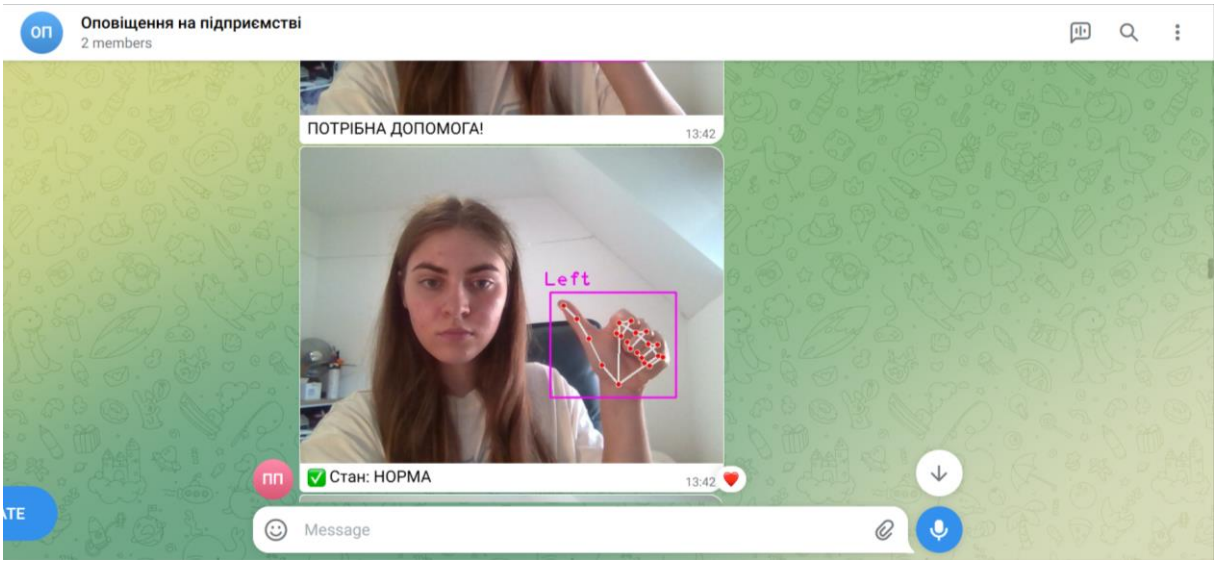

Рисунок 3.12 – Відправлення фото до Телеграм каналу з жестом «Норма»

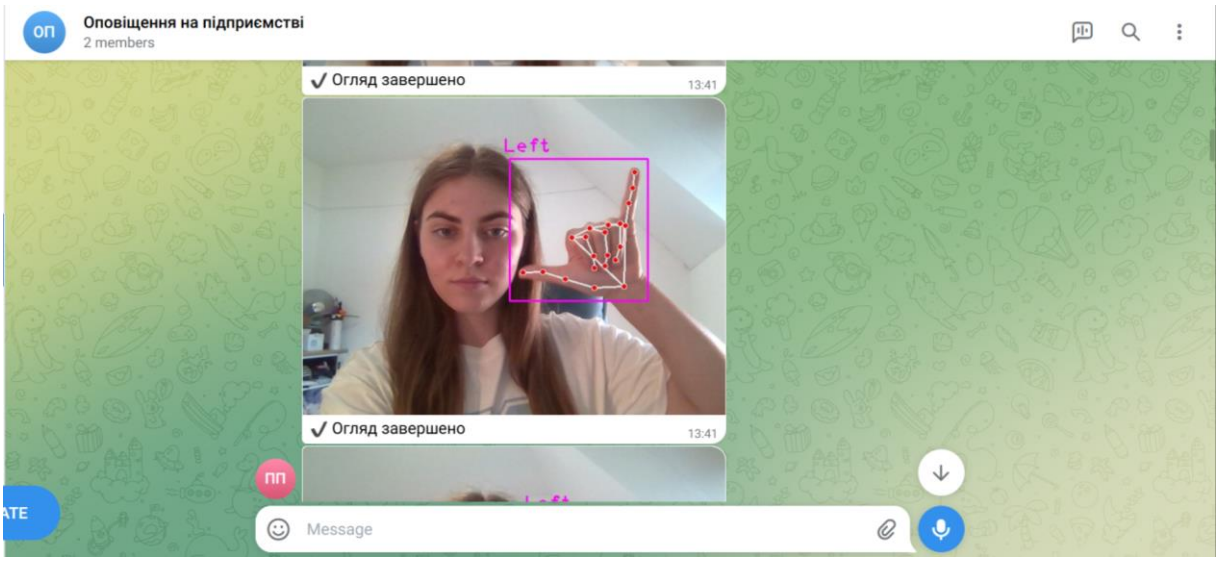

Рисунок 3.13 – Відправлення фото до Телеграм каналу з жестом «Огляд завершено»

