ДОДАТОК А

Текст програми

Base.py:

```
import cv2
import mediapipe as mp
import numpy as np
import pickle
import os
import tkinter as tk
from tkinter import simpledialog, messagebox

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1, min_detection_confidence=0.7)
mp_draw = mp.solutions.drawing_utils


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

    def fit(self, coords, labels):
        self.coords = np.array(coords)
        unique_labels = sorted(set(labels))
```

```
        self.label_to_int = {label: i for i, label in enumerate(unique_labels)}
        self.int_to_label = {i: label for label, i in self.label_to_int.items()}
        self.labels = np.array([self.label_to_int[label] for label in labels])

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

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

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


knn = KNN(k=3)
data = {'coords': [], 'labels': []}

root = tk.Tk()
root.withdraw()
```

```
# Функція для додавання нового жесту (з кількома варіаціями)
def add_gesture(label, coords):
    data['coords'].extend(coords)
    data['labels'].extend([label] * len(coords))
    knn.fit(data['coords'], data['labels'])



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

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

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

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

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

            # Збираємо 21 ключову точку
```

```
                coords = []
                for lm in hand_landmarks.landmark:
                    coords.append(lm.x)
                    coords.append(lm.y)

                coords = np.array(coords)

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

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

            cv2.imshow("Hand Gesture Recognition", img)

            key = cv2.waitKey(1) & 0xFF

            # Почати запис нового жесту
            if key == ord('n'):
                active_label = simpledialog.askstring("Новий жест", "Введіть назву
нового жесту:")
                if active_label:
                    captured_coords = []
                    recording = True
```

```
            messagebox.showinfo("Запис жесту", f"Жест '{active_label}' записується. Натисніть 's' для завершення.")
        else:
            messagebox.showwarning("Увага", "Запис нового жесту скасовано.")

    # Додаємо позиції для наявного жесту
    if key == ord('a'):
        active_label = simpledialog.askstring("Додавання позицій", "Введіть назву жесту:")
        if active_label and active_label in data['labels']:
            captured_coords = []
            recording = True
            messagebox.showinfo("Додавання позицій", f"Додавання позицій для жесту '{active_label}'.")
        else:
            messagebox.showwarning("Увага", f"Жест '{active_label}' не знайдено!" if active_label else "Скасовано.")

    # Зупинка запису
    if key == ord('s') and active_label and recording:
        if len(captured_coords) > 0:
            add_gesture(active_label, captured_coords)
            messagebox.showinfo("Запис завершено", f"Жест '{active_label}' додано!")
        else:
            messagebox.showwarning("Увага", "Координати не були записані.")
        active_label = None
        captured_coords = []
```

```
            recording = False

        # Збереження даних у файл
        if key == ord('w'):
            try:
                with open("gestures.pkl", "wb") as f:
                    pickle.dump(data, f)
                messagebox.showinfo("Збереження", "Дані збережено.")
            except Exception as e:
                messagebox.showerror("Помилка", f"Помилка збереження: {e}")

        # Завантаження даних із файлу
        if key == ord('l'):
            if os.path.exists("gestures.pkl"):
                try:
                    with open("gestures.pkl", "rb") as f:
                        data = pickle.load(f)
                        knn.fit(data['coords'], data['labels'])
                        messagebox.showinfo("Завантаження", "Дані завантажено.")
                except Exception as e:
                    messagebox.showerror("Помилка", f"Помилка завантаження:
{e}")
            else:
                messagebox.showwarning("Увага", "Файл не знайдено!")

        # Завершення роботи
        if key == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

Program.py:

```
import cv2
import mediapipe as mp
import numpy as np
import pickle
import time
import os
from pynput.mouse import Button, Controller
import ctypes
from tkinter import filedialog, messagebox, simpledialog
import tkinter as tk
import subprocess

# Ініціалізація MediaPipe
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2, min_detection_confidence=0.9,
                       min_tracking_confidence=0.5)
mp_draw = mp.solutions.drawing_utils

