

IMPLEMENTASI MEDIPIPE & MACHINE LEARNING UNTUK  
INTERAKSI MANUSIA-KOMPUTER

# Kontrol Game Subway Surfers dengan Gerakan Kepala

Politeknik Negeri Bandung

2A-D3

---

# Team



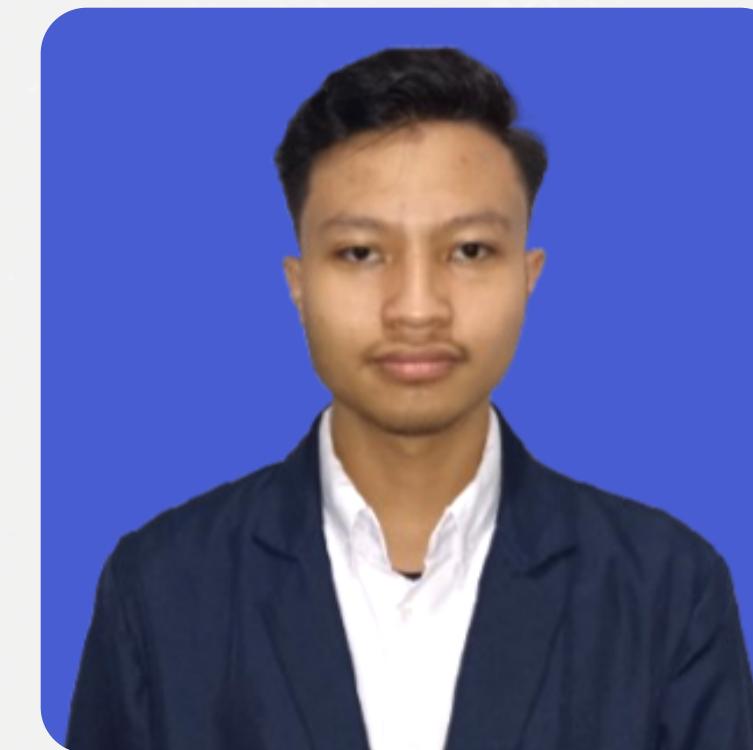
**Geraldin Gysrawa**

231511011



**Ikhwan Zuhri Al Ghifary**

231511015



**Muhammad Harish Al-R.**

231511020

---

# Table of Contents

3	4	5
Latar Belakang	Rumusan Masalah	Arsitektur & Teknologi Utama
6	7	8
Alur Kerja Sistem	Tahapan Proyek	Performa Model
9	10	11
Tantangan & Solusi	DEMO APLIKASI	Kesimpulan

# Latar Belakang

**Manfaatkan solusi  
MediaPipe untuk kasus  
spesifik**

1

Interaksi Manusia-Komputer (HCI) kini lebih dari sekadar keyboard & mouse

2

Computer vision

3

Game adalah area yang ideal untuk menerapkan kontrol gestur yang responsif

# Rumusan Masalah

1

Bagaimana mengubah gerakan kepala menjadi data yang bisa diproses? (Yaw, Pitch, Roll).

2

Bagaimana menciptakan model yang akurat untuk mengenali gestur spesifik? (Kiri, Kanan, Lompat, dll.)

