Program Studi Teknik Informatika
Institut Teknologi Sumatera

Mata Kuliah: Sistem / Teknologi Multimedia - IF25-40305 **Tugas: Final Project**

Nama Anggota:
1. Sulthan Fatih Pradewa (122140183), (sulthan122140183)
2. Muhammad Fauzi Azizi (122140106), (Fauzi_122140106)
3. Ihya Razky Hidayat (122140167). (Ecstarssyy)

Tanggal: 12-12-2025

1 Deskripsi Proyek

Proyek ini bernama **CineTune**, yaitu sebuah permainan edukatif yang memungkinkan pengguna menebak judul film menggunakan **k**endali berbasis gesture tangan. Aplikasi ini memanfaatkan teknologi *computer vision* untuk mendeteksi pose tangan (hand landmarks) melalui kamera, kemudian mengklasifikasikan gesture tersebut menjadi pilihan jawaban A, B, C, atau D.

CineTune dirancang untuk menghadirkan pengalaman bermain yang unik tanpa perangkat input tradisional seperti mouse atau keyboard. Seluruh interaksi dilakukan menggunakan gesture tangan yang diproses secara real-time. OpenCV dan MediaPipe dengan bahasa python digunakan sebagai library pengenalan pola landmark model jari-jari tangan manusia yang terdeteksi pada kamera/webcam secara realtime [1].

Aplikasi ini menampilkan serangkaian pertanyaan berupa gambar cuplikan film (movie still), lengkap dengan empat opsi jawaban. Pengguna harus mengangkat gesture tangan tertentu untuk memilih jawaban. Sistem akan melakukan *gesture holding detection* untuk memastikan jawaban tidak terbaca secara tidak sengaja. Setelah gesture dipertahankan beberapa saat, jawaban secara otomatis terekam dan aplikasi menampilkan umpan balik berupa informasi benar/salah lengkap dengan suara efek pendukung.

2 Tujuan Proyek

1. Membangun sistem pendeteksi tangan (hand tracking) yang stabil dan real-time menggunakan teknologi seperti MediaPipe

2. Mengintegrasikan modul gesture recognition dengan antarmuka aplikasi *CineTune* sehingga pengguna dapat memilih jawaban kuis hanya dengan menggunakan gerakan tangan.

3. Meningkatkan pengalaman pengguna (user experience) melalui interaksi yang inovatif, menarik, dan lebih inklusif terutama bagi pengguna yang ingin bermain tanpa menggunakan perangkat input tradisional.

3 Alat dan Bahasa Pemrograman

3.1 Bahasa Pemrograman / Tools

Pada pengembangan sistem deteksi gesture dan aplikasi interaktif CineTune, menggunakan bahasa python sebagai bahasa utama.

3.2 Library

1. pygame : Untuk antarmuka game (tampilan), window, event loop, dan pemutaran audio.
2. OpenCV : Mengambil inputan secara real-time dari camera, menampilkan video ke layar,

mengambil hasil deteksi landmark.
3. Numpy : Digunakan untuk melakukan operasi matematis pada koordinat landmark
4. Mediapipe : Untuk operasi array numerik (konversi frame kamera ke surface pygame, generate beep audio).
5. Time Library : Mengatur delat atau jeda
6. OS dan Sys : Digunakan untuk import struktur atau medeteksi gambar pada penyimpanan

3.3 Metode dan Algoritma

3.3.1 Alur Kerja

Filter CineTune dimulai ketika pengguna menjalankan program utama main.py, yang akan langsung menjalankan fungsi utama dari kelas CineTuneApp pada modul frame_utama.py. Secara umum, langkah-langkah dalam memulai program adalah sebagai berikut.