# Реалізація класифікатора KNN
class KNNClassifier:
    def __init__(self, k=3):
        self.k = k
        self.coords = np.array([])
        self.labels = []
        self.label_to_int = {}  # Словник для перетворення текстових міток у числа
        self.int_to_label = {}  # Словник для перетворення чисел у текстові мітки
```

```
        def fit(self, coords, labels):
            self.coords = np.array(coords)
            # Унікальні мітки перетворюємо на ціле значення
            unique_labels = sorted(set(labels))
            self.label_to_int = {label: i for i, label in enumerate(unique_labels)}
            self.int_to_label = {i: label for label, i in self.label_to_int.items()}
            self.labels = np.array([self.label_to_int[label] for label in labels])

        def predict(self, samples):
            predictions = []
            for sample in samples:
                # Обчислення відстаней між sample та всіма точками у даних
                distances = np.linalg.norm(self.coords - sample, axis=1)
                # Отримання індексів найближчих сусідів
                nearest_indices = distances.argsort()[:self.k]
                # Мітки найближчих сусідів
                nearest_labels = self.labels[nearest_indices]
                # Вибір мітки з найбільшою кількістю
                predicted_label = np.bincount(nearest_labels).argmax()
                predictions.append(self.int_to_label[predicted_label])                  #
Перетворення назад у текстову мітку
            return predictions


    # Ініціалізація MediaPipe
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2,
min_detection_confidence=0.9,
                           min_tracking_confidence=0.5)
```

```
mp_draw = mp.solutions.drawing_utils

# Ініціалізація класифікатора жестів (власний KNN)
knn = KNNClassifier(k=3)
data = {'coords': [], 'labels': []}
gesture_actions = {}

CONFIG_FILE = "gesture_config.pkl"
root = tk.Tk()
root.withdraw()



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



# Збереження конфігурації жестів
def save_gesture_config():
    with open(CONFIG_FILE, "wb") as config_file:
```

```
            pickle.dump(gesture_actions, config_file)
        messagebox.showinfo("Конфігурація", "Конфігурація жестів успішно збережена.")