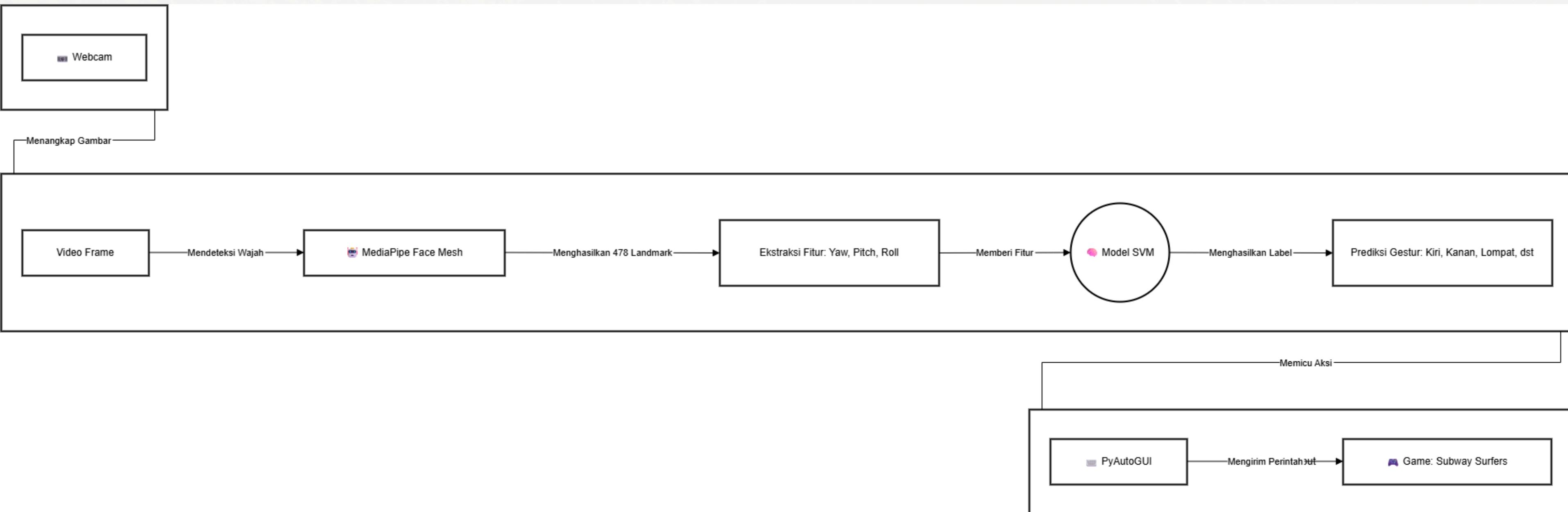
3

Bagaimana mengintegrasikan semua ini untuk mengontrol game secara real-time?

# Arsitektur dan Teknologi

Teknologi	Peran dalam Proyek
Deteksi & Pelacakan Wajah	Mendeteksi 478 landmark wajah secara real-time untuk mengekstrak data pose kepala.
Pelatihan Model	Menggunakan Support Vector Machine (SVM) untuk melatih model klasifikasi yang mengenali setiap gestur.
Input Video	Mengambil feed video dari webcam.
Kontrol Game	Mengirimkan perintah penekanan tombol (up, down, left, right) ke sistem operasi.

