Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВЗАЄМОДІЇ З КОМП'ЮТЕРОМ ЗА ДОПОМОГОЮ ЖЕСТІВ З ВИКОРИСТАННЯМ PYTHON, MEDIAPIPE ТА OPENCV

DEVELOPMENT OF AN APPLICATION FOR COMPUTER INTERACTION USING GESTURES WITH PYTHON, MEDIAPIPE AND OPENCV

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Міронов Нікіта Олександрович

Керівник:
Самчук Людмила Михайлівна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Міронову Нікіті Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка застосунку для взаємодії з комп'ютером за допомогою жестів з використанням Python, MediaPipe та OpenCV»*
Керівник роботи: *к.т.н., доцент, Самчук Л. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу, документація засобів та інструментів розробки, допоміжні бібліотеки, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз та огляд сучасного стану проблеми, формулювання вимог і функціоналу застосунку і проектування системи, вибір інструментів розробки, розробка застосунку, тестування роботи і функціональності програмного рішення, створення виконуваного файлу та інсталяційного пакету, дослідження результатів точності розпізнавання, аналіз шляхів розвитку та рекомендації для вдосконалення системи.*
5. Перелік графічного матеріалу: *22 рисунки, 9 таблиць, 4 додатки*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Самчук Л. М.*		
Специфікація вимог до розробленої системи	*Самчук Л. М.*		
Розробка об'єкта проектування	*Самчук Л. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	16,64%		
Академічна доброчесність	*Самчук Л. М.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	виконано
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування	до 29.03.2025 р.	виконано
5	Практична реалізація об'єкта проектування	до 26.04.2025 р.	виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	виконано

Здобувач вищої освіти _____ Нікіта МІРОНОВ

Керівник кваліфікаційної роботи _____ Людмила САМЧУК

АНОТАЦІЯ

Міронов Н. О. Розробка застосунку для взаємодії з комп’ютером за допомогою жестів з використанням Python, MediaPipe та OpenCV. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається з вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено комплексний аналіз сучасного стану технології розпізнавання жестів, досліджено ключові досягнення та перспективи розвитку цієї галузі. Проаналізовано практичне застосування технології в різних сферах, від медицини до військової промисловості. Сформульовані завдання на кваліфікаційну роботу бакалавра.

У другому розділі проведено детальний аналіз технологій розпізнавання жестів та визначено основні функціональні вимоги до системи. Розроблено архітектуру програмного забезпечення з використанням UML діаграм для візуалізації алгоритму роботи системи. Обґрунтовано вибір технологічного стеку, зокрема мови програмування Python та бібліотек MediaPipe для відстеження рухів рук, OpenCV для комп’ютерного зору тощо.

У третьому розділі описано практичну реалізацію застосунку. Проведено тестування функцій програми, включаючи перевірку базових операцій миші й регулювання гучності. Створено виконуваний файл та інсталяційний пакет програми. Виконано перевірку точності розпізнавання жестів користувачами з різним обладнанням й умовами освітлення. Обґрунтовано перспективи та визначено напрямки вдосконалення програмного забезпечення.

Ключові слова: розпізнавання жестів, безконтактне керування, машинне навчання, комп’ютерний зір, Python, MediaPipe, OpenCV, pynput, pycaw, модульна архітектура, UML-діаграми, тестування програмного забезпечення.

ABSTRACT

Mironov N. Development of an Application for Computer Interaction Using Gestures with Python, MediaPipe and OpenCV. Manuscript. Qualification work of the bachelor's degree program "Software Engineering", specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's thesis consists of an introduction, three chapters, conclusions and suggestions, a list of references and appendices.

The first chapter provides a comprehensive analysis of the current state of gesture recognition technology, examines the key achievements and prospects for the development of this field. The practical application of the technology in various fields, from medicine to the military industry, is analyzed. The tasks for the bachelor's thesis are formulated.

The second chapter provides a detailed analysis of gesture recognition technologies and identifies the main functional requirements for the system. The software architecture is developed using UML diagrams to visualize the system's algorithm. The choice of the technology stack is substantiated, including the Python programming language and MediaPipe libraries for tracking hand movements, OpenCV for computer vision, etc.

The third chapter describes the practical implementation of the application. The program functions were tested, including basic mouse operations and volume control. The executable file and installation package of the program were created. The accuracy of gesture recognition by users with different equipment and lighting conditions was tested. Prospects and directions for improving the software are substantiated.

Keywords: gesture recognition, computer vision, Python, MediaPipe, OpenCV, pynput, pycaw, contactless control, machine learning, modular architecture, UML diagrams, software testing.

ЗМІСТ

ВСТУП

Актуальність теми. У наш час форми і можливості взаємодії з комп'ютером зазнають фундаментальних змін. Хоча використання клавіатури та миші ще досі лишається як найбільш оптимальний та класичний спосіб керування ЕОМ, дедалі зростає потреба переходу до більш природних та інтуїтивних форм взаємодії із системою.

Можливість розпізнавання жестів, завдяки технологіям комп'ютерного зору (Computer Vision, CV), є перспективним рішенням, яке знаходить практичне застосування на різних повсякденних пристроях – від маніпуляцій з курсором миші до ігрових інтерфейсів і систем керування розумним будинком.

Розпізнавання жестів вирішує реальні проблеми – від полегшення доступу до повсякденних пристроїв для людей з обмеженими фізичними можливостями до створення стерильних умов контролю в медичних установах.

У даній роботі поставлено за мету розробити програмний застосунок, який дозволятиме користувачеві взаємодіяти з комп'ютером за допомогою жестів для безконтактного керування мишею і регулювання гучності системи.

Об'єктом дослідження є застосунок для взаємодії з комп'ютером жестами.

Предметом дослідження є концепція, методи та алгоритми розпізнавання та інтерпретації рухів та жестів рук у реальному часі, функціональні можливості розроблюваного програмного застосунку.

Засобом досягнення мети є вирішення наступних завдань:

– обґрунтувати актуальність технології розпізнавання жестів та провести огляд сучасного стану проблеми;

– сформулювати вимоги до функціоналу застосунку та спроектувати концепцію системи розпізнавання жестів;

– обрати інструменти для розробки програмного забезпечення;

– розробити застосунок для жестової взаємодії з комп'ютером;

– провести тестування роботи і функціональності продукту;

– створити виконуваний файл та інсталяційний пакет програми;

– дослідити результати точності розпізнавання жестів;

– проаналізувати можливі шляхи розвитку системи та запропонувати рекомендації для вдосконалення функціонування застосунку.

Проведено апробації отриманих результатів роботи:

1) Міронов Н. О., Самчук Л. М. Дослідження характеристик та методології системи розпізнавання жестів для безконтактної взаємодії з комп’ютером з використанням технологій Computer Vision. Комп’ютерно-інтегровані технології: освіта, наука, виробництво. 2024. № 57. С. 111-119. URL: https://doi.org/10.36910/6775-2524-0560-2024-57-13 (додаток А);

2) Міронов Н. О., Самчук Л. М. Розробка застосунку для взаємодії з комп’ютером за допомогою жестів з використанням технологій Computer Vision. Комп’ютерно-інтегровані технології: освіта, наука, виробництво. 2025. № 58. С. 80-92. URL: https://cit.lntu.edu.ua/index.php/cit/issue/view/43 (додаток Б);

3) Міронов Н. О., Самчук Л. М. Тенденції застосування інструментів безконтактної взаємодії з пристроями за допомогою жестів з використанням технологій комп’ютерного зору і машинного навчання. Матеріали міжнародної науково-практичної конференції «Цифрова трансформація: виклики та стратегії», 25 лютого 2025 р., м. Луцьк: ЛНТУ, 2025. С. 166-168. URL: https://dtcs.lntu.edu.ua/tezy-konferencziyi/ (додаток В);

4) Міронов Н. О., Самчук Л. М. Застосунок для безконтактної взаємодії з пристроями за допомогою жестів. Тези доповідей X міжнародної науково-практичної конференції «Інформаційні технології в освіті, науці і виробництві», 23-24 травня 2025 р., м. Луцьк: ЛНТУ, 2025. С. 363-367. URL: https://itonv.lntu.edu.ua/files/2025/zbirnyk_itonv_2025.pdf (додаток Г).

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЖЕСТІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Комп’ютеризація в нашому повсякденному житті стає все більш помітною та значущою, а отже потреба в більш природних, інтуїтивно зрозумілих інтерфейсах зростає пропорційно. Механічні методи введення виявляються непрактичними в багатьох випадках, особливо в сценаріях, де важливо працювати без допомоги рук або де традиційні інтерфейси не передбачені. Водночас, зростаюча обчислювальна потужність пристроїв нарешті наздогнала обчислювальні вимоги розпізнавання жестів у реальному часі, що робить можливим широке розгортання технології.

Розпізнавання жестів – це технологія, що дозволяє комп’ютерним системам інтерпретувати рухи й жести рук для здійснення керування, виконання команд чи взаємодії з пристроями [1]. Вона надає можливість комп’ютерам реагувати на певні рухи, створюючи спосіб безконтактної взаємодії з технікою.

Ключовою технологією для реалізації розпізнавання жестів у реальному часі є комп’ютерний зір (CV) – галузь штучного інтелекту, що займається розробкою алгоритмів для аналізу та інтерпретації візуальної інформації. Технологія обробляє цифрові зображення або відеопотоки, виділяючи у них об’єкти та відстежуючи їх рух, а також просторові взаємозв’язки між ними. Це дозволяє комп’ютерам аналізувати й інтерпретувати візуальні дані аналогічно до того, як людський мозок обробляє інформацію, отриману зоровою системою.

Робота таких систем ґрунтується на алгоритмах [2], які здатні виявляти та інтерпретувати жести, зіставляючи їх із визначеними командами. Наприклад, CV може застосовувати комбінацію сенсорів і камер для відстеження положення та руху руки користувача. За допомогою методів машинного навчання такі системи навчаються розрізняти різні жести на основі аналізу великих наборів зображень з прикладами різних жестів у різних умовах. З часом система здатна навчитися з

високою точністю розпізнавати певні жести, забезпечуючи надійну та точну взаємодію користувача з пристроєм.

Розпізнавання жестів, завдяки інтуїтивності й наближеності до природної людської поведінки, відкриває нові можливості для безконтактної взаємодії. Наразі різні галузі, зокрема в автомобільній індустрії та сфері віртуальної і доповненої реальності, активно вивчають та впроваджують цю технологію. Сучасні автомобілі можуть бути оснащені даною можливістю для безпечного і зручного керування різними функціями. Водії можуть виконувати такі команди, як регулювання гучності стереозвуку, зміна налаштувань кондиціонера тощо, мінімізуючи відволікаючі фактори та підвищуючи зосередженість на водінні.

У середовищах віртуальної (VR) і доповненої (AR) реальності користувачі можуть маніпулювати об’єктами, переміщатися по меню, керувати програмами за допомогою жестів, створюючи більш захоплюючий та інтерактивний підхід до взаємодії з віртуальним світом без потреби в фізичних контролерах. Такі пристрої, як Microsoft Kinect, змінили ігрову індустрію, забезпечуючи керуванням рухом персонажів (рис. 1.1). Гравці використовують рухи свого тіла для взаємодії з віртуальним середовищем, що додає грі більшого занурення у неї.

Рисунок 1.1 – Застосування Microsoft Kinect в іграх [3]

Пандемія та війна прискорили розвиток і використання цих технологій, оскільки потреба в безконтактних інтерфейсах стала не просто питанням зручності, але й здоров’я громадян і, в умовах військових дій, полегшення доступу взаємодії з пристроями для людей з обмеженими матеріальними і фізичними можливостями.

Так само знаходить використання системи розпізнавання жестів у військовій справі. Одне з найдоцільніших застосувань – знешкодження бомб. Задля того, щоб не піддавати ризику людей-саперів, компанія DataGlove розробила спеціалізовані рукавички, які під’єднані до роботів-маніпуляторів. Рукавички (рис. 1.2) відстежують точне положення і обертання рук оператора, дозволяючи йому виконувати операції [4] з вибуховими речовинами і снарядами віддалено, без загрози за власне життя.

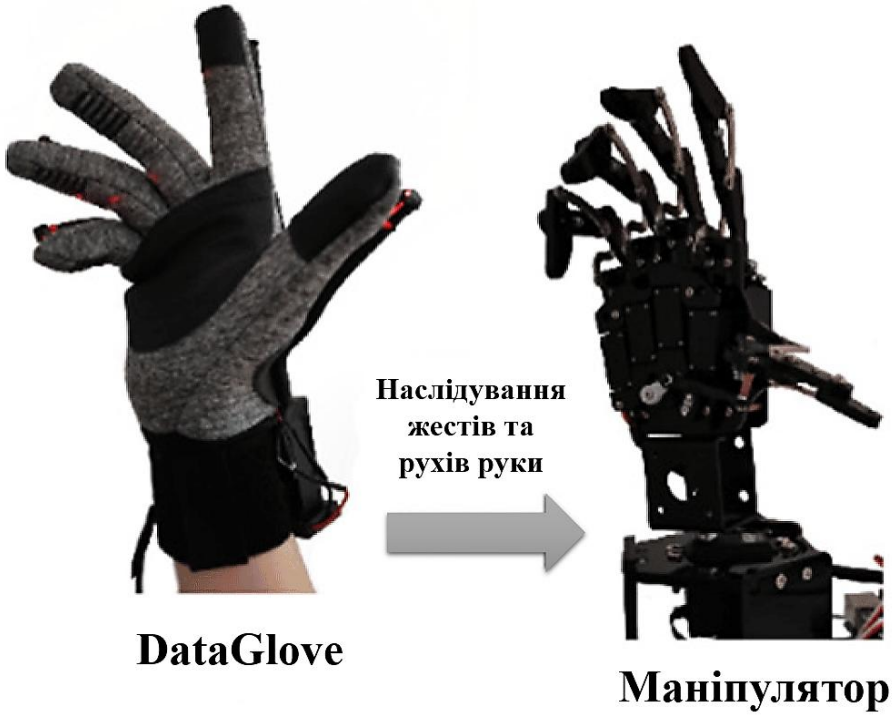


Рисунок 1.2 – DataGlove [4]

Ще одним прикладом використання технології в критичних випадках може слугувати медицина. Розпізнавання жестів може зробити революцію у виконанні та оцінці хірургічних процедур за допомогою роботизованих асистентів [5]. Замість того, щоб кликати медсестру до кожного інструменту, хірург просто

робить певні жести руками (рис. 1.3), які віртуальний асистент миттєво розпізнає і оперативно доставляє потрібний інструмент до нього. Це не лише економить дорогоцінний час, але й підтримує стерильне середовище в операційній.

Рисунок 1.3 – Технологія розпізнавання жестів у медицині [6]

Багато наукових праць відзначають переваги безконтактних інтерфейсів, особливо в умовах обмеженого простору або підвищеного ризику інфекцій. Так, у період COVID-19 дослідниками розроблено безконтактні комп'ютерні інтерфейси для зменшення ризику передачі вірусу [7]. Це підкреслює важливість технології у сучасному світі та потенціал для безпечної взаємодії з пристроями.

У розумних будинках розпізнавання жестів дозволяє користувачеві керувати всією домашньою технікою. Наприклад, помахуючи рукою, можна вмикати або вимикати світло, регулювати термостат та інші системи, легко інтегруючись із технологією розумного дому для підвищення зручності та ефективності. Із розвитком комп'ютерного зору надалі можна прогнозувати, що можливість розпізнавання жестів стане частиною повсякденної взаємодії з пристроями, роблячи її більш природною та інтуїтивною.

Технології розпізнавання жестів використовують алгоритми машинного навчання (Machine Learning, ML) для інтерпретації та класифікації людських жестів з високою точністю. Ці алгоритми тренуються на великих наборах даних

про шаблони жестів [8] і адаптуються для розпізнавання та навчання людських рухів. Алгоритми ML включають:

- згорткові нейронні мережі (Convolutional Neural Networks, CNN);
- опорні векторні машини (Support Vector Machines, SVM);
- приховані марковські моделі (Hidden Markov Models, HMM);
- дерева рішень (Decision Trees).

CNN застосовуються для розпізнавання жестів завдяки своїм можливостям аналізу візуальних даних і виявлення просторових ієрархій. Вони структурують зображення у вигляді багаторівневих фільтрів, де на перших визначаються основні елементи, такі як контури й текстури, а на вищих рівнях – складніші структури, такі як, власне, жести та рухи.

SVM, як правило, використовуються для класифікації жестів на основі виділених ознак, завдяки чому забезпечується точне розмежування між різними їх категоріями. Вони створюють оптимальні межі між різними категоріями жестів, ефективно перетворюючи складні дані, витягнуті з рухів рук, у чіткі, розпізнавані команди, які може інтерпретувати комп’ютер.

HMM виявляються ідеальними для розпізнавання послідовностей у жестах, оскільки здатні моделювати часові залежності та перехідні стани. Використовуючи ймовірнісні розподіли, вони можуть враховувати перехід від одного стану до іншого, що є важливим для розпізнавання жестів, що включають складні рухи.

Дерева рішень дозволяють реалізувати просту й зрозумілу класифікацію жестів, визначаючи категорію жесту на основі конкретних характеристик, таких як форма чи напрямок руху. Підхід використовує ієрархічну структуру, що дозволяє швидко обирати рішення на основі заданих умов або критеріїв.

Поточні дослідження і розробки в сфері розпізнавання жестів є своєчасними з огляду на збіг кількох факторів: вдосконалення алгоритмів комп’ютерного зору і машинного навчання, зростання потужних можливостей мобільних процесорів і поява нових технологій, де недоцільне використання традиційних інтерфейсів. Це створює оптимальне середовище для просування

систем розпізнавання жестів від перспективної технології до практичного, повсякденного інструменту.

Однак залишаються ще значні виклики. Хоча вже освоєно базові функції розпізнавання жестів у контрольованому середовищі, створення надійних систем, які б працювали в різних умовах освітлення та на різних відстанях, залишається непростою задачею. Сучасне покоління систем все ще бореться за точність у складних сценаріях.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Сформовано мету роботи, якою є розробка програмного забезпечення для взаємодії з комп’ютером за допомогою жестів рук, функціональність якої буде зосереджена на управлінні мишею та регулюванні гучності системи без фізичного контакту, задовольняючи потребу в безконтактних інтерфейсах та використовуючи технології комп’ютерного зору.

Відповідно до мети, визначено завдання, які виконуватимуться під час роботи над проєктом кваліфікаційної роботи бакалавра:

– обґрунтувати актуальність технології розпізнавання жестів та провести огляд сучасного стану проблеми;

– сформулювати вимоги до функціоналу застосунку та спроектувати концепцію системи розпізнавання жестів;

– обрати інструменти для розробки програмного забезпечення;

– розробити застосунок для жестової взаємодії з комп’ютером;

– провести тестування роботи і функціональності продукту;

– створити виконуваний файл та інсталяційний пакет програми;

– дослідити результати точності розпізнавання жестів;

– проаналізувати можливі шляхи розвитку системи та запропонувати рекомендації для вдосконалення функціонування застосунку.

Висновки до розділу 1

Було здійснено аналіз поточного стану технології розпізнавання жестів, визначено ключові аспекти і досягнення в цій галузі, наведено напрямки для подальшого вдосконалення і розвитку технології.

Розпізнавання жестів пропонує природний міст між людськими рухами та цифровим управлінням пристроїв, не застосовуючи механічних девайсів. Технологія дійшла до практики завдяки розвитку комп'ютерного зору та зростаючій обчислювальній потужності сучасних пристроїв. На сьогоднішній день галузь досягла кількох значних успіхів. Технології перейшли від простого виявлення руху до складної інтерпретації жестів, яка може розрізняти складні рухи пальців і положення рук.

Різноманітні застосування розпізнавання жестів в життя демонструють її універсальність і потенційний вплив у різних галузях. Від допомоги хірургам у підтримці стерильності в операційних до безпечної взаємодії з автомобільними системами під час руху – ці впровадження демонструють, як технологія може вирішувати реальні проблеми. Ці фактори перетворили розпізнавання жестів з цікавої технічної новинки на концепцію, що спричинило швидкий розвиток як базової технології, так і її практичних застосувань.

На основі аналізу, визначено мету роботи, зосередившись на розробці програмного забезпечення, яке дозволяє взаємодіяти з комп'ютером завдяки жестам. Окреслено конкретні завдання для реалізації рішення, який включає огляд проблеми й аргументування актуальності технології, в тому числі, аналіз концепції розпізнавання жестів, вибір відповідних інструментів для розробки, визначення вимог, а також функціональних можливостей розроблюваного продукту, створення застосунку та проведення ретельного тестування.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Проаналізувавши технологію розпізнавання жестів та зростаючий попит на безконтактні системи керування, вирішено розробити відповідний застосунок для керування комп'ютером. В основі лежить виявлення та відстеження рухів рук у реальному часі, а також розпізнавання та інтерпретування різних жестів одночасно. Тому система має працювати з відносно плавною частотою кадрів для забезпечення безперебійної й комфортної для користувача роботи.

Програмне забезпечення повинне підтримувати основні функції миші, від переміщення курсору та натискання як на ліву, так і на праву кнопки, до більш складних взаємодій, таких як перетягування об'єктів, прокручування сторінок вікон тощо. Особливістю даного програмного рішення є впровадження у систему додаткового функціоналу, такого як регулювання гучності комп'ютера.

Розпізнавання жестів вимагає можливостей відстеження положення пальців, здатних точно ідентифікувати й відстежувати їх положення одночасно, зберігаючи при цьому стабільність в різних умовах освітлення та навколишнього середовища. Важливим є впровадження алгоритмів відображення координат, які можуть плавно переводити положення рук у тривимірному просторі у двовимірні координати екрану, фільтруючи при цьому природне тремтіння рук і ненавмисні рухи.

Провівши аналіз поширених рухів рук і жестів, визначено, що жест одним пальцем буде основою для базової навігації, тоді як комбінації з кількох пальців дозволяють здійснювати більш складні взаємодії. Тому для керування курсором, як основної функції застосунку, обрано витягнутий вказівний палець (рис. 2.1).

Рисунок 2.1 – Вказівний палець

Логіка вибору пояснюється природним інстинктом людини вказувати на об’єкти, які її цікавлять. Система розпізнавання жестів відстежує кінчик вказівного пальця для визначення положення курсору, застосовуючи алгоритми згладжування для забезпечення стабільного руху.

Поєднання витягнутих вказівного та середнього пальців обрано для операції лівого кліку миші (рис. 2.2). Також вирішено, що захоплення та перетягування файлів і папок має бути реалізовано таким самим жестом, але стиснутими між собою пальцями, оскільки це нагадує природний хапальний рух.

Рисунок 2.2 – Вказівний та середній пальці

Функціональність правої кнопки повинна бути з використанням подібного жесту, але з додаванням безіменного пальця, що створює чітке розмежування між операціями клацання лівою і правою кнопкою миші (рис. 2.3).

Рисунок 2.3 – Вказівний, середній та безіменний пальці

Функція прокручування має бути жестом зімкнутого кулака з положенням великого пальця, що визначає напрямок (рис. 2.4). Це забезпечує інтуїтивний спосіб прокрутки вмісту вікон, не вимагаючи складних рухів рукою.

Рисунок 2.4 – Великий палець

Для регулювання гучності, вертикальний рух руки має змінювати цей рівень звуку, використовуючи всі витягнуті пальці, окрім великого (рис. 2.5). Цей жест відмінний від інших команд через застосування для функції всієї руки і водночас відповідає концепції збільшення або зменшення величини гучності.

Рисунок 2.5 – Вказівний, середній, безіменний пальці та мізинець

Реалізація системи управління на основі жестів створює унікальні ситуації та варіанти в логіці переходів між функціями. Вибрані жести для відповідної команди повинні утворювати цілісну мову жестової взаємодії, яку користувачі можуть швидко освоїти та використовувати у своїй повсякденній роботі.

Для нотації послідовності дій користувача та представлення етапів процесу взаємодії за допомогою жестів вибір зроблено на користь UML діаграми станів. Це обумовлено здатністю діаграми моделювати переходи між процесами і одночасну поведінку системи, притаманну архітектурі розпізнавання жестів у реальному часі.

Як ілюструє рисунок 2.6, коли користувач запускає застосунок, камера активується. У цей момент програма постійно аналізує відеопотік, шукаючи руку користувача і визначаючи її ключові точки.

Коли рука потрапляє в поле зору камери, система починає відстежувати її положення і конфігурацію пальців, після чого користувач виконує команди керування за допомогою відповідних жестів.

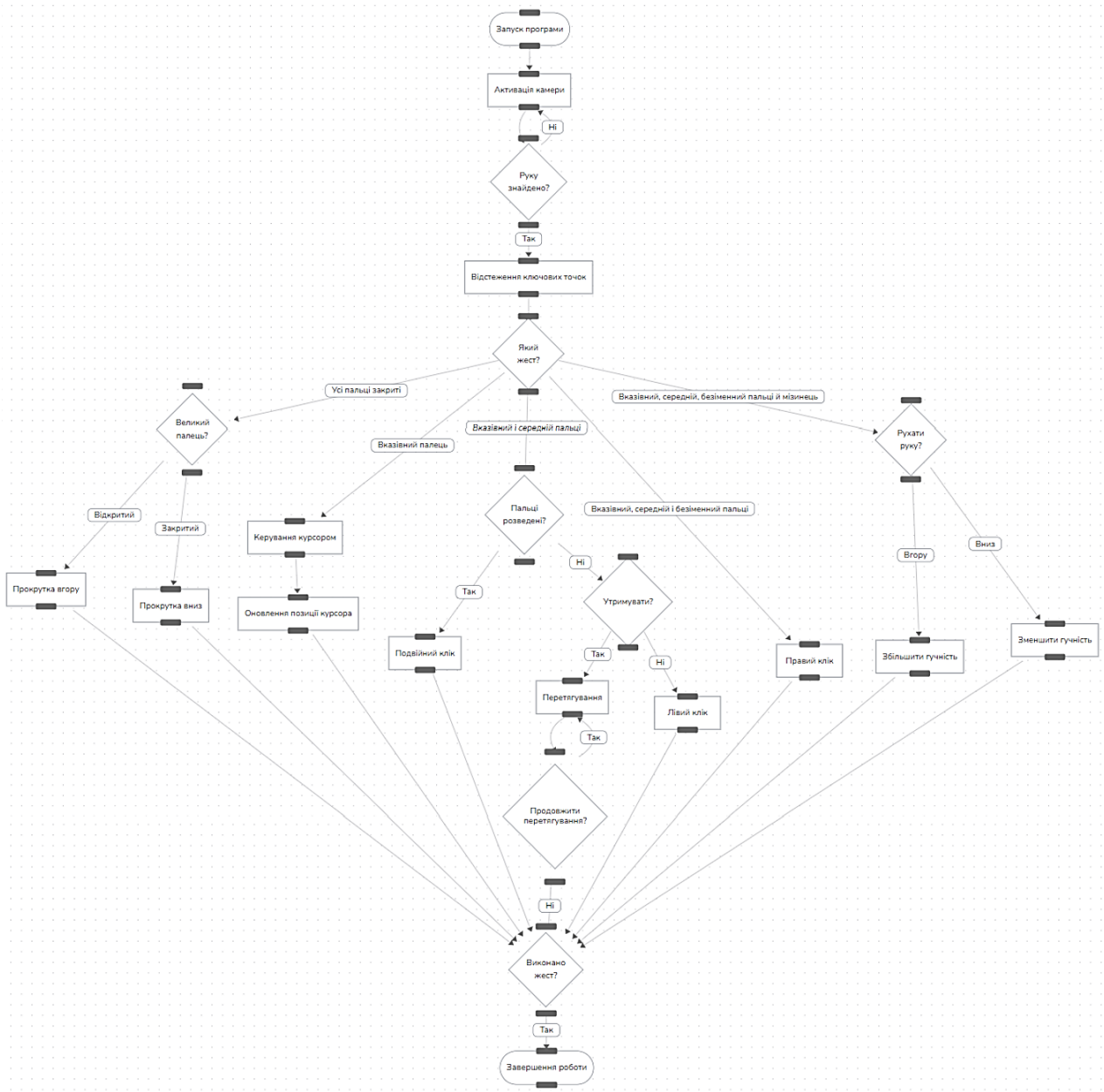


Рисунок 2.6 – Діаграма станів керування комп’ютером на основі жестів

Діаграма показує, що керування курсором активується підняттям вказівного пальця при зігнутих інших пальцях. Для натискання лівої кнопки миші користувач піднімає вказівний і середній пальці, а для правої додатково безіменний палець. Прокручування здійснюється жестом із закритими пальцями,

де положення великого пальця визначає напрямок прокрутки. Регулювання гучності відбувається підняттям чотирьох пальців, а вертикальне переміщення руки змінює рівень звуку.

