ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет
радіоелектроніки

МЕТОД РОЗПІЗНАВАННЯ ЖЕСТІВ ДЛЯ ІНТЕРАКТИВНОГО КЕРУВАННЯ КОМП'ЮТЕРОМ З УРАХУВАННЯМ КОНТЕКСТНОЇ АДАПТАЦІЇ

Виконав:
студент групи СПм-23-4 Білоусов І.А.

Науковий керівник:
доц. Бологова Н.М.

МЕТА

Розробити метод розпізнавання жестів, який автоматично адаптує інтерпретацію рухів до активної програми, працює на типовій веб-камері й не потребує спеціалізованого сенсора.

- опрацювати сучасні підходи CV та DL для gesture-UI;
- вибрати бібліотеки та сформувати архітектуру системи;
- створити математичну модель;
- реалізувати класифікатор;
- забезпечити контекстну адаптацію;

КЛАСИФІКАЦІЯ ЖЕСТІВ

За характером:

- Статичні, дозволяють миттєво реагувати на прості команди;
- Динамічні, дають змогу реалізувати прокрутку чи перелистування.

За кількістю рук:

- Одноручні жести важливі на мобільних і портативних пристроях;
- Дворучні забезпечують високу надійність для критичних операцій.

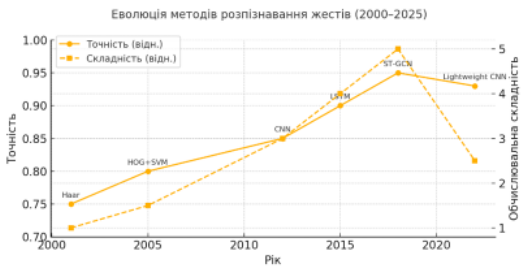
За функціональною семантикою:

- Маніпуляційні жести відповідають дії миші,
- Навігаційні — прокручуванню та масштабуванню, а системні — запуску або зупинці процесів.

3

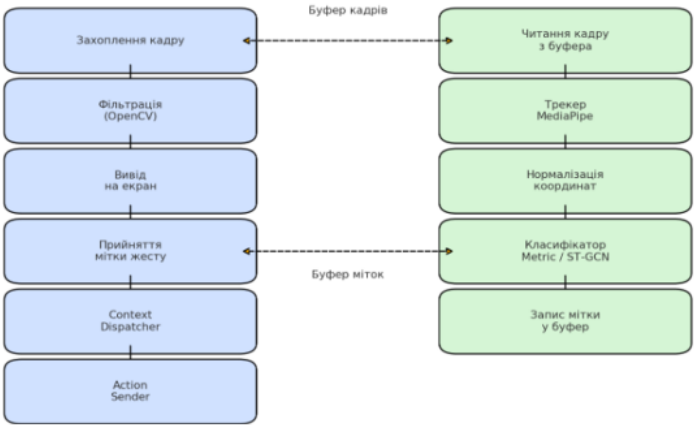
ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

- Viola–Jones (Haar-каскади)
- HOG + SVM
- Skeletal Heuristics (Ionescu 2005)
- CNN 2D/3D
- RNN / LSTM
- ST-GCN (Yan 2018)
- Lightweight CNN + MediaPipe (Alas 2022)



4

ЗАГАЛЬНА АРХІТЕКТУРА ЗАСТОСУНКУ

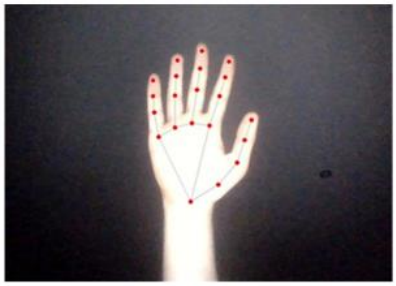


5

МАТЕМАТИЧНА МОДЕЛЬ НОРМАЛІЗАЦІЇ ЛЕНДМАРОК

1. Трансляція: за опорну точку обрано WRIST.
Для кожної лендмарки p_i обчислюємо вектор $v_i = p_i - p_0$.
2. Масштабування:
базовою довжиною береться відстань між WRIST і MIDDLE_MCP (ідекс 9):
$d = \|p_0 - p_9\|$;
$v_i' = v_i / d$.
3. Опціональне вирівнювання:
кут повороту кисті у площині XY визначається за вектором $p_0 \rightarrow p_5$ (ось долоні) і компенсується матрицею $R(-\theta)$.