# Alur Kerja Sistem



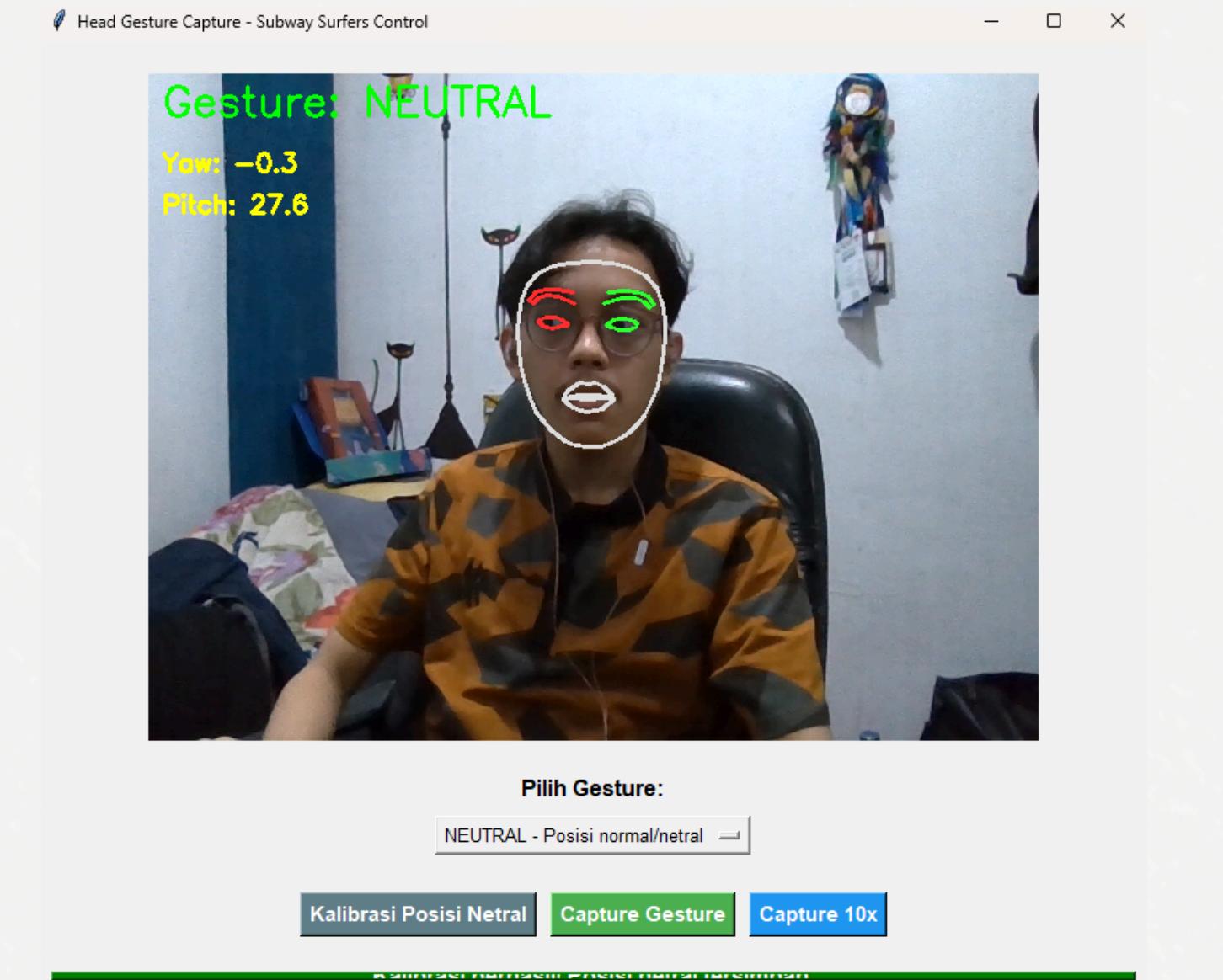
# Tahapan Proyek

1. Akuisisi Data	2. Pelatihan Model	3. Implementasi Real-Time	4. Antarmuka Pengguna
Membuat aplikasi GUI dengan Tkinter untuk mengumpulkan data gestur kepala yang bersih dan berlabel.	Melatih model SVM menggunakan data yang telah dikumpulkan. Dievaluasi dengan 5-Fold Cross Validation.	Mengintegrasikan model ke dalam skrip Python yang membaca webcam dan mengirim perintah keyboard.	Membangun aplikasi demo menggunakan <b>Streamlit</b> untuk visualisasi dan kontrol yang mudah.

# Akuisisi Data

Proses:

1. Antarmuka GUI: Pengguna memilih label gestur yang ingin direkam (misal: "JUMP").
2. Deteksi Pose: Saat tombol "Capture" ditekan, MediaPipe mendeteksi landmark wajah dan fungsi kami menghitung nilai Yaw, Pitch, dan Roll.
3. Penyimpanan Data: Nilai-nilai ini, bersama dengan label gesturnya, disimpan dalam format JSON.
4. Hasil: Terbentuk sebuah dataset kustom berisi 450 sampel yang siap digunakan untuk pelatihan.



```
Jumlah sampel data: 450
Bentuk matriks fitur (X): (450, 3)
Bentuk vektor label (y): (450,)
Kelas unik: ['DUCK' 'JUMP' 'LEFT' 'NEUTRAL' 'RIGHT']
```

# Pelatihan Model

Melatih model yang dapat membedakan 5 kelas gestur berdasarkan 3 fitur (Yaw, Pitch, Roll).

Proses:

1. Pra-pemrosesan:

- StandardScaler: Menyamakan skala nilai Yaw, Pitch, dan Roll agar setiap fitur memiliki bobot yang setara.
- LabelEncoder: Mengubah label teks (e.g., "JUMP") menjadi angka yang bisa diproses model.