1. Inisialisasi program dari file main.py
   Program utama memanggil:

```
from ui.frame_utama import CineTuneApp
if __name__ == "__main__":
    app = CineTuneApp()
    app.run()
```

Kode 1: Program Pemanggil Utama

   Saat kode di atas dijalankan, objek CineTuneApp dibuat dan fungsi run() dipanggil untuk memulai seluruh alur aplikasi.
2. Pemanggilan modul utama aplikasi Di dalam konstruktor CineTuneApp, sistem akan:
   - Memuat data soal kuis dari data/questions.csv dan pemetaan gesture dari data/gestures.csv melalui data_loader.py.
   - Menginisialisasi pengelola permainan (GameManager), pengelola audio (AudioPlayer), serta antarmuka pengguna (GameUI di tampilan.py).
   - Mengaktifkan kamera dengan cv2.VideoCapture(0) dan menyiapkan modul deteksi gesture (GestureDetector dan GestureMapper).

3. Pengaktifan kamera dan pemrosesan input gesture Saat pengguna menekan tombol "Mulai" pada menu utama, status permainan diubah ke mode game. Sistem kemudian:
   - Mengambil frame dari kamera secara terus-menerus.
   - Mengirim frame tersebut ke GestureDetector untuk dideteksi landmark tangannya.
   - Meneruskan hasil landmark ke GestureMapper untuk dikonversi menjadi label gesture (misalnya A, B, C, atau D).

4. Pemetaan gesture menjadi aksi dalam permainan Hasil klasifikasi gesture yang dikembalikan oleh GestureMapper akan dianggap sebagai:
   - Pilihan jawaban yang sesuai dengan opsi A, B, C, atau D pada soal yang sedang aktif.
   - Input yang harus ditahan selama beberapa saat (gesture hold time) sebelum benar-benar tercatat sebagai jawaban, agar mengurangi kesalahan baca.

5. Pembaruan status permainan secara berkelanjutan Program terus:
    - Membaca input dari kamera.
    - Mengupdate tampilan Pygame (gambar film, teks soal, pilihan jawaban).
    - Menilai jawaban berdasarkan gesture yang diterima dan memperbarui skor.

    Permainan akan berlanjut sampai seluruh soal selesai, kemudian sistem menampilkan layar hasil (game over) dan memberikan opsi untuk mengulang atau kembali ke menu utama.

3.3.2 Deteksi Gesture Tangan

1. Deteksi Tangan Deteksi tangan merupakan tahapan awal untuk mengenali posisi dan gerakan tangan sebagai input utama dalam permainan. Proses yang dilakukan adalah:
    - Kamera menangkap gambar tangan pengguna dalam bentuk frame video.
    - Frame dikirim ke modul GestureDetector yang memanfaatkan MediaPipe Hands.
    - MediaPipe mendeteksi 21 landmark pada tangan dan mengembalikan koordinat setiap titik (x, y) relatif terhadap ukuran frame.
    - Landmark ini kemudian digambar kembali di frame (annotated frame) untuk memberikan indikasi visual posisi tangan pada layar.
2. Pencocokan Pose Tangan dengan Gesture Pilihan Jawaban Setelah landmark tangan diperoleh, langkah berikutnya adalah mencocokkan pose tangan tersebut dengan gesture yang telah didefinisikan untuk setiap opsi jawaban. Prosesnya meliputi:
    - Pengumpulan dan Observasi Pola Gesture
      Pengembang menentukan pola pose untuk masing-masing gesture, misalnya:
        - Jempol ke atas untuk opsi A.
        - Telapak terbuka untuk opsi B.
        - Dua jari (telunjuk dan tengah) ke atas untuk opsi C.
        - Genggaman tangan untuk opsi D.
    - Perancangan Aturan Klasifikasi Gesture

      Pada modul GestureMapper, koordinat beberapa titik penting (ujung jempol, ujung telunjuk, ujung jari tengah, dan seterusnya) dibandingkan satu sama lain.
        - Jika perbedaan posisi antar ujung jari kecil, tangan diasumsikan menggenggam (gesture D).
        - Jika jempol berada paling atas dibanding jari lain, diasumsikan sebagai gesture A.
        - Kombinasi kondisi posisi jari lain digunakan untuk membedakan gesture B dan C.
    - Klasifikasi Gesture Secara Real-time

      Pada setiap frame:

        - Hasil landmark dimasukkan ke GestureMapper.
        - Fungsi map() mengembalikan label A, B, C, D, atau None jika tidak ada gesture yang dikenali.
        - Label yang stabil selama durasi tertentu dianggap sebagai jawaban pengguna.
3. Sistem Verifikasi Gesture dan Interaksi Visual Sistem verifikasi memastikan bahwa gesture yang dikenali benar-benar diinginkan pengguna sebelum diproses sebagai jawaban. Mekanismenya antara lain:
    - Sistem menyimpan gesture terakhir dan waktu mulai ketika gesture tersebut pertama kali muncul.
    - Jika gesture berubah, timer direset; jika gesture yang sama dipertahankan lebih lama dari gesture_hold_time, gesture tersebut dianggap valid.
    - Setelah valid, gesture dikirim ke submit_answer() untuk dinilai sebagai jawaban soal saat ini.

4. Di sisi antarmuka visual:
    - Gambar film dan opsi jawaban ditampilkan dalam layout seperti tampilan aplikasi video pendek.
    - Pengguna dapat melihat perubahan tampilan ketika jawaban diproses (misalnya berpindah ke layar hasil atau soal berikutnya).
5. Penggunaan Feedback Audio Untuk memperkaya interaksi, CineTune menambahkan feedback audio pada beberapa titik proses:
    - Audio Pertanyaan
        - Setiap soal memiliki berkas audio (misalnya cuplikan suara film atau narasi singkat) yang diputar melalui AudioPlayer.
        - Audio ini membantu pengguna menebak judul film selain dari gambar yang ditampilkan.
    - Audio Feedback Jawaban
        - Setelah jawaban diproses, jika jawaban benar, sistem memutar suara positif (misalnya efek “benar”).
        - Jika jawaban salah, sistem memutar suara berbeda sebagai penanda kesalahan.
    - Integrasi dengan Status Game
        - Audio pertanyaan akan dihentikan ketika pengguna sudah menjawab dan sistem berpindah ke layar hasil.
        - Hal ini mencegah tumpang tindih antara audio pertanyaan dan audio feedback.

Dengan kombinasi deteksi gesture tangan berbasis landmark, aturan klasifikasi sederhana, serta integrasi visual dan audio, metode dan algoritma yang digunakan pada CineTune memungkinkan pengguna untuk bermain kuis tebak film hanya dengan gerakan tangan di depan kamera.

4 Implementasi

4.1 Instalasi Library

4.1.1 Menggunakan Virtual Environment

File konfigurasi otomatis yang dibuat saat membuat virtual environment dengan venv. File ini menyimpan informasi seperti versi Python dan pengaturan environment.

```bash
python -m venv venv
```

Kode 2: Perintah create virtual environment

4.1.2 Menggunakan Requirements.txt

Daftar library yang digunakan dalam proyek. File ini digunakan untuk menginstal semua dependensi secara otomatis dengan perintah pip install -r requirements.txt.

```bash
pip install -r requirements.txt
```

Kode 3: Perintah install requirements.txt

Dependensi utama proyek CineTune:

- opencv-python==4.10.0.84
- mediapipe==0.10.21
- pygame==2.6.1
- numpy==1.26.4
- scipy==1.15.3
- sounddevice==0.5.3
- matplotlib==3.10.7
- pillow==12.0.0

4.2 Modul data_loader.py

```python
import csv
import os

# ========================================
# RESOLVE PATHS (Lebih aman & fleksibel)
# ========================================
BASE_DIR = os.path.abspath(os.path.join(os.path.
dirname(__file__), "../../"))
QUESTIONS_CSV = os.path.join(BASE_DIR, "data",
"questions.csv")
GESTURES_CSV  = os.path.join(BASE_DIR, "data",
"gestures.csv")
```

Kode 4: Import Library dan Resolusi Path

4.2.2 Fungsi Load Questions

Memuat data pertanyaan dari file CSV yang berisi:

- ID pertanyaan
- Path gambar film
- Path audio
- 4 opsi jawaban (A, B, C, D)
- Jawaban benar

```
def load_questions():
    questions = []
    try:
        with open(QUESTIONS_CSV, newline='', encoding='utf-8') as f:
            reader = csv.DictReader(f)
            for row in reader:
                questions.append({
                    "id": int(row["id"]),
                    "image": os.path.join(BASE_DIR, row["image_path"]),
                    "audio": os.path.join(BASE_DIR, row["audio_path"]),
                    "options": {
                        "A": row["option_a"],
                        "B": row["option_b"],
                        "C": row["option_c"],
                        "D": row["option_d"],
                    },
                    "answer": row["answer"].sfrip().upper(),
                })
    except Exception as e:
        print("[ERROR] Gagal load questions:", e)
    return questions
```

Kode 5: Fungsi Load Questions

4.2.3 Fungsi Load Gesture Map

Memuat pemetaan gesture ke jawaban dari file CSV.

```
def load_gesture_map():
    gestures = {}
    try:
        with open(GESTURES_CSV, newline='', encoding='utf-8') as f:
            reader = csv.DictReader(f)
            for row in reader:
                gestures[row["gesture_name"]] = row["answer"].strip().
upper()
        print("[DEBUG] Gesture map loaded:", gestures)
    except Exception as e:
        print("[ERROR] Gagal load gestures:", e)
    return gestures
```

Kode 6: Fungsi Load Gesture Map

4.3 Modul gesture_detector.py

4.3.1 Import Library

```python
import cv2
import mediapipe as mp

class GestureDetector:
    def __init__(self):
        self.hands = mp.solutions.hands.Hands(
            max_num_hands=1,
            min_detection_confidence=0.7,
            min_tracking_confidence=0.7
        )
        self.drawer = mp.solutions.drawing_utils
```

Kode 7: Inisialisasi MediaPipe Hand Detection

4.3.2 Fungsi Detect Landmarks

Mendeteksi dan menggambar 21 landmark tangan dari frame.

```python
def detect(self, frame):
    """
    Input: frame BGR (opencv)
    Output:
        - landmarks: list berisi 21 titik (x, y)
        - frame: frame dengan landmark digambar
    """

    # Convert BGR → RGB
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Proses tangan
    results = self.hands.process(rgb)

    landmarks = None

    # Jika ada tangan
    if results.multi_hand_landmarks:
        handLms = results.multi_hand_landmarks[0]

        # Draw landmarks di frame
        self.drawer.draw_landmarks(
            frame,
            handLms,
            mp.solutions.hands.HAND_CONNECTIONS
        )

        # Extract koordinat setiap titik
        h, w, _ = frame.shape
        landmarks = [(int(lm.x * w), int(lm.y * h)) for lm in handLms.landmark]

    return landmarks, frame
```

Kode 8. Fungsi Detect Landmarks

4.4 Modul Gestur Mapper

Modul ini berfungsi untuk memetakan landmark tangan ke gesture tertentu menggunakan klasifikasi berbasis rule-based atau machine learning.

```
class GestureMapper:
    def map(self, landmarks):
        """
        Input: landmarks (list dari 21 titik tangan)
        Output: 'A', 'B', 'C', 'D', atau None
        """

        if landmarks is None:
            return None

        # Ambil titik penting
        thumb_tip  = landmarks[4]   # ujung jempol
        index_tip  = landmarks[8]   # ujung telunjuk
        middle_tip = landmarks[12]  # ujung jari tengah
        ring_tip   = landmarks[16]
        pinky_tip  = landmarks[20]
```

Kode 9. Inisiai modul gestur mapper

4.4.1 Fitur Utama

- Menghitung jarak dan sudut antar landmark
- Mendeteksi posisi jari terbuka atau tertutup
- Mendeteksi gesture khusus (Thumbs Up, Pointing, V-Sign, dll)
- Memetakan gesture ke jawaban (A, B, C, D)