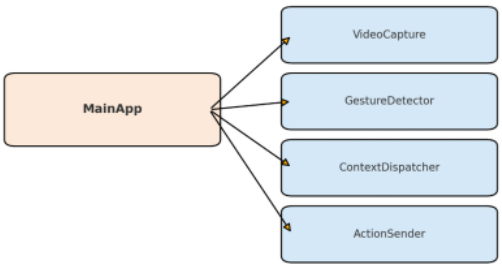
$q_i = R(-\theta) \cdot (p_i - p_0) / d \rightarrow$ нормалізований вектор (x', y', z')



6

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ

main.py запускає два потоки: один малює інтерфейс, другий обробляє кадри.
Зібрати додаткові приклади можна утилітою create_data_gestures.py; вона додає нові записи без перезапису існуючого JSON.
На UML-діаграмі видно, що ContextDispatcher не залежить від конкретного алгоритму розпізнавання — він працює лише з міткою й заголовком вікна.



7

ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА СИСТЕМИ

На слабких пристроях можна отримати майже миттєву реакцію, а на сучасних ноутбуках користувач матиме кращу точність без відчутної втрати швидкодії. Система гнучко адаптується до ресурсів, виконуючи головну мету роботи — забезпечити практичний баланс між якістю та продуктивністю.

8

ПІДСУМКОВІ ПЕРЕВАГИ РОЗРОБЛЕНОЇ СИСТЕМИ

1. Працює на звичайній веб-камері – жодних глибинних сенсорів або рукавичок.
2. Реальний час – середня затримка < 20 мс; інтерфейс не «мерехтить» завдяки паралельним потокам.
3. Контекстна адаптація – один жест виконує різні дії у різних програмах; додавання правил без перекомпіляції.
4. Гнучкий класифікатор – автоматичне перемикання між «економним» та «точним» режимами під апаратні ресурси.
5. Відкрите ПЗ та розширюваність – код на Python, структура модулів дозволяє легко інтегрувати нові жести й моделі.

9

ВИСНОВКИ

1. Реалізовано прототип, що працює на звичайній веб-камері й забезпечує точність 97–98 % при затримці < 20 мс.
2. Запропоновано універсальну нормалізацію лендмарок і дворівневий класифікатор, який автоматично балансує «якість ↔ швидкодія».
3. Впроваджено контекстну адаптацію, що дозволяє одному жесту виконувати різні дії залежно від активної програми.
4. Результати дослідження опубліковано у статті Бологова Н.М., Білоусов І.А. Метод розпізнавання жестів для інтерактивного керування комп'ютером з урахуванням контекстної адаптації / Системи управління, навігації та зв'язку. 2025. Вип.2 (80). С. 83-89.