Опис дій алгоритму даної системи керування комп’ютером на основі жестів наведено в таблиці 2.1.

Таблиця 2.1 – Опис дій системи діаграми станів

Дія	Опис
Запуск програми	Запускається застосунок керування жестами
Активація камери	Вмикається камера для відстеження рук
Руку знайдено?	Система перевіряє наявність рук в кадрі
Відстеження ключових точок	Система визначає ключові точки руки
Який жест?	Система аналізує поточне положення руки для визначення жесту
Великий палець?	Перевіряється положення великого пальця для визначення напрямку прокрутки
Прокрутка вгору	Виконується прокрутка сторінки вгору
Прокрутка вниз	Виконується прокрутка сторінки вниз
Керування курсором	Відслідковується рух вказівного пальця для переміщення курсору
Оновлення позиції курсору	Оновлюються координати курсору відповідно до руху руки
Пальці розведені?	Перевіряється відстань між пальцями для визначення команди
Подвійний клік	Проводиться команда подвійного кліку лівої кнопки миші
Утримувати?	Перевіряється жест для перетягування
Перетягування	Проводиться команда перетягування
Продовжити перетягування?	Перевіряється, чи продовжується перетягування
Лівий клік	Проводиться команда лівої кнопки миші
Правий клік	Проводиться команда правої кнопки миші
Рухати руку?	Перевіряється наявність руху руки вгору/вниз
Збільшити гучність	Підвищується гучність при відповідному жесті
Зменшити гучність	Знижується гучність при відповідному жесті
Виконано жест?	Перевіряється завершення виконання жесту
Завершення роботи	Завершується робота програми

Модульна конструкція системи забезпечує можливості для розширення в майбутньому. Універсальність та доступність системи для більшого кола користувачів можна забезпечити за допомогою кросплатформенного рішення для різних операційних систем. Інтеграція з додатковими елементами керування комп'ютером та реалізація додаткових функцій взаємодії на основі жестів – це логічні шляхи розвитку можливостей системи.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки застосунку, який дозволятиме взаємодіяти з комп'ютером за допомогою жестів, можна використовувати різні мови програмування і відповідні їм бібліотеки. Щоб визначити, які інструменти є найбільш оптимальними та ефективними для обробки й розпізнавання жестів, проведено огляд кількох таких мов, що дозволяють реалізувати дане програмне рішення.

Досліджено наступні мови програмування: C++, JavaScript, Java, та Python. Огляд, переваги та недоліки кожного з них наведено далі.

C++ являє собою мову системного програмування, високопродуктивних додатків та програмного забезпечення, що вимагають тонкого контролю над пам'яттю та обробки даних [9]. Вона має підтримку комп'ютерного зору, а отже її можна використовувати для взаємодії на основі жестів.

Однак, розробка на C++, зазвичай, займає більше часу через її складність і часто вимагає написання додаткового коду для керування пам'яттю та виконання завдань системного рівня, які в інших мовах вирішуються автоматично. Крім того, інтеграція складних функцій комп'ютерного зору може бути складним завданням, оскільки вимагає ретельного управління залежностями.

Переваги C++:

- висока продуктивність і швидкість роботи;
- повний контроль над розподілом пам'яті та керування нею;
- велика кількість бібліотек та багаточисельна спільнота розробників.

Недоліки C++:

- складність мови та синтаксису для початківця;
- більше рядків коду для написання певних функцій;
- довший час розробки відносно інших мов;
- складна та трудомістка для крос-платформного прототипування.

Java – об’єктно-орієнтована мова [10], яка дозволяє запускати застосунки на пристроях, що мають віртуальну машину Java (JVM). Вона використовується для розробки крос-платформних додатків і завдяки бібліотекам для обробки зображень та комп’ютерного зору, таких як JavaCV, Java може бути варіантом для програм, заснованих на жестах, особливо тих, які працюють на декількох платформах. Здатність інтегруватися з базами даних і внутрішніми системами також робить її гарним вибором для додатків, які можуть потребувати обробки на стороні сервера або додаткових можливостей керування даними.

Проте JVM вносить певні накладні витрати, які можуть вплинути на продуктивність завдань обробки відео. Досягнення плавної роботи для відстеження жестів може вимагати значних зусиль з оптимізації. Вона може добре працювати для додатків, де швидкість роботи в реальному часі не є найвищим пріоритетом.

Переваги Java:

- крос-платформна підтримка додатків;
- інтеграція та взаємодія з базами даних;
- обширна документація та підтримка спільноти.

Недоліки Java:

- обмежена продуктивність у реальному часі;
- складний для початківця синтаксис;
- довший час розробки;
- обмежена кількість спеціалізованих бібліотек.

JavaScript, в поєднанні з HTML5, CSS, а також фреймворками, переважно використовують для веб-розробки. Однак, з розвитком можливостей браузерів і таких бібліотек, як TensorFlow.js [11], вона стала реальним варіантом для

створення інтерактивних браузерних додатків, навіть у тому числі й тих, що використовують розпізнавання жестів. Подієво-орієнтована архітектура мови дозволяє обробляти взаємодію з користувачем, що робить його придатним для зворотного зв’язку з жестами в браузері в режимі реального часу.

Однак він може мати проблеми з високошвидкісним розпізнаванням жестів через обмеження продуктивності в середовищі браузера. Для легких додатків або прототипів JavaScript може підійти, але для більш складних додатків він дещо поступається іншим доступним рішенням. Так само доступ до камери через браузер може бути непослідовним, що може створювати проблеми для роботи додатків у режимі реального часу.

Переваги JavaScript:

– крос-платформенна доступність;
– простота розгортання та оновлення застосунків;
– велика кількість бібліотек.

Недоліки JavaScript:

– обмеження продуктивності;
– складнощі в доступі до елементів керування системного рівня;
– зазвичай, додатки потребують постійний доступ до мережі;
– можливі проблеми з доступом до камери;
– залежність від сумісності з браузером.

Python – це інтерпретована мова програмування, яка, завдяки своїй великій кількості бібліотек і підтримці спільноти розробників [12], набула популярності у сферах машинного навчання та комп’ютерного зору. Гнучкість мови і простота інтеграції з різними бібліотеками дозволяють швидко розробляти застосунки, які потребують складної функціональності. Python добре підходить для створення програм на основі жестів, які передбачають складну обробку даних та машинне навчання [13]. Такі бібліотеки, як MediaPipe та OpenCV, дозволяють розробити дані системи відносно швидко.

Переваги Python:

– широка підтримка з боку спільноти і велика кількість бібліотек;

– простий та зрозумілий для новачка синтаксис;
– менше рядків коду для написання функцій;
– швидке створення прототипу і розгортання готового продукту;
– крос-платформна сумісність.

Недоліки Python:

– низька швидкість виконання відносно інших мов, таких як C++ ;
– залежність від зовнішніх бібліотек та проблеми з їх сумісністю;
– менш ефективне використання пам’яті.

Щоб узагальнити сильні та слабкі сторони кожного інструменту і зробити остаточний вибір мови програмування для розробки за відповідними критеріями складено порівняльну таблицю (табл. 2.2).

Таблиця 2.2 – Порівняння мов програмування за критеріями

Критерій	C++	Java	JavaScript	Python
Висока продуктивність	+	–	–	–
Обробка відео в реальному часі	+	–	–	+
Крос-платформна сумісність	–	+	+	+
Простота розробки	–	–	+	+
Потужні CV-бібліотеки	+	+	–	+
Простота розгортання	–	–	+	+
Швидке створення прототипів	–	+	+	+

Таким чином, описавши і визначивши плюси та мінуси, а також виконавши порівняння розглянутих засобів, прийнято рішення, що застосунок для взаємодії з комп’ютером на основі жестів буде розроблено мовою програмування Python. Її бібліотеки машинного навчання та комп’ютерного зору дозволять легко створити систему розпізнавання жестів. MediaPipe та OpenCV оптимізовані для обробки зображень та відео і дозволяють ефективно розпізнавати жести.

Крім того, Python дозволяє швидко створювати прототипи, а це означає, що можна тестувати функціонал коду і швидко його коригувати. Така гнучкість дуже корисна на ранніх стадіях розробки. Простий синтаксис мови полегшує

створення прототипу і дозволяє постійне вдосконалення та налаштування програми. А також існує велика кількість ресурсів, документації та бібліотек, що полегшує подолання проблем, які можуть виникнути під час розробки.

Як зазначалося раніше, Python підтримує величезну кількість бібліотек, в тому числі, для розпізнавання жестів та взаємодії з комп'ютером технологіями CV. Основні бібліотеки, що використано для розробки програмного забезпечення, включають: MediaPipe – аналізує та відстежує рухи рук і тіла в режимі реального часу, OpenCV – методами комп'ютерного зору захоплює та інтерпретує відео з камери, pynput – дозволяє перетворювати розпізнані жести на програмне керування пристроями. Більш детальний огляд даних бібліотек наведено далі.

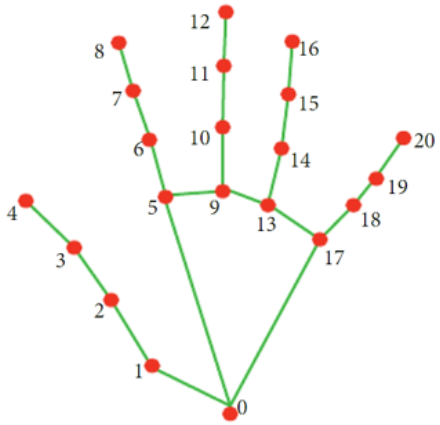
Основою системи керування комп'ютером на основі жестів є бібліотека з відкритим вихідним кодом [14] MediaPipe, завдяки якій застосунок здатен досить точно розпізнавати та відстежувати положення рук в режимі реального часу. Процес починається з виявлення руки на зображенні, що надходить з камери, після чого бібліотека застосовує алгоритми для розпізнавання руки та її ключових точок.

MediaPipe використовує метод сегментації, який розподіляє зображення на окремі області для виявлення ділянки, що може містити руку. Застосовується алгоритм обмежувальних рамок, що окреслює руку на зображенні, ізолюючи її від інших об'єктів у кадрі.

Наступним кроком є визначення ключових точок руки. Для цього MediaPipe використовує метод регресії, що побудований на основі глибоких навчальних моделей, здатних точно передбачити положення кожної з цих точок. Відбувається прогноз координат кожної ключової точки на основі зображення і вхідних даних з попередніх кадрів.

MediaPipe надає потужні інструменти для визначення до 21 ключової точки на руці, включаючи суглоби пальців та основу зап'ястя, що дозволяє створити деталізовану модель руки. Ця модель (рис. 2.7) використовується програмою для інтерпретації жестів користувача, що дозволяє перетворювати

фізичні рухи в інтуїтивні команди для комп’ютера.



0. Зап’ясток
1. Основна фаланга великого пальця
2. Середня фаланга великого пальця
3. Проксимальна фаланга великого пальця
4. Кінчик великого пальця
5. Основна фаланга вказівного пальця
6. Середня фаланга вказівного пальця
7. Проксимальна фаланга вказівного пальця
8. Кінчик вказівного пальця
9. Основна фаланга середнього пальця
10. Середня фаланга середнього пальця
11. Проксимальна фаланга середнього пальця
12. Кінчик середнього пальця
13. Основна фаланга безіменного пальця
14. Середня фаланга безіменного пальця
15. Проксимальна фаланга безіменного пальця
16. Кінчик безіменного пальця
17. Основна фаланга мізинця
18. Середня фаланга мізинця
19. Проксимальна фаланга мізинця
20. Кінчик мізинця

Рисунок 2.7 – Модель руки MediaPipe

Бібліотека використовує методи ML для обробки зображень і працює як зі статичними даними, так і з безперервним потоком [15]. Завдяки цьому можна отримувати координати рук на зображеннях, визначати праву чи ліву руку, категорії жестів для різних рук тощо.

В той час, як MediaPipe забезпечує відстеження рук, OpenCV [16], завдяки технологіям комп’ютерного зору, захоплює рухи та виконує операції, пов’язані з обробкою зображень, такі як виявлення контурів, сегментація картини, розпізнавання об’єктів тощо. Вона інтерпретує отриманий відеопотік для передачі та аналізу цих даних надалі іншим складовим системи.

Для відстеження рук і жестів, OpenCV використовує алгоритми оптичного потоку, зокрема метод Лукаса-Канаде [17]. Цей метод визначає переміщення ключових точок (наприклад, пальців чи контурів руки) між послідовними кадрами відео. Він розраховує вектор зміщення для кожної виділеної точки, що

дозволяє системі відстежувати траєкторію руху рук у просторі.

Бібліотека надає інструменти для обробки зображень, які дозволяють маніпулювати та готувати відеокадри для подальшого аналізу. Сюди входять перетворення колірного простору, зменшення шуму та інші кроки попередньої обробки, які оптимізують зображення для алгоритмів відстеження рухів.

Після відстеження рухів рук і розпізнавання жестів, лишається інтегрувати їх з фактичною функціональністю управління комп'ютером. Це дозволяє реалізувати бібліотека pynput [18], яка забезпечує доступ до низькорівневих подій введення та передає команди системі введення-виведення комп'ютера.

Наприклад, бібліотека дозволяє програмі позиціонувати курсор у точці, на яку вказує користувач у реальному світі, і робить це, перетворюючи координати, виявлені MediaPipe, на позиції на екрані. Pynput відстежує рухи курсору та реагує на швидкі зміни в позиціях пальців, забезпечуючи реалістичне відображення дій користувача на екрані.

Окрім позиціонування курсору, є можливість симулювати натискання кнопок миші. Після того, як система розпізнає відповідний жест, pynput генерує подію натискання або відпускання кнопки. Це дозволяє системі, наприклад, імітувати правий або лівий клік, залежно від заданого жесту, і, таким чином, інтегрувати жести з функціями управління комп'ютером.

У сукупності, ці три бібліотеки Python – MediaPipe, OpenCV та pynput – утворюють взаємозв'язок системи керування жестами в застосунку. Таким чином, MediaPipe виконує розпізнавання та відстеження рухів рук, OpenCV відповідає за обробку відео, звідки отримано розпізнані рухи, а pynput забезпечує передачу розпізнаних жестів до системи керування курсором та симуляції кліків.

Даний зв'язок між бібліотеками, який дозволяє користувачеві взаємодіяти з комп'ютером за допомогою жестів, представлений у вигляді UML-діаграми прецедентів. Вона ілюструє етапи процесу взаємодії, починаючи з ініціації користувачем і закінчуючи виконанням фактичних комп'ютерних команд, що були сформовані на основі аналізу рухів руки.

Діаграма (рис. 2.8) зображує процес, у якому користувач виконує команди

за допомогою жестів, а система розпізнає та обробляє ці сигнали. Дії на кшталт керування курсором, натискання кнопок миші, скролінг або регулювання гучності описані через взаємодію з відповідними функціями застосунку [19].

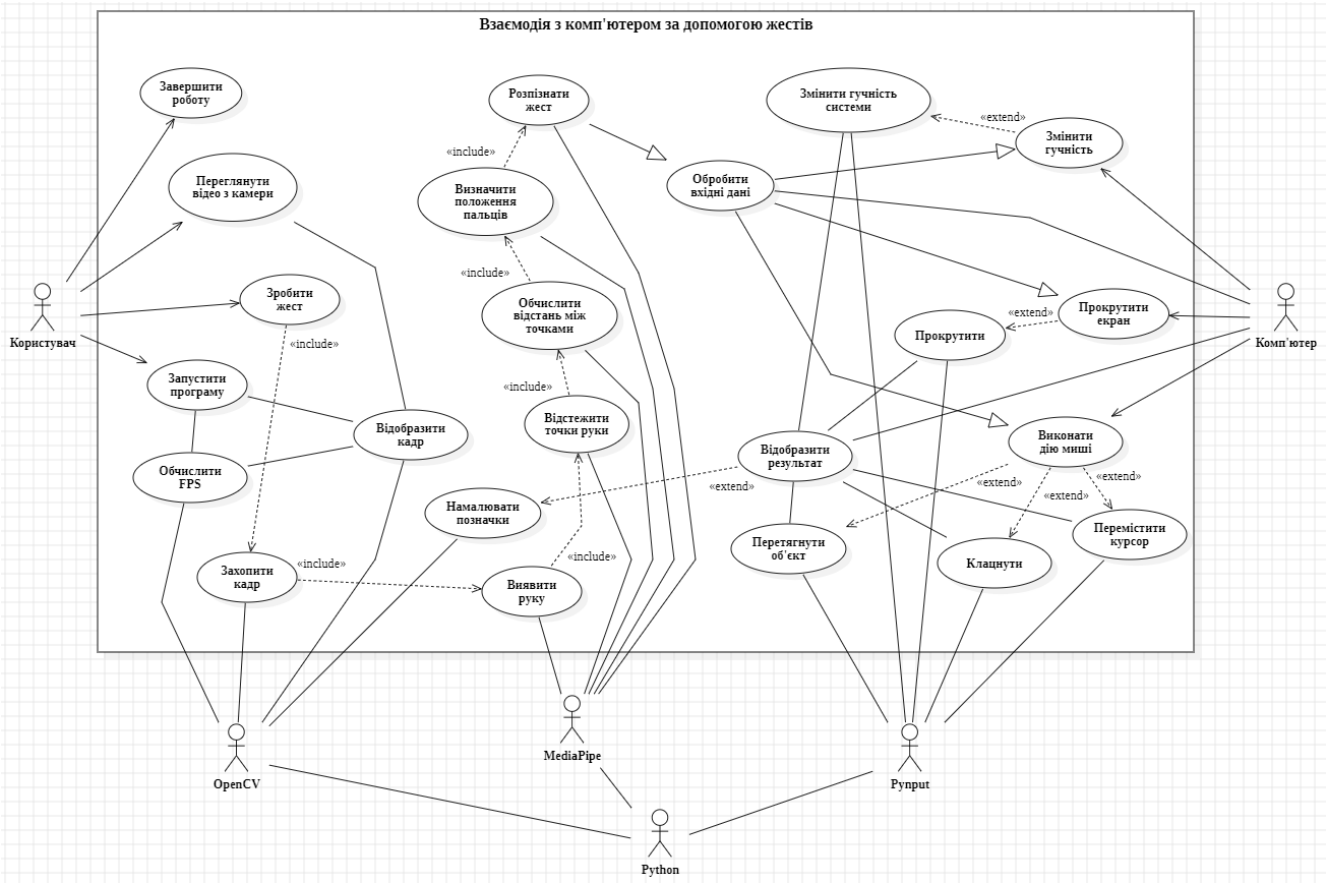


Рисунок 2.8 – Діаграма прецедентів процесу взаємодії з комп’ютером за допомогою жестів

За допомогою діаграми прецедентів, деталізовано основні етапи взаємодії користувача із системою, починаючи з ініціалізації застосунку та захоплення кадру, через покрокову обробку сигналів (детекція руки, відстеження точок, класифікація позицій пальців), до інтерпретування розпізнаного жесту як команди операційній системі – наприклад, для переміщення курсору, виконання регулювання рівня гучності.

Кожен із цих прецедентів описує окремий функціональний модуль, що забезпечує точність обробки жестових даних на всіх рівнях системи. Усі ключові сценарії взаємодії у системі узагальнено в таблиці 2.3.

Таблиця 2.3 – Опис прецедентів системи

Прецедент	Опис прецеденту
Запустити програму	Ініціалізація, що включає активацію камери, завантаження бібліотек та підготовку системи до розпізнавання жестів
Зробити жест	Виконання руху рукою перед камерою для інтерпретації як команди надалі
Переглянути відео з камери	Відображення відео в реальному часі з камери, що дозволяє системі «бачити» руки та жести
Завершити роботу	Завершення роботи системи, що включає вивільнення ресурсів, закриття відеопотоку тощо
Обробити вхідні дані	Аналізу відеопотоку з камери, виділення кадрів та їх попередня обробка для аналізу жестів надалі
Виконати дію миші	Реалізація основних команд миші, таких як переміщення курсору, клік, подвійний клік або перетягування об’єктів
Змінити гучність	Регулювання системної гучності на основі жестів
Прокрутити екран	Реалізація прокрутки вмісту вікна на основі жестів
Відобразити результат	Візуалізація виконаних дій на екрані
Виявити руку	Ідентифікація наявності руки в кадрі з камери
Відстежити точки руки	Визначення положення точок руки у просторі
Визначити положення пальців	Аналіз розташування і положення точок для визначення, які пальці підняті, а які опущені
Розпізнати жест	Інтерпретація руки та пальців як конкретного жесту, що відповідає певній команді
Обчислити відстань між точками	Розрахунок відстаней між точками руки для визначення жестів
Захопити кадр	Отримання поточного кадру з камери для обробки
Намалювати позначки	Додавання візуальних елементів на кадр для відображення розпізнаних рук і точок
Відобразити кадр	Показ кадру з візуальними позначками
Обчислити FPS	Розрахунок кількості кадрів за секунду та відображення цієї інформації
Перемістити курсор	Зміна позиції курсору на екрані відповідно до руху руки
Клацнути	Виконання кліку на основі розпізнаного жесту
Перетягнути об’єкт	Імітація утримання кнопки миші для переміщення об’єктів
Прокрутити	Виконання прокрутки вмісту активного вікна
Змінити гучність системи	Регулювання рівня гучності комп’ютера

Окрім прецедентів, наведено ключові ролі акторів у системі взаємодії з комп’ютером за допомогою жестів. До них належать: ініціатор взаємодії (користувач), виконавча складова (комп’ютер), мова програмування Python з її бібліотеками MediaPipe, OpenCV та Pynput.

Кожен актор у діаграмі прецедентів відповідає за окремий елемент процесу, забезпечуючи цілісність і узгодженість взаємодії. Опис акторів системи зведено в таблиці 2.4.

Таблиця 2.4 – Опис ролі акторів системи

Актор	Опис ролі
Користувач	Виконує жести перед камерою, які система інтерпретує як команди для керування комп'ютером
Комп'ютер	Виконує команди, такі як рух курсору, кліки, прокрутка, зміна гучності тощо, а також надає зворотний зв'язок через інтерфейс
Python	Основа для інтеграції обробки даних та реалізації логіки управління, відповідає за координацію компонентів і забезпечує функціонування програми; у цій системі Python розділений на три основні бібліотеки: MediaPipe, OpenCV та pynput
MediaPipe	Бібліотека, яка відповідає за розпізнавання жестів, аналізує потік з камери, виявляє руки, визначає положення точок, інтерпретує жести
OpenCV	Бібліотека, яка використовується для захоплення та обробки відео з камери, малювання на кадрах та відображення результатів
Pynput	Бібліотека для взаємодії з інтерфейсом, виконує фактичні дії та команди на комп'ютері, на основі розпізнаних жестів

Додатково, буде доречно згадати бібліотеку, яка надає можливість керувати гучністю комп'ютера за допомогою жестів. Це дозволяє реалізувати бібліотека pycaw, яка взаємодіє з налаштуваннями аудіо безпосередньо в операційній системі. Завдяки ній програма визначає поточний стан аудіосистеми та безпосередньо змінює його на основі відповідних жестів.

Основний метод, на якому базується робота бібліотеки – це взаємодія з Windows Core Audio API [20], який забезпечує доступ до аудіосесій ОС, включно з можливістю управління параметрами звуку. Це дозволяє регулювати гучність системи без необхідності ручного налаштування через додаткові інструменти або стороннє програмне забезпечення.

Аналізуючи можливі середовища розробки для Python, було розглянуто два варіанти – Visual Studio Code (VS Code) та PyCharm. Перший вирізняється високою гнучкістю та широкими можливостями налаштування завдяки безлічі доступних плагінів, що дозволяє адаптувати його під різні мови програмування. Крім того, VS Code [21] є легким і швидким середовищем, яке чудово підходить для швидкого розроблення проєктів на менш потужних комп'ютерах.

З іншого боку, PyCharm – це професійний інструмент, орієнтований на Python-розробку, що пропонує розширені можливості для роботи з мовою [22]. Завдяки інтегрованому аналізу коду, вбудованим інструментам для тестування

та відлагодження, PyCharm забезпечує високу якість та ефективність програмування. Однак, ця IDE є важчою та вимогливою до системних ресурсів, що може впливати на швидкість роботи, особливо на слабших апаратах.

У кінцевому підсумку, вибір зроблений на користь VS Code, зважаючи на його гнучкість, універсальність, швидкість компіляції, легкість в роботі й освоєнні, а також простоту в інтеграції з іншими технологіями та інструментами, що робить його оптимальним варіантом для розробки.

Висновки до розділу 2

Проаналізувавши технології розпізнавання жестів та затребувані дії користувача для взаємодії з комп'ютером, вирішено, що дана система повинна забезпечувати функціональність для управління курсором миші, клацання лівою і правою кнопкою миші, прокручування сторінок, а також більш просунуті дії, такі як перетягування об'єктів і регулювання гучності комп'ютера. Базова навігація реалізована за допомогою жесту витягнутого вказівного пальця, в той час як комбінації пальців використовуються для більш складних операцій.

Для моделювання системи використано UML діаграму станів, оскільки вона дозволяє представити переходи станів, алгоритм роботи та реакції на різні жести. Крім цього, архітектура спільної роботи компонентів системи та процес жестового керування комп'ютером також представлені у вигляді UML діаграми прецедентів, яка ілюструє як користувач взаємодіє з системою.

Після аналізу і порівняння різних мов програмування для розробки програми, обрано Python. Вона пропонує бібліотеки для CV та ML, такі як MediaPipe, OpenCV та pynput, які дозволяють реалізувати розпізнавання та інтерпретацію жестів. MediaPipe використана для реалізації відстеження рухів рук і розпізнавання жестів, OpenCV для захоплення та обробки відеопотоку з виявленими руками, pynput для інтеграції розпізнаних жестів з функціями управління комп'ютером, pycaw для регулювання гучності. Visual Studio Code використано як середовище розробки програмного застосунку.