4.4.2 Struktur Klasifikasi Gesture

```
class GestureMapper:
    def __init__(self):
        # Dictionary untuk menyimpan hasil deteksi gesture
        self.last_gesture = None
        self.confidence = 0.0

    def classify(self, landmarks):
        """
        Input: landmarks (21 titik)
        Output: gesture_name (string), confidence (float)
        """
        if landmarks is None or len(landmarks) < 21:
            return None, 0.0

        # Proses klasifikasi gesture
        # ... logika deteksi gesture ...

        return gesture_name, confidence
```

Kode 9: Struktur Klasifikasi Gesture

4.5 Modul game_manager.py

Modul ini mengatur logika permainan, termasuk pengelolaan state, skor, dan timer.

4.5.1 Enum Game Phase

```python
from enum import Enum

class GamePhase(Enum):
    IDLE = 0
    WAITING_ANSWER = 1
    SHOWING_RESULT = 2
    GAME_OVER = 3
```

Kode 10: Game Phase Enum

4.5.2 Inisialisasi Game Manager

```python
class GameManager:
    def __init__(self, questions):
        """
        Initialize game manager

        Args:
            questions: List of question dicts
        """
        self.questions = questions if questions else [
        self.current_question_idx = 0
        self.score = 0
        self.answered_count = 0
        self.phase = GamePhase.IDLE

        # Konfigurasi timer per soal
        self.question_duration = 10.0  # detik per soa
        self.current_question_start_time = None

        # Shuffle questions
        if self.questions:
            random.shuffle(self.questions)
```

Kode 11: Inisialisasi Game Manager

4.5.3 Fungsi Start Game

```python
def start_game(self):
    """Start the game"""
    self.current_question_idx = 0
    self.score = 0
    self.answered_count = 0
    self.phase = GamePhase.WAITING_ANSWER

    if self.questions:
        random.shuffle(self.questions)
        self.start_question()
```

Kode 12: Fungsi Start Game

4.5.4 Fungsi Check Answer

```
def check_answer(self, player_answer):
    """
    Periksa jawaban pemain

    Args:
        player_answer: string (A/B/C/D)

    Returns:
        bool: True jika benar, False jika salah
    """
    if self.current_question is None:
        return False

    is_correct = (player_answer == self.current_question['answer'])

    if is_correct:
        self.score += 1
        self.phase = GamePhase.SHOWING_RESULT

    self.answered_count += 1
    return is_correct
```

Kode 13: Fungsi Check Answer

4.6 Modul audio_player.py

Modul ini mengatur pemutaran audio untuk feedback pertanyaan dan jawaban.

4.6.1 Inisialisasi Audio Player

```
import sounddevice as sd
import soundfile as sf

class AudioPlayer:
    def __init__(self):
        """Initialize audio player"""
        self.is_playing = False

    def play_audio(self, audio_path):
        """
        Play audio file

        Args:
            audio_path: string path ke file audio
        """
        try:
            data, samplerate = sf.read(audio_path)
            self.is_playing = True
            sd.play(data, samplerate)
            sd.wait()
            self.is_playing = False
        except Exception as e:
            print(f"[ERROR] Gagal play audio: {e}")
```

Kode 14: Inisialisasi dan Fungsi Audio Player

4.7 Modul tampilan.py (GameUI)

Modul ini menangani render UI menggunakan pygame.

4.7.1 Enum Game State

```python
from enum import Enum

class GameState(Enum):
    MENU = 0
    PLAYING = 1
    RESULT = 2
    GAME_OVER = 3
```

Kode 15: Game State Enum

4.7.2 Inisialisasi GameUI

```python
class GameUI:
    def __init__(self, width=None, height=None):
        """
        Initialize game UI dengan responsive resolution

        Args:
            width: lebar screen (default: 1280)
            height: tinggi screen (default: 800)
        """
        # Set default resolution yang lebih compatible untuk berbagai device
        self.width = width if width else 1280
        self.height = height if height else 800

        # Gunakan window yang dapat di-resize untuk compatibility lebih baik
        self.screen = pygame.display.set_mode((self.width, self.height), pygame.RESIZABLE)
        pygame.display.set_caption("CineTune - Tebak Film Lewat Gesture")

        # Load fonts dengan scaling berbasis height
        self.font_title = pygame.font.Font(None, int(self.height * 0.08))
        self.font_text = pygame.font.Font(None, int(self.height * 0.06))
        self.font_small = pygame.font.Font(None, int(self.height * 0.04))

        # Initialize state
        self.state = GameState.MENU
```