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


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

```
# Завантаження структурованих даних жестів
file_path = filedialog.askopenfilename(
    title="Виберіть файл жестів",
    filetypes=(("PKL файли", "*.pkl"), ("Усі файли", "*.*"))
)
if file_path:
    with open(file_path, "rb") as f:
        loaded_data = pickle.load(f)
        if loaded_data['coords'] and loaded_data['labels']:
            data = loaded_data
            knn.fit(data['coords'], data['labels'])
            messagebox.showinfo("Успішно", "Дані жестів завантажено.")
else:
    messagebox.showerror("Помилка", "Файл не вибрано. Програма завершить роботу.")
    exit()

# Ініціалізація налаштувань для миші
mouse = Controller()
user32 = ctypes.windll.user32
wScr, hScr = user32.GetSystemMetrics(0), user32.GetSystemMetrics(1)
smoothening = 7
frameReduction = 100
pLocX, pLocY = 0, 0
cLocX, cLocY = 0, 0

# Основні змінні
enable_gestures = False
ok_active_time = None
gesture_timer = {}  # Таймер для жестів
```

```
gesture_execution_delay = 5  # Затримка для виконання жесту (в секундах)
next_gesture_time = 0  # Час, коли можна виконати наступний жест
gesture_timeout = 8  # Таймаут між жестами

load_gesture_config()

# Відеозахоплення
cap = cv2.VideoCapture(0)

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

    img = cv2.flip(img, 1)
    h, w, _ = img.shape

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

    multi_hand_types = []
    multi_hand_coords = []
    recognized_gesture = None  # Змінна для виведення назви розпізнаного жесту

    # Обробка результатів MediaPipe
    if results.multi_hand_landmarks:
        for i, hand_landmarks in enumerate(results.multi_hand_landmarks):
            hand_type = results.multi_handedness[i].classification[0].label  # Тип руки (Left/Right)
```

```
                multi_hand_types.append(hand_type)

                coords = []
                for lm in hand_landmarks.landmark:
                    coords.append(lm.x)
                    coords.append(lm.y)
                multi_hand_coords.append(coords)

                mp_draw.draw_landmarks(img, hand_landmarks,
mp_hands.HAND_CONNECTIONS)

            if multi_hand_coords:
                for hand_type, coords in zip(multi_hand_types, multi_hand_coords):
                    coords_array = np.array(coords)
                    predicted_label = knn.predict([coords_array])[0]

                    # Ввімкнення/вимкнення режиму розпізнавання жестів
                    if hand_type == "Left":
                        if predicted_label == "OK":
                            if ok_active_time is None:
                                ok_active_time = time.time()

                            if time.time() - ok_active_time > 3:
                                enable_gestures = not enable_gestures
                                ok_active_time = None
                                state = "увімкнено" if enable_gestures else "вимкнено"
                                messagebox.showinfo("Стан", f"Розпізнавання жестів
{state}.")
                        else:
                            ok_active_time = None
```

```
            # Відображення таймера для "OK"
            if ok_active_time:
                remaining = int(3 - (time.time() - ok_active_time))
                cv2.putText(img, f"Hold 'OK': {remaining}s", (20, 70),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)

            # Виконання дій тільки після ввімкнення режиму
            if enable_gestures:
                # Прив'язка до лівої руки для розпізнавання жестів
                if hand_type == "Left":
                    recognized_gesture = predicted_label
                    # Таймаут між жестами
                    current_time = time.time()
                    if current_time < next_gesture_time:
                        # Показуємо, скільки часу залишилося до наступного
жесту
                        remaining_cooldown = int(next_gesture_time - current_time)
                        cv2.putText(img, f"Gesture cooldown:
{remaining_cooldown}s", (20, 150),
                                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255),
2)
                        continue

                    # Якщо жест знайдено в конфігурації