РОЗДІЛ 3
РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВЗАЄМОДІЇ З КОМП'ЮТЕРОМ ЗА ДОПОМОГОЮ ЖЕСТІВ

3.1 Практична реалізація об'єкта проєктування

У процесі практичної реалізації програмного забезпечення для керування комп'ютером за допомогою жестів застосовано середовище розробки Visual Studio Code як основну платформу для написання, тестування та налагодження коду. Базовим інтерпретатором обрано Python версії 3.8.5. Попередньому етапу розробки передувала підготовча фаза, яка включала інсталяцію Python, конфігурування VS Code та імпортування необхідних залежностей і бібліотек.

Запустивши Visual Studio Code та перейшовши до папки проєкту, створено головний файл «main.py». Тут також створено папку «modules», у якій знаходяться додаткові файли для підключення до основного. Модулі відповідають за розпізнавання та відстеження руки «hand_detector.py», налаштування екрана і вікна програми «screen_settings.py», управління мишею «mouse_control.py» та керування гучністю «volume_control.py». Початкова структура каталогів і файлів застосунку показана на рисунку 3.1.

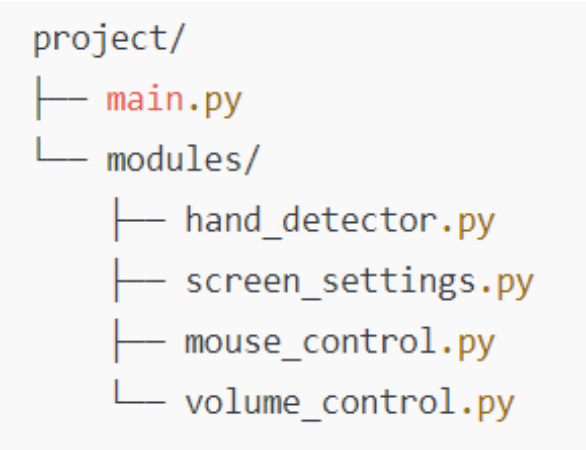


Рисунок 3.1 – Початкова структура каталогів і файлів програми

Відкривши головний файл програми, для початку, виконано імпорт бібліотек. Бібліотека комп'ютерного зору OpenCV, яка підключається як «cv2»,

відповідає за обробку відеопотоку, а «keyboard» необхідна для відстеження натискань клавіш. У файлі «main.py», окрім бібліотек, також імпортовано модулі з відповідної папки.

Далі створено функцію «main()», у якій прописано основний код цього файлу. У функції ініціалізовано компоненти підключених модулів. Для деяких компонентів передбачено конфігурацію (ліст. 3.1), як-от для «HandDetector()» налаштовано виявлення однієї руки та оптимізовано метод «MouseControl()», з урахуванням фактичних розмірів екрану.

Лістинг 3.1 – Імпорт та ініціалізація компонентів у файлі «main.py»

```
import cv2
import keyboard
from modules.hand_detector import HandDetector
from modules.mouse_control import MouseControl
from modules.volume_control import VolumeControl
from modules.screen_settings import ScreenSettings

def main():
    screen_settings = ScreenSettings()
    detector = HandDetector(maxHands=1, detectionCon=0.5)
    mouse_control = MouseControl(screen_settings.wScr,
screen_settings.hScr)
    volume_control = VolumeControl()
```

кінець лістингу 3.1

У функції створено цикл обробки, який реалізує безперервне отримання та аналіз відеопотоку. Кожен кадр проходить через процес нормалізації, включаючи підтримку коректного співвідношення сторін. Для отримання поточного розміру вікна та його ручної зміни, встановлено відповідні змінні.

Таким же чином підключаються елементи інтерфейсу, такі як показ частоти кадрів (FPS) та шкала гучності для візуалізації зміни рівня звуку. Встановлено і налаштовано прямокутну рамку для відображення області керування жестами. Код з підключенням змінних у циклі функції та рамки наведено в лістингу 3.2.

Лістинг 3.2 – Цикл обробки кадрів у файлі «main.py»

```
while True:
        img, frameReduction = screen_settings.get_frame()
        if img is None:
            break
        screen_settings.enforce_aspect_ratio()
        window_height = screen_settings.get_window_height()
        cv2.rectangle(img, (frameReduction, frameReduction),
                           (screen_settings.wCam - frameReduction,
screen_settings.hCam - frameReduction),
                           (255, 0, 255), 2)
        img = volume_control.draw_volume_interface(img,
window_height)
        img = screen_settings.draw_fps(img)
        cv2.imshow(screen_settings.window_name, img)
        screen_settings.position_window()
```

кінець лістингу 3.2

У циклі обробки система виконує виявлення рук та визначення положення точок пальців. Також ініціалізовано пальці руки (ліст. 3.3).

Лістинг 3.3 – Виявлення рук та ініціалізація пальців у файлі «main.py»

```
img = detector.findHands(img)
lmList = detector.findPosition(img, draw=False)
if len(lmList) != 0:
            x0, y0 = lmList[4][1:]
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]
            x3, y3 = lmList[16][1:]
            x4, y4 = lmList[20][1:]
            fingers = detector.fingersUp()
```

кінець лістингу 3.3

Підключено команди, кожна з яких активується тим чи іншим жестом. Переміщення курсору миші активується підняттям вказівного пальця. Кліки та перетягування відбуваються при піднятті вказівного та середнього пальців. Прокрутка активується, коли всі пальці опущені, а напрямок контролюється великим пальцем. Контроль гучності стає доступним при піднятті всіх пальців крім великого. Код призначення команд певному жесту показано в лістингу 3.4.

Лістинг 3.4 – Призначення команд певному жесту у файлі «main.py»

```
if fingers[1] == 1 and fingers[2] == 0 and fingers[3] == 0:
                coords = mouse_control.handle_mouse_movement(x1, y1,
frameReduction, screen_settings.wCam, screen_settings.hCam)
                cv2.circle(img, coords, 7, (255, 0, 255), cv2.FILLED)
if fingers[1] == 1 and fingers[2] == 1:
                img = mouse_control.handle_clicking_dragging(
                    fingers, detector, img, frameReduction,
 screen_settings.wCam, screen_settings.hCam)
else: mouse_control.reset_dragging()
if fingers[1] == 0 and fingers[2] == 0 and fingers[3] == 0 and fingers[4]
== 0:
                cv2.circle(img, (x0, y0), 7, (255, 0, 0), cv2.FILLED)
                mouse_control.handle_scrolling(fingers[0] == 1)
if fingers[1] == 1 and fingers[2] == 1 and fingers[3] == 1 and fingers[4]
== 1: volume_control.handle_volume_gesture(lmList)
```

кінець лістингу 3.4

У файлі «hand_detector.py» підключено бібліотеки для комп'ютерного зору, обчислень при відстеженні рук та координат розташування пальців, та «mediapipe», який дає змогу відстежувати руки. Створено клас «HandDetector()», у якому здійснено конфігурацію параметрів відстеження. Також визначено індекси кінчиків пальців, які використовуються для жестів (ліст. 3.5).

Лістинг 3.5 – Конфігурація відстеження рук у файлі «hand_detector.py»

```
class HandDetector:
    def __init__(self, mode=False, maxHands=1, detectionCon=0.5,
trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon
        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(
            static_image_mode=self.mode,
            max_num_hands=self.maxHands,
            min_detection_confidence=self.detectionCon,
            min_tracking_confidence=self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]
```

кінець лістингу 3.5

Процес виявлення рук реалізований через метод «findHands()», який виконує попередню обробку зображення. Після обробки система відображає з'єднання між ключовими точками руки. Визначення позицій ключових точок руки здійснюється функцією «findPosition()». Вимірювання відстаней між будь-якими двома точками на руці виконує метод «findDistance()». Код для вказаних методів відстеження рук показано в лістингу 3.6.

Лістинг 3.6 – Методи відстеження рук у файлі «hand_detector.py»

```
def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
 self.mpHands.HAND_CONNECTIONS)
        return img
def findPosition(self, img, draw=False):
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[0]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 7, (255, 0, 255),
cv2.FILLED)
        return self.lmList
def findDistance(self, p1, p2, img, draw=True, r=7, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)
        return length, img, [x1, y1, x2, y2, cx, cy]
```

кінець лістингу 3.6

У файлі «screen_settings.py» підключено бібліотеки, що відповідають за обробку відео, обчислення частоти кадрів та доступ до розмірів екрану. Клас «ScreenSettings()» інкапсулює всю логіку керування вікном та відеопотоком.

Встановлено параметри відображення, включаючи роздільну здатність. Система визначає розміри екрану через Windows API для позиціонування вікна програми, коли при її запуску вікно автоматично розміщується в правому нижньому куті екрану, для ергономіки робочого простору на екрані (ліст. 3.7).

Лістинг 3.7 – Параметри відображення у файлі «screen_settings.py»

```
class ScreenSettings:
    def __init__(self):
        self.wCam = 640
        self.hCam = 480
        self.user32 = ctypes.windll.user32
        self.wScr = self.user32.GetSystemMetrics(0)
        self.hScr = self.user32.GetSystemMetrics(1)
        self.window_name = 'Hand Virtual Mouse'
        self.first_display = True
        self.pTime = 0
        self.aspect_ratio = 4.0 / 3.0
        self.prev_width = self.wCam
        self.prev_height = self.hCam
        self.threshold = 5
        self.cap = cv2.VideoCapture(0)
        self.cap.set(3, self.wCam)
        self.cap.set(4, self.hCam)
        self._initialize_window()
    def _initialize_window(self):
        cv2.namedWindow(self.window_name, cv2.WINDOW_NORMAL)
        cv2.resizeWindow(self.window_name, 400, 300)
    def position_window(self):
        if self.first_display:
            x_position = self.wScr - 400
            taskbar_height = GetSystemMetrics(SM_CYSCREEN) -
GetSystemMetrics(SM_CYFULLSCREEN)
            y_position = self.hScr - 300 - taskbar_height
            cv2.moveWindow(self.window_name, x_position, y_position)
            self.first_display = False
```

кінець лістингу 3.7

Процес захоплення та обробки відео реалізовано через «get_frame()», який забезпечує отримання кадру з камери та його адаптацію до поточних розмірів

вікна. Впроваджено механізм відстеження та коригування розмірів вікна, який активується при спробах змінити розміри вікна вручну (ліст. 3.8).

Лістинг 3.8 – Відстеження розмірів вікна у файлі «screen_settings.py»

```
def get_frame(self):
        success, img = self.cap.read()
        if not success:
            return None, None
        window_rect = cv2.getWindowImageRect(self.window_name)
        window_width = window_rect[2]
        window_height = window_rect[3]
        self.wCam = window_width
        self.hCam = window_height
        frameReduction = int(window_width * 0.16)
        if window_width > 0 and window_height > 0:
            img = cv2.resize(img, (window_width, window_height))
        return img, frameReduction
def draw_fps(self, img):
        font_scale = self.hCam / 480.0
        cTime = time.time()
        fps = 1 / (cTime - self.pTime)
        self.pTime = cTime
        cv2.putText(img, str(int(fps)),
                    (10, int(self.hCam * 0.15)),
                    cv2.FONT_HERSHEY_TRIPLEX,
                    font_scale * 2,
                    (255, 0, 255), 2)
        return img
def enforce_aspect_ratio(self):
        window_rect = cv2.getWindowImageRect(self.window_name)
        current_width = window_rect[2]
        current_height = window_rect[3]
        delta_width = abs(current_width - self.prev_width)
        delta_height = abs(current_height - self.prev_height)
        if delta_width > delta_height + self.threshold:
            new_height = int(current_width / self.aspect_ratio)
            cv2.resizeWindow(self.window_name, current_width,
 new_height)
        elif delta_height > delta_width + self.threshold:
            new_width = int(current_height * self.aspect_ratio)
            cv2.resizeWindow(self.window_name, new_width,
 current_height)
        self.prev_width = current_width
        self.prev_height = current_height
```

кінець лістингу 3.8

У файлі «mouse_control.py» використовуються бібліотеки: «pynput» для контролю миші, «numpy» для обчислень та «cv2» для комп'ютерного зору. В класі «MouseControl()» встановлено параметри для оптимальної роботи та коефіцієнти згладжування руху. Можливість руху курсору реалізована через метод «handle_mouse_movement()» (ліст. 3.9).

Лістинг 3.9 – Параметри роботи миші у файлі «mouse_control.py»

```
class MouseControl:
    def __init__(self, screen_width, screen_height):
        self.mouse = Controller()
        self.wScr = screen_width
        self.hScr = screen_height
        self.smoothening = 7
        self.click_smoother = 0
        self.pLocX, self.pLocY = 0, 0
        self.cLocX, self.cLocY = 0, 0
        self.RMB_pressed = False
        self.LMB_pressed = False
        self.dragging = False
    def handle_mouse_movement(self, x1, y1, frame_reduction,
window_width, window_height):
        x5 = np.interp(x1, (frame_reduction, window_width -
frame_reduction), (0, self.wScr))
        y5 = np.interp(y1, (frame_reduction, window_height -
frame_reduction), (0, self.hScr))
        self.cLocX = self.pLocX + (x5 - self.pLocX) / self.smoothening
        self.cLocY = self.pLocY + (y5 - self.pLocY) / self.smoothening
        self.mouse.position = (self.wScr - self.cLocX, self.cLocY)
        self.pLocX, self.pLocY = self.cLocX, self.cLocY
        if self.click_smoother >= 30:
            self.RMB_pressed = False
            self.LMB_pressed = False
        self.click_smoother += 1
        return (x1, y1)
```

кінець лістингу 3.9

Клацання та перетягування виконуються в «handle_clicking_dragging()», який аналізує положення пальців і відстань між ними. Таким чином, в залежності від розміщення точок, визначається чи буде виконано звичайне натискання, або натискання з утриманням кнопки, або подвійне натискання миші (ліст. 3.10).

Лістинг 3.10 – Клацання і перетягування у файлі «mouse_control.py»

```
def handle_clicking_dragging(self, fingers, detector, img,
frame_reduction, window_width, window_height):
        if fingers[1] == 1 and fingers[2] == 1 and fingers[3] == 1 and
fingers[4] == 1:
            if self.dragging:
                self.mouse.release(Button.left)
                self.dragging = False
            self.LMB_pressed = False
            self.RMB_pressed = False
            return img
     length_im, img, line_info_im = detector.findDistance(8, 12, img)
     length_mr, img, line_info_mr = detector.findDistance(12, 16, img)
        if fingers[3] == 1 and length_mr < 50 and length_im < 50:
            img = self._handle_right_click(img, line_info_im,
line_info_mr)
        else:   img = self._handle_left_click_and_drag(
                img, length_im, line_info_im, frame_reduction,
                window_width, window_height)
        self.click_smoother = 0
        return img
def  _handle_left_click_and_drag(self,  img,  length_im,  line_info_im,
frame_reduction, window_width, window_height):
        self.RMB_pressed = False
        if not self.dragging and length_im < 25:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 255, 0), cv2.FILLED)
            self.mouse.press(Button.left)
            self.dragging = True
            self.LMB_pressed = True
        elif self.dragging:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 0, 255), cv2.FILLED)
            x3 = np.interp(line_info_im[4], (frame_reduction,
window_width - frame_reduction), (0, self.wScr))
            y3 = np.interp(line_info_im[5], (frame_reduction,
window_height - frame_reduction), (0, self.hScr))
            self.cLocX = self.pLocX + (x3 - self.pLocX) /self.smoothening
            self.cLocY = self.pLocY + (y3 - self.pLocY) /self.smoothening
            self.mouse.position = (self.wScr - self.cLocX, self.cLocY)
            self.pLocX, self.pLocY = self.cLocX, self.cLocY
            if length_im > 25:
                self.mouse.click(Button.left, 2)
                self.LMB_pressed = True
                self.dragging = False
        return img
```

кінець лістингу 3.10

Якщо, окрім вказівного і середнього, з’являється ще й безіменний палець, спрацьовує метод «_handle_right_click()». Таким чином, відбувається правий клік миші. Функціонал скролінгу знаходиться у «handle_scrolling()», який інтерпретує положення великого пальця для визначення напрямку прокрутки. Код для правого клацання й прокручування показано в лістингу 3.11.

Лістинг 3.11 – Правий клік та скролінг у файлі «mouse_control.py»

```
def _handle_right_click(self, img, line_info_im, line_info_mr):
        if self.dragging:
            self.mouse.release(Button.left)
            self.dragging = False
            self.LMB_pressed = False
        if not self.RMB_pressed:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 255, 255), cv2.FILLED)
            cv2.circle(img, (line_info_mr[4], line_info_mr[5]),
                       7, (0, 255, 255), cv2.FILLED)
            self.mouse.click(Button.right, 1)
            self.RMB_pressed = True
            self.LMB_pressed = False
        return img
def handle_scrolling(self, thumb_up):
        if thumb_up:
            self.mouse.scroll(0, -0.5)
        else:
            self.mouse.scroll(0, 0.5)
```

кінець лістингу 3.11

У файлі «volume_control.py» встановлено з’єднання з Windows API через бібліотеку «pycaw», що забезпечує доступ до системних налаштувань звуку. В класі «VolumeControl()» включено доступ до аудіопристрою за замовчуванням та визначено допустимий діапазон гучності.

Система зберігає поточний рівень звуку та розраховує початкові значення для візуального відображення надалі. Обробка жестів реалізована через метод «handle_volume_gesture()», а візуальний інтерфейс шкали гучності через «draw_volume_interface()». Числові показники гучності відображено у відсотках. Система динамічно обчислює розміри та позиції елементів інтерфейсу залежно

від розмірів вікна, що забезпечує адаптивність відображення. Код конфігурацій гучності надано в лістингу 3.12.

Лістинг 3.12 – Конфігурація гучності у файлі «volume_control.py»

```
class VolumeControl:
    def __init__(self):
        devices = AudioUtilities.GetSpeakers()
        interface     =     devices.Activate(IAudioEndpointVolume._iid_,
CLSCTX_ALL, None)
        self.volume = cast(interface, POINTER(IAudioEndpointVolume))
        self.volRange = self.volume.GetVolumeRange()
        self.minVol, self.maxVol = self.volRange[0], self.volRange[1]
        self.current_volume = self.volume.GetMasterVolumeLevelScalar()
        self.volume_bar = int(np.interp(self.current_volume, [0, 1], [400,
150]))
        self.volume_percentage = int(self.current_volume * 100)
    def handle_volume_gesture(self, lmList):
        fingertips = [lmList[tip] for tip in [8, 12, 16, 20]]
        avg_y = sum(tip[2] for tip in fingertips) / len(fingertips)
        vol = np.interp(avg_y, [80, 150], [self.maxVol, self.minVol])
        current_volume = np.interp(vol, [self.minVol, self.maxVol], [0,
1])
        self.volume.SetMasterVolumeLevelScalar(current_volume, None)
        self.volume_bar = np.interp(avg_y, [80, 150], [150, 400])
        self.volume_percentage = int(np.interp(avg_y, [80, 150], [100,
0]))
    def draw_volume_interface(self, img, window_height):
        height, width = img.shape[:2]
        volume_x = int(width * 0.1)
        volume_top = int(height * 0.3)
        volume_bottom = int(height * 0.8)
        volume_width = int(width * 0.05)
        cv2.rectangle(img,(volume_x,volume_top),(volume_x + volume_width,
volume_bottom),(0, 255, 0), 3)
        current_volume_height = int(np.interp(self.volume_bar, [150, 400],
                                              [volume_top, volume_bottom]))
        cv2.rectangle(img, (volume_x, current_volume_height),
(volume_x + volume_width, volume_bottom), (0, 255, 0), cv2.FILLED)
```

кінець лістингу 3.12

Таким чином, вся система працює в режимі реального часу, забезпечуючи миттєвий відгук на жести користувача, не лише оновлюючи кадри відеопотоку, але й одразу ж виконуючи команди миші та змінюючи системну гучність.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Після завершення практичної реалізації системи, постала необхідність у проведенні комплексного тестування розробленого застосунку задля верифікації коректності його функціонування у відповідності встановленим вимогам. Visual Studio Code надає вбудовані засоби для налагодження і попереднього тестування програмного продукту до етапу його компіляції у виконуваний файл. Тестування передбачає систематичну перевірку кожного жесту з метою виявлення можливих невідповідностей у виконанні команд.

Для запуску налагоджувача необхідно відкрити головний файл програми, далі в розділі «Run» обрати пункт «Start Debugging» або натиснути кнопку F5. Якщо не було виявлено помилок в коді, відбудеться запуск додатку. При запуску вмикається камера системи та з’являються елементи інтерфейсу, такі як число частоти кадрів у FPS та шкала гучності системи, під якою значення шкали у відсотках, що показує поточний рівень звуку. Для початку, проведено тест на відображення рамки зони дії керування жестами (рис. 3.2).

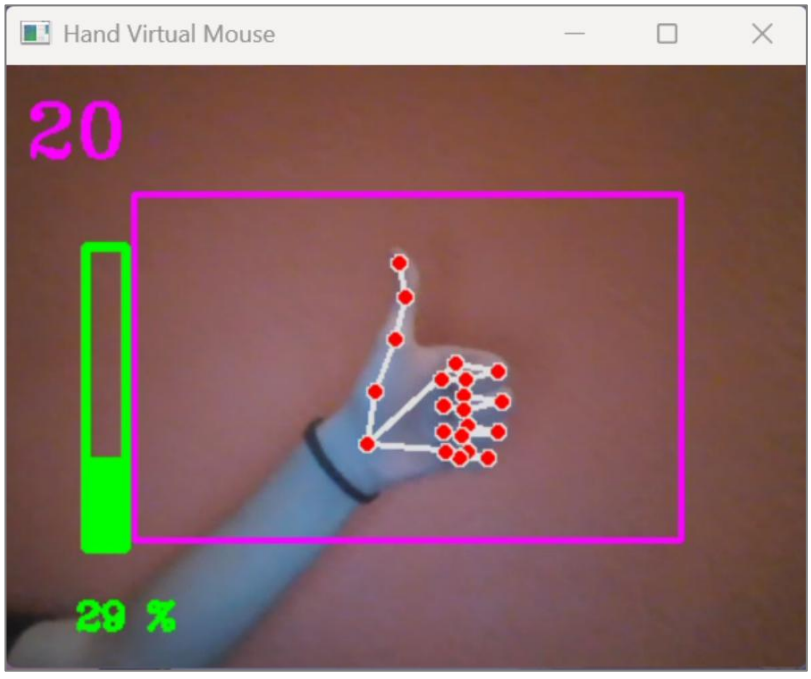


Рисунок 3.2 – Інтерфейс програмного застосунку

Рамка відображається, коли користувач розміщує руку у відповідній області. Якщо забрати руку з цієї зони, вона зникає. При зміні розмірів вікна рамка адаптується і зберігає свої пропорції, відповідно до розміру екрану.

Наступним етапом є перевірка виконання функціоналу застосунку. Якщо підняти вказівний палець і почати водити ним по площині рамки, система розпізнає кінчик цього пальця та інтерпретує в команду переміщення курсору миші по екрану комп’ютера. Перевірка даної функції продемонстрована на рисунку 3.3.

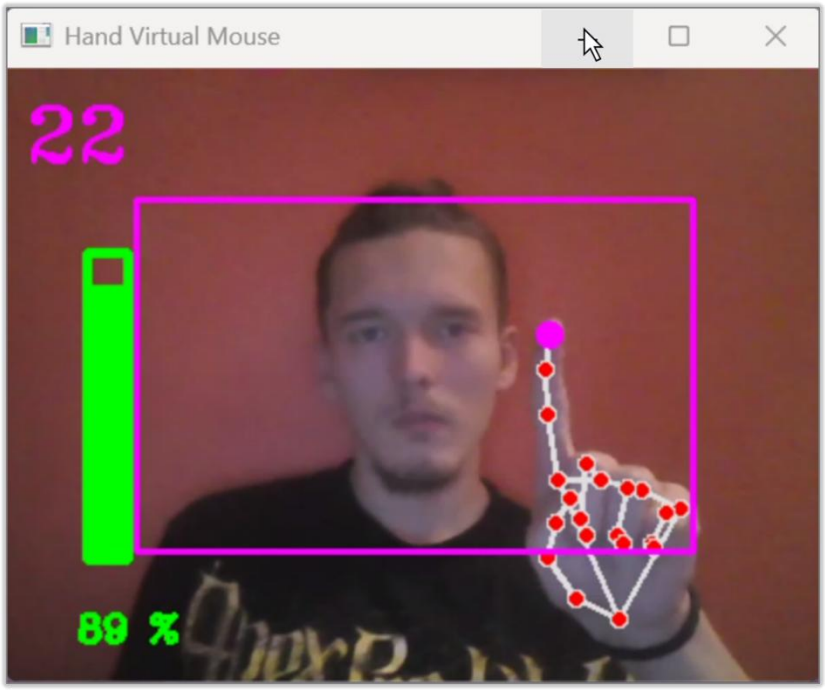


Рисунок 3.3 – Тестування переміщення курсору

Для переміщення об’єктів (файлів, папок тощо) потрібно підняти додатково середній палець і стиснути його з вказівним. Окрім переміщення, також можна виконувати виділення тих самих об’єктів. Комбінація вказівного та середнього пальців, у даному програмному забезпеченні, надають можливість виконання додаткових функцій. До прикладу, якщо робити зближення пальців, відбудеться команда лівого клацання миші (рис. 3.4). Якщо ж розвести два пальці на більшу відстань, то здійснюється подвійне натискання лівої кнопки миші.

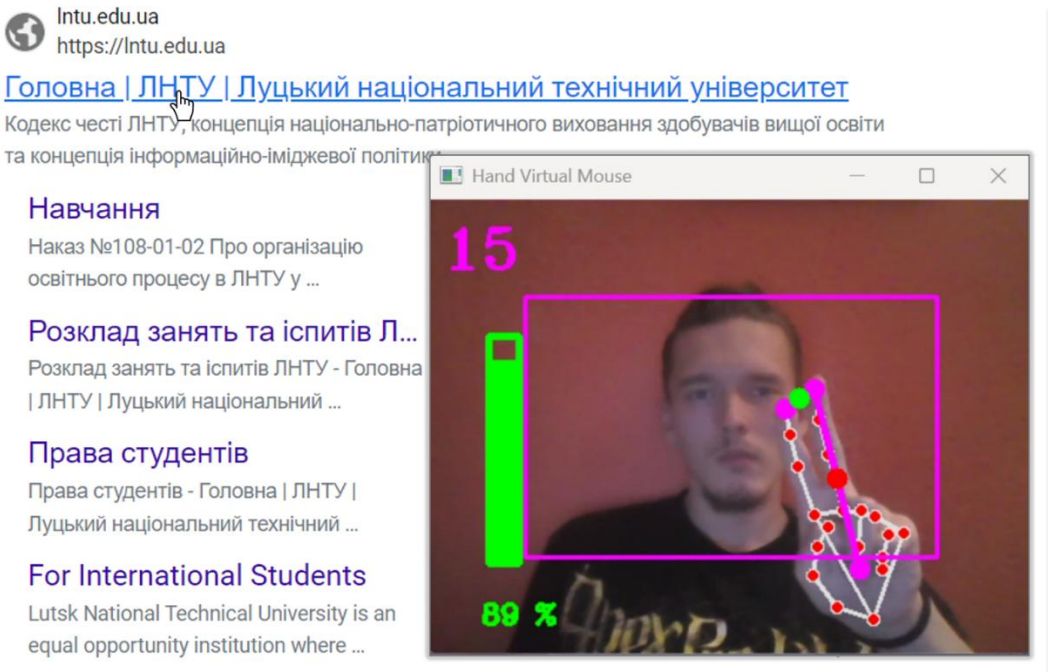


Рисунок 3.4 – Тестування натискання лівої кнопки миші

Застосування до цих пальців додатково безіменного дозволяє виконувати функцію клацання правої кнопки миші (рис. 3.5).