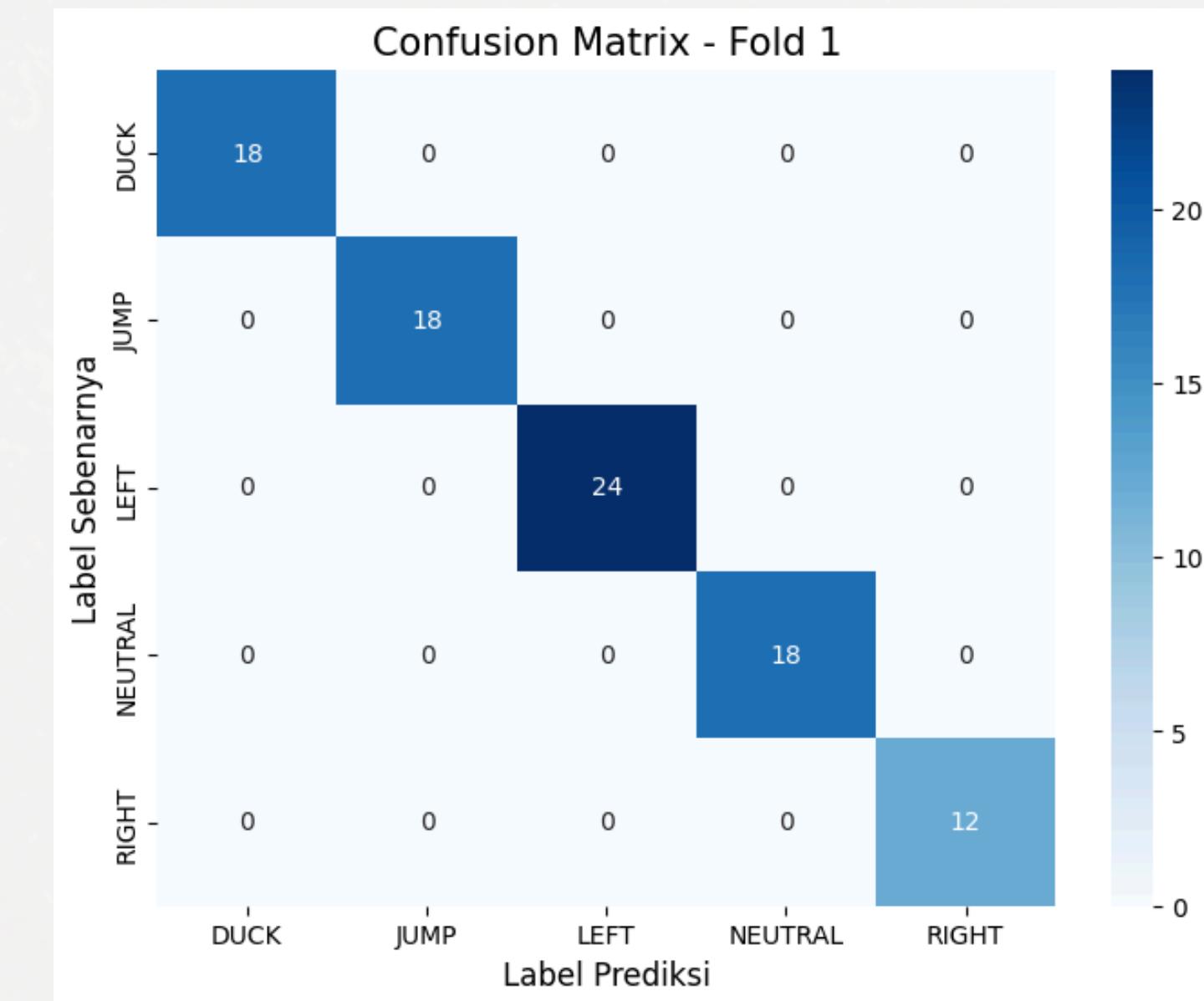
2. Pemilihan Model: Kami menggunakan Support Vector Machine (SVM) dengan kernel RBF, yang sangat efektif untuk menemukan batasan keputusan yang kompleks.