10

ДОДАТОК Б

Вихідний код застосунку

Б.1 Модуль розпізнавання жестів

```
import tkinter as tk
from tkinter import messagebox
import time
import json
import os
import threading  # Додано для потоків
import keyboard

import cv2
import numpy as np
from PIL import Image, ImageTk

import mediapipe as mp

import win32gui
import win32con
import pyautogui

# ---------- Налаштування та глобальні змінні ----------
JSON_FILE = "gestures_data.json"   # Файл із жестами
THRESHOLD = 0.06                   # Поріг порівняння
running = False
cap = None
hands_processor = None
last_gesture_time = 0
is_playing = False

# Зберігаємо попередні координати пальця (для r_twofingers...)
prev_finger_x = None
prev_finger_y = None

# Список слів, що вказують, що вікно є плеєром (YouTube, VLC
тощо)
PLAYERS = [
    "YouTube", "VLC", "Windows Media Player", "Spotify",
    "Netflix", "AIMP", "Kinopoisk", "Winamp", "Okko", "ivi",
"Google"
]

# Глобальний словник для жестів
gestures_db = {}

# Глобальні змінні для обміну даними між потоками
```

```
current_frame = None                # Останній кадр з камери
current_gesture_label = None        # Розпізнаний жест
current_coords = None               # Координати (21,3)
frame_lock = threading.Lock()       # Блокування для доступу до
current_frame
result_lock = threading.Lock()      # Блокування для доступу до
current_gesture_label
stop_event = threading.Event()      # Для зупинки потоку

mp_hands = mp.solutions.hands


# ---------- 1. Автокорекція яскравості/контрасту ----------
def auto_adjust_brightness_contrast(image, clip_hist_percent=1):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = cv2.calcHist([gray],[0],None,[256],[0,256])  # shape
= (256,1)
    hist_size = len(hist)

    accumulator = [float(hist[0,0])]
    for i in range(1, hist_size):
        accumulator.append(accumulator[i-1] + float(hist[i,0]))

    maximum = accumulator[-1]
    clip_hist_percent *= (maximum / 100.0)
    clip_hist_percent /= 2.0

    min_gray = 0
    while min_gray < hist_size and accumulator[min_gray] <
clip_hist_percent:
        min_gray += 1

    max_gray = hist_size - 1
    while max_gray >= 0 and accumulator[max_gray] >= (maximum -
clip_hist_percent):
        max_gray -= 1

    if max_gray <= min_gray:
        return image

    alpha = 255.0 / (max_gray - min_gray)
    beta = -min_gray * alpha

    auto_result = cv2.convertScaleAbs(image, alpha=alpha,
beta=beta)
    return auto_result


# ---------- 2. Завантаження жестів із JSON ----------
def load_gestures_from_json(json_file):
    if not os.path.exists(json_file):
        print(f"[WARN] Файл {json_file} не знайдено.")
        return {}
```

```
    try:
        with open(json_file, "r", encoding="utf-8") as f:
            data = json.load(f)
    except (json.JSONDecodeError, OSError):
        print(f"[WARN] Файл {json_file} порожній або
пошкоджений.")
        return {}

    gestures_dict = {}
    for entry in data:
        label = entry.get("label", "")
        hands_arr = entry.get("hands", [])
        if not hands_arr:
            continue
        first_hand = hands_arr[0]
        lm_list = first_hand.get("landmarks", [])
        if len(lm_list) != 21:
            continue

        arr_coords = []
        for lm in lm_list:
            x, y, z = lm["x"], lm["y"], lm["z"]
            arr_coords.append([x, y, z])
        arr_coords = np.array(arr_coords, dtype=np.float32)

        if label not in gestures_dict:
            gestures_dict[label] = []
        gestures_dict[label].append(arr_coords)

    print(f"[INFO] Жести завантажено з {json_file}:
{list(gestures_dict.keys())}")
    return gestures_dict


# ---------- 3. Нормалізація та порівняння ----------
def normalize_landmarks(landmarks_21):
    coords = landmarks_21.copy()  # shape=(21,3)
    wrist = coords[0,:].copy()
    coords -= wrist

    dist_sq = np.sum(coords * coords)
    dist = np.sqrt(dist_sq) if dist_sq > 1e-6 else 1e-6
    coords /= dist
    return coords

def compare_to_label(coords_norm, label):
    if label not in gestures_db:
        return 999.0

    dists = []
    for ref in gestures_db[label]:
        ref_norm = normalize_landmarks(ref)
```

```
        dist = np.mean(np.sqrt(np.sum((coords_norm -
ref_norm)**2, axis=1)))
        dists.append(dist)

    mean_dist = float(np.mean(dists))
    return mean_dist


# ---------- 4. Розпізнавання жесту ----------
def detect_gesture_local(frame):
    """
    Локальна функція, яка виконує розпізнавання жесту (не змінює
глобальних змінних).
    Повертає (gesture_label, coords).
    """
    if not gestures_db:
        return (None, None)

    filtered = auto_adjust_brightness_contrast(frame,
clip_hist_percent=1)
    rgb = cv2.cvtColor(filtered, cv2.COLOR_BGR2RGB)
    results = hands_processor.process(rgb)
    if not results.multi_hand_landmarks:
        return (None, None)

    hand_lm = results.multi_hand_landmarks[0]
    coords = []
    for lm in hand_lm.landmark:
        coords.append([lm.x, lm.y, lm.z])
    coords = np.array(coords, dtype=np.float32)

    coords_norm = normalize_landmarks(coords)

    best_label = None
    best_dist = 999.0
    for lbl in gestures_db:
        dist = compare_to_label(coords_norm, lbl)
        if dist < best_dist:
            best_dist = dist
            best_label = lbl

    print(f"[DEBUG] best_label={best_label},
best_dist={best_dist:.4f}")
    if best_dist < THRESHOLD:
        return (best_label, coords)
    else:
        return (None, coords)
```