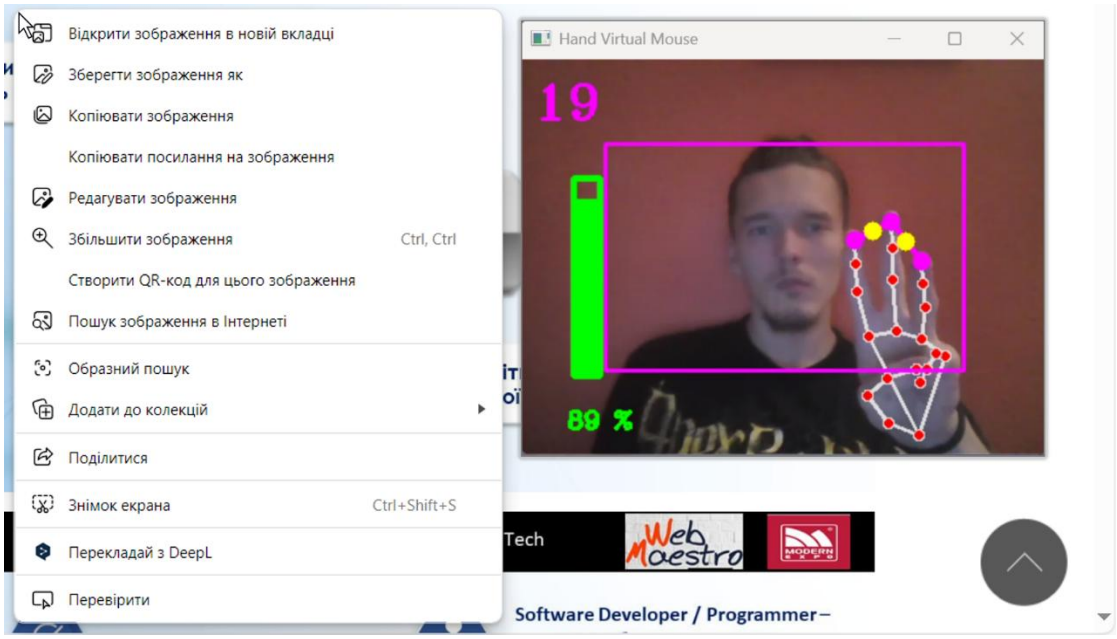


Рисунок 3.5 – Тестування натискання правої кнопки миші

Для перевірки скролінгу потрібно закрити усі пальці, а всі дії виконуються великим пальцем. Якщо відкрити будь-який документ або інтернет-сторінку,

використання цієї команди дозволяє прогортати сторінки. Коли піднято великий палець, відбувається прокручування контенту вгору. Опустивши великий палець, відбувається прокручування сторінки вниз. Тестування команди прокручування, а також виявлення великого пальця у даній функції показано на рисунку 3.6.

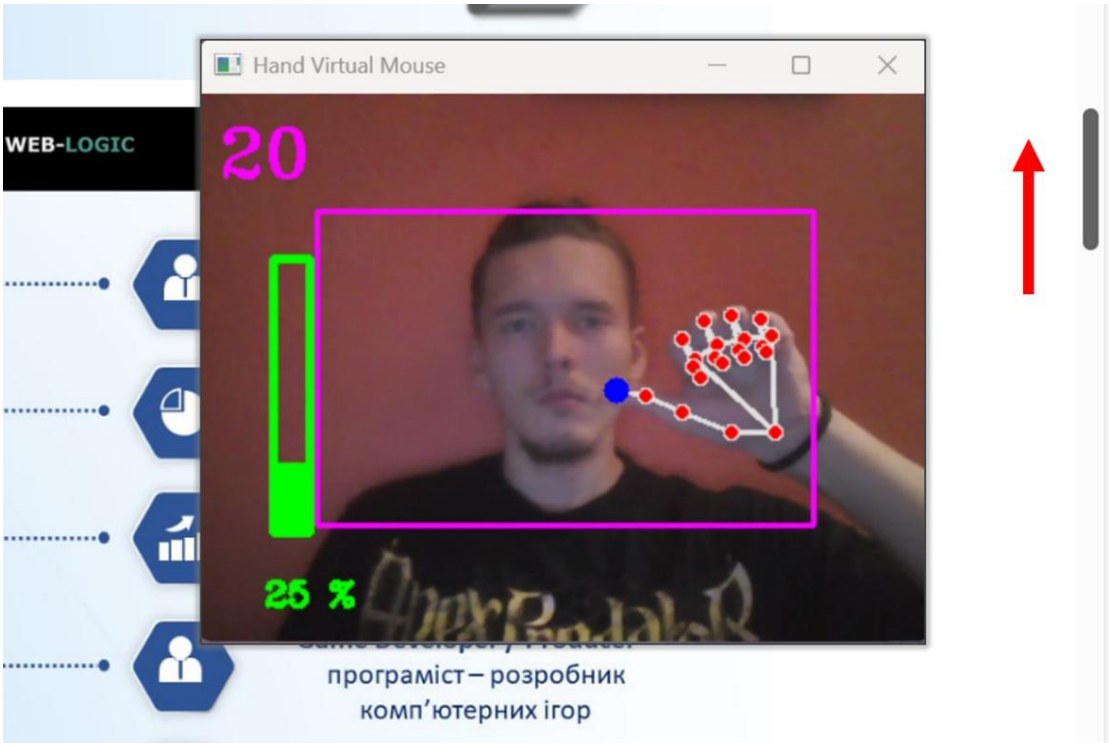


Рисунок 3.6 – Тестування скролінгу

Останньою командою для тестування стала перевірка регулювання гучності системи вертикальними рухами руки. Для цього потрібно відкрити усі пальці, крім великого. Далі необхідно робити рухи рукою по вертикалі екрану. Якщо поспостерігати за шкалою гучності, можна побачити, що вона змінюється, і також змінюється числове значення рівня звуку у відсотках. При підніманні руки вгору, гучність збільшується (рис. 3.7). Відповідно, при опусканні руки донизу, гучність стає меншою.

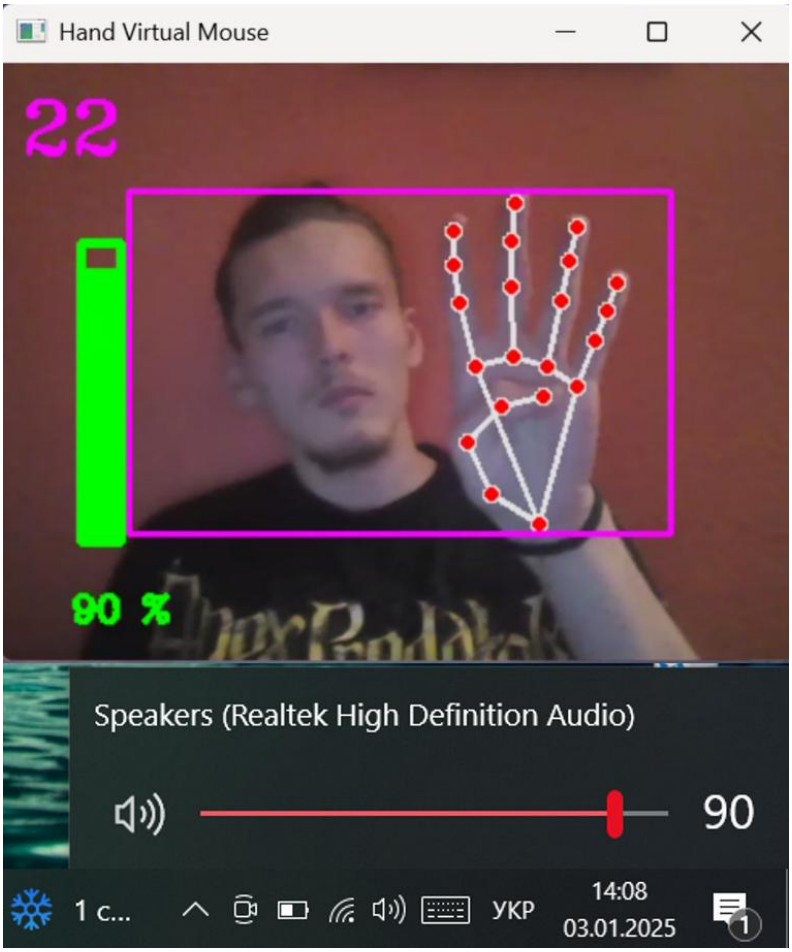


Рисунок 3.7 – Тестування зміни рівня гучності

Отже, поточна версія програмного продукту забезпечує надійне виконання основних операцій керування комп’ютером і розпізнавання жестів. Слід також зазначити, що в застосунку керування комп’ютером можна виконувати жестами, використовуючи для цього як праву, так і ліву руку.

3.3 Розробка виконуваного файлу та інсталяційного пакета

Провівши тестування роботи програми, функціонування основних команд та упевнившись у коректному виконанні завдань, наступним етапом є компіляція усіх файлів проєкту в єдиний виконуваний файл формату «.exe».

Таким чином, додаток буде, практично, на завершальному етапі розробки. Його можна запускати одним файлом лише за декілька кліків, без необхідності запуску за допомогою середовища VS Code. Виконуваним файлом можливо

також ділитися із іншими юзерами з метою тестування застосунку різними користувачами, аналізу та покращення в наступних версіях програми.

Перед початком процесу компіляції програмного продукту у виконуваний файл, створено спеціальний конфігураційний файл «virtual_mouse.spec», куди внесено необхідні залежності, бібліотеки з модулів та скрипт перетворення в exe-файл, включно з параметрами, такими як назва, піктограма тощо.

Скрипт дозволяє короткою командою, без необхідності прописувати в командному рядку залежності вручну, створити виконуваний файл (ліст. 3.13).

Лістинг 3.13 – Скрипт перетворення програми в exe-файл

```
block_cipher = None
a = Analysis(['main.py'], pathex=[], binaries=[], datas=[],
    hiddenimports=['mediapipe', 'cv2', 'numpy', 'pynput', 'keyboard',
        'ctypes', 'win32api', 'win32con', 'comtypes', 'pycaw.pycaw'],
    hookspath=['.'],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
    noarchive=False,)
pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)
exe = EXE(pyz, a.scripts, a.binaries, a.zipfiles, a.datas, [],
    name=’VirtualMouse’,
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=False,
    disable_windowed_traceback=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
    icon='icon.ico')
```

кінець лістингу 3.13

Далі, запустивши командний рядок, інстальовано бібліотеку «pyinstall», яка дозволяє перетворити Python-файли в програму exe-формату. У командному

рядку, перейшовши в папку проєкту, введено «pyinstaller virtual_mouse.spec», після чого розпочався процес компіляції продукту у виконуваний файл (рис. 3.8).

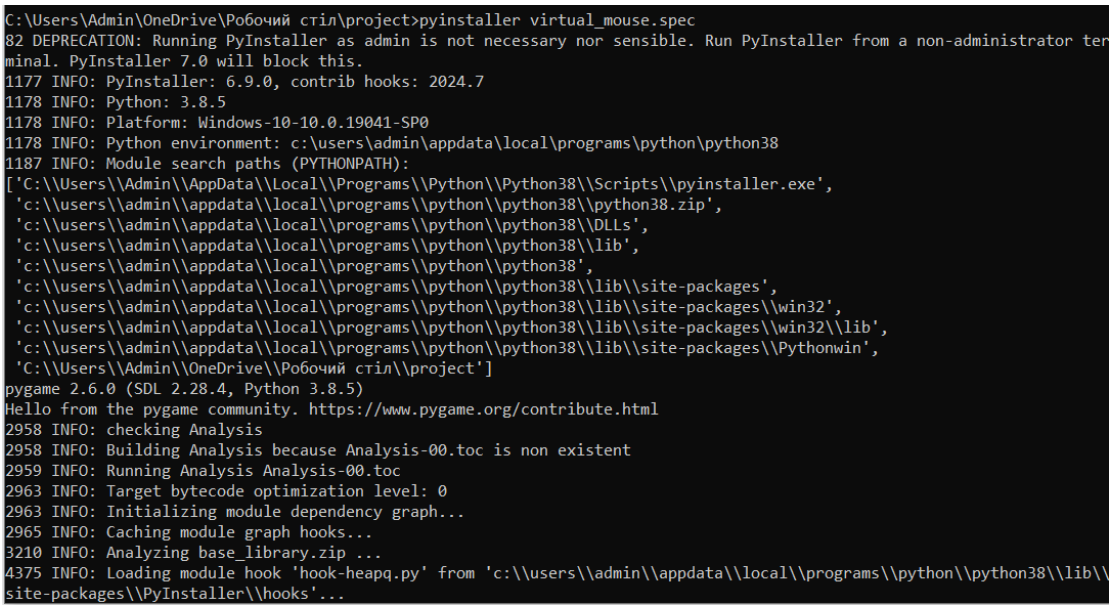


Рисунок 3.8 – Перетворення програми в exe-файл

По завершенню компіляції, в папці проєкту з’являються папки «build» та «dist», в останній і знаходиться виконуваний файл. Створений exe-файл можна запускати, переміщати в інші каталоги та відправляти користувачам без проблем і побоювань, що вона може не розгорнутися в іншому середовищі.

Використовуючи спеціальне програмне забезпечення для пакування файлів у інсталяційний пакет «Inno Setup», виконано процес створення скрипту інсталяції виконуваного файлу у відповідний пакет. Застосунок дозволяє в процесі створення вказати власника проєкту, шляхи до директорії, можливість створювати ярлики на робочому столі й панелі завдань, додати файл ліцензії та запит про запуск програми одразу по завершенню інсталяції на комп’ютер користувача.

По завершенню конфігурації параметрів інсталяції створено скрипт, який показано в лістингу 3.14. Запустивши цей скрипт, відбувається створення інсталяційного пакета.

Лістинг 3.14 – Скрипт інсталяційного пакета

```
AppName=VirtualMouse
AppVersion=1.0
AppPublisher=Use Potrokhu, Inc.
AppPublisherURL=https://github.com/nekitNekitnekit
AppSupportURL=https://github.com/nekitNekitnekit
AppUpdatesURL=https://github.com/nekitNekitnekit
DefaultDirName={pf}\VirtualMouse
DefaultGroupName=VirtualMouse
AllowNoIcons=yes
LicenseFile=C:\...\project\License.txt
OutputDir=C:\...\project
OutputBaseFilename=VirtualMouseInstaller
Compression=lzma
SolidCompression=yes
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}";
GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked
Filename: "{app}\VirtualMouse.exe"; Description:
"{cm:LaunchProgram,VirtualMouse}"; Flags: nowait postinstall
skipifsilent
```

кінець лістингу 3.14

Після створення інсталяційного пакета, автоматично відбувається запуск даного файлу. При запуску можна обрати мову для інсталяції, після чого з’являється стандартний майстер встановлення програм на комп’ютер (рис. 3.9).

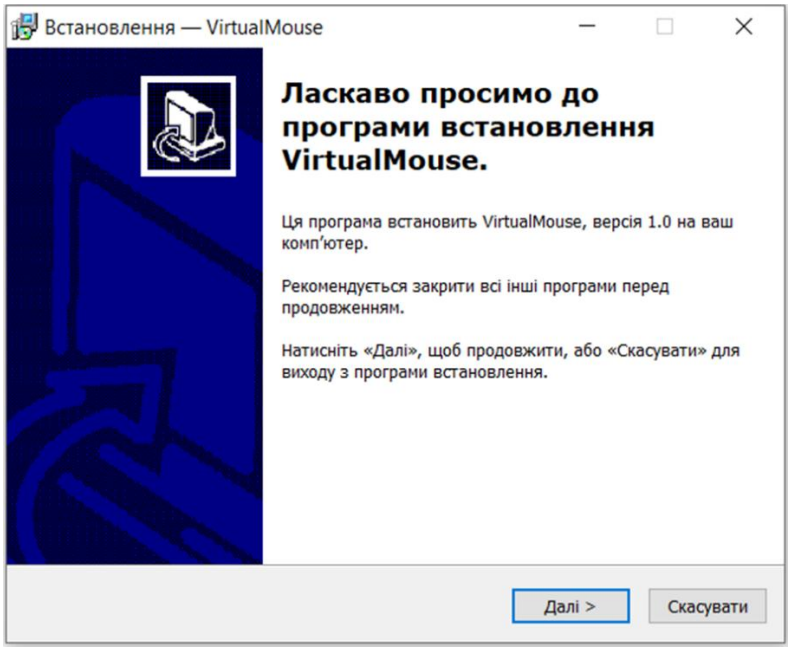


Рисунок 3.9 – Майстер встановлення застосунку

Як було зазначено раніше, що в процесі створення пакета можна додати свій ліцензійний файл для захисту інтелектуальної власності розробника додатку. Якщо його було додано, то під час інсталяції продукту буде розділ, який надає користувачеві ознайомитися з ліцензійною угодою (рис. 3.10). Якщо юзер погоджується з її умовами, він обирає відповідну опцію, після чого можливий наступний етап інсталяції.

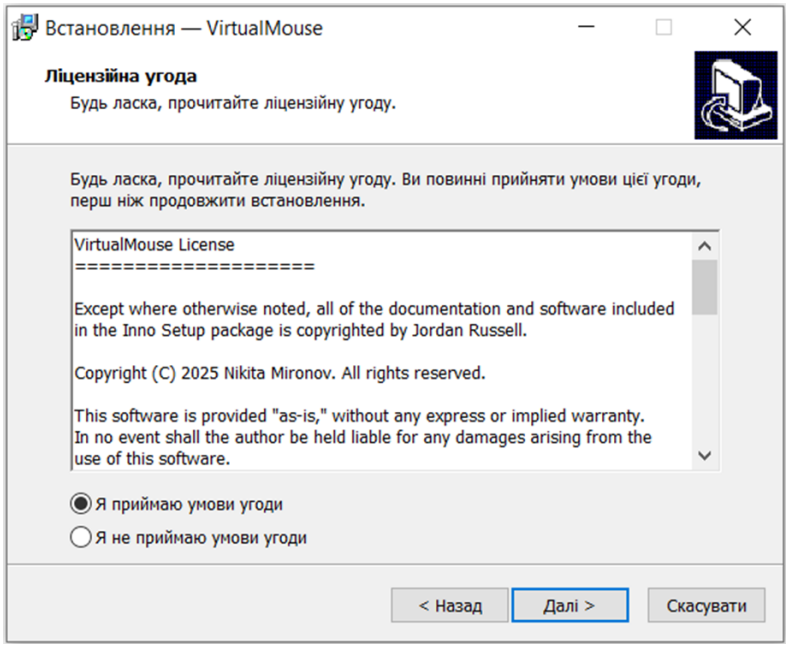


Рисунок 3.10 – Ліцензійна угода

Виконавши всі етапи майстра інсталяції, відбувається, власне, встановлення програмного застосунку в каталог, який обрав користувач. Після встановлення програми з’являється вікно з успішною інсталяцією, де можна обрати опцію про запуск застосунку одразу після закриття інсталяційного пакета.

У каталозі, куди користувач встановив програму, знаходяться кілька файлів: власне розроблюваний програмний продукт в exe-форматі та два файли для видаленням (рис. 3.11). Якщо запустити файл «unins000.exe», відкриється майстер деінсталяції застосунку, після проходження всіх етапів якого додаток видалиться з комп’ютера користувача.

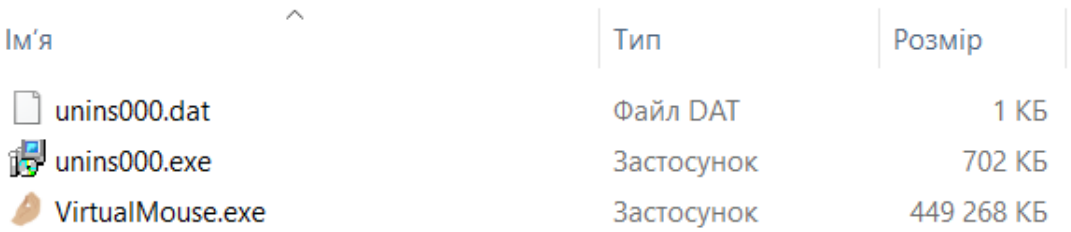


Рисунок 3.11 – Файли програми після інсталяції

Таким чином, створений інсталяційний пакет дозволяє розгорнути додаток на інших пристроях, знижуючи при цьому можливі помилки під час встановлення. Пакет можна інтегрувати у свій сайт, репозиторій, звідки можуть його завантажити інші користувачі. Так само можна відправляти юзерам цей файл, щоб вони змогли без проблем встановити програмний продукт на власний комп'ютер та на своєму досвіді спробувати керування системою жестами.

3.4 Аналіз результатів точності розпізнавання жестів

З метою оцінки точності в розпізнаванні жестів та ефективності програмного забезпечення, проведено аналіз результатів користувачами, які особисто отримали виконуваний файл продукту або встановили програмний застосунок за допомогою інсталяційного пакета.

Для проведення тестування залучено 8 користувачів, з яких 5 встановили додаток за допомогою інсталяційного пакета, решта отримали одразу виконуваний файл програми, який запустили на своєму пристрої. Варто зазначити, що в обох випадках код файлів продукту не змінювався від моменту створення виконуваного файлу до моменту його пакування в інсталятор. Перевірка розпізнавання жестів виконувалася з використанням різного апаратного забезпечення, в тому числі як стаціонарних комп'ютерів, так і ноутбуків різних розмірів.

Так само використано в тестуванні як вбудовані (переважно для ноутбуків), так і зовнішні веб-камери з різною роздільною здатністю. Враховано також відстані від камери, при яких виконувалися ті чи інші жести. Важливим

критерієм є умови освітлення під час проведення тесту, які відрізнялися у кожного користувача. Таким чином, під час проведення тестування враховано наступні метрики:

– спосіб встановлення застосунку на пристрій користувача;
– використання зовнішньої (З) або вбудованої (В) камери під час тесту;
– роздільна здатність камери;
– відстань від камери під час тестування;
– умови освітлення (хороше, помірне, слабке).

Отримані дані критеріїв у випадках для кожного користувача акумульовано в таблицю 3.1 для аналізу результатів точності розпізнавання жестів надалі.

Таблиця 3.1 – Умови проведення тестування

Умови	Користувачі							
	1	2	3	4	5	6	7	8
Встановлено за допомогою інсталятора	Ні	Ні	Ні	Так	Так	Так	Так	Так
Камера	З	З	В	З	В	В	В	З
Роздільна здатність	720p	1080p	720p	4K	480p	1080p	720p	720p
Відстань від камери	60 см	65 см	40 см	90 см	55 см	55 см	50 см	70 см
Освітлення	Слабке	Слабке	Помірне	Хороше	Хороше	Помірне	Слабке	Хороше

Маючи вичерпну інформацію про умови проведення тестування, отримано та проаналізовано результати перевірок за наступними критеріями:

– точність розпізнавання;
– час реакції на розпізнавання жесту;
– стабільність роботи команди, яку виконує той чи інший жест.

На кожен жест було виділено по 10 спроб на його виконання, після чого вираховувалося середнє значення результату для кожного критерію.

Результати точності розпізнавання жестів представлено в таблиці 3.2, де оцінювання здійснювалося, спираючись на те, що камера відстежить руку та

зчитає показаний жест або рух. На основі цього точність переведено у відсотки.

Таблиця 3.2 – Аналіз результатів точності розпізнавання жестів

Жест	Користувачі							
	1	2	3	4	5	6	7	8
Переміщення курсору миші	97,4 %	98,1 %	98,7 %	99,3 %	97,2 %	98,5 %	97,6 %	97,8 %
Лівий клік	95,8 %	95,6 %	96,9 %	96,4 %	96,5 %	96,7 %	95,8 %	97,1 %
Подвійний лівий клік	90,1 %	90,2 %	90,8 %	90,4 %	90,2 %	90,7 %	90,3 %	90,6 %
Правий клік	94,2 %	94,1 %	94,9 %	93,8 %	94,6 %	93,8 %	94,7 %	93,9 %
Перетягування	96,5 %	96,2 %	95,9 %	96,4 %	96,3 %	96,6 %	95,7 %	96,7 %
Скролінг	96,9 %	97,1 %	96,7 %	97,3 %	97,2 %	96,5 %	96,6 %	96,8 %
Регулювання гучності	97,1 %	97,6 %	97,9 %	98,4 %	98,2 %	97,8 %	98,1 %	98,6 %

Аналіз показав, що програма демонструє високу ефективність із середньою точністю понад 95 % для всіх жестів. Найкращі результати виявилися для простих рухів, таких як переміщення курсору та регулювання гучності. В той же час було важче з розпізнаванням жесту для подвійного клацання.

Результати часу реакції на розпізнавання жестів показано в таблиці 3.3.

Таблиця 3.3 – Аналіз результатів часу реакції жестів

Жест	Користувачі							
	1	2	3	4	5	6	7	8
Переміщення курсору миші	47 мс	46 мс	40 мс	39 мс	42 мс	40 мс	45 мс	38 мс
Лівий клік	43 мс	42 мс	36 мс	38 мс	38 мс	39 мс	41 мс	37 мс
Подвійний лівий клік	50 мс	54 мс	48 мс	45 мс	51 мс	49 мс	53 мс	47 мс
Правий клік	45 мс	44 мс	38 мс	39 мс	49 мс	38 мс	43 мс	37 мс
Перетягування	49 мс	48 мс	43 мс	42 мс	44 мс	43 мс	48 мс	42 мс
Скролінг	42 мс	41 мс	44 мс	37 мс	39 мс	44 мс	41 мс	43 мс
Регулювання гучності	42 мс	47 мс	34 мс	41 мс	43 мс	45 мс	46 мс	40 мс

Спираючись на результати, застосунок показав швидкодію всіх функцій із

середнім часом у межах 36-54 мс, що є прийнятним для роботи в реальному часі.

Аналіз результатів стабільності в роботі жестів демонструє таблиця 3.4, де кожна спроба виконання всіх команд оцінювалася від 1 до 10, після чого отримано середнє арифметичне цих чисел, округлених до десятих.