Kode 16: Inisialisasi GameUI (Responsive)

4.7.2.1 Optimasi Resolusi untuk Berbagai Device

Untuk memastikan UI tidak terpotong di berbagai device, gunakan:

```python
import pygame
pygame.init()
info = pygame.display.Info()
screen_width = info.current_w
screen_height = info.current_h

# Tentukan resolusi optimal (85% dari resolusi layar)
optimal_width = int(screen_width * 0.85)
optimal_height = int(screen_height * 0.85)

# Batasi agar tidak terlalu besar (max 1920x1080)
max_width = min(optimal_width, 1920)
max_height = min(optimal_height, 1080)

# Batasi agar tidak terlalu kecil (min 1024x768)
min_width = max(max_width, 1024)
min_height = max(max_height, 768)

ui = GameUI(width=min_width, height=min_height)
```

Kode 16a: Deteksi dan Optimasi Resolusi Otomatis

4.7.3 Fungsi Render Pertanyaan

```python
def render_question(self, screen, image_surface, options, timer_text, score_text):
    """
    Render tampilan pertanyaan dengan layout responsive

    Args:
        screen: pygame surface
        image_surface: gambar film
        options: dict {A: text, B: text, C: text, D: text}
        timer_text: teks timer
        score_text: teks skor
    """
    screen.fill((0, 0, 0))

    # Display image dengan padding responsive
    padding = int(self.width * 0.04)
    if image_surface:
        screen.blit(image_surface, (padding, padding))

    # Display options dengan spacing responsive
    y_offset = int(self.height * 0.65)
    option_spacing = int(self.height * 0.08)
    for key, text in options.items():
        option_text = self.font_text.render(f"{key}: {text}", True, (255, 255, 255))
        screen.blit(option_text, (padding, y_offset))
        y_offset += option_spacing

    # Display timer and score dengan positioning responsive
    timer_surf = self.font_small.render(timer_text, True, (255, 200, 0))
    score_surf = self.font_small.render(score_text, True, (0, 255, 0))
    screen.blit(timer_surf, (1000, 50))
    screen.blit(score_surf, (1000, 100))

    pygame.display.flip()
```

Kode 17: Fungsi Render Pertanyaan

4.8 Modul frame_utama.py (CineTuneApp)

Modul utama yang mengintegrasikan semua komponen.

4.8.1 Inisialisasi Aplikasi

```python
class CineTuneApp:
    def __init__(self):
        """Initialize the application"""
        self.base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

        # Initialize pygame
        pygame.init()

        # Initialize components
        print("[INIT] Loading data...")
        self.questions = load_questions()
        self.gesture_map = load_gesture_map()

        # Initialize managers
        self.game_manager = GameManager(self.questions)
        self.audio_player = AudioPlayer()
        self.ui = GameUI()

        # Initialize vision components
        print("[INIT] Initializing gesture detection...")
        self.gesture_detector = GestureDetector()
        self.gesture_mapper = GestureMapper()
        self.cap = cv2.VideoCapture(0)
```

Kode 18: Inisialisasi CineTuneApp

4.8.2 Main Game Loop

- Aplikasi menjalankan loop utama yang:
- Membaca frame dari kamera
- Mendeteksi gesture tangan
- Memperbarui UI dengan frame dan informasi pertanyaan
- Mengevaluasi jawaban berdasarkan gesture yang terdeteksi