Б.2 Модуль контекстної адаптації

```
# ---------- 5. Робота з вікнами Windows ----------
def get_active_window_title():
    hwnd = win32gui.GetForegroundWindow()
    title = win32gui.GetWindowText(hwnd)
    return hwnd, title
def is_player_window(title):
    if not title:
        return False
    low_title = title.lower()
    for kw in PLAYERS:
        if kw.lower() in low_title:
            return True
    return False


# ---------- 6. Керування відтворенням ----------
def start_playback():
    global is_playing
    keyboard.press('space')
    keyboard.release('space')
    is_playing = True
    print("[INFO] -> старт відтворення")
def stop_playback():
    global is_playing
    keyboard.press('space')
    keyboard.release('space')
    is_playing = False
    print("[INFO] -> зупинка відтворення")


# ========== 7. Управління курсором (дві точки) ==========

prev_finger_x = None
prev_finger_y = None
def handle_two_fingers_and_click_relative(gesture_label, coords,
frame_shape, screen_size=(1920, 1080), scale_factor=1.0):
    global prev_finger_x, prev_finger_y
    global last_gesture_time
    if coords is None or coords.shape != (21, 3):
        return
    finger_x = coords[8, 0]
    finger_y = coords[8, 1]
    if prev_finger_x is None or prev_finger_y is None:
        prev_finger_x = finger_x
        prev_finger_y = finger_y
        return
    dx_norm = (finger_x - prev_finger_x)
    dy_norm = (finger_y - prev_finger_y)
    screen_w, screen_h = screen_size
    dx_pixels = dx_norm * screen_w * scale_factor * -1
    dy_pixels = dy_norm * screen_h * scale_factor
    cur_x, cur_y = pyautogui.position()
```

```
    new_x = cur_x + dx_pixels
    new_y = cur_y + dy_pixels
    if gesture_label == "r_twofingers_palm":
        pyautogui.moveTo(round(new_x), round(new_y), duration=0)
    elif gesture_label == "r_twofingersoneclick_palm":
        now = time.time()
        if now - last_gesture_time > 2:
            pyautogui.click(button="left")
            last_gesture_time = now
    prev_finger_x = finger_x
    prev_finger_y = finger_y


# ---------- Потокова функція для обробки кадрів ----------
def gesture_detection_thread():
    """
    Цей потік постійно читає кадр (current_frame) і виконує
detect_gesture_local,
    результати записує у глобальні (current_gesture_label,
current_coords).
    """
    global current_frame, current_gesture_label, current_coords
    global stop_event
    while not stop_event.is_set():
        # Беремо кадр з current_frame (захист через frame_lock)
        frame_lock.acquire()
        temp_frame = None
        if current_frame is not None:
            temp_frame = current_frame.copy()
        frame_lock.release()
        if temp_frame is not None:
            # Викликаємо локальну функцію розпізнавання
            gesture_label, coords =
detect_gesture_local(temp_frame)
            # Записуємо результат (захист через result_lock)
            result_lock.acquire()
            current_gesture_label = gesture_label
            current_coords = coords
            result_lock.release()
        time.sleep(0.01)  # Невелика пауза, щоб зменшити
навантаження


# ---------- 8. Основний цикл (Tkinter) ----------
def show_frame():
    global running, cap, last_gesture_time
    global prev_finger_x, prev_finger_y
    global current_frame, current_gesture_label, current_coords
    if not running:
        return
    ret, frame = cap.read()
    if not ret or frame is None:
        label_video.config(text="Немає сигналу з камери")
```

```
            root.after(500, show_frame)
            return
        # Записуємо останній зчитаний кадр у глобальну змінну
(захищено lock)
        frame_lock.acquire()
        current_frame = frame
        frame_lock.release()
        # Зчитуємо результати розпізнавання (захищено result_lock)
        result_lock.acquire()
        gesture_label = current_gesture_label
        coords = current_coords
        result_lock.release()
        # Використовуємо gesture_label, coords для логіки:
        if gesture_label is not None:
            try:
                hwnd, title = get_active_window_title()
                label_window_name.config(text=f"Активне вікно:
{title if title else '-'}")

                now = time.time()
                if gesture_label in ["r_open_palm", "r_close_palm"]:
                    if now - last_gesture_time > 2:
                        if is_player_window(title):
                            if gesture_label == "r_open_palm":
                                start_playback()
                            elif gesture_label == "r_close_palm":
                                stop_playback()
                        last_gesture_time = now
                if gesture_label in ["r_twofingers_palm",
"r_twofingersoneclick_palm"]:
                    handle_two_fingers_and_click_relative(
                        gesture_label,
                        coords,
                        frame.shape[:2],
                        pyautogui.size(),
                        scale_factor=1.0
                    )
                else:
                    prev_finger_x = None
                    prev_finger_y = None
            except Exception as e:
                status_label.config(text="Помилка", fg="red")
                messagebox.showerror("Помилка", str(e))
                stop_program()
                return
        else:
            # Немає жесту -> скидаємо
            prev_finger_x = None
            prev_finger_y = None
        # Показуємо картинку (не фільтровану, а оригінал frame)
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        imgtk = ImageTk.PhotoImage(image=Image.fromarray(frame_rgb))
        label_video.imgtk = imgtk
```

```
    label_video.config(image=imgtk)
    root.after(30, show_frame)
def poll_active_window():
    if not running:
        return
    try:
        hwnd, title = get_active_window_title()
        label_window_name.config(text=f"Активне вікно: {title if
title else '-'}")
    except:
        label_window_name.config(text="Активне вікно: -")

    root.after(1000, poll_active_window)
```