Таблиця 3.4 – Аналіз результатів стабільної роботи жестів

Жест	Користувачі							
	1	2	3	4	5	6	7	8
Переміщення курсору миші	9,6	9,5	9,7	9,9	9,5	9,7	9,6	9,6
Лівий клік	8,4	8,5	8,7	8,6	8,5	8,8	8,3	8,5
Подвійний лівий клік	7,4	7,3	7,4	7,3	7,5	7,4	7,4	7,2
Правий клік	8,5	8,4	8,8	8,8	8,7	8,6	8,5	8,7
Перетягування	8,7	8,6	8,8	8,7	9,1	9,2	8,7	8,6
Скролінг	8,4	8,6	8,5	8,2	8,4	8,5	8,3	8,4
Регулювання гучності	9,2	9,1	9,3	9,5	9,3	9,4	9,1	9,4

Таблиця результатів стабільності свідчить про достатню надійність додатку, із середніми показниками кожного жесту в межах 7,4-9,6 бала. Кращу стабільність забезпечують ті ж самі жести, що відповідають за рух миші та регулювання звуку. Команда подвійного лівого клацання, так само, показала найнижчий результат, що вказує на потребу оптимізації функції. Також можливою причиною такої невисокої стабільності операції є складність в проведенні відповідного жесту для подвійного кліку та плутанина в схожості з проведенням одного клацання лівої кнопки миші. Таким чином, варто зробити акцент у вдосконаленні алгоритмів розпізнавання, що дозволить підвищити точність і надійність обробки складних жестів.

Загалом, результати аналізу трьох таблиць підтверджують, що застосунок достатньо точно, швидко і стабільно виконує свої основні функції. Звичайно, є можливості для покращення та оптимізації функціонування системи. Для легшого сприйняття складено загальну таблицю результатів (табл. 3.5).

Таблиця 3.5 – Узагальнені результати аналізу функціонування жестів

Жест	Критерії		
	Точність розпізнавання (%)	Час реакції (мс)	Стабільність (від 1 до 10)
Переміщення курсору миші	98	42	9,6
Лівий клік	96	39	8,5
Подвійний лівий клік	90	50	7,4
Правий клік	94	42	8,6
Перетягування	96	45	8,8
Скролінг	97	41	8,4
Регулювання гучності	98	42	9,3

Зробивши аналіз загальної таблиці, можна дійти висновку, що програма має збалансовані результати в точності розпізнавання та реакції на жест, що свідчить про ефективність системи. Особливо виділяються команди переміщення курсору і зміни гучності з високими показниками в усіх критеріях. Однак, для підвищення якості функціонування можна зосередитися на покращенні точності та стабільності жестів, які мають нижчі показники, особливо це стосується подвійного клацання.

3.5 Перспективи подальшого розвитку застосунку

Здійснивши етапи розробки програми, перетворення продукту у виконуваний файл та інсталяційний пакет, а також тестування як з боку розробника, так і з боку кінцевих користувачів додатку, проаналізовано можливі шляхи розвитку системи та рекомендації для вдосконалення функціонування застосунку, зокрема в питанні виконання різних команд відповідними жестами. Звичайно, розроблене програмне забезпечення виконує поставлені перед ним завдання, але завжди можна і варто прагнути покращити свій продукт.

Проведені в підрозділі 3.4 аналізи показали, що найменш ефективним і стабільним у роботі є виконання жесту для подвійного лівого клацання миші.

Тому слід оптимізувати алгоритми команди для забезпечення точнішого розпізнавання та стабільнішого виконання. Для цього можна буде переробити логіку виклику функції подвійного клацання або призначити для цієї дії інший жест, який буде простішим у виконанні та інтуїтивно зрозумілим.

Не тільки подвійний лівий клік миші, а й алгоритм роботи усіх інших жестів також можна покращити для досягнення максимальних результатів ефективності. Оскільки результати тестів напряму залежали від таких зовнішніх чинників, як роздільна здатність камери та освітленість середовища, тому слід приділити увагу розробці методів, які будуть оптимізувати стабільність роботи системи у випадках нечітких та швидких рухів руками та в умовах різного освітлення. Так, для кращої роботи програмного застосунку в нагоді стане й веб-камера з високою роздільною здатністю і можливістю адаптуватися до будь-якого освітлення.

Саме собою, майбутнє вдосконалення застосунку включає також розширення можливостей розпізнавання шляхом впровадження підтримки додаткових жестів для широкого набору системних команд, наприклад, масштабування документів та фото, перемикання між елементами контенту (фото, відео, музика тощо), зупинка або відтворення музики та відео, регулювання яскравості системи, конкретні дії з об’єктами (копіювання, вставлення, видалення тощо) і так далі.

Оскільки застосунок розроблено за модульним принципом, стає легким розширення функціоналу через створення додаткових модулів відповідних дій та їх імпорт в корінь проєкту. Разом із цим, можливим стає інтеграція інструментів, таких як голосовий помічник, віртуальна клавіатура тощо, які розширять взаємодію користувача з системою, виключаючи використання фізичних пристроїв, що може бути дуже корисним і вкрай необхідним для людей з обмеженими можливостями, пов’язаними з порушеннями рухової активності.

Можна також розробити механізм, який, коли застосунок було запущено, активує управління жестами програми лише після виконання спеціального руху або жесту. В іншому випадку основний функціонал додатку буде вимкнений, що

забезпечує захист від виконання ненавмисних дій випадковими жестами. Так само можна впровадити підтримку одночасного розпізнавання двох рук, що відкриє можливості для реалізації комплексних команд керування системою.

Інтерфейс програмного забезпечення теж вартий уваги та потребує розвитку надалі. Доцільно розглянути розробку навігаційної довідки користувача, інструкції (гід) з використанням застосунку або системи візуальних підказок, що допоможуть краще орієнтуватися у доступних жестах та їх призначенні. Для розміщення довідкових матеріалів або підказок в самій програмі, можна реалізувати панель меню, яка, окрім прикріплених інструкцій, може мати можливості індивідуального налаштування застосунку та додатковий функціонал, що забезпечить як зручний доступ до інформації, так і підвищить ефективність взаємодії з системою.

Дане програмне рішення розраховане на користувачів, у яких на комп’ютері встановлена ОС Windows. Увага буде приділена розширенню системної інтеграції застосунку шляхом реалізації підтримки додаткових операційних систем, таких як Linux та macOS, що, в свою чергу, дозволить охопити ширшу аудиторію.

Таким чином, аналіз розробленого програмного забезпечення для керування комп’ютером за допомогою жестів виявив значний потенціал для вдосконалення й розширення функціональності системи надалі. Реалізація окреслених напрямків розвитку дозволить створити багатофункціональний та зручний інструмент для безконтактного керування пристроями, що зможе задовольнити потреби широкого кола користувачів.

Висновки до розділу 3

Було виконано практичну реалізацію програмного застосунку для жестового керування комп’ютером в середовищі розробки VS Code, використовуючи мову Python та її бібліотеки, в тому числі, для відстеження рук, комп’ютерного зору, програмного керування системою тощо. Застосунок

реалізовано за модульним принципом, де основна логіка розділена на окремі компоненти: розпізнавання та відстеження руки, налаштування екрану та вікна програми, управління командами миші та керування гучністю системи.

Проведено тестування та аналіз результатів роботи основного функціоналу програми, включаючи переміщення курсору, клацання лівою та правою кнопками миші, перетягування об’єктів, прокручування сторінок та регулювання гучності системи. Перевірка, в цілому, підтвердила коректну роботу всіх команд. Опісля програмний застосунок успішно трансформовано у виконуваний файл та створено інсталяційний пакет. Таке рішення було здійснено з метою зручного розгортання продукту на комп’ютерах користувачів.

Проведений аналіз користувачами, що встановили додаток на свій пристрій, з різними конфігураціями обладнання та умовами освітлення, показав ефективність системи. Застосунок показав високу точність та швидкодію в реальному часі, а середня стабільність роботи всіх жестів оцінено у 8,7 балів з 10, що свідчить про достатню надійність програмного забезпечення. Найкраще себе показали функції переміщення курсору та регулювання гучності, тоді як складніші жести, зокрема подвійне клацання, потребують вдосконалення й оптимізації алгоритмів надалі.

Визначено перспективні напрямки розвитку програмного забезпечення, які включають оновлення алгоритмів розпізнавання складних жестів, покращення роботи в умовах різного освітлення, розширення набору підтримуваних команд, інтеграцію додаткових інструментів взаємодії з системою, розробку панелі меню з користувацькою довідкою і можливістю індивідуального налаштування програми, забезпечення підтримки інших операційних систем тощо. Оскільки програма має модульну структуру, зручно підтримувати код проєкту та розширювати його через впровадження додаткових модулів та інструментів у ядро системи.

ВИСНОВКИ

У результаті виконання роботи було створено програмне забезпечення для керування комп’ютером за допомогою жестів, використовуючи Python і бібліотеки, які спеціалізуються на розпізнаванні й відстеженні рук та комп’ютерному зорі, такі як MediaPipe та OpenCV відповідно. В ході розробки виконано поставлені задачі та досягнуто мету кваліфікаційної роботи бакалавра, проаналізовано технології розпізнавання жестів, їх застосування й актуальність, визначено засоби для реалізації проєкту, спроектовано структуру системи та реалізовано програмний застосунок.

На підставі проведеного аналітичного огляду, можна стверджувати, що технологія розпізнавання жестів є перспективним напрямком розвитку людино-машинної взаємодії, пропонуючи природний спосіб керування пристроями без фізичного їх застосування. Технологія демонструє універсальність та практичну цінність у різних областях – в ігровій індустрії, медицині, військовій справі тощо.

Концепція розпізнавання жестів базується на використанні різних алгоритмів машинного навчання, зокрема CNN, SVM, HMM та дерев рішень. Методи постійно вдосконалюються, проте все ще зіштовхуються з викликами щодо точності роботи й розпізнавання в складних умовах освітлення та відстані, що спонукає до пошуку альтернативних підходів для покращення цих показників в різних умовах.

На основі аналізу технологій, визначено функціонал системи, що включає операції переміщення курсором миші за допомогою вказівного пальця, клацання кнопками миші, використовуючи вказівний, середній та безіменний пальці, перетягування файлів двома пальцями, скролінг великим пальцем і регулювання гучності системи вертикальними рухами руки. Архітектуру змодельовано за допомогою UML-діаграм станів і прецедентів, які дозволяють визначити взаємодію компонентів та алгоритм роботи системи.

Для реалізації обрано мову програмування Python та набір спеціалізованих бібліотек: MediaPipe для відстеження рухів рук, OpenCV для комп’ютерного

зору, pynput для інтеграції з системними функціями керування, pycaw для управління звуком тощо. Виконання розробки проведено у середовищі VS Code.

Здійснено практичну реалізацію програмного застосунку для жестової взаємодії з комп’ютером. Структуру реалізовано за модульним принципом, що забезпечує гнучкість розробки та можливість розширення функціоналу надалі.

Проведено валідацію додатку та його функцій, виконуючи ті чи інші команди відповідними жестами. Застосунок запускається, команди, яким були призначені жести, виконуються справно.

Після успішного тестування, продукт перетворено у виконуваний файл та інсталяційний пакет. Таким чином, програмний застосунок можна передавати між користувачами та легко розгортати на їхніх пристроях.

При різних системних налаштуваннях пристроїв та умовах проведення, здійснено тестування точності, часу реакції та стабільного розпізнавання жестів зі сторони користувачів застосунку. Проаналізовано отримані результати, які показують високу ефективність системи з точністю розпізнавання жестів понад 95 % та середньою стабільністю роботи 8,7 балів з 10.

Було визначено перспективи і шляхи розвитку програмного застосунку. Надалі його можна вдосконалювати за рахунок покращення наявних алгоритмів розпізнавання жестів, а також розширювати функціональні можливості через додавання нових жестів та команд. Модульна архітектура програми створює основу для інтеграції допоміжних засобів взаємодії з комп’ютером (голосовий помічник, віртуальна клавіатура тощо), та дозволяє в майбутньому розробити можливість підтримки програмного забезпечення на різних операційних системах та пристроях.

Таким чином, розроблене рішення досягнуло мети із забезпечення керування комп’ютером жестами, демонструючи високу ефективність та потенціал для подальшого вдосконалення. Система може бути особливо корисною у ситуаціях, де використання миші незручне або неможливе, а також для людей з обмеженими руховими можливостями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Gesture Recognition? URL: https://www.geeksforgeeks.org/what-is-gesture-recognition/ (date of access: 05.01.2025).

2. Як технології комп’ютерного зору застосовуються в ритейлі. URL: https://www.imena.ua/blog/computer-vision-technologies-in-retail/ (дата звернення: 09.01.2025).

3. Yasen M., Jusoh S. A systematic review on hand gesture recognition techniques, challenges and applications. URL: https://peerj.com/articles/cs-218/ (date of access: 16.01.2025).

4. Yu H., Zheng D., Liu Y., Chen S., Wang X., Peng W. Low-Cost Self-Calibration Data Glove Based on Space-Division Multiplexed Flexible Optical Fiber Sensor. URL: https://www.mdpi.com/2073-4360/14/19/3935 (date of access: 23.01.2025).

5. Nourelhoda M. M., Fouad H., Soliman A. M. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex & Intelligent Systems. URL: https://link.springer.com/article/10.1007/s40747-020-00194-9 (date of access: 27.01.2025).

6. How Gesture Recognition Technology Shaping our future? URL: https://www.how2shout.com/technology/how-gesture-recognition-technology-shaping-our-future.html (date of access: 02.02.2025).

7. Prakash J. S., Prakash J. A., Pławiak P., Samantray S. Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network. URL: https://www.mdpi.com/1424-8220/22/3/706 (date of access: 10.02.2025).

8. Machine Learning Tutorial. URL: https://www.geeksforgeeks.org/machine-learning/ (date of access: 18.02.2025).

9. C++ Programming Language: Power, Versatility, and Considerations. URL: https://devcommunityhub.com/languages/c/c-programming-language-power-versatility-and-considerations/ (date of access: 26.02.2025).

10. Introduction to Java Programming. URL: https://www.w3docs.com/learn-java/java-intro.html (date of access: 01.03.2025).

11. TensorFlow.js is a library for machine learning in JavaScript. URL: https://www.tensorflow.org/js (date of access: 01.03.2025).

12. The Python Community. URL: https://www.python.org/community/ (date of access: 14.03.2025).

13. Gesture recognition. Python. Gesture recognition guide for Python. URL: https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer/python (date of access: 14.03.2025).

14. MediaPipe Solutions guide. URL: https://chuoling.github.io/mediapipe/ (date of access: 25.03.2025).

15. Gesture recognition. Gesture recognition task guide for MediaPipe. URL: https://shorturl.at/edg7O (date of access: 25.03.2025).

16. OpenCV documentation. URL: https://pypi.org/project/opencv-python/ (date of access: 04.04.2025).

17. Optical Flow in OpenCV. URL: https://learnopencv.com/optical-flow-in-opencv/ (date of access: 04.04.2025).

18. Pynput package. URL: https://pynput.readthedocs.io/en/latest/index.html (date of access: 16.04.2025).

19. Міронов Н. О., Самчук Л. М. Дослідження характеристик та медотології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій Computer Vision. Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк, 2024, № 57. С. 111-119.

20. Pycaw package. URL: https://pycaw.readthedocs.io/en/latest/index.html (date of access: 20.04.2025).

21. Visual Studio Code Docs. URL: https://code.visualstudio.com/docs (date of access: 22.04.2025).

22. PyCharm documentation. URL: https://www.jetbrains.com/help/pycharm/ (date of access: 22.04.2025).

ДОДАТКИ

Додаток А

Апробація у науковому журналі «Комп'ютерно-інтегровані технології: освіта, наука, виробництво» № 57

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ:
ОСВІТА, НАУКА, ВИРОБНИЦТВО

НАУКОВИЙ
ЖУРНАЛ

Головний редактор – професор, д.т.н., Гордєєв О.О.

№57 2024

м. Луцьк

ЗМІСТ

Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво" 111
Луцьк, 2024. Випуск № 57

DOI: https://doi.org/10.36910/6775-2524-0560-2024-57-13
УДК 004.5
Міронов Нікіта Олександрович, здобувач вищої освіти
Самчук Людмила Михайлівна, канд. техн. наук, доцент
https://orcid.org/0000-0003-2516-045X
Луцький національний технічний університет, м. Луцьк, Україна

ДОСЛІДЖЕННЯ ХАРАКТЕРИСТИК ТА МЕДОТОЛОГІЇ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ БЕЗКОНТАКТНОЇ ВЗАЄМОДІЇ З КОМП'ЮТЕРОМ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ COMPUTER VISION

Міронов Н.О., Самчук Л.М. Дослідження характеристик та методології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій Computer Vision. В роботі досліджено методи розпізнавання жестів для безконтактної взаємодії з комп'ютером, що використовують технології комп'ютерного зору та машинного навчання. Наведено приклади застосування в різних сферах діяльності людини, де вони забезпечують більшу безпеку, ефективність та зручність у використанні. Запропоновано методологію розпізнавання жестів рук у реальному часі. Система базується на алгоритмах, що використовують сегментацію зображень, виявлення контурів та класифікацію рухів рук. Основний акцент зроблено на застосуванні бібліотек OpenCV для обробки відеопотоків та MediaPipe для точного відстеження позицій пальців і рук. Мовою програмування для розробки системи є Python, який використовує дані та додаткові бібліотеки, що дозволяють інтегрувати різні функції. Розроблено діаграму прецедентів системи, визначено взаємодії компонентів, візуалізації сценаріїв використання та запропоновано шляхи застосування даної діаграми. Результатом дослідження є програмний продукт, який дозволяє керувати курсором миші за допомогою жестів рук. Система підтримує такі функції, як переміщення курсора, лівий та правий кліки, подвійний лівий клік, перетягування елементів, прокручування та регулювання гучності.
Ключові слова: розпізнавання жестів, комп'ютерний зір, віртуальна миша, машинне навчання, Python, MediaPipe, OpenCV, безконтактна взаємодія, обробка зображень, управління курсором.

Mironov N., Samchuk L. Study of characteristics and methodology of gesture recognition system for contactless interaction with a computer using Computer Vision technologies. The paper investigates methods of gesture recognition for contactless interaction with a computer using computer vision and machine learning technologies. Examples of application in various fields of human activity are given, where they provide greater safety, efficiency and ease of use. A methodology for recognizing hand gestures in real time is proposed. The system is based on algorithms that use image segmentation, contour detection, and hand motion classification. The main emphasis is placed on the use of OpenCV libraries for processing video streams and MediaPipe for precise tracking of finger and hand positions. The programming language used to develop the system is Python, which uses data and additional libraries to integrate various functions. A system precedent diagram was developed, the interactions of components were identified, use cases were visualized, and ways to use this diagram were proposed. The result of the study is a software product that allows you to control the mouse cursor using hand gestures. The system supports such functions as moving the cursor, left and right clicks, double left click, dragging elements, scrolling, and volume control.
Keywords: gesture recognition, computer vision, virtual mouse, machine learning, Python, MediaPipe, OpenCV, contactless interaction, image processing, cursor control.

Постановка наукової проблеми. Жести відіграють важливу роль у повсякденному житті людини як природний засіб спілкування. Вони використовуються для передачі емоцій, вираження думок і команд у взаємодії з іншими людьми [1]. Простота та універсальність жестів роблять їх ефективним засобом комунікації, зрозумілим незалежно від мови чи культурного контексту. Ця природна здатність до невербального спілкування відкриває можливості для її застосування і в технічних системах, зокрема для взаємодії з комп'ютерами.

Системи розпізнавання жестів для безконтактної взаємодії з комп'ютером набувають все більшої популярності завдяки можливості використання комп'ютерного зору та методів машинного навчання. Важливим аспектом таких систем є можливість зменшити потребу в фізичних пристроях, таких як миші та клавіатури, що особливо корисно для людей з обмеженими можливостями. Окрім того, такі технології відкривають нові перспективи для застосування в ігрових та віртуальних середовищах, де користувачі можуть управляти віртуальними об'єктами за допомогою рухів рук, подібно до взаємодії з фізичними предметами.

Аналіз досліджень. Системи розпізнавання жестів базуються на використанні технологій комп'ютерного зору, що дозволяє виявляти та аналізувати рухи користувача в реальному часі. Однією з найбільш використовуваних технологій є OpenCV, яка надає інструменти для обробки зображень та відеопотоків. Наприклад, у системах на основі OpenCV фіксуються рухи пальців та рук для симуляції натискання кнопок миші або для управління курсором [2].

Через OpenCV та обробку зображень можна використовувати систему маркерів, яка дозволяє контролювати комп'ютер за допомогою кольорових точок на кінчиках пальців, що забезпечує більш

стабільний контроль курсора і виконання команд. Цей метод використовує обробку зображень для виявлення відмінностей у кольорі та положенні пальців, що дозволяє системі визначати різні команди, такі як лівий або правий клік.

Дослідження показують, що використання методів глибокого навчання, таких як згорткові нейронні мережі (CNN), дозволяє досягати високої точності в розпізнаванні жестів. В одній із робіт, що дослідили дане питання [3], система на основі CNN дозволила здійснювати реальний контроль над комп'ютером за допомогою жестів рук, забезпечуючи точне відстеження та аналіз рухів. Такі моделі дозволяють системам адаптуватися до різних типів жестів і зменшують вплив зовнішніх факторів, таких як зміни освітлення або позицій користувача.

У багатьох дослідженнях підкреслюється перевага систем розпізнавання жестів, особливо в умовах обмеженого простору або під час пандемій, коли важливо мінімізувати фізичний контакт із пристроями. Наприклад, під час розпалу COVID-19 розроблено систему безконтактної взаємодії з комп'ютером для зменшення поширення вірусу [4]. Це підтверджує актуальність та практичність застосування таких технологій у реальному житті.

Мета роботи. Метою та одним із напрямків дослідження є розробка ефективних алгоритмів для точного розпізнавання жестів, що використовують методи обробки зображень для перетворення рухів рук у команди. Важливим питанням є також забезпечення достатньої швидкодії системи при мінімальних вимогах до апаратного забезпечення.

Виклад основного матеріалу й обгрунтування отриманих результатів дослідження. Однією з ключових проблем, яка потребує вирішення, є точність розпізнавання жестів у складних умовах, наприклад, при змінному освітленні або у випадку часткового перекриття рук. Деякі системи пропонують використовувати електроміографічні (EMG) сигнали [5], що дозволяє підвищити точність розпізнавання жестів, аналізуючи сигнали від м'язів передпліччя. Це особливо корисно в середовищах, де камера не може забезпечити достатньо точне зчитування.

Різні системи розпізнавання жестів використовують різноманітну техніку, включаючи звичайні веб-камери, інфрачервоні сенсори та спеціалізовані пристрої, такі як Leap Motion. Останній забезпечує точне виявлення рухів пальців та рук людини [6]. Leap Motion дозволяє використовувати інтуїтивні жести для виконання різних завдань, таких як переміщення курсора, прокручування сторінок, регулювання гучності на комп'ютері та інших команд.

Системи розпізнавання жестів також знаходять застосування у спеціалізованих середовищах, таких як програми для моделювання в CAD-системах. Наприклад, використання EMG-сигналів для керування програмами, такими як Solidworks, дозволяє інженерам взаємодіяти з інтерфейсом безпосередньо за допомогою жестів.

Технології розпізнавання жестів також використовуються в комп'ютерних іграх і системах доповненої та віртуальної реальності для більш природної взаємодії з віртуальними об'єктами. Kinect, Leap Motion, VR-шоломи та інші подібні системи дозволяють користувачам маніпулювати віртуальними об'єктами за допомогою рухів не лише рук, а й, в окремих випадках, усього тіла, що створює більш інтерактивний досвід. Це дозволяє користувачам діяти у віртуальних середовищах так, як вони б взаємодіяли з реальними об'єктами.

Системи розпізнавання жестів все дедалі частіше використовуються і в інших сферах життя людини. До прикладу, в сучасних автомобілях вони дозволяють знизити відволікання водія під час керування, підвищують безпеку і комфорт. Вони дозволяють керувати мультимедійними та навігаційними пристроями, що забезпечує безконтактну взаємодію під час руху. Прикладом є компанії BMW та Mercedes-Benz, які впроваджують ці технології у свої преміальні автомобілі, дозволяючи водіям керувати основними функціями автомобіля за допомогою жестів. Ці інновації не тільки підвищують зручність водіння, але й знижують ризики аварій.

У військових операціях, особливо в інженерно-саперній справі, розпізнавання жестів стає ефективним інструментом безпеки. Віддалене керування роботами з маніпуляторами дає змогу виконувати небезпечні операції з мінімальною загрозою для життя [7]. Наприклад, рукавички DataGlove передають положення рук у реальному часі, дозволяючи керувати роботами з високою точністю. Це знижує ризики при розмінуванні та інших небезпечних завданнях.

Для навчання та розпізнавання команд у військовій справі використовуються системи на базі Kinect, які визначають рухи рук і передають команди. Ці системи дозволяють виявляти команди, що робить їх ефективними у військових операціях. Наприклад, логістична модель на базі Kinect продемонструвала ефективність розпізнавання жестів з точністю 96,75% [8].

Військові дослідження зосереджуються на створенні систем, здатних не тільки розпізнавати навчальні команди, але й вивчати нові за допомогою глибокого навчання. Такі системи використовують нейронні мережі, що дозволяє роботизованим партнерам адаптуватися до нових ситуацій та команд. Цей підхід має потенціал значно розширити можливості автоматизованих систем у бойових умовах, що, своєю чергою, в умовах військової ситуації на території України, є вкрай необхідним та дуже актуальним під час виконання операцій.

Хоча багато сучасних систем демонструють високу точність, вони ще мають низку обмежень, таких як обмежена кількість розпізнаваних жестів та складність адаптації до нових сценаріїв. Для подальшого розвитку необхідно зосередитись на створенні адаптивних систем, здатних вивчати нові жести та пристосовуватися до різних умов використання.

Пропонована система розпізнавання жестів використовує технологію комп'ютерного зору, що дозволяє керувати курсором миші та виконувати основні дії, такі як кліки, прокручування та перетягування об'єктів, без фізичного дотику до пристрою. Система базується на використанні веб-камери для захоплення рухів рук, які обробляються та інтерпретуються програмним забезпеченням для виконання відповідних команд.

Технологія комп'ютерного зору (Computer Vision, CV) є основою системи розпізнавання жестів і дозволяє комп'ютеру «бачити» та інтерпретувати візуальну інформацію з камер або інших сенсорів. Комп'ютерний зір має в собі набір алгоритмів та методів, що обробляють та аналізують зображення чи відео для розпізнавання об'єктів, рухів та інших характеристик сцени [9]. У контексті системи розпізнавання жестів, Computer Vision використовується для виявлення рук користувача, визначення їхнього положення у просторі та розпізнавання специфічних жестів.

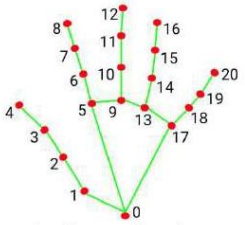
Процес включає кілька етапів:

– захоплення зображення;
– попередня обробка (фільтрація, покращення якості зображення);
– сегментація (виділення області рук);
– виявлення ключових точок (пальців та суглобів);
– інтерпретація жестів для перетворення їх у команди для управління комп'ютером.

Основною мовою програмування для розробки цієї системи є Python, що надає широкий спектр бібліотек для реалізації алгоритмів CV, обробки зображень та машинного навчання. Python був обраний завдяки його гнучкості, великій кількості доступних інструментів для обробки даних та простоті інтеграції з різними бібліотеками. У проєкті використовуються ключові бібліотеки, такі як OpenCV та MediaPipe, які забезпечують розпізнавання жестів у реальному часі.