Menampilkan hasil dan melanjutkan ke pertanyaan berikutnya

```python
def run(self):
    """Main game loop"""
    clock = pygame.time.Clock()
    running = True

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

        # Capture frame dari kamera
        ret, frame = self.cap.read()
        if not ret:
            continue

        # Deteksi gesture
        landmarks, frame_with_landmarks = self.gesture_detector.detect(frame)
        gesture, confidence = self.gesture_mapper.classify(landmarks)

        # Update UI
        # ... render frame, pertanyaan, dan feedback ...

        clock.tick(30)  # 30 FPS
```

Kode 19: Main Game Loop

4.8.3 Cleanup Resources

```python
def cleanup(self):
    """Release resources"""
    if self.cap is not None:
        self.cap.release()
    cv2.destroyAllWindows()
    pygame.quit()
    print("[INFO] Application cleanup complete")
```

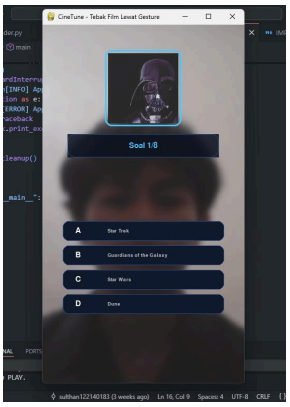
Kode 20: Cleanup Resources

4.9 Struktur Folder Proyek

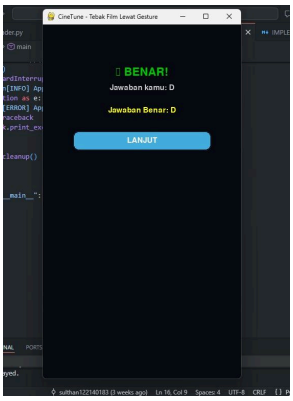
```
CineTune/
├── src/
│   ├── main.py              # Entry point aplikasi
│   ├── test.py              # Testing scripts
│   ├── core/
│   │   ├── audio_player.py     # Pengelolaan audio
│   │   ├── data_loader.py      # Load data dari CSV
│   │   ├── game_manager.py     # Logika game
│   │   └── __pycache__/
│   ├── ui/
│   │   ├── frame_utama.py      # Main app class
│   │   ├── tampilan.py         # UI rendering
│   │   └── __pycache__/
│   └── vision/
│       ├── gesture_detector.py # MediaPipe detection
│       ├── gesture_mapper.py   # Gesture classification
│       └── __pycache__/
├── assets/
│   ├── audio/               # File audio pertanyaan
│   └── images/              # Gambar film untuk pertanyaan
├── data/
│   ├── gestures.csv         # Pemetaan gesture ke jawaban
│   └── questions.csv        # Data pertanyaan
├── requirements.txt         # Dependensi Python
├── README.md
```

5 Hasil

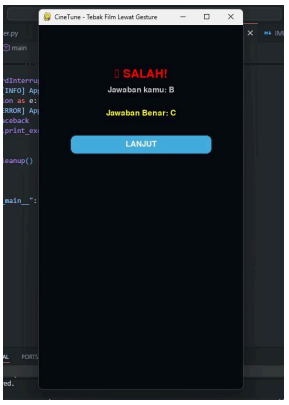
(a) Tampilan Awal(Start Screen)



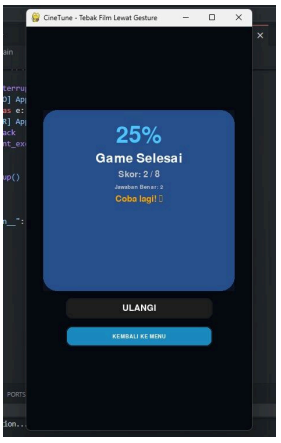
(b) Tampilan Soal(Game Screen)



(c) Tampilan Benar(Correct Answer Screen)



(d) Tampilan Salah(Wrong Answer Screen)



(e) Tampilan Selesai(Game Over Screen)

Implementasi aplikasi CineTune - Tebak Film Lewat Gesture berhasil dijalankan dengan mengintegrasikan input kamera secara real-time ke dalam antarmuka grafis berbasis Pygame. Logika utama sistem memanfaatkan modul gesture_detector.py dan gesture_mapper.py untuk menangkap titik koordinat tangan menggunakan teknologi MediaPipe Hands, yang kemudian diterjemahkan menjadi opsi jawaban (A, B, C, D) untuk menggantikan input keyboard konvensional. Saat aplikasi dimulai, pengguna disuguhkan instruksi visual mengenai pemetaan gesture, yaitu Thumb Up (Jempol Naik) untuk opsi A, Open Palm (Telapak Terbuka) untuk opsi B, Two Fingers (Dua Jari) untuk opsi C, dan Fist (Genggaman) untuk opsi D.