Б.3 Модуль збереження жестів

```
import cv2
import mediapipe as mp
import numpy as np
import json
import os
import time


def auto_adjust_brightness_contrast(image, clip_hist_percent=1):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Обчислимо гістограму
    hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
    hist_size = len(hist)

    accumulator = [float(hist[0])]
    for i in range(1, hist_size):
        accumulator.append(accumulator[i - 1] + float(hist[i]))

    maximum = accumulator[-1]
    clip_hist_percent *= maximum / 100.0
    clip_hist_percent /= 2.0

    # Знаходимо лівий і правий кроп
    min_gray = 0
    while accumulator[min_gray] < clip_hist_percent:
        min_gray += 1

    max_gray = hist_size - 1
    while accumulator[max_gray] >= maximum - clip_hist_percent:
        max_gray -= 1

    # Розрахунок α та β для convertScaleAbs()
    alpha = 255 / (max_gray - min_gray)
```

```
    beta = -min_gray * alpha

    auto_result = cv2.convertScaleAbs(image, alpha=alpha,
beta=beta)
    return auto_result


def collect_two_hands_gesture_data(gesture_name: str,
                                   output_json: str =
"gestures_data.json",
                                   max_num_hands: int = 2):
    # 1. Завантажуємо наявні дані з JSON, щоб не перезаписати
файл
    if os.path.exists(output_json):
        try:
            with open(output_json, "r", encoding="utf-8") as f:
                data = json.load(f)
            print(f"[INFO]  JSON: {output_json}, записів:
{len(data)}")
        except (json.JSONDecodeError, OSError):
            print(f"[WARN] Файл {output_json} пошкоджений або
порожній. Починаємо заново.")
            data = []
    else:
        data = []
        print("[INFO] Створюємо новий JSON-файл.")

    mp_hands = mp.solutions.hands
    mp_drawing = mp.solutions.drawing_utils

    hands = mp_hands.Hands(
        static_image_mode=False,
        max_num_hands=max_num_hands,
        min_detection_confidence=0.5,
        min_tracking_confidence=0.5
    )

    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("[ERROR] Не вдалося відкрити камеру.")
        return

    print("[INFO] Керування:")
    print("  - Натисніть ПРОБІЛ, щоб зберегти координати
поточного жесту.")
    print("  - Натисніть ESC, щоб вийти з програми.")
    print(f"[INFO] Збираємо жест: {gesture_name}")

    while True:
        ret, frame = cap.read()
        if not ret:
            print("[ERROR] Не вдалося зчитати кадр з камери.")
            break
```

```
        # 2. Фільтрація кадру (автокорекція яскравості й
контрасту)
        frame_filtered = auto_adjust_brightness_contrast(frame,
clip_hist_percent=1)

        # Перетворюємо BGR → RGB для MediaPipe
        rgb_frame = cv2.cvtColor(frame_filtered,
cv2.COLOR_BGR2RGB)
        results = hands.process(rgb_frame)

        # Відображаємо ключові точки для наочності
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    frame_filtered,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS
                )

        # Текст на екрані
        cv2.putText(frame_filtered, f"Gesture: {gesture_name}",
(10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        cv2.imshow("Collecting Gestures (Two Hands)",
frame_filtered)

        key = cv2.waitKey(1) & 0xFF
        if key == 27:  # ESC
            print("[INFO] Вихід із режиму збору даних.")
            break
        elif key == 32:  # ПРОБІЛ
            if results.multi_hand_landmarks:
                # Формуємо один запис для всіх знайдених рук
                hands_data = []
                for i, hand_landmarks in
enumerate(results.multi_hand_landmarks):
                    landmark_points = [{"x": lm.x, "y": lm.y,
"z": lm.z}
                                       for lm in
hand_landmarks.landmark]

                    hands_data.append({
                        "hand_index": i,      # 0 або 1
                        "landmarks": landmark_points
                    })

                record = {
                    "label": gesture_name,
                    "hands": hands_data,
                    "timestamp": int(time.time())
                }
```

```
                data.append(record)
                print(f"[INFO] Запис збережено (кількість рук =
{len(hands_data)}). Усього: {len(data)}.")
            else:
                print("[WARN] Руки не знайдено — запис не
додано.")

    cap.release()
    cv2.destroyAllWindows()
    hands.close()

    # 3. Зберігаємо все в JSON
    with open(output_json, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

    print(f"[INFO] Дані збережено у {output_json}. Усього
записів: {len(data)}")


if __name__ == "__main__":
    gesture_name_input = input("Введіть назву жесту (наприклад,
BOTH_HANDS_OK): ")
    collect_two_hands_gesture_data(gesture_name_input)
```