OpenCV (Open Source Computer Vision Library) є однією з найважливіших бібліотек у системі. Вона дозволяє захоплювати та обробляти зображення з камери, а також виконувати різноманітні операції, пов'язані з обробкою зображень, такі як виявлення контурів, сегментація та розпізнавання об'єктів. У даній системі OpenCV відповідає за зчитування відеопотоку з камери, обробку кадрів та підготовку даних для подальшого аналізу.

MediaPipe – це бібліотека для обробки мультимедійних потоків у реальному часі, яка широко використовується для розпізнавання рухів тіла, обличчя та рук [10]. У даній розробці MediaPipe виконує функцію трекінгу рухів рук та пальців, що дозволяє системі точно розпізнавати жести користувача. Використання бібліотеки MediaPipe дозволяє досягти високої точності та швидкості у розпізнаванні жестів без необхідності використання спеціалізованого апаратного забезпечення. Даний фреймворк дозволяє використати попередньо навчену модель для точної ідентифікації 21 різних точок на руці, включаючи пальці та суглоби долоні (рис. 1).



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Рисунок 1 – Ідентифікація точок на руці навченої моделі бібліотеки MediaPipe

Окрім OpenCV та MediaPipe, у системі також застосовуються NumPy, pynput та інші бібліотеки. NumPy використовується для роботи з масивами даних та виконання математичних операцій. У контексті системи розпізнавання жестів ця бібліотека дозволяє ефективно обробляти числові дані, отримані з веб-камери, та виконувати обчислення, необхідні для аналізу рухів рук. Pynput відповідає за взаємодію з інтерфейсом користувача. Вона дозволяє програмно керувати мишею, виконуючи такі дії, як кліки, прокручування сторінок, переміщення курсора тощо.

Запустивши виконувану програму, система використовує функцію cv2.VideoCapture() з бібліотеки OpenCV для підключення до веб-камери. Ця функція забезпечує постійний потік кадрів у реальному часі, які будуть оброблятися програмою для виявлення жестів. Кожен кадр є окремим зображенням, яке програма аналізує. Кожен кадр з відеопотоку проходить попередню обробку. В Зображення може бути перетворене в інший колірний простір для точнішого виявлення об'єктів, таких як шкіра рук. Це дозволяє ефективніше розрізняти фон та руки користувача.

Для розпізнавання рук використовується навчена модель MediaPipe, яка дозволяє визначити необхідні точки на руці. Фреймворк забезпечує функцію виявлення та відстеження положення рук у кадрі незалежно від їхнього руху.

Коли система ідентифікує руку, вона починає аналізувати положення пальців для визначення жестів. Наприклад, якщо всі пальці розтягнуті, а долоня рухається по вертикальній осі, у цьому випадку, це інтерпретується як команда для регулювання гучності (рис. 2).

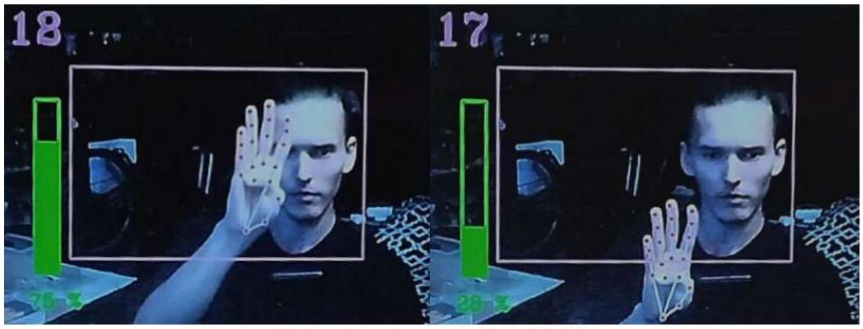


Рисунок 2 – Регулювання гучності

Якщо ж лише один вказівний палець піднятий, це інтерпретується як жест для руху курсора миші (рис. 3). Важливою частиною є розпізнавання просторових відносин між ключовими точками.

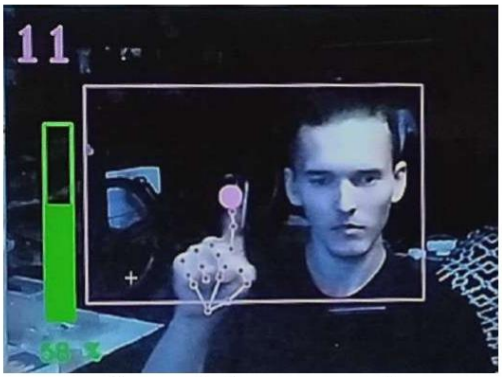


Рисунок 3 – Рух курсора миші

Використовуючи дані про позиції рук і пальців, програма передає ці координати бібліотеці pynput для керування курсором миші. Pynput може виконувати різні дії, такі як переміщення курсора на екрані залежно від положення долоні або виконання кліків при виявленні специфічних жестів (наприклад, розмикання пальців для кліку).

Для виконання дій, таких як кліки або прокручування сторінки, програма використовує додаткові функції pynput. Наприклад, при виявленні жесту розмикання вказівного і середнього пальців, програма виконує лівий клік миші (рис. 4).

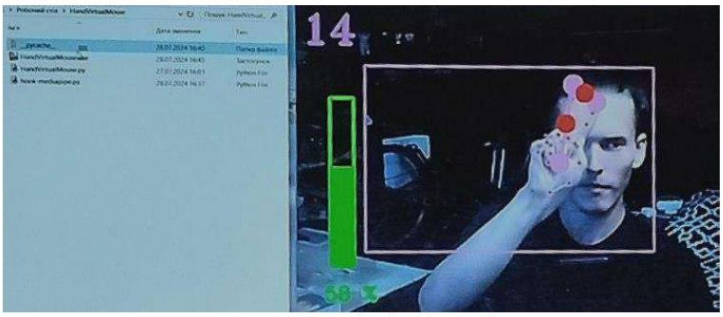


Рисунок 4 – Лівий клік миші

Прокручування реалізоване через положення великого пальця (рис. 5). Кожен жест вимагає специфічного обчислення на основі координат ключових точок на руці, що дозволяє програмі визначити точну дію.

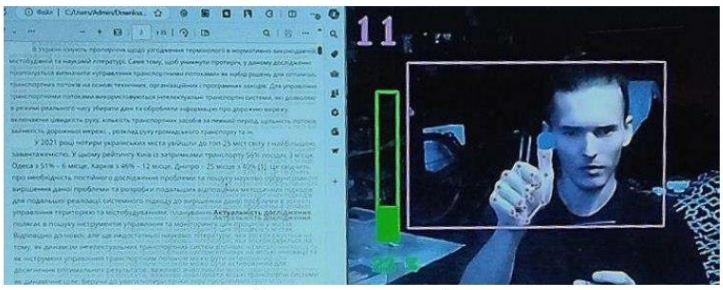


Рисунок 5 – Прокручування сторінок документа

Щоб уникнути «дрижання» курсору через незначні коливання рук, програма використовує методи фільтрації. Це включає згладжування координат або використання алгоритмів, що запобігають надто чутливій реакції системи на незначні рухи користувача. Плавне та згладжене пересування курсора миші досягається завдяки використанню функцій бібліотеки pynput. Це дозволяє системі реагувати на жести користувача з більшим рівнем стабільності та плавності під час переміщення курсора.

Після обробки кожного кадру програма виводить результат на екран, показуючи положення руки або виконані дії. Коли програма завершена або камера більше не потрібна, використовується функція для закриття відеопотоку та звільнення ресурсів системи.

У процесі розробки системи розпізнавання жестів важливим етапом є моделювання основних сценаріїв використання. Для цього ефективним інструментом є UML діаграми прецедентів (Use Case Diagrams), які дозволяють візуалізувати взаємодію користувача із системою та визначити ключові функції, які вона підтримує. Ці діаграми є особливо корисними на етапі проєктування, оскільки дають змогу краще зрозуміти, як користувачі будуть взаємодіяти з програмою та які компоненти відповідають за реалізацію певних дій.

UML діаграма прецедентів демонструє взаємодію між актором (користувачем або іншим зовнішнім елементом) та системою через конкретні прецеденти використання (use cases), що описують дії, які може виконати користувач [11]. У випадку даної системи, прецедентами можуть бути сценарії виконання певних команд за допомогою жестів, таких як переміщення курсора, виконання кліку тощо. Кожен з цих сценаріїв показує, як система відповідає на конкретні команди користувача.

Діаграми прецедентів також допомагають виявити основні компоненти системи, що беруть участь у виконанні цих дій. Наприклад, компонент, що відповідає за розпізнавання рухів рук, отримує дані з камери, а потім передає їх для обробки модулем керування інтерфейсом, який взаємодіє з операційною системою для виконання дій, таких як рух курсора або кліки.

Таким чином, UML діаграми дозволяють не лише відобразити взаємодію користувача з системою, але й деталізувати її внутрішні компоненти, показуючи, які модулі відповідають за конкретні функції. Для роз'яснення сценаріїв використання та компонентів системи розпізнавання жестів, використано UML діаграму прецедентів, що допомагає візуалізувати основні функціональні можливості системи та взаємодію користувача з нею (рис. 6). Опис ролі акторів та прецедентів системи наведено в табл. 1 і табл. 2 відповідно.

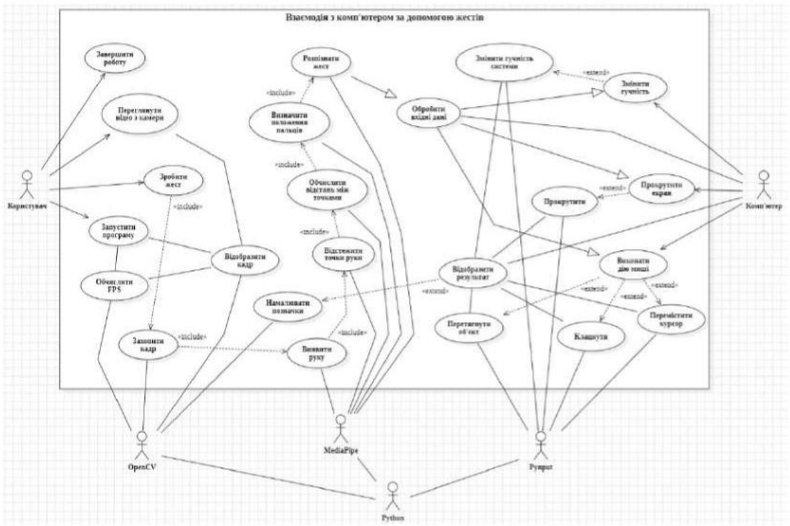


Рисунок 6 – Діаграма прецедентів процесу взаємодії з комп'ютером за допомогою жестів

Таблиця 1 – Опис ролі акторів системи

Актор	**Опис ролі**
Користувач	Особа, яка взаємодіє з системою. Користувач виконує жести перед камерою, які система інтерпретує як команди для керування комп'ютером. Він може запускати та завершувати роботу програми, а також виконувати різні дії на комп'ютері за допомогою жестів.
Комп'ютер	Пристрій, яким керує користувач за допомогою жестів. Комп'ютер виконує команди, такі як рух курсора, кліки, прокрутка, зміна гучності тощо. Він також надає зворотний зв'язок користувачеві через візуальний інтерфейс.
Python	Мова програмування, яка використовується для розробки та виконання системи. Python забезпечує основу для інтеграції різних компонентів системи, обробки даних та реалізації логіки управління. Відповідає за координацію роботи всіх компонентів та забезпечує загальну функціональність програми. У цій

	системі Python розділений на три основні бібліотеки: MediaPipe, OpenCV та pynput.
MediaPipe	Бібліотека, яка відповідає за розпізнавання та відстеження рук користувача. MediaPipe аналізує відеопотік з камери, виявляє руки, визначає положення ключових точок руки та інтерпретує жести.
OpenCV	Бібліотека обробки зображень, яка використовується для захоплення та обробки відео з камери. OpenCV забезпечує функціональність для перетворення кольорового простору, малювання на кадрах та відображення результатів користувачеві.
Pynput	Бібліотека для автоматизації взаємодії з графічним інтерфейсом користувача. Pynput виконує фактичні дії на комп'ютері, такі як переміщення курсора, кліки, прокрутка та зміна гучності, на основі розпізнаних жестів.

Таблиця 2 – Опис прецедентів системи

Прецедент	**Опис прецеденту**
Запустити програму	Ініціалізація програмного забезпечення, що включає активацію камери, завантаження необхідних бібліотек та підготовку системи до розпізнавання жестів.
Зробити жест	Процес виконання користувачем певного руху рукою перед камерою. Цей жест буде захоплено камерою та проаналізовано системою для подальшої інтерпретації як команди для комп'ютера.
Переглянути відео з камери	Відображення користувачеві відео в реальному часі з камери, що дозволяє йому бачити, як система «бачить» його руки та жести.
Завершити роботу	Коректне завершення роботи системи, що включає вивільнення ресурсів, закриття відеопотоку тощо.
Обробити вхідні дані	Процес аналізу відеопотоку з камери, виділення кадрів та їх попередня обробка для подальшого аналізу жестів.
Виконати дію миші	Реалізація команд, пов'язаних з керуванням курсором миші, таких як переміщення курсора, клік, подвійний клік або перетягування об'єктів.
Змінити гучність	Процес регулювання системної гучності комп'ютера на основі розпізнаних жестів користувача.
Прокрутити екран	Виконання прокрутки вмісту активного вікна вгору або вниз на основі жестів користувача.
Відобразити результат	Візуалізація результатів розпізнавання жестів та виконаних дій на екрані для надання користувачеві зворотного зв'язку.
Виявити руку	Процес ідентифікації наявності руки в кадрі відеопотоку з камери.
Відстежити точки руки	Визначення положення ключових точок руки (таких як суглоби пальців) у тривимірному просторі.
Визначити положення пальців	Аналіз взаємного розташування точок руки для визначення, які пальці підняті, а які опущені.
Розпізнати жест	Інтерпретація конфігурації руки та руху пальців як конкретного жесту, що відповідає певній команді.
Обчислити відстань між точками	Розрахунок відстаней між ключовими точками руки для визначення специфічних жестів, наприклад, для розрізнення кліків миші.
Захопити кадр	Отримання поточного кадру з відеопотоку камери для подальшої обробки.
Намалювати позначки	Додавання візуальних елементів на кадр для відображення

	розпізнаних рук, ключових точок тощо.
Відобразити кадр	Показ кадру з візуальними позначками користувачеві для зворотного зв'язку.
Обчислити FPS	Розрахунок кількості кадрів за секунду (FPS) для оцінки продуктивності системи та відображення цієї інформації користувачеві.
Перемістити курсор	Зміна позиції курсора на екрані відповідно до руху руки користувача.
Клацнути	Виконання кліку (лівою або правою кнопкою миші) на основі розпізнаного жесту.
Перетягнути об'єкт	Імітація натискання та утримання кнопки миші для переміщення об'єктів на екрані.
Прокрутити	Виконання вертикальної або горизонтальної прокрутки вмісту активного вікна.
Змінити гучність системи	Регулювання рівня системної гучності комп'ютера.

Як видно на рисунку 6, з UML діаграми прецедентів системи розпізнавання жестів для взаємодії з комп'ютером, цей вид моделювання дозволяє ефективно візуалізувати ключові етапи взаємодії користувача з системою. У діаграмі відображено, як користувач ініціює дії за допомогою жестів, і як система розпізнає та обробляє ці команди. Операції, такі як переміщення курсора, зміна гучності, прокручування екрана і так далі, деталізовано через взаємодію з конкретними функціями програми. Це дає чітке уявлення про те, як різні компоненти програми, зокрема бібліотеки OpenCV, MediaPipe та pynput, об'єднуються для досягнення поставленої мети.

Діаграма прецедентів також допомагає визначити всі ключові компоненти системи, які відповідають за обробку жестів та взаємодію з комп'ютером. Завдяки візуалізації процесу, розробники можуть краще розуміти, як реалізуються основні функції системи. Крім того, це сприяє фіксації вимог до системи та покращенню її надійності. Діаграма дозволяє бачити взаємозв'язки між компонентами, зокрема, як система розпізнає положення пальців, обчислює відстань між точками та відображає результати у вигляді дій миші.

Застосування UML діаграми для даної системи також відкриває можливість для подальшого її розвитку та вдосконалення. Використовуючи такі діаграми, можна покращити комунікацію між розробниками й тестувальниками, а також задокументувати всі процеси для полегшення майбутніх змін у системі. UML діаграми можуть бути застосовані для моделювання нових сценаріїв використання, таких як інтеграція нових жестів або підключення додаткових пристроїв вводу, що дозволить адаптувати систему до нових викликів та потреб користувачів.

Висновки. Розробка віртуальної миші на основі розпізнавання жестів демонструє значний прогрес у сфері безконтактної взаємодії з комп'ютером. Застосування комп'ютерного зору та алгоритмів машинного навчання забезпечує високу точність розпізнавання жестів у реальному часі.

Створено програмний продукт, який дозволяє керувати курсором за допомогою жестів рук, використовуючи веб-камеру. Система підтримує базові функції миші, такі як лівий і правий клік, переміщення курсора, прокручування та перетягування елементів, а також функцію регулювання гучності на комп'ютері.

Для покращення системи доцільно впровадити вдосконалені алгоритми виявлення кінчиків пальців для підвищення точності та швидкості взаємодії. Також варто звернути увагу на забезпечення стабільної роботи системи в різних умовах освітлення, що дозволить використовувати її в ширшому діапазоні ситуацій. Додатково можна інтегрувати підтримку багатопальцевих жестів та динамічних рухів для розширення функціональних можливостей.

Майбутні вдосконалення можуть включати розробку адаптивних систем, які зможуть динамічно підлаштовуватись під умови користувача та середовища. Інтеграція голосових команд та відстеження рухів очей разом з жестами рук також може підвищити зручність використання. Крім того, важливо забезпечити кросплатформну сумісність, що дозволить системі бути інтегрованою в різні операційні системи та пристрої.

Список бібліографічного опису

1. Roshnee Matlani, Roshan Dadlani, Sharv Dumbre, Shruti Mishra, Abha Tewari. Virtual Mouse using Hand Gestures. International Conference on Technological Advancements and Innovations. 2021. P. 12-13.

2. Mr.E.Sankar, B.Nitish Bharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Research in Engineering and Management. Vol. 7. No. 5. May 2023. P. 2-4.
3. C. Cao, Y. Gao, S. Zhang. Real-Time Hand Gesture Detection and Recognition Using Convolutional Neural Networks. 2017. P. 30-42.
4. S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, P. Ajay. Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. Journal of Healthcare Engineering. 2021. P. 4-7.
5. Ala-Addin Nabulsi. Hand Gesture Recognition via Electromyographic (EMG) Armband for CAD Software control. 2018. P. 49-59.
6. Prutha Atre, Sahil Bhagat, Nevil Pooniwala, Payal Shah. Efficient and Feasible Gesture Controlled Robotic Arm. 2018. P. 34-54.
7. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex & Intelligent Systems. 2020. P. 7-12.
8. Michael Hamilton, Patrick Mead, Megan Kozub, Alexander Felid. Gesture Recognition Model for Robotic Systems of Military Squad Commands. 2016. P. 23-25.
9. Як технології комп'ютерного зору застосовуються в ритейлі. URL: https://www.imena.ua/blog/computer-vision-technologies-in-retail/
10. MediaPipe. Framework Concepts. URL: https://chuoling.github.io/mediapipe/
11. Documentation Plant UML. URL: https://plantuml.com/

References

1. Roshnee Matlani, Roshan Dadlani, Sharv Dumbre, Shruti Mishra, Abha Tewari. Virtual Mouse using Hand Gestures. International Conference on Technological Advancements and Innovations. 2021. P. 12-13.
2. Mr.E.Sankar, B.Nitish Bharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Research in Engineering and Management. Vol. 7. No. 5. May 2023. P. 2-4.
3. C. Cao, Y. Gao, S. Zhang. Real-Time Hand Gesture Detection and Recognition Using Convolutional Neural Networks. 2017. P. 30-42.
4. S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, P. Ajay. Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. Journal of Healthcare Engineering. 2021. P. 4-7.
5. Ala-Addin Nabulsi. Hand Gesture Recognition via Electromyographic (EMG) Armband for CAD Software control. 2018. P. 49-59.
6. Prutha Atre, Sahil Bhagat, Nevil Pooniwala, Payal Shah. Efficient and Feasible Gesture Controlled Robotic Arm. 2018. P. 34-54.
7. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex & Intelligent Systems. 2020. P. 7-12.
8. Michael Hamilton, Patrick Mead, Megan Kozub, Alexander Felid. Gesture Recognition Model for Robotic Systems of Military Squad Commands. 2016. P. 23-25.
9. How computer vision technologies are used in retail. URL: https://www.imena.ua/blog/computer-vision-technologies-in-retail/
10. MediaPipe. Framework Concepts. URL: https://chuoling.github.io/mediapipe/
11. Documentation Plant UML. URL: https://plantuml.com/

Додаток Б

Апробація у науковому журналі «Комп'ютерно-інтегровані технології: освіта, наука, виробництво» № 58

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ:
ОСВІТА, НАУКА, ВИРОБНИЦТВО

НАУКОВИЙ
ЖУРНАЛ

Головний редактор – професор, д.т.н., Гордєєв О.О.

№58 2025

м. Луцьк

ЗМІСТ

DOI: https://doi.org/10.36910/6775-2524-0560-2025-58-10
УДК 004.5
Міронов Нікіта Олександрович, здобувач вищої освіти
Самчук Людмила Михайлівна, к.т.н, доцент
https://orcid.org/0000-0003-2516-045X
Луцький національний технічний університет, м. Луцьк, Україна

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ВЗАЄМОДІЇ З КОМП'ЮТЕРОМ ЗА ДОПОМОГОЮ ЖЕСТІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ COMPUTER VISION

Міронов Н.О., Самчук Л.М. Розробка застосунку для взаємодії з комп'ютером за допомогою жестів з використанням технологій ComputerVision. У роботі досліджено методи безконтактного керування комп'ютером за допомогою жестів рук на основі технологій комп'ютерного зору та машинного навчання. Наведено приклади застосування технологій в різних сферах життя, де вони показують переважну ефективність та зручність над традиційними інтерфейсами взаємодії з системою. Описано засоби та інструменти, використані під час розробки програмного застосунку. Запропоновано модульну архітектуру системи, яка забезпечує розпізнавання жестів у реальному часі, включаючи сегментацію зображень, відстеження ключових точок руки та класифікацію рухів. Реалізовано функціонал для переміщення курсора, виконання лівих та правих кліків, подвійних клацань, прокручування сторінок і регулювання гучності. Результатом роботи є програмний продукт, який підтримує переміщення курсора миші, лівий і правий кліки, подвійний клік, перетягування об'єктів, скролінг та регулювання гучності. Проведено тестування як функціонування основних команд застосунку, так і роботи на різних апаратних конфігураціях в реальному часі. Запропоновано рішення для вдосконалення надалі й розширення функціоналу програмного забезпечення.
Ключові слова: розпізнавання жестів, безконтактна взаємодія, комп'ютерний зір, машинне навчання,Python, MediaPipe, OpenCV, безконтактна взаємодія, обробка зображень, відеопотік, управління і виконання функційміші, кросплатформність, адаптивні алгоритми.

Mironov N., Samchuk L. Development of an application for interaction with a computer using gestures using Computer Vision technologies. The work investigates methods for contactless computer control using hand gestures based on computer vision and machine learning technologies. Examples of the application of technologies in various areas of life are given, where they show superior efficiency and convenience over traditional interfaces for interacting with the system. The tools and instruments used in the development of the software application are described. A modular architecture of the system is proposed that provides real-time gesture recognition, including image segmentation, tracking of key hand points and classification of movements. Functionality for moving the cursor, performing left and right clicks, double clicks, scrolling pages and adjusting the volume is implemented. The result of the work is a software product that supports moving the mouse cursor, left and right clicks, double clicks, dragging objects, scrolling and adjusting the volume. Both the functioning of the main application commands and the operation on different hardware configurations in real time were tested. Solutions for further improvement and expansion of the software functionality were proposed.
Keywords: gesture recognition, contactless interaction, computer vision, machine learning, Python, MediaPipe, OpenCV, contactless interaction, image processing, video stream, mouse control and execution, cross-platform, adaptive algorithms.

Постановка наукової проблеми. Сучасні технології взаємодії з комп'ютером, такі як клавіатури та миші, залишаються основним інструментом введення даних. Однак можна визначити ситуації, коли вони можуть бути не зовсім доречні. Наприклад, фізичний контакт з пристроями може бути небажаним в умовах підвищених вимог до гігієни або під час пандемій, а для людей з обмеженими руховими можливостями такі девайси часто є недоступними або незручними. Крім того, у спеціалізованих установах медицини, промисловості, або в сфері віртуальної реальності, зростає потреба в інтуїтивній безконтактній взаємодії, яка імітує природні рухи людини.

Актуальність розробки системи жестового керування комп'ютером полягає в необхідності створення універсальної та ефективної системи, здатної замінити традиційні пристрої введення. Така система має забезпечувати високу точність розпізнавання жестів навіть у несприятливих зовнішніх умовах, таких як змінне освітлення. Крім того, важливим є мінімізація затримки для забезпечення роботи в реальному часі, що дозволить досягти природної плавності керування.

Вирішення цієї проблеми відкриває нові можливості для застосування технології в медицині (керування обладнанням без фізичного контакту), освіті (інтерактивні презентації), військовій промисловості (віддалене виконання безпечних операцій) та для користувачів з обмеженими можливостями. Таким чином, розробка стабільної та доступної системи керування жестами є критично важливою для прогресу в галузі людино-машинної взаємодії надалі.

Аналіз досліджень. Сучасні системи безконтактного керування спираються на технології комп'ютерного зору, які дають змогу розпізнавати та інтерпретувати рухи без затримок. Серед ключових інструментів використовується бібліотека OpenCV, що надає функціонал для маніпуляцій із зображеннями та аналізу відеоданих. Зокрема, за її допомогою реалізуються алгоритми

відстеження положення рук і пальців, що дозволяє імітувати дії миші, такі як пересування курсору або виконання натискань.

Окремим підходом є використання маркерних систем, де на пальцях розміщуються кольорові маркери. Ця методика, заснована на OpenCV, покращує стабільність керування завдяки точній ідентифікації кольорових точок [1]. Алгоритми обробки зображень аналізують зміни у відтінках та просторовому розташуванні маркерів, що дає змогу розрізняти конкретні команди (наприклад, лівий або правий клік). Таке рішення усуває неоднозначність у розпізнаванні жестів, особливо в умовах швидких рухів або проблемного освітлення.

Ще одним перспективним підходом є система, запропонована у роботі [2], яка використовує згорткові нейронні мережі (CNN) для високоточного розпізнавання жестів у реальному часі. Дослідження демонструє ефективність поєднання мови Python та глибокого навчання для створення безконтактного інтерфейсу, здатного до роботи в різних складних умовах. Методика передбачає попередню обробку зображень, сегментацію області руки та вилучення ключових ознак для подальшої класифікації жестів. Особливістю цього підходу є розширення навчальної вибірки за допомогою методів аугментації, що дозволило підвищити точність розпізнавання з 96,9% до 99,2%. Запропоноване рішення було протестоване в сценаріях управління роботизованим маніпулятором, що підтвердило його придатність для промислового використання.