135	08.06.2025 23:33	Подія: Потрібна допомога
136	08.06.2025 23:33	Подія: ТРИВОГА
137	08.06.2025 23:33	Подія: Огляд завершено
138	08.06.2025 23:33	Подія: Стан: НОРМА
139	08.06.2025 23:45	Подія: Потрібна допомога
140	08.06.2025 23:45	Подія: Огляд завершено
141	08.06.2025 23:45	Подія: Стан: НОРМА
142	08.06.2025 23:46	Подія: ТРИВОГА
143		
144		
145		
146		
147		
148		
149		
150		
151		
152		
153		
154		
155		

Рисунок 3.14 – Логування подій

3.3 Охорона праці

Робота з комп’ютером належить до видів діяльності, яка супроводжується зоровим і психоемоційним навантаженням, тому відноситься до легкої фізичної праці категорії Іа-Іб згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Через тривале перебування за монітором можуть виникати втома очей, загальне виснаження й проблеми з опорно-руховим апаратом. Тому важливо подбати про належні умови роботи та дотримання режиму праці й відпочинку.

Забезпечення належних умов праці починається з дотримання параметрів мікроклімату в приміщенні. Мікроклімат має відповідати вимогам, визначеним у ДСН 3.3.6.042-99. Для робіт категорії Іа–Іб оптимальні значення наведено в таблиці 3.1.

Таблиця 3.1 – Оптимальні параметри мікроклімату

Параметри	Холодний період	Теплий період
Температура повітря, ˚С	22-24	23-25
Відносна вологість повітря, %	60-40	60-40
Швидкість руху повітря, м/с	до 0,1	до 0,1

Необхідний мікроклімат підтримується завдяки роботі централізованого опалення в холодну пору року, а в теплий період за рахунок природного провітрювання або кондиціонування повітря (за наявності відповідної системи).

Не менш важливим є освітлення робочого місця. Воно має бути достатнім як за рахунок природного, так і штучного світла. Під час організації штучного освітлення слід орієнтуватися на вимоги ДБН В.2.5-28:2018, забезпечуючи освітленість на робочій поверхні в межах 300 – 500 лк.

Якщо рівень шуму в приміщенні перевищує допустимі значення, встановлені ДСН 3.3.6.037-99, застосовуються заходи для його зниження: екранування, облицювання стін або використання звукопоглинальних матеріалів.

Приміщення, де виконується робота, за своїм призначенням та наявністю горючих матеріалів (таких як папір чи пластикові корпуси техніки) відноситься до категорії «В» – пожежонебезпечне, згідно з НАПБ Б.03.002-2007. Відповідно до ПУЕ, за класом пожежонебезпеки воно належить до П-IIа, що характерно для зон з наявністю горючих рідин із температурою спалаху понад 61 °С або пилу, здатного до займання. Будівля, в якій розміщене приміщення, передбачається як така, що має II ступінь вогнестійкості – наприклад, із цегляними стінами, відповідно до ДБН В.1.1.7:2016.

Серед основних причин виникнення пожежі можна віднести коротке замикання в електромережі або обладнанні, перевантаження мережі, а також

несправності самих пристроїв. Щоб уникнути загоряння та забезпечити безпеку, важливо дотримуватися правил пожежної безпеки: не використовувати несправні електроприлади й саморобні подовжувачі, утримувати евакуаційні проходи вільними. Приміщення має бути оснащене первинними засобами пожежогасіння, а саме вуглекислотним (ВВК) або порошковим (ВП) вогнегасником із зарядом, достатнім для гасіння ймовірного осередку вогню (наприклад, ВВК-1,4 або ВП-2).

ВИСНОВКИ

Під час написання першого розділу атестаційної роботи було проведено аналіз літератури за темою розробки підсистеми підтримки технічних рішень працівника з використанням елементів доповненої реальності, а саме:

– проведено аналіз концепції Індустрії 5.0, зокрема її ключові принципи, орієнтацію на людиноцентричний підхід у виробництві та важливість ефективної співпраці людини й машини;

– проаналізовано, як сьогодні розвиваються інтелектуальні системи підтримки працівників. Особливу увагу приділено цифровим помічникам на основі штучного інтелекту та колаборативним роботам, які вже використовуються на сучасних виробництвах;

– проведено аналіз основних напрямків розвитку штучного інтелекту та роль інтелектуальних персональних помічників у різних в різних сферах життя;

– розглянуто технології комп’ютерного зору, а саме: де вони застосовуються, як навчаються відповідні моделі, і з якими викликами сьогодні стикається ця галузь.

За підсумками написання другого розділу було здійснено вибір та обґрунтування технічних засобів, які необхідні для створення підсистеми.

Було визначено необхідне апаратне забезпечення та було детально описано конфігурацію програмного середовища: мову програмування Python, середовище розробки PyCharm і ключові бібліотеки: OpenCV, CVZone, MediaPipe, які використовуються для реалізації функцій комп'ютерного зору.

Також було розроблено функціональну схему роботи програми, яка показує, як взаємодіють її основних компонентів. Окрему увагу було приділено процесу розпізнавання та ідентифікації долоні: проаналізовано функціональні

можливості бібліотек OpenCV, MediaPipe та CVZone, а також наведено приклад програмної реалізації цього модуля.