                    if recognized_gesture in gesture_actions:
                        # Запуск таймеру для цього жесту
                        if recognized_gesture not in gesture_timer:
                            gesture_timer[recognized_gesture] = time.time() #
Початок таймера
```

```
                    # Обчислюємо, скільки часу утримується жест
                    elapsed_time = time.time() - gesture_timer[recognized_gesture]

                    # Якщо жест утримується достатньо часу, виконуємо дію
                    if elapsed_time >= gesture_execution_delay:

execute_gesture_action(gesture_actions[recognized_gesture])
                        del gesture_timer[recognized_gesture]  # Скидуємо таймер після виконання
                        next_gesture_time = time.time() + gesture_timeout  # Встановлюємо таймаут
                else:
                    # Якщо жест не в конфігурації, скидаємо його таймер
                    if recognized_gesture in gesture_timer:
                        del gesture_timer[recognized_gesture]
            # Керування мишею: права рука
            if hand_type == "Right":
                lm_list = np.array(coords).reshape((21, 2)) * [w, h]
                x1, y1 = lm_list[8]  # Вказівний палець
                x2, y2 = lm_list[12]  # Середній палець

                # Рух миші
                if y1 < y2:
                    x3 = np.interp(x1, (frameReduction, w - frameReduction), (0, wScr))
                    y3 = np.interp(y1, (frameReduction, h - frameReduction), (0, hScr))
                    cLocX = pLocX + (x3 - pLocX) / smoothening
```

```
                    cLocY = pLocY + (y3 - pLocY) / smoothening
                    mouse.position = (cLocX, cLocY)
                    pLocX, pLocY = cLocX, cLocY
                    cv2.circle(img, (int(x1), int(y1)), 15, (255, 0, 255), cv2.FILLED)

                # Ліва кнопка миші
                if y1 > y2:
                    mouse.click(Button.left, 1)
                    cv2.circle(img, (int(x1), int(y1)), 15, (0, 255, 0), cv2.FILLED)

        # Виведення індикатора активного жесту
        if enable_gestures and recognized_gesture:
            if recognized_gesture in gesture_actions and recognized_gesture in gesture_timer:
                elapsed_time = time.time() - gesture_timer[recognized_gesture]
                remaining_time = max(0, gesture_execution_delay - elapsed_time)
                cv2.putText(img, f"Gesture: {recognized_gesture} ({remaining_time:.1f}s)", (20, 120),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)
            else:
                cv2.putText(img, f"Gesture: {recognized_gesture}", (20, 120), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)

        cv2.imshow("Hand Gesture and Mouse Control", img)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('c'):
            gesture = simpledialog.askstring("Налаштування жесту", "Введіть назву жесту:")
```

```
        if gesture:
            configure_gesture_action(gesture)

    if key == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

ДОДАТОК Б

Демонстраційний матеріал

ДОДАТОК В

Апробація результатів дослідження

SECTION: AUTOMATION AND ROBOTICS

СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТІВ В АВТОМАТИЗАЦІЇ ВИРОБНИЦТВА

Ніконенко Едуард Романович
здобувач вищої освіти
Харківський національний університет радіоелектроніки

Системи розпізнавання жестів відіграють дедалі важливішу роль у сучасних інтелектуальних виробничих середовищах. Вони забезпечують природну, безконтактну форму взаємодії між людиною та машиною, що особливо актуально в умовах, де фізичний контакт обмежений або небажаний. У контексті автоматизації виробництва такі системи дозволяють підвищити ефективність, зменшити час реакції та знизити ризик помилок оператора.

Основою більшості систем розпізнавання жестів є технології комп'ютерного зору, які аналізують зображення з відеокамер для виявлення та інтерпретації рухів. Використовуються як традиційні RGB-камери, так і більш складні пристрої –глибинні камери, інфрачервоні сенсори, ультразвукові датчики. На основі цих даних алгоритми машинного навчання розпізнають шаблони руху, класифікують жести та генерують відповідні команди [4].