У дослідженні [3] запропоновано підхід до розпізнавання жестів для безконтактної взаємодії з комп'ютером у реальному часі, де автори використовують комбінацію бібліотек OpenCV та MediaPipe. Розроблена методологія передбачає сегментацію зображень, виявлення контурів рук і точне визначення положення пальців, що забезпечує розпізнавання жестів. Функціонал системи включає управління курсором, виконання натискань, прокручування, перетягування об'єктів та регулювання гучності. Особливістю запропонованого рішення є його ефективність у реальному часі за умови мінімальних обчислювальних ресурсів, що робить систему придатною для широкого спектра застосувань.

Наукові праці неодноразово акцентують переваги технологій розпізнавання жестів, зокрема у сценаріях, де зменшення фізичного контакту з поверхнями є критичним – наприклад, у замкнутих просторах або під час спалахів інфекційних захворювань. Ілюстрацією цього слугує система[4], розроблена на етапі активного поширення COVID-19, яка дозволяє керувати комп'ютером через жести без необхідності дотику до пристроїв, тим самим мінімізуючи ризик передачі патогенів. Подібні ініціативи підкреслюють не лише теоретичний потенціал таких рішень, але й їхню практичну цінність у сучасних умовах.

У контексті військових операцій, зокрема при виконанні інженерно-саперних завдань, технології розпізнавання жестів відіграють вирішальну роль у забезпеченні безпеки персоналу. Використання дистанційно керованих роботів з функціональними маніпуляторами[5] дозволяє виконувати операції з обмеженим безпосереднім втручанням людини, що знижує прямий ризик для життя. Яскравим прикладом є система DataGlove, яка фіксує та передає координати рук оператора в режимі реального часу, забезпечуючи управління робототехнічними комплексами. Такі інновації значно зменшують імовірність помилок під час знешкодження вибухових пристроїв, де кожен міліметр контролю має фатальне значення.

Мета роботи. Метою дослідження є розробка ефективної системи безконтактного керування комп'ютером за допомогою жестів. Основним завданням є створення програмного рішення на базі технологій комп'ютерного зору (OpenCV) та відстеження рухів (MediaPipe), здатного замінити пристрої введення, такі як миша або клавіатура. Система має забезпечувати реалізацію базових функцій миші та роботу в реальному часі. Окрім технічних аспектів, робота спрямована на аналіз точності та стабільності системи через тестування з участю користувачів у різних сценаріях.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Сучасні системи розпізнавання жестів, попри високий рівень точності, стикаються з суттєвими обмеженнями, зокрема обмеженою кількістю підтримуваних жестів та труднощами адаптації до нових сценаріїв використання. Наприклад, багато рішень не здатні динамічно впроваджувати нові жести без повної переробки алгоритмів, що обмежує їх застосування в складних умовах. Для подолання недоліків у даній роботі пропонується система, яка інтегрує технології комп'ютерного зору (CV) для створення інтуїтивного безконтактного інтерфейсу керування.

Для реалізації системи обрано мову Python, як інтерпретовану мову програмування, що є ключовим інструментом у сфері машинного навчання та комп'ютерного зору завдяки своїй гнучкості та великій кількості бібліотек [6]. Синтаксис мови, зрозумілий навіть новачкам, і

можливість швидкого прототипування роблять його ідеальним для розробки системи розпізнавання жестів.

Python розширює набір доступних інструментів, та крос-платформну сумісність, що дозволяє запускати код на різних ОС без змін. Проте мова має обмеження: порівняно з C++, швидкість виконання нижча, а залежність від зовнішніх бібліотек іноді ускладнює сумісність версій. Незважаючи на це, Python залишається оптимальним вибором для швидкої ітерації та тестування ідей, особливо на ранніх етапах проекту. Використання бібліотек MediaPipe та OpenCV дає змогу створювати системи, які аналізують рухи рук у реальному часі, перетворюючи їх у команди для керування комп'ютером.

MediaPipe, розроблена Google, є основним інструментом для розпізнавання жестів у даній системі. Вона використовує попередньо навчені моделі машинного навчання, щоб ідентифікувати 21 ключову точку на руці [7], включаючи суглоби пальців та основу зап'ястя. Процес починається з сегментації зображення, де алгоритми виділяють область, що містить руку, ізолюючи її від фону за допомогою обмежувальних рамок. Далі, за допомогою глибоких нейронних мереж, система прогнозує точні координати кожної точки, враховуючи дані з попередніх кадрів для забезпечення плавності відстеження. Наприклад, коли користувач згинає вказівний палець, модель MediaPipe визначає зміну кута між фалангами та передає ці дані для подальшої обробки.

OpenCV відповідає за обробку відеопотоку. Бібліотека застосовує такі методи, як фільтрація шумів, корекція кольорового балансу та перетворення зображень у простір HSV, щоб підготувати дані для точного аналізу. Важливу роль відіграє метод Лукаса-Канаде, який визначає оптичний потік [8]– вектор зміщення ключових точок між послідовними кадрами. Це дозволяє системі відстежувати не лише статичні позиції руки, але й її динамічні рухи.

Для перетворення розпізнаних жестів у конкретні дії використовується бібліотека pynput. Вона взаємодіє з системним API [9]. Керування гучністю реалізоване через бібліотеку pycaw, яка використовує Windows CoreAudio API [10], що дозволяє програмно змінювати рівень звуку без затримок.

Розробка програмного забезпечення для безконтактної взаємодії з комп'ютером розпочалася з вибору інструментів. Як середовище розробки обрано VisualStudioCode, що забезпечило зручність написання, тестування та налагодження коду. Базою став інтерпретатор Python 3.8.5, який був інстальований і налаштований на етапі підготовки. Додатково встановлено необхідні залежності, включаючи бібліотеки для роботи з відео, обробки зображень та емуляції дій миші.

Архітектура системи реалізована за модульним принципом, що дозволяє розділити функціонал на незалежні компоненти. Наприклад, один з компонентів відповідає за виявлення рук, ідентифікацію ключових точок та розрахунок відстаней між пальцями за допомогою алгоритмів MediaPipe, тоді як другий перетворює координати рук у команди миші, а третій дозволяє налаштувати параметри звуку і регулювання гучності системи. Таким чином, модульність системи дозволяє легко додавати нові функції та жести для масштабування керування пристроями, без змін у базовому коді.

Основним компонентом системи є файл main.py, який ініціалізує роботу всіх модулів. Для роботи з відео потоком імпортовано бібліотеку OpenCV, що забезпечує захоплення кадрів з веб-камери та їх подальшу обробку. У функції main() реалізовано безперервний цикл, який аналізує кожен кадр: виконує корекцію кольорового балансу, зміну розміру зі збереженням пропорцій та додавання елементів інтерфейсу, таких як прямокутна рамка для зони керування, шкалу гучності та індикатор частоти кадрів (FPS). Код реалізації даного циклу продемонстровано в лістингу 1.

Лістинг 1 – Реалізація кадру

```
whileTrue:
        img, frameReduction = screen_settings.get_frame()
        if img is None:
            break
        screen_settings.enforce_aspect_ratio()
        img = detector.findHands(img)
        lmList = detector.findPosition(img, draw=False)
        if len(lmList) != 0:
            fingers = detector.fingersUp()
            cv2.rectangle(img, (frameReduction, frameReduction),
```

```
                        (screen_settings.wCam - frameReduction,
screen_settings.hCam - frameReduction),
(255, 0, 255), 2)
img = volume_control.draw_volume_interface(img, window_height)
img = screen_settings.draw_fps(img)
```

кінець лістингу 1

Використовуючи бібліотеку MediaPipe, створено модуль, що відповідає за розпізнавання рук. Клас HandDetector() включає три ключові методи: find Hands() для попередньої обробки зображення та виділення контурів рук, find Position() для визначення координат кінчиків пальців і find Distance() для розрахунку відстаней між точками (ліст. 2). Ця логіка є основою для перетворення фізичних рухів у цифрові команди.

Лістинг 2 – Розпізнавання рук

```
classHandDetector:
def __init__(self, mode=False, maxHands=1, detectionCon=0.5,
 trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon
        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(
            static_image_mode=self.mode,
            max_num_hands=self.maxHands,
            min_detection_confidence=self.detectionCon,
            min_tracking_confidence=self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]
def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
 self.mpHands.HAND_CONNECTIONS)
        return img
    def findPosition(self, img, draw=False):
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[0]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 7, (255, 0, 255),
 cv2.FILLED)
        return self.lmList
    def findDistance(self, p1, p2, img, draw=True, r=7, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
```

```
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)
return length, img, [x1, y1, x2, y2, cx, cy]
```

кінець лістингу 2

Інтегровано модуль, який відповідає за конфігурацію вікна програми та адаптацію відеопотоку до різних умов відображення. Клас Screen Settings() використовує Windows API для отримання метрик екрана користувача, що дозволяє автоматично розміщувати вікно програми в конкретному місці екрана при запуску. Метод get_frame() забезпечує захоплення кадрів з веб-камери та їх адаптацію до поточних розмірів вікна. Під час зміни розмірів вручну система автоматично коригує пропорції зображення, зберігаючи співвідношення сторін 4:3. Це запобігає деформації відеопотоку та забезпечує стабільність розпізнавання жестів. Для візуалізації інтерфейсу додано індикатор FPS. Код з налаштування вікна програми показано в лістингу 3.

Лістинг 3 – Налаштування вікна

```
class ScreenSettings:
    def __init__(self):
        self.wCam = 640
        self.hCam = 480
        self.user32 = ctypes.windll.user32
        self.wScr = self.user32.GetSystemMetrics(0)
        self.hScr = self.user32.GetSystemMetrics(1)
        self.pTime = 0
        self.aspect_ratio = 4.0 / 3.0
        self.cap = cv2.VideoCapture(0)
        self.cap.set(3, self.wCam)
     self.cap.set(4, self.hCam)
        self._initialize_window()
    def _initialize_window(self):
        cv2.namedWindow(self.window_name, cv2.WINDOW_NORMAL)
        cv2.resizeWindow(self.window_name, 400, 300)
    def position_window(self):
        if self.first_display:
            x_position = self.wScr - 400
            taskbar_height = GetSystemMetrics(SM_CYSCREEN) -
 GetSystemMetrics(SM_CYFULLSCREEN)
            y_position = self.hScr - 300 - taskbar_height
            cv2.moveWindow(self.window_name, x_position, y_position)
            self.first_display = False
    def get_frame(self):
        success, img = self.cap.read()
        if not success:
            return None, None
        window_rect = cv2.getWindowImageRect(self.window_name)
        window_width = window_rect[2]
        window_height = window_rect[3]
        self.wCam = window_width
        self.hCam = window_height
        frameReduction = int(window_width * 0.16)
        if window_width > 0 and window_height > 0:
            img = cv2.resize(img, (window_width, window_height))
        return img, frameReduction
    def draw_fps(self, img):
        font_scale = self.hCam / 480.0
```

```
        cTime = time.time()
        fps = 1 / (cTime - self.pTime)
        self.pTime = cTime
return img
```

кінець лістингу 3

Для модуля емуляції дій миші використано бібліотеку pynput. Метод handle_mouse_movement() перетворює координати руки, отримані з модуля розпізнавання рук, у позицію курсору на екрані, враховуючи реальні розміри дисплея для точного позиціонування. Рух вказівного пальця активізує переміщення курсору, тоді як одночасне підняття вказівного та середнього пальців ініціює лівий клік або перетягування. Додавання безіменного пальця до комбінації викликає правий клік, а положення великого пальця визначає напрямок прокрутки – вгору або вниз (ліст. 4). Для зменшення «дрижання» курсору застосовано алгоритми згладжування руху на основі лінійної інтерполяції. Це дозволило досягти плавності навіть при швидких рухах руки.

Лістинг 4 – Емуляція дій миші

```
class MouseControl:
    def __init__(self, screen_width, screen_height):
        self.mouse = Controller()
        self.smoothening = 7
        self.click_smoother = 0
    def    handle_mouse_movement(self,    x1,    y1,    frame_reduction,
window_width,
 window_height):
        x5 = np.interp(x1, (frame_reduction, window_width -
 frame_reduction), (0, self.wScr))
        y5 = np.interp(y1, (frame_reduction, window_height -
 frame_reduction), (0, self.hScr))
        self.cLocX = self.pLocX + (x5 - self.pLocX) / self.smoothening
        self.cLocY = self.pLocY + (y5 - self.pLocY) / self.smoothening
        self.mouse.position = (self.wScr - self.cLocX, self.cLocY)
        self.pLocX, self.pLocY = self.cLocX, self.cLocY
        if self.click_smoother >= 30:
            self.RMB_pressed = False
            self.LMB_pressed = False
        self.click_smoother += 1
        return (x1, y1)
def _handle_left_click_and_drag(self, img, length_im, line_info_im,
frame_reduction, window_width,window_height):
        self.RMB_pressed = False
        if not self.dragging and length_im < 25:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 255, 0), cv2.FILLED)
            self.mouse.press(Button.left)
            self.dragging = True
            self.LMB_pressed = True
        elif self.dragging:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 0, 255), cv2.FILLED)
    def handle_scrolling(self, thumb_up):
        if thumb_up:
    self.mouse.scroll(0, -0.5)
        else:
    self.mouse.scroll(0, 0.5)
    def _handle_right_click(self, img, line_info_im, line_info_mr):
```

```
        if self.dragging:
            self.mouse.release(Button.left)
            self.dragging = False
            self.LMB_pressed = False
        if not self.RMB_pressed:
            cv2.circle(img, (line_info_im[4], line_info_im[5]),
                       7, (0, 255, 255), cv2.FILLED)
            cv2.circle(img, (line_info_mr[4], line_info_mr[5]),
                       7, (0, 255, 255), cv2.FILLED)
            self.mouse.click(Button.right, 1)
            self.RMB_pressed = True
            self.LMB_pressed = False
return img
```

кінець лістингу 4

Регулювання гучності реалізовано в окремому модулі через інтеграцію бібліотеки pycaw, яка взаємодіє з аудіосистемою Windows. Метод handle_volume_gesture() аналізує вертикальні рухи руки: підняття долоні збільшує гучність, а опускання – зменшує. Візуальна шкала гучності, створена за допомогою OpenCV, динамічно адаптується до розмірів вікна програми, відображаючи поточний рівень звуку у відсотках (ліст. 5).

Лістинг 5 – Регулювання гучності

```
classVolumeControl:
def __init__(self):
        devices = AudioUtilities.GetSpeakers()
        interface = devices.Activate(IAudioEndpointVolume._iid_,
 CLSCTX_ALL, None)
        self.volume = cast(interface, POINTER(IAudioEndpointVolume))
        self.volRange = self.volume.GetVolumeRange()
        self.minVol, self.maxVol = self.volRange[0], self.volRange[1]
        self.current_volume = self.volume.GetMasterVolumeLevelScalar()
        self.volume_bar  =  int(np.interp(self.current_volume,  [0,  1],
[400,
 150]))
        self.volume_percentage = int(self.current_volume * 100)
    def handle_volume_gesture(self, lmList):
        fingertips = [lmList[tip] for tip in [8, 12, 16, 20]]
        avg_y = sum(tip[2] for tip in fingertips) / len(fingertips)
        vol = np.interp(avg_y, [80, 150], [self.maxVol, self.minVol])
        current_volume = np.interp(vol, [self.minVol, self.maxVol], [0,
 1])
        self.volume.SetMasterVolumeLevelScalar(current_volume, None)
        self.volume_bar = np.interp(avg_y, [80, 150], [150, 400])
        self.volume_percentage = int(np.interp(avg_y, [80, 150], [100,
 0]))
    def draw_volume_interface(self, img, window_height):
        height, width = img.shape[:2]
        volume_x = int(width * 0.1)
        volume_top = int(height * 0.3)
        volume_bottom = int(height * 0.8)
        volume_width = int(width * 0.05)
        cv2.rectangle(img, (volume_x, volume_top),
                      (volume_x + volume_width, volume_bottom),
                      (0, 255, 0), 3)
        current_volume_height  =  int(np.interp(self.volume_bar,  [150,
400],
```

```
                                                [volume_top,
volume_bottom]))
        cv2.rectangle(img, (volume_x, current_volume_height),
                      (volume_x + volume_width, volume_bottom),
                      (0, 255, 0), cv2.FILLED)
return img
```

кінець лістингу 5

Після завершення розробки програмного забезпечення для керування комп'ютером жестами, ініційовано комплексне тестування, метою якого стала перевірка відповідності системи встановленим вимогам. Тестування охопило послідовну перевірку кожного жесту, щоб забезпечити коректність виконання всіх команд та мінімізувати ризик неправильної інтерпретації рухів.

У разі відсутності критичних помилок у коді, програма успішно запускалася, активуючи вебкамеру та відображаючи інтерфейсі елементи: індикатор FPS у реальному часі та шкалу гучності з числовим відображенням рівня звуку у відсотках. Перший етап тестування зосередився на перевірці відображення рамки зони керування, яка мала адаптуватися до змін розмірів вікна програми. Рамка з'являлася лише тоді, коли користувач розміщував руку в заданій області, і зникала при її видаленні, що підтвердило коректність роботи алгоритмів відстеження (рис. 1).

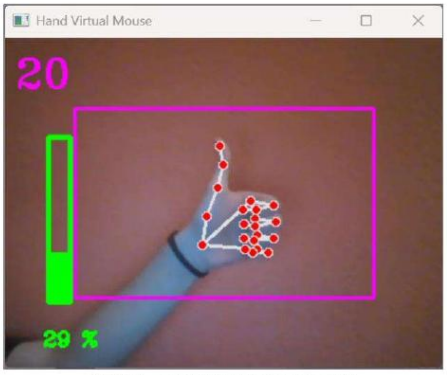


Рис.1 – Інтерфейс програми

Наступним кроком стала перевірка базових функцій. Підняття вказівного пальця та його переміщення в межах рамки активувало рух курсору миші по екрану (рис. 2).

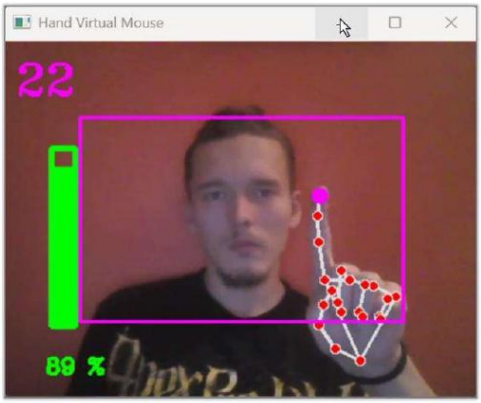


Рис.2 – Рух курсору

Для виконання складніших дій, таких як перетягування файлів або виділення об'єктів, потрібно було підняти вказівний і середній пальці одночасно (рис.3). Зближення цих пальців ініціювало лівий клік, тоді як їхнє розведення на більшу відстань викликало подвійне клацання.

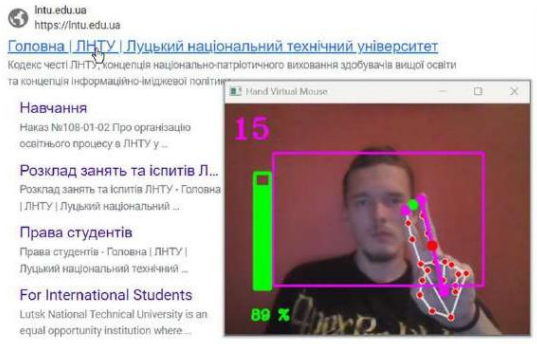


Рис.3 – Лівий клік миші

Додавання безіменного пальця до комбінації забезпечувало правий клік (рис.4), що розширювало функціональність системи.

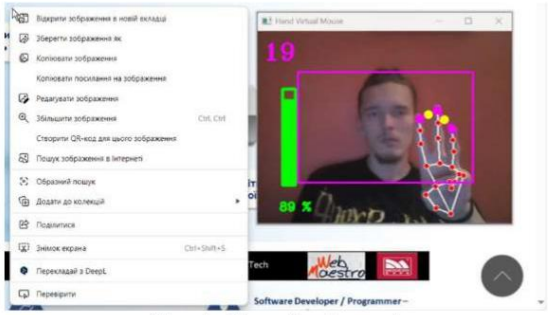


Рис. 4 – Правий клік миші

Для прокручування сторінок вгору або вниз користувач мав закрити всі пальці, крім великого, рух якого визначав напрямок (рис.5).

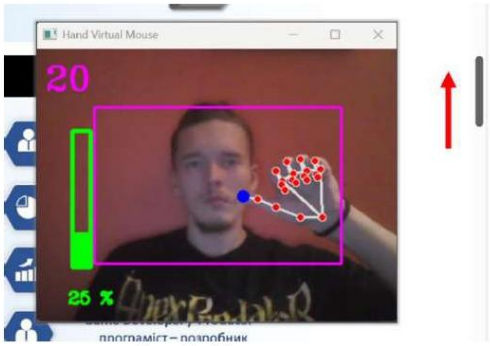


Рис. 5 – Прокручування

Вертикальні рухи руки з відкритими пальцями, крім великого, дозволяли змінювати гучність: підняття руки збільшувало рівень звуку, опускання – зменшувало. Ці зміни візуалізувалися на шкалі та підтверджувалися відображенням на панелі завдань Windows (рис.6).

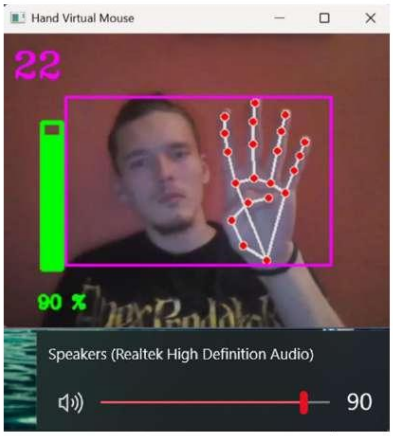


Рис.6 – Регулювання гучності

Окремим результатом тестування стала підтвердження універсальності системи: вона коректно працювала як з лівою, так і з правою рукою, надаючи користувачам свободу вибору. Це забезпечило інтуїтивність взаємодії та зручність у різних сценаріях використання.

Для оцінки продуктивності розробленого програмного забезпечення було проведено серію тестів за участю 8 користувачів. Важливо зазначити, що код програми залишався незмінним на всіх етапах, що дозволило виключити сторонні фактори впливу на результати. Тестування охопило різні апаратні конфігурації: стаціонарні комп'ютери, ноутбуки, вбудовані та зовнішні камери з різною роздільною здатністю. Додатково враховано відстань від веб-камери та умови освітлення, щоб імітувати реальні сценарії використання.

Кожен учасник виконував 10 спроб для кожного жесту, що дозволило отримати середні значення для трьох показників: точності розпізнавання, часу реакції системи та стабільності виконання команд. Точність вимірювалася як відсоток успішних спроб, коли система коректно інтерпретувала жест. Час реакції фіксувався від моменту виконання жесту до ініціювання відповідної дії на комп'ютері. Стабільність оцінювалася за 10-бальною шкалою, де користувачі відзначали, наскільки послідовно система реагувала на однакові рухи.

Програмне забезпечення продемонструвало середню точність понад 95% для базових операцій, таких як переміщення курсору та регулювання гучності, тоді як подвійне клацання, яке вимагає синхронного зведення та розведення пальців, показало меншу точність. Час реакції системи коливався в межах 36–54 мс, що відповідає вимогам роботи в реальному часі.

Стабільність жесту оцінювалася через консистентність результатів у різних умовах. Відповідно як і до точності, переміщення курсору та регулювання гучності отримали найкращі результати, тоді як подвійне клацання показало себе гірше. Низька стабільність останнього пояснюється чутливістю до швидкості руху пальців та схожістю жесту з простим клацанням, що іноді призводило до помилкової інтерпретації. Отримані результати всіх показників було узагальнено та зведено в таблицю 1.

Таблиця 1 – Результати показників аналізу функціонування жестів

Жест	**Критерії**		
	Точність (%)	**Реакція (мс)**	**Стабільність (від 1 до 10)**
Переміщення курсору миші	98	42	9,6

Лівий клік	96	39	8,5
Подвійний лівий клік	90	50	7,4
Правий клік	94	42	8,6
Перетягування	96	45	8,8
Скролінг	97	41	8,4
Регулювання гучності	98	42	9,3

Обґрунтування результатів також підтверджується порівнянням з аналогічними рішеннями. Наприклад, система на основі LeapMotion демонструє точність 98% [11], але вимагає спеціалізованого обладнання, тоді як запропонований підхід забезпечує 95% при використанні звичайної веб-камери.

Отримані дані підтверджують, що система ефективно виконує основні функції, але потребує оптимізації для складніших жестів. Наприклад, подвійне клацання можна покращити шляхом впровадження алгоритмів, які аналізують не лише відстань між пальцями, а й динаміку руху. Також варто додати можливість калібрування чутливості під індивідуальні особливості користувачів.

Процес тестування програми виявив ключові напрями для вдосконалення надалі, зокрема, оптимізацію роботи зі складними командами, такими як подвійний лівий клік. Для вирішення цієї проблеми пропонується переглянути логіку обробки цієї команди: замінити жест на більш інтуїтивний або впровадити додаткові алгоритми, що аналізують тривалість та траєкторію руху. Також варто враховувати зовнішні фактори, такі як якість камери та освітлення, розробивши адаптивні механізми, які автоматично коригують параметри розпізнавання.

Майбутні оновлення передбачають розширення функціоналу шляхом додавання нових жестів для керування медіа (пауза, перемикання, масштабування) та системних налаштувань (наприклад, яскравість). Модульна архітектура програми дозволяє легко інтегрувати ці можливості через створення окремих компонентів. Крім того, планується впровадження інструментів, таких як голосовий помічник та віртуальна клавіатура, що зробить систему ще більш універсальною.

Для підвищення безпеки та зручності розробляється механізм активації жестового керування через спеціальну комбінацію рухів, що запобігає випадковим діям. Підтримка розпізнавання двох рук одночасно відкриє можливість виконання складних команд, наприклад, одночасного перетягування об'єктів і регулювання параметрів. Інтерфейс програми також потребує розвитку: планується додати інтерактивний гід із жестами, контекстні підказки та персоналізовані налаштування, доступні через панель меню. Це спростить освоєння системи для новачків і зробить взаємодію більш ефективною. Важливим рішенням стане розширення підтримки на операційні системи Linux та macOS, що збільшить аудиторію користувачів. Для цього буде використано кросплатформні бібліотеки та адаптовано існуючі модулі під специфіку цих ОС.

Таким чином, система безконтактного керування жестами має потенціал для перетворення на універсальний інструмент, який поєднує високу точність, зручність та доступність. Реалізація цих ініціатив не лише покращить поточні функції, але й відкриє нові сценарії використання – від побутових задач до професійних середовищ, де безконтактне керування є більш доцільним, аніж використання фізичних пристроїв введення.

Висновки. Розроблено програмний продукт для безконтактного керування комп'ютером за допомогою жестів демонструє високу ефективність у реалізації базових функцій миші, таких як переміщення курсору, виконання натискань, прокрутка та регулювання гучності. Використання технологій комп'ютерного зору OpenCV та відстеження рухів MediaPipe дозволило створити рішення, що працює в режимі реального часу з частотою обробки кадрів до 30 кадрів у секунду, забезпечуючи достатньо плавну взаємодію. Модульна архітектура системи забезпечує гнучкість у розширенні функціоналу, наприклад, додаванні нових жестів або інтеграцію додаткових дій та інструментів, що робить її адаптивною до різних сценаріїв використання.