В результаті написання третього розділу було безпосередньо розроблено програмне забезпечення для підтримки технічних рішень працівника на основі розпізнавання жестів. У процесі розробки було:

– розроблено блок-схему роботи програми;
– реалізовано програмний код;
– проведено тестування розробленого програми;
– розглянуто питання охорони праці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Системна інженерія» [Електронний ресурс] / упоряд. : І. Ш. Невлюдов, О. М. Цимбал, О. В. Токарєва, А. І. Бронніков ; М-во освіти і науки України, ХНУРЕ. – електрон. вид. – Харків : ХНУРЕ, 2024. – 68 с.

2. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

3. Ривак Н. О. Індустрія 5.0: перехід до стійкої та орієнтованої на людину промисловості / Н. О. Ривак // Соціально-економічні проблеми сучасного періоду України. – 2022. – Вип. 3 (155). – С. 41–45.

4. Collaborative Robot Safety / Robots.com [Електронний ресурс]. – Режим доступу: https://www.robots.com/articles/collaborative-robot-safety.

5. George A. S., George A. S. H. Industrial Revolution 5.0: The Transformation of the Modern Manufacturing Process to Enable Man and Machine to Work Hand in Hand // The Seybold Report. – 2020. – Vol. 15, Issue 9. – P. 214–228.

6. Семененко Ю. Роль інформаційних технологій та інструментів штучного інтелекту в підвищенні ефективності підбору, навчання та адаптації працівників / Ю. Семененко // Галицький економічний вісник. – 2024. – № 2 (87). – С. 20–29.

7. Microsoft Teams Chatbot Integration / BotPenguin [Електронний ресурс]. – Режим доступу: https://botpenguin.com/glossary/microsoft-teams-chatbot-integration.

8. Maedche, A., Legner, C., Benlian, A., Berger, B., Gimpel, H., Hess, T., Hinz, O., Morana, S., & Söllner, M. (2019). AI-based digital assistants: Opportunities, threats, and research perspectives. Business & Information Systems Engineering, 61(4),

535–544.

9. Vernim, S., Bauer, H., Rauch, E., Ziegler, M. T., & Umbrello, S. (2022). A value sensitive design approach for designing AI-based worker assistance systems in manufacturing. Procedia Computer Science, 200, 505–516.

10. Демиденко М.А. Системи підтримки прийняття рішень: Навч. посіб. / М.А. Демиденко; Нац. гірн. ун-т. - Електрон. текст. дані. Д. 2016. - 104 с.

11. Phillips-Wren, G. Intelligent Decision Support Systems // This is a Book Title Name of the Author/Editor. – John Wiley & Sons, Ltd, 2013. – P. 1-14.

12. Talukder M. Major applications of AI / ResearchGate [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Major-applications-of-AI_fig2_370351166.

13. Пчелянський Д.П., Воінова С.А. Штучний інтелект: перспективи та тенденції розвитку // Автоматизація технологічних і бізнес-процесів. – 2019. – Т. 11, № 3.

14. Goksel-Canbek, N., Mutlu, M.E. On the track of Artificial Intelligence: Learning with Intelligent Personal Assistants // International Journal of Human Sciences. – 2016. – V. 13, № 1.

15. Balcı, E., (2019), Overview of Intelligent Personal Assistants. Acta Infologica, 3(1), 22-33.

16. Gupta A. Diverse Applications of Computer Vision Across Various Sectors / ResearchGate [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/figure/Diverse-Applications-of-Computer-Vision-Across-Various-Sectors_fig2_377143161.

17. Синеглазов В.М. Прикладні системи штучного інтелекту: комп'ютерний зір // Вісник НАН України. – 2024. – № 6.

18. Biks Y., Aleksishin K. The assessment of envelopes energy efficiency by multicriteria decision analysis methods // About the problems of science and practice, tasks and ways to solve them. – 2020.

19. What is OpenCV? / Roboflow Blog [Електронний ресурс]. – Режим доступу: https://blog.roboflow.com/what-is-opencv/.

20. Learning OpenCV / OpenCV [Електронний ресурс]. – Режим доступу: https://opencv.org/blog/learning-opencv/?ref=blog.roboflow.com.

21. What is MediaPipe? / Roboflow Blog [Електронний ресурс]. – Режим доступу: https://blog.roboflow.com/what-is-mediapipe/.

22. Ajiteshvarun J. Computer Vision using MediaPipe and OpenCV / Medium [Електронний ресурс]. – Режим доступу: https://medium.com/@joeajiteshvarun/computer-vision-using-mediapipe-and-opencv-4b44b902b918.

23. Sunitha K., Jeevan N. Y., Karthik P., Maithreya T. M., Prajwal S. Hand Gesture Recognition using CVZONE // Journal of Emerging Technologies and Innovative Research (JETIR). – 2024. – Т. 11, № 5. – С. f37–f42.

24. Gautam A. Hand Recognition using OpenCV / Medium [Електронний ресурс]. – Режим доступу: https://gautamaditee.medium.com/hand-recognition-using-opencv-a7b109941c88.