Dalam alur permainannya, antarmuka menampilkan pertanyaan berupa cuplikan gambar film dengan latar belakang feed kamera pengguna. Sistem memberikan respons umpan balik secara langsung berdasarkan hasil analisis gesture; jika gerakan tangan pengguna sesuai dengan kunci jawaban, layar akan menampilkan status "BENAR!" berwarna hijau, sedangkan jika tidak sesuai, sistem menampilkan peringatan "SALAH!" berwarna merah beserta kunci jawaban yang seharusnya. Setelah seluruh pertanyaan selesai dijawab, aplikasi akan menutup sesi dengan menampilkan layar "Game Selesai" yang memuat akumulasi skor akhir pengguna.

Meskipun sistem dapat berjalan sesuai logika permainan, terdapat beberapa kendala teknis yang ditemukan selama pengujian, terutama terkait akurasi deteksi model. Sistem terkadang mengalami kesalahan interpretasi (misclassification) atau "salah tangkap" pada gesture yang memiliki kemiripan bentuk geometris, seperti sulitnya membedakan antara gesture Two Fingers (V-Sign) dengan Open Palm apabila jari-jari pengguna tidak terentang atau tertutup dengan sempurna. Selain itu, performa deteksi landmark tangan sangat sensitif terhadap kondisi pencahayaan; pada lingkungan dengan cahaya rendah, respon deteksi menjadi tidak stabil, sehingga pengguna sangat disarankan untuk melakukan permainan di ruangan dengan pencahayaan yang terang dan kontras yang cukup agar gesture dapat dikenali dengan presisi.

6 Kesimpulan

Proyek CineTune berhasil mengimplementasikan sistem deteksi gesture tangan berbasis computer vision yang dapat digunakan sebagai media interaktif dalam permainan tebak film. Melalui pemanfaatan OpenCV dan MediaPipe, aplikasi mampu mendeteksi posisi jari dan menganalisis gesture pengguna secara real–time dengan tingkat responsivitas yang baik.

Sistem gesture recognition yang dibangun dapat mengidentifikasi empat pola gesture berbeda (A, B, C, dan D) yang masing-masing mewakili pilihan jawaban. Proses klasifikasi gesture berjalan secara konsisten berkat pemrosesan landmark dari MediaPipe serta logika pemetaan yang diatur dalam modul GestureMapper.

Dari sisi antarmuka, penggunaan Tkinter membuat aplikasi dapat dioperasikan dengan mudah, ringan, dan user-friendly. Integrasi antara modul kamera, deteksi gesture, dan tampilan GUI juga berjalan stabil sesuai kebutuhan aplikasi kuis interaktif.

Secara keseluruhan, proyek ini menunjukkan bahwa teknologi visi komputer dapat digunakan tidak hanya untuk keperluan teknis, tetapi juga sebagai media hiburan dan pembelajaran yang menarik. Sistem ini dapat dikembangkan lebih lanjut dengan menambah variasi gesture, meningkatkan akurasi klasifikasi, serta memperluas fitur permainan untuk menghasilkan pengalaman pengguna yang lebih baik.

.

References

[1] Budiman, S. N., Lestanti, S., Evvandri, S. M., & Putri, R. K. (2022). Pengenalan Gestur Gerakan Jari Untuk Mengontrol Volume Di Komputer Menggunakan Library Opencv Dan Mediapipe. *Antivirus: Jurnal Ilmiah Teknik Informatika*, *16*(2), 223-232.