Однією з ключових переваг таких систем є можливість адаптації до широкого спектра застосувань. Наприклад, у виробничих лініях оператор може запускати або зупиняти обладнання за допомогою простих жестів, не торкаючись поверхонь, що особливо важливо для чистих або стерильних зон. Водночас, у випадках надзвичайних ситуацій, система може ідентифікувати критичні жести, які сигналізують про потребу зупинити технологічний процес.

Важливу роль у цьому процесі відіграє правильне формування бази навчальних даних. Жести можуть мати індивідуальні особливості, тому система повинна мати механізми адаптації до конкретного користувача або групи користувачів. Крім того, для підвищення надійності необхідно враховувати зовнішні фактори, зокрема рівень освітлення, тло та наявність шумів [4].

Розроблена в рамках дослідження система ґрунтується на поєднанні традиційної обробки зображень і нейронних мереж. Завдяки використанню глибинної камери вдалося значно зменшити кількість помилкових спрацювань, пов'язаних з фоновими об'єктами. Алгоритм класифікації було оптимізовано для роботи в реальному часі, що забезпечує зручність і безперервність взаємодії.

Світовий ринок систем розпізнавання жестів демонструє стабільне зростання. Згідно з даними дослідницької компанії Grand View Research, обсяг ринку становив $21,04 млрд у 2023 році та очікується, що досягне $70,18 млрд до 2030 року, із середньорічним темпом зростання 18,8% у період з 2023 по 2030 роки.

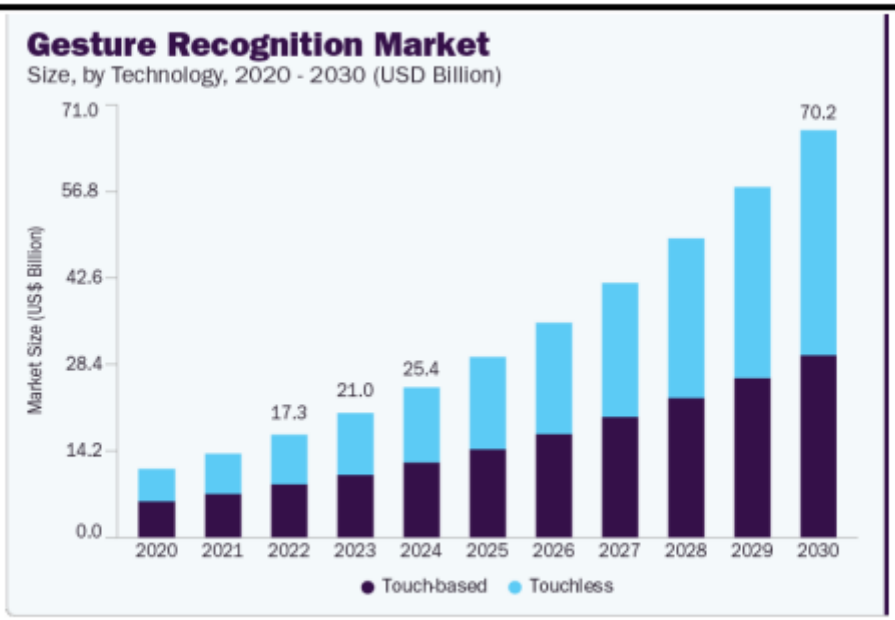

Рисунок 1 – Динаміка зростання ринку систем розпізнавання жестів за технологіями у 2020–2030 роках (за даними Grand View Research)

Це зростання зумовлене поширенням сенсорних пристроїв, підвищенням інтересу до безконтактних інтерфейсів у зв'язку з санітарними вимогами, а також розвитком технологій доповненої реальності, робототехніки й індустрії розваг. У промисловості особливий акцент робиться на впровадженні таких систем у виробничі та складські процеси, де важливе значення мають ергономіка, швидкість реакції та безпека. Застосування подібних систем у виробництві дозволяє перейти на якісно новий рівень автоматизації – інтерактивної, гнучкої та орієнтованої на людину. Подальші дослідження спрямовані на покращення стабільності роботи систем в умовах змінного освітлення, а також на розширення набору розпізнаваних жестів з урахуванням складних контекстів виробничого середовища.

Це зростання зумовлене поширенням сенсорних пристроїв, підвищенням інтересу до безконтактних інтерфейсів у зв'язку з санітарними вимогами, а також розвитком технологій доповненої реальності, робототехніки й індустрії розваг. У промисловості особливий акцент робиться на впровадженні таких систем у виробничі та складські процеси, де важливе значення мають ергономіка, швидкість реакції та безпека.

Застосування подібних систем у виробництві дозволяє перейти на якісно новий рівень автоматизації – інтерактивної, гнучкої та орієнтованої на людину. Подальші дослідження спрямовані на покращення стабільності роботи систем в умовах змінного освітлення, а також на розширення набору розпізнаваних жестів з урахуванням складних контекстів виробничого середовища.

Список використаних джерел

1. Grand View Research. Gesture Recognition Market Size, Share & Trends Analysis Report By Technology, By Application, By Region, And Segment Forecasts, 2024 - 2030. URL: https://www.grandviewresearch.com/industry-analysis/gesture-recognition-market
2. Mitra S., Acharya T. Gesture Recognition: A Survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2007, 37(3): 311-324.
3. Liang W., Zhao Y., Lv Z. Gesture Interaction Based on Machine Vision: A Survey. Human-centric Computing and Information Sciences, 2022, 12(1): 1-28.
4. Гриценко О.О. Системи взаємодії людина-комп'ютер: сучасні тренди та перспективи. Інформаційні технології і комп'ютерна інженерія, 2021, №1(59): 10–17.

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.505900.047 ПЗ	Пояснювальна записка	А4, 62 с.
		Додаткові матеріали	
2		Демонстраційний матеріал у	А4, 12 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.505900.047 ВД			
Розроб.		Ніконенко Е.Р.			Розроблення підсистеми автоматичного керування виробничим обладнанням з використанням розпізнавання жестів оператора	Літера	Аркуш	Аркушів
Перевір.		Бронніков А.І.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н.П.						
Затв.		Невлюдов І.Ш.						