Експериментальні тести за участю користувачів підтвердили точність розпізнавання жестів на рівні 95% для базових операцій, таких як переміщення курсору та регулювання гучності. Однак складніші дії, зокрема подвійний клік, потребують подальшого вдосконалення алгоритмів для підвищення стабільності. Важливим досягненням є сумісність системи зі звичайними веб-камерами, що значно знижує вартість рішення та робить його доступним для широкого застосування.

Перспективи розвитку системи надалі включають впровадження методів глибокого навчання для підвищення точності розпізнавання складних і комбінованих жестів. Важливим кроком є розширення підтримки на різних операційних системах, що дозволить залучити ширшу аудиторію користувачів. Оптимізація алгоритмів для роботи в умовах нерівномірного освітлення або при частковому перекритті рук підвищить стабільність системи у реальних сценаріях, де ідеальні умови рідко досяжні.

Практична цінність розробки полягає в її потенціалі для використання в промисловості, медицині, віртуальній реальності та для користувачів з обмеженими руховими можливостями. Наприклад, в умовах пандемій система може зменшити ризик передачі інфекцій через мінімізацію контакту з поверхнями. Таким чином, система керування комп'ютером за допомогою жестів є перспективним кроком у напрямку створення інтуїтивних інтерфейсів людино-машинної взаємодії, що відкриває нові можливості для застосування у різних галузях, від побутового використання до критично важливих операцій.

Список бібліографічного опису

1. Mr.E.Sankar, B.Nitish Bharadwaj, A.V.Vignesh.Virtual Mouse Using Hand Gesture. International Journal of Scientific Research in Engineering and Management. Vol. 7. No. 5. May 2023. P. 2-4.
2. P. J. Ezigbo, O. C. Nosiri, E. S. Mbonu, V. Ofor, J. Obichere. Hand Gesture Recognition and Control for Human-Robot Interaction Using Deep Learning. International Journal of Electrical and Electronic Engineering&Telecommunications.Vol. 12.No. 6.November 2023.P. 433-441.
3. Міронов Н. О., Самчук Л. М. Дослідження характеристик та медотології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій ComputerVision. Комп'ютерно-інтегровані технології: освіта, наука, виробництво, Луцьк, 2024. № 57. С. 115-118.
4. S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, P. Ajay. Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. Journal of Healthcare Engineering. 2021. P. 4-7.
5. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex&IntelligentSystems. 2020. P. 7-12.
6. The Python Community. URL:https://www.python.org/community/
7.MediaPipe Solutions guide. URL: https://chuoling.github.io/mediapipe/
8.Optical Flow in OpenCV. URL: https://learnopencv.com/optical-flow-in-opencv/
9.Pynputpackage. URL: https://pynput.readthedocs.io/en/latest/index.html
10.Pycawpackage. URL: https://pycaw.readthedocs.io/en/latest/index.html
11. PruthaAtre, SahilBhagat, Nevil Pooniwala, PayalShah. Efficient and Feasible Gesture Controlled Robotic Arm. 2019.P. 34-54.

References

1. Mr.E.Sankar, B.NitishBharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Research in Engineering and Management. Vol. 7. No. 5. May 2023. P. 2-4.
2. P. J. Ezigbo, O. C. Nosiri, E. S. Mbonu, V. Ofor, J. Obichere. Hand Gesture Recognition and Control for Human-Robot Interaction Using Deep Learning. International Journal of Electrical and Electronic Engineering&Telecommunications.Vol. 12.No. 6.November 2023.P. 433-441.
3. Mironov N.O., Samchuk L.M. Study of characteristics and methodology of gesture recognition system for contact less interaction with a computer using Computer Vision technologies: education, science, production, Lutsk, 2024. No 57. P. 115-118.
4. S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, P. Ajay. Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. Journal of Healthcare Engineering. 2021. P. 4-7.
5. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex&IntelligentSystems. 2020. P. 7-12.
6. The Python Community. URL:https://www.python.org/community/
7.MediaPipe Solutions guide. URL: https://chuoling.github.io/mediapipe/
8.Optical Flow in Open CV. URL: https://learnopencv.com/optical-flow-in-opencv/
9.Pynputpackage. URL: https://pynput.readthedocs.io/en/latest/index.html
10.Pycawpackage. URL: https://pycaw.readthedocs.io/en/latest/index.html
11. PruthaAtre, SahilBhagat, Nevil Pooniwala, PayalShah. Efficient and Feasible Gesture Controlled Robotic Arm. 2019.P. 34-54.

Додаток В

Участь у міжнародній науково-практичній конференції «Цифрова трансформація: виклики та стратегії»

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
Lutsk National Technical University (Ukraine), National Technical University "Kharkiv Polytechnic Institute" (Ukraine), National Technical University "Dnipro Polytechnic" (Ukraine), Lublin University of Technology (Poland), Ahlia University (Bahrain), Polytechnic Institute of Braganca (Portugal), Vytautas Magnus University (Lithuania), Marie Curie-Sklodowska University (Poland), OWL University of Applied Sciences and Arts (Germany), Batumi Navigation Teaching University (Georgia), Dunarea de Jos University of Galati (Romania), Western Science Center (Ukraine), Akkon University of Human Sciences (Germany)

МАТЕРІАЛИ
Міжнародної науково-практичної конференції
«Цифрова трансформація: виклики та стратегії»
https://dtcs.lntu.edu.ua/

м. Луцьк, Україна
25 лютого 2025 року

MATERIALS of the
International scientific and practical conference
"Digital Transformation: Challenges and Strategies"
https://dtcs.lntu.edu.ua/

Lutsk, Ukraine
February 25, 2025

Список використаних джерел:

1. Communication Today.URL:https://communicationtoday.net/2014/03/31/internet-as-a-medium-of-communication-in-modern-society (Last accessed: 22.05.2024).
2. 4 ways the web has changed our lives – and will shape our future. URL: https://surl.li/otlacg (Last accessed: 22.05.2024).
3. UML Use Case Diagram Example. Social Networking Sites Project. URL: https://surl.li/zibaxl (Lastaccessed: 22.05.2024).
4. The Expressive Power of UML-basedWebEngineering: chrome-extension. URL: https://surl.li/jmhgyn (Last accessed: 22.05.2024).
5. Unified Modeling Language.URL:https://surl.li/xrwskf (Last accessed:22.05.2024).

НікітаМІРОНОВ
здобувач вищої освіти факультету комп'ютерних та інформаційних технологій
Луцький національний технічний університет, Україна
Людмила САМЧУК
к.т.н., доцент
Луцький національний технічний університет, Україна

ТЕНДЕНЦІЇ ЗАСТОСУВАННЯ ІНСТРУМЕНТІВ БЕЗКОНТАКТНОЇ ВЗАЄМОДІЇ З ПРИСТРОЯМИ ЗА ДОПОМОГОЮ ЖЕСТІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ КОМП'ЮТЕРНОГО ЗОРУ І МАШИННОГО НАВЧАННЯ

Традиційні методи введення даних, такі як клавіатури та миші, мають обмеження у випадках, коли фізичний контакт з пристроєм є небажаним або неможливим. Це актуально в умовах гігієнічних обмежень або в складних виробничих середовищах. У сфері медицини, зокрема, під час хірургічних операцій або в стерильних зонах [1, с. 1254], використання жестового управління може значно підвищити ефективність роботи персоналу.

Розвиток технологій комп'ютерного зору відкриває нові можливості для створення адаптивних інтерфейсів, які можуть навчатися та підлаштовуватися під особливості користувачів. Впровадження алгоритмів машинного навчання дозволяє підвищити точність розпізнавання жестів навіть у несприятливих умовах, таких як слабке освітлення. У зв'язку з цим виникає необхідність у розробці ефективних і доступних систем, які забезпечуватимуть високу точність розпізнавання жестів у реальному часі. Наукові праці розглядають можливості та перспективи застосування розпізнавання жестів за допомогою комп'ютерного зору. Багато рішень використовують OpenCV для аналізу відеопотоку та відстеження ключових точок рук. Наприклад, у роботі [2, с. 113] досліджено методику розпізнавання жестів на основі поєднання OpenCV та MediaPipe, що дозволяє досягати високої точності навіть у реальному часі. Окрім цього, застосування нейронних мереж (CNN) значно підвищує точність розпізнавання складних жестів. У дослідженні [3, с. 3]описано ефективність використання CNN для аналізу відеопотоку та ідентифікації жестів, що дає змогу адаптувати систему до різних користувачів і умов зовнішнього середовища.

Використання навчених моделей, таких як MediaPipeHands [4], дозволяє отримати стабільні результати навіть у несприятливих умовах, таких як змінне освітлення або перешкоди перед камерою. У роботі [5, с. 436] представлено систему, яка поєднує аналіз відеопотоку з алгоритмами навчання, що дозволяє значно знизити рівень помилкових спрацьовувань та підвищити точність розпізнавання жестів. Інструментом для написання програмного забезпечення, що дозволяє розпізнавати жести, став Python [6], завдяки своїй гнучкості та широкому вибору бібліотек для машинного навчання та комп'ютерного зору. В основі лежить MediaPipe, який використовує попередньо навчені моделі машинного навчання, що дозволяєідентифікувати 21 точку на руці. Процес розпізнаваннявключає сегментацію зображення, ізоляцію руки від фону та прогнозування позиції пальців руки за

допомогою нейронних мереж [7]. Наприклад, при згинанні пальця алгоритм розпізнає зміну кута між фалангами та передає відповідну команду. Ще одна фундаментальна бібліотека в системі, OpenCV [8], використовується для обробки зображення, забезпечуючи фільтрацію шумів, корекцію кольорового балансу та попередню обробку відео. Інструмент, який застосовує алгоритми технологій комп'ютерного зору, дозволяє відстежувати динамічні рухи руки та їхню зміну в просторі.

Після того, як розроблена програма виявляє руку, вона аналізує положення пальців для визначення конкретних жестів. Наприклад, для керування курсором миші потрібно лише підняти вказівний палець і рухати ним по площині зони керування жестами (рис. 1).Можна поєднувати команду з прокручуванням вмісту сторінок, використовуючи великий палець.

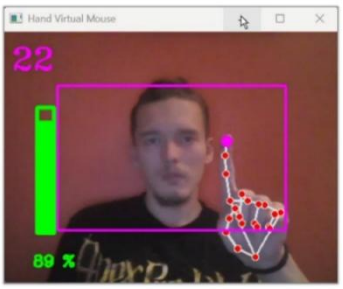


Рисунок 1 – Курсор

Виконувати клацання кнопками миші можна, використовуючи комбінації з кількох пальців. Наприклад, для виконання лівого кліку (рис. 2), використовується жест розмикання вказівного та середнього пальців. Для правого клацання до додається безіменний палець.

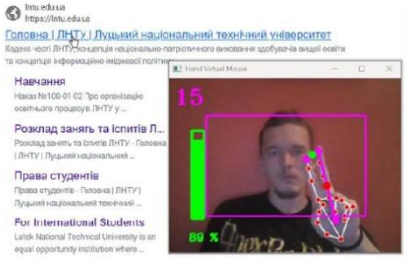


Рисунок 2 – Ліва кнопка

Якщо пальці розтягнуті і долоня рухається по вертикалі, це сприймається як команда для зміни гучності звуку (рис. 2).

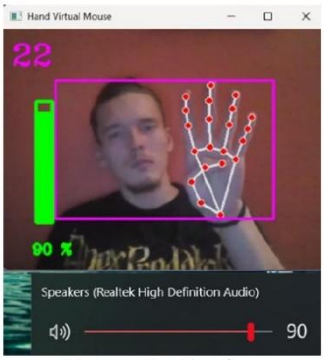


Рисунок 3 – Гучність

Для перевірки ефективності системи проведено тестування за участю користувачів, які виконували серію жестів для керування комп'ютером. Кожен учасник виконував декілька спроб кожної команди, а результати оцінено за трьома критеріями: точність розпізнавання, швидкість реакції та стабільність виконання команд. Результати тестів узагальнено і зведено у відповідну таблицю (табл. 1).

Таблиця 1 – Показники тестування функціонування команд

Жест	Критерії		
	Точність (%)	Реакція (мс)	Стабільність (від 1 до 10)
Переміщення курсору	98	42	9,6
Лівий клік	96	39	8,5
Подвійний клік	90	50	7,4
Правий клік	94	42	8,6
Перетягування	96	45	8,8
Прокручування	97	41	8,4
Зміна гучності	98	42	9,3

Експериментальні тести показали, що використання методів глибокого навчання для розпізнавання жестів підвищує точність до 95%, проте складніші дії, такі як подвійний клік, потребують подальшого вдосконалення алгоритмів. Для підвищення стабільності роботи системи необхідно враховувати зовнішні фактори, включаючи якість камери та освітлення. Розвиток технологій безконтактної взаємодії з пристроями за допомогою жестів є перспективним напрямом у сфері людино-машинної взаємодії. Надалі дослідження мають бути спрямовані на зменшення затримок у роботі системи, розширення спектра доступних жестів та адаптацію алгоритмів донових сценаріїв використання. Перспективним напрямом є також впровадження мультимодального управління, яке поєднує жести, голосові команди та зворотний зв'язок, що дозволить підвищити рівень інтерактивності та точності взаємодії.У майбутньому інтеграція таких рішень у смарт-системи та робототехніку відкриє нові можливості для розвитку інтуїтивних інтерфейсів, зокрема у сфері дистанційного управління пристроями, автоматизованих систем моніторингу

Список використаних джерел:

1. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smart healthcare solutions using the internet of medical things for hand gesture recognition system. Complex&Intelligent Systems. 7. 2021, 1253–1264.URL:https://doi.org/10.1007/s40747-020-00194-9

2.Міронов Н. О., Самчук Л. М. Дослідження характеристик та медотології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій ComputerVision. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 57. 2024, 111-119.URL:https://doi.org/10.36910/6775-2524-0560-2024-57-13

3.Mr.E.Sankar, B.NitishBharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Researchin Engineering and Management. 7, 5. 2023, 2-4. URL:https://www.researchgate.net/publication/372165002

4.MediaPipe HandsDocs.URL: https://chuoling.github.io/mediapipe/solutions/hands.html

5. P. J. Ezigbo, O. C. Nosiri, E. S. Mbonu, V. Ofor, J. Obichere. Hand Gesture Recognition and Control for Human-Robot Interaction Using Deep Learning. International Journal of Electrical and Electronic Engineering&Telecommunications. 12,6. 2023, 433-441. URL: https://www.researchgate.net/publication/375422572

6.Python Docs. URL: https://www.python.org/doc/

7.Gesturerecognition. Overview. Gesturere cognition task guide for MediaPipe. URL: https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer

8.OpenCVDocs. URL: https://pypi.org/project/opencv-python/

Владислав Терещук,
магістр
Світлана Лавренчук,
к.т.н., доцент
Луцький національний технічний університет, Україна

РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЯ ЖЕСТІВ ДЛЯ УПРАВЛІННЯ ПРИСТРОЯМИ

Розпізнавання жестів є важливим напрямком у сфері взаємодії людини з пристроями. Завдяки розвитку алгоритмів штучного інтелекту та сенсорних технологій сучасні системи жестового управління демонструють високу точність і швидкість розпізнавання. У роботі розглядаються два підходи до розпізнавання жестів: комп'ютерний зір із використанням глибоких нейронних мереж та сенсорні системи з ємнісними датчиками. Аналізуються їхні переваги, виклики та можливості застосування.

Методи розпізнавання жестів базуються на двох основних підходах. Один із них використовує комп'ютерний зір та нейронні мережі, що дозволяє одночасно виконувати кілька завдань, таких як розпізнавання статичних і динамічних жестів, локалізація об'єктів та опис сцени. Ефективність цього методу досягається завдяки розподілу обчислювальних ресурсів між різними компонентами нейромережі. На рисунку 1 зображено жести, які використовуються для взаємодії з пристроями. Основними з них є «вказування», «лупа» та «щипок», які застосовуються для роботи з віртуальними об'єктами, зміни масштабу зображень або отримання додаткової інформації про об'єкт. Вказування, перетягування, лупа та підняття великого пальця є статичними жестами, тоді як щипання та помахування є динамічними.

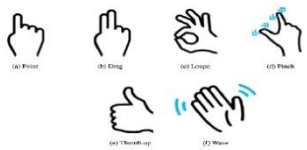


Рисунок 1. Графічний опис запропонованих жестів руками

Додаток Г

Участь у X міжнародній науково-практичній конференції «Інформаційні технології в освіті, науці і виробництві»

Експериментальне тестування сервісу показало стабільну роботу навіть з великою кількістю елементів схеми. Це підтверджує доцільність вибраних технологій та ефективність реалізованих алгоритмів оптимізації рендерингу.

Отже, створений сервіс є корисним інструментом як для початківців, так і для досвідчених розробників, викладачів та студентів, які працюють з базами даних. У подальшому можливе розширення функціоналу шляхом додавання спільного редагування, інтеграції з хмарними сервісами та підтримки реверсивного інжинірингу існуючих баз даних.

Список використаних джерел

1. Elmasri R., Navathe S. B. (2017). Fundamentals of Database Systems (7th ed.). Pearson Education.

2. React Flow Documentation URL: https://reactflow.dev/ (дата звернення: 28.04.2025).

3. TypeScript Documentation. URL: https://www.typescriptlang.org/ (дата звернення: 28.04.2025).

УДК 004.5

ЗАСТОСУНОК ДЛЯ БЕЗКОНТАКТНОЇ ВЗАЄМОДІЇ З ПРИСТРОЯМИ ЗА ДОПОМОГОЮ ЖЕСТІВ

Міронов Нікіта Олександрович

Луцький національний технічний університет, здобувач вищої освіти факультету комп'ютерних та інформаційних технологій, mironov.n3004@lntu.edu.ua

Самчук Людмила Михайлівна

Луцький національний технічний університет, кандидат технічних наук, доцент кафедри прикладної механіки та мехатроніки,Samchuk204@gmail.com

AN APPLICATION FOR CONTACTLESS INTERACTION WITH DEVICES USING GESTURES

Mironov Nikita Oleksandrovych

Lutsk National Technical University, higher education student of the Faculty of Computer and Information Technologies, mironov.n3004@lntu.edu.ua

Samchuk Liudmyla Mykhailivna

Lutsk National Technical University, PhD inTechnical Sciences, Associate Professor, Department of Applied Mechanics and Mechatronics Samchuk204@gmail.com

Trends in the use of tools for contactless interaction with devices through gestures using computer vision and machine learning technologies are considered. The use of the OpenCV and MediaPipe libraries for processing the video stream and detecting key points of the hands is analyzed.

***Keywords**: contactless interaction, gestures, computer vision, machine learning, gesture recognition, MediaPipe, OpenCV, human-machine interaction, neural networks, device control.*

Звичайні способи введення інформації, як-от клавіатури та миші, виявляються недостатніми в ситуаціях, де безпосередній контакт з обладнанням є неможливим. Це особливо важливо в умовах суворих гігієнічних вимог або в складних промислових умовах. У медичній галузі, зокрема під час хірургічних втручань або в стерильних приміщеннях [1], застосування управління за допомогою жестів може суттєво покращити продуктивність та якість роботи медичного персоналу.

Більшість рішень покладаються на комп'ютерний зір для обробки відеопотоку та виявлення основних точок на руках. У дослідженні [2] вивчено підхід до розпізнавання жестів, що базується на комбінації OpenCV і MediaPipe, який забезпечує високу точність у реальному часі. Крім того, використання згорткових нейронних мереж (CNN) істотно покращує точність виявлення складних жестів. У дослідженні [3] продемонстровано, наскільки ефективно CNN можуть аналізувати зображенняй розпізнавати жести, що дозволяє системі пристосовуватися до різних зовнішніх умов.

Застосування навчених моделей, як MediaPipeHands [4], забезпечує надійні результати навіть у складних умовах, таких як нестабільне освітлення та/або наявність перешкод перед камерою. У дослідженні [5] описано систему, яка інтегрує аналіз відеопотоку з алгоритмами машинного навчання, що значно зменшує кількість помилкових спрацьовувань. Для розробки застосунку, здатного розпізнавати жести, обрано мову програмування Python, завдяки її гнучкості та багатому набору бібліотек. Основою системи є MediaPipe, який використовує навчені моделі для виявлення 21 точки на руці. Процес розпізнавання складається з сегментації зображення, виділення руки з фону та передбачення положення пальців. Ще однією

бібліотекою в системі є OpenCV, яка відповідає за обробку зображень, забезпечуючи усунення шумів, корекцію кольорів та попередню обробку відеоданих. Після того, як програма виявляє руку, вона аналізує позиції пальців для ідентифікації конкретних жестів. Наприклад, щоб керувати курсором миші, достатньо підняти вказівний палець і переміщати його в межах зони управління жестами (рис. 1).

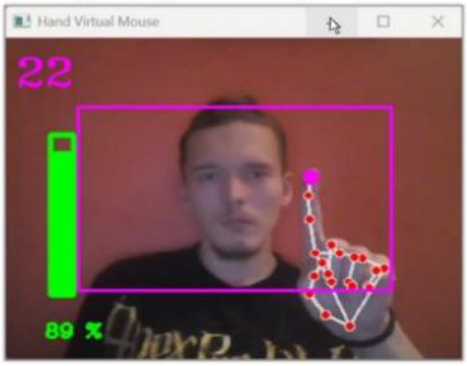


Рисунок 1 – Курсор

Для імітації натискань кнопок миші використовуються комбінації кількох пальців. Так, для лівого клацання (рис. 2) застосовується розведення вказівного та середнього пальців. Для правого кліку до них додається безіменний палець.

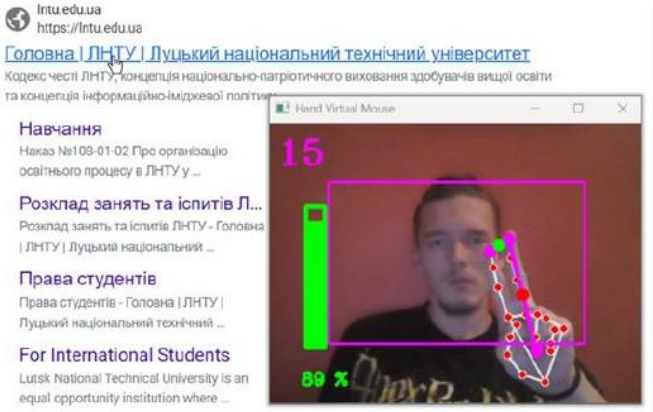


Рисунок 2 – Ліва кнопка

Якщо пальці витягнуті, а долоня рухається вертикально, це розпізнається як команда для регулювання гучності (рис. 3).

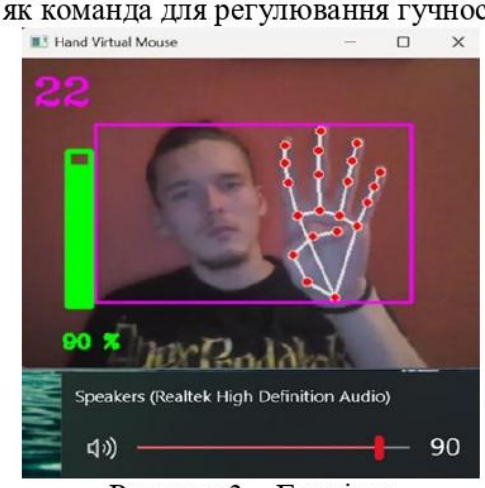


Рисунок 3 – Гучність

Розвиток методів безконтактної взаємодії з пристроями за допомогою жестів є багатообіцяючим напрямом у галузі людино-машинної взаємодії. Подальші дослідження зосереджені на скороченні затримок у роботі системи, розширенні набору розпізнаваних жестів та адаптації алгоритмів.

Ще одним напрямом є впровадження мультимодального управління, що поєднує жести, голосові команди та зворотний зв'язок, що сприятиме підвищенню інтерактивності та точності взаємодії. У майбутньому впровадження таких рішень створить нові можливості для розробки інтуїтивних інтерфейсів, зокрема для дистанційного керування пристроями та автоматизованих систем спостереження.

Список використаних джерел

1. Mahmoud, Nourelhoda, Fouad, Hassan, Soliman, Ahmed. Smarthealthcaresolutionsusingtheinternetofmedicalthingsforhandgesturerecognitionsystem. Complex&IntelligentSystems. 7. 2021, 1253–1264. URL:https://doi.org/10.1007/s40747-020-00194-9.

2. Міронов Н. О., Самчук Л. М. Дослідження характеристик та медотології системи розпізнавання жестів для безконтактної взаємодії з комп'ютером з використанням технологій ComputerVision. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 57. 2024, 111-119. URL:https://doi.org/10.36910/6775-2524-0560-2024-57-13.

3. Mr.E.Sankar, B.NitishBharadwaj, A.V.Vignesh. Virtual Mouse Using Hand Gesture. International Journal of Scientific Researchin Engineering and Management. 7, 5. 2023, 2-4. URL:https://www.researchgate.net/publication/372165002.

4. MediaPipe Hands.https://mediapipe/solutions/hands.html

5. P. J. Ezigbo, O. C. Nosiri, E. S. Mbonu, V. Ofor, J. Obichere. HandGestureRecognitionandControl for Human-RobotInteractionUsingDeepLearning. International Journal of Electrical and Electronic Engineering&Telecommunications. 12, 6. 2023, 433-441. URL: https://www.researchgate.net/publication/375422572

УДК 004.738.5:316.42

РОЛЬ ВЕБТЕХНОЛОГІЙ У РОЗВИТКУ ГРОМАДСЬКИХ ІНІЦІАТИВ

Надашкевич Анастасія Геннадіївна
Луцький національний технічний університет,
студентка 4 курсу, nastia.nadashkevich@gmail.com
Вознюк Анастасія Вадимівна
Луцький національний технічний університет, асистент кафедри інженерії програмного забезпечення, a.vozniuk@lutsk-ntu.com.ua

THE ROLE OF WEB TECHNOLOGIES IN THE DEVELOPMENT OF CIVIC INITIATIVES

Nadashkevych Anastasiia Hennadiivna
Lutsk National Technical University, 4th-year student,
nastia.nadashkevich@gmail.com
Vozniuk Anastasiia Vadymivna
Lutsk National Technical University, Assistant Professor, Department of Software Engineering, a.vozniuk@lutsk-ntu.com.ua

СЕРТИФІКАТ

Кафедра цифрових освітніх технологій

Кафедра інженерії програмного забезпечення

Даний сертифікат засвідчує, що

Міронов Нікіта Олександрович

брав(-ла) участь у

X Міжнародній науково-практичній конференції
з проблем вищої освіти і науки
"Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)"
загальним обсягом 15 годин (0,5 кредитів ECTS),

яка проводилася 23-24 травня 2025 року у м.Луцьк
на базі Луцького національного технічного університету

Ректор ЛНТУ _____ **Ірина ВАХОВИЧ**

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

№ ІТОНВ-2025-104

Програма конференції: https://itonv.lntu.edu.ua/files/2025/program-itonv-2025.pdf