3. Validasi Ketat: Untuk memastikan model andal, kami menggunakan 5-Fold Cross-Validation. Dataset dibagi 5, lalu model dilatih dan diuji 5 kali, di mana setiap bagian data mendapat giliran menjadi data uji.

4. Hasil: Proses ini menghasilkan model final yang telah dilatih pada seluruh data dan siap untuk digunakan.

# Performa Model

Metrik	Nilai	Interpretasi
Precision	1.00	Jika model memprediksi sebuah gestur, prediksinya 100% benar.
Recall	1.00	Model berhasil mengidentifikasi 100% dari semua gestur yang ada.
F1-Score	1.00	Keseimbangan sempurna antara Precision dan Recall.



Akurasi Rata-rata: 100% dari 5-Fold Cross-Validation

## Analisis Singkat:

- Model SVM berhasil memisahkan 5 kelas gestur tanpa kesalahan pada data validasi.
- Ini menunjukkan fitur (Yaw, Pitch, Roll) sangat distingatif dan kualitas data latih sangat baik.

# Integrasi ke Game

Model kita tidak "berbicara" langsung dengan game. Ia berbicara dengan sistem operasi, dan sistem operasi yang berbicara dengan game. PyAutoGUI adalah penerjemah.

1. Model SVM mengeluarkan prediksi, misalnya string 'JUMP'.
2. Kode Python kita menerima string ini.
3. Sebuah blok if-elif sederhana memetakan string prediksi ke perintah PyAutoGUI.
4. PyAutoGUI menjalankan perintah tersebut, misalnya pyautogui.press('up').
5. Sistem operasi menerima ini seolah-olah tombol panah atas benar-benar ditekan pada keyboard fisik.
6. Jendela game Subway Surfers yang sedang aktif merespons penekanan tombol tersebut, dan karakter pun melompat.

```
if results.multi_face_landmarks:  
    for face_landmarks in results.multi_face_landmarks:  
        pose = calculate_head_pose(face_landmarks.landmark)  
        if pose:  
            fitur = np.array([[pose['yaw'], pose['pitch'], pose['roll']]])  
            fitur_scaled = scaler.transform(fitur)  
            pred = model.predict(fitur_scaled)  
            gesture = label_encoder.inverse_transform(pred)[0]  
            gesture_text = gesture  
  
            # Kontrol game dengan cooldown 0.5 detik  
            if gesture_text != last_gesture and gesture_text in ["JUMP", "LEFT", "RIGHT", "DUCK"]:  
                if time.time() - last_time > 0.5:  
                    if gesture_text == "JUMP":  
                        pyautogui.press('up')  
                    elif gesture_text == "LEFT":  
                        pyautogui.press('left')  
                    elif gesture_text == "RIGHT":  
                        pyautogui.press('right')  
                    elif gesture_text == "DUCK":  
                        pyautogui.press('down')  
  
                    print(f"Aksi: {gesture_text}")  
                    last_gesture = gesture_text  
                    last_time = time.time()  
  
                elif gesture_text == "NEUTRAL":  
                    last_gesture = None  
  
            # Tampilkan hasil prediksi di frame  
            cv2.putText(frame, f"Gesture: {gesture_text}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)  
            cv2.imshow('Kontrol Subway Surfers - Tekan ESC untuk Keluar', frame)
```

# Tantangan & Solusi

Tantangan	Solusi
Kontrol Terlalu Sensitif: Satu gerakan kepala memicu banyak aksi (lompat berkali-kali).	Implementasi Cooldown Timer. Setelah sebuah aksi dijalankan, sistem akan mengabaikan perintah yang sama selama 0.5 detik. Hasil: Kontrol menjadi lebih stabil, presisi, dan terasa disengaja oleh pengguna.

# Demo Aplikasi

---

# Kesimpulan

1

- sukses mengembangkan sistem kontrol game berbasis gerakan kepala yang fungsional, akurat, dan responsif

2

- Proyek ini membuktikan kelayakan MediaPipe & SVM untuk aplikasi HCI yang inovatif dan praktis

# Saran pengembangan

A

- Gestur Tambahan: Menggunakan gestur mulut atau kedipan mata untuk power-up

# The End

THANK YOU FOR LISTENING

SuMiCi