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 функціймиші, 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 веб-камеру 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).

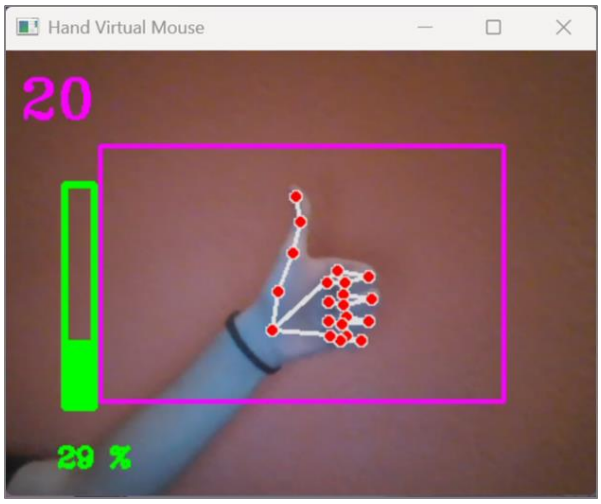

Рис.1 – Інтерфейс програми

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

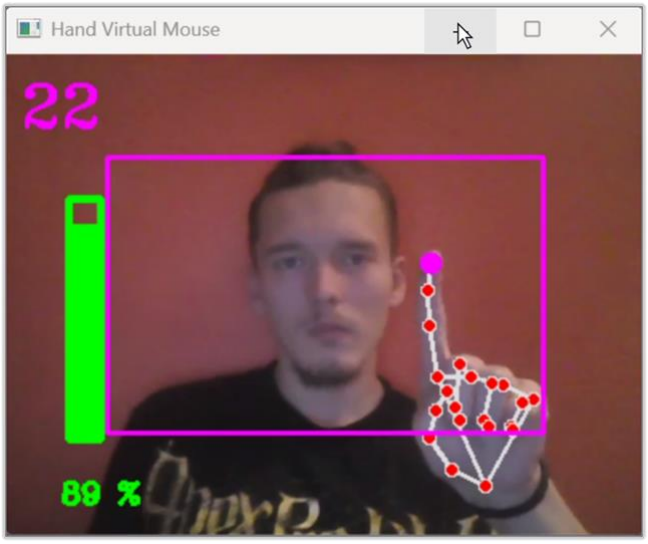

Рис.2 – Рух курсору

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.

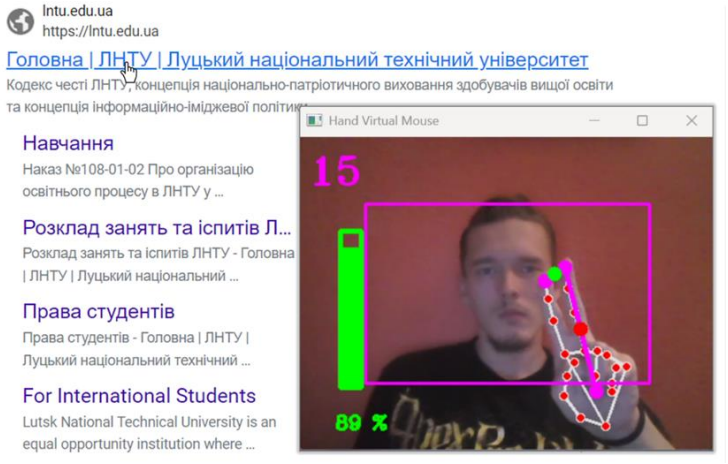

Рис.3 – Лівий клік миші

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

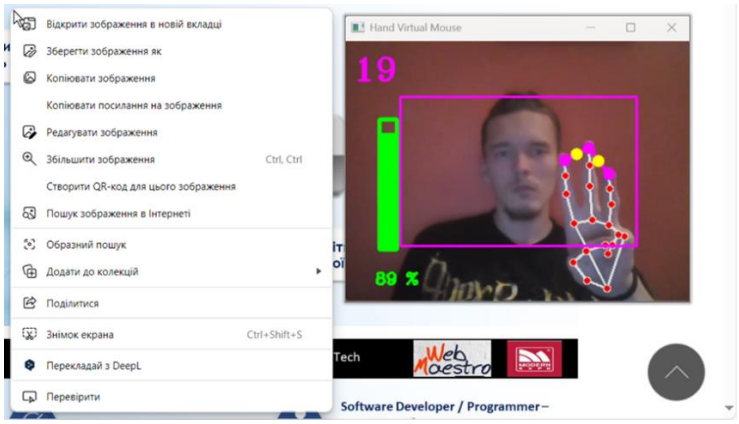

Рис. 4 – Правий клік миші

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

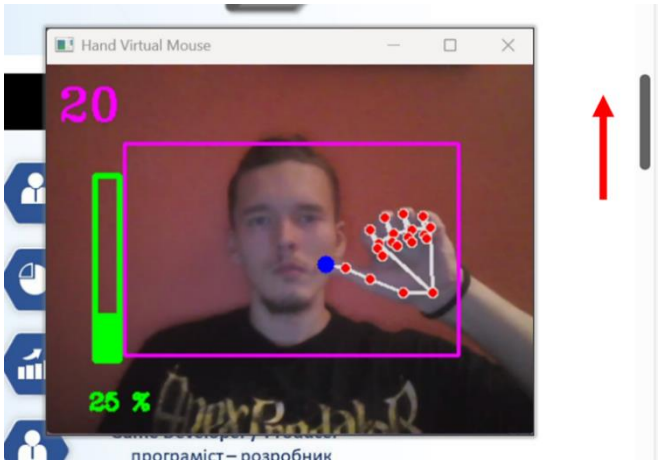

Рис. 5 – Прокручування

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).

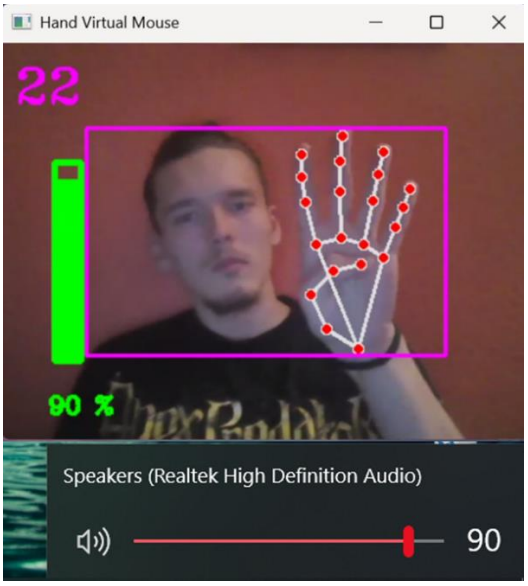

Рис.6 – Регулювання гучності

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.