Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **M. Fasya Atthoriq, Hayyatul Fajri (122140107, 122140103)** Tugas Ke: **Tugas Besar**
Mata Kuliah: **Sistem Teknologi Multimedia** Tanggal: 12 Desember 2025

1 Pendahuluan

Interaksi tanpa sentuhan telah menjadi area riset dan aplikasi yang berkembang pesat saat ini, khususnya untuk aksesibilitas dan pengalaman pengguna baru. Dalam tugas besar ini kami mengembangkan *Blink Beat*, yaitu sebuah aplikasi yang menerjemahkan kedipan mata menjadi pemicu suara drum. Tujuan proyek adalah membuktikan bahwa sinyal sederhana dari wajah (kedipan) dapat dipetakan menjadi kontrol musik real-time dengan respons yang cukup cepat dan andal.

2 Tujuan

Tujuan tugas besar ini adalah:

- Merancang dan mengimplementasikan sistem deteksi kedipan mata secara real-time.
- Menghubungkan kedipan dengan pemutaran suara drum (kick, snare, cymbal).
- Menyajikan antarmuka visual sederhana (kotak indikator mata dan visualizer partikel).
- Mengevaluasi respons sistem terhadap kedipan pada kondisi pencahayaan normal.

3 Dasar Teori

3.1 Face Landmark dan MediaPipe Face Mesh

MediaPipe Face Mesh menyediakan 468 titik landmark pada wajah yang memungkinkan ekstraksi posisi mata secara presisi. Landmark ini mempermudah perhitungan jarak antara titik-titik kunci yang dipakai dalam formula EAR.

3.2 Eye Aspect Ratio (EAR)

EAR (Eye Aspect Ratio) adalah rasio antara jarak vertikal dan jarak horizontal mata yang digunakan untuk mengukur tingkat keterbukaan mata. Secara sederhana, EAR dihitung dari beberapa pasangan titik pada kontur mata; ketika mata menutup, nilai EAR turun drastis sehingga dapat digunakan untuk mendeteksi kedipan.

3.3 Pygame untuk Audio

Pygame menyediakan modul `pygame.mixer` untuk pemutaran audio sederhana. Dengan memuat file WAV (PCM 16-bit, 44.1 kHz) dan memanggil metode `Sound.play()`, kita dapat memainkan suara secara non-blocking dalam loop real-time.

4 Perancangan Sistem

4.1 Arsitektur

Sistem terdiri dari modul-modul utama:

- **Acquisition** — Webcam + OpenCV menangkap frame video.
- **Detection** — MediaPipe Face Mesh mengekstrak landmark wajah; modul blink detector menghitung EAR dan menentukan event kedipan.
- **Audio** — Pygame memuat dan memainkan file suara ketika event terdeteksi.
- **Visualizer** — Kotak kecil di sekitar mata dan sistem partikel yang memunculkan animasi saat suara diputar.

4.2 Flowchart Singkat

1. Inisialisasi webcam, Pygame mixer, dan MediaPipe.
2. Loop menangkap frame, memproses landmark, menghitung EAR.
3. Jika EAR di bawah threshold selama beberapa frame, set event blink.
4. Trigger audio sesuai jenis blink (kiri → snare, kanan → cymbal, kedua → kick) dan spawn partikel.
5. Tampilkan frame dengan indikator, tunggu input 'q' untuk keluar.

5 Implementasi

Berikut potongan kode penting yang merepresentasikan inti dari implementasi. Silakan lihat file proyek lengkap untuk kode penuh.

5.1 Deteksi EAR dan Blink (potongan)

```python
# src/blink_detector.py (intisari)
import mediapipe as mp
import cv2
import numpy as np

class BlinkDetector:
    def __init__(self):
        self.face_mesh = mp.solutions.face_mesh.FaceMesh(
            max_num_faces=1,
            refine_landmarks=True,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

        self.LEFT_EYE = [33, 160, 158, 133, 153, 144]
        self.RIGHT_EYE = [263, 387, 385, 362, 380, 373]
        self.EAR_THRESHOLD = 0.25

    def get_EAR(self, eye_points):
        v1 = np.linalg.norm(eye_points[1] - eye_points[5])
        v2 = np.linalg.norm(eye_points[2] - eye_points[4])
        h = np.linalg.norm(eye_points[0] - eye_points[3])
        return (v1 + v2) / (2.0 * h)
```

Kode 1: Fungsi deteksi blink (ringkasan)

5.2 Audio Manager (potongan)

```python
# src/audio_manager.py
import pygame
import os

class DrumAudio:
    def __init__(self):
        pygame.mixer.init()
        base_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))
        sounds_dir = os.path.join(base_dir, "assets", "sounds")

        self.sounds = {
            "kick": pygame.mixer.Sound(os.path.join(sounds_dir,"kick.wav")),
            "snare": pygame.mixer.Sound(os.path.join(sounds_dir,"snare.wav")),
            "cymbal": pygame.mixer.Sound(os.path.join(sounds_dir,"cymbal.wav")),
        }

    def play(self, key):
        if self.sounds.get(key):
            self.sounds[key].play()
```

Kode 2: Pemuat audio menggunakan pygame

5.3 Main Loop (ringkasan)

```python
# Kick  : kedua mata tertutup
# Snare : mata kiri
# Cymbal: mata kanan

if both_blink:
    audio.play("kick")
    particles.spawn(400, 300)
elif left_blink:
    audio.play("snare")
    particles.spawn(300, 300)
elif right_blink:
    audio.play("cymbal")
    particles.spawn(500, 300)
```

Kode 3: Main loop: panggil detektor, trigger suara, render

6 Pengujian dan Hasil

6.1 Setup Pengujian

Pengujian dilakukan pada laptop dengan webcam internal, pencahayaan ruangan kantor (tidak gelap), dan file suara WAV PCM 16-bit 44.1kHz. Kami menguji tiga skenario: kedipan mata kiri, kedipan mata kanan, dan kedipan kedua mata bersamaan.

6.2 Hasil

- **Deteksi EAR:** Nilai EAR saat mata terbuka berkisar 0.28–0.34; saat mata menutup turun ke kisaran 0.15–0.22 (bergantung posisi kamera dan pencahayaan).
- **Respons audio:** Sistem memicu suara sesuai aturan; latency mendekati real-time (tergantung FPS webcam, biasanya 25–30 fps).

- **Stabilitas:** Dengan konfigurasi threshold 0.22 dan konfirmasi 2–3 frame, false positive berkurang signifikan.

6.3 Catatan Praktis

- Pastikan file audio berada pada folder `assets/sounds/` dan bernama `kick.wav`, `snare.wav`, `cymbal.wav`.
- Jika suara terdengar aneh, konversi file ke PCM 16-bit 44.1kHz (misal menggunakan Audacity atau `pydub` + `ffmpeg`).
- Jika EAR tidak turun cukup saat menutup mata, sesuaikan threshold (contoh 0.20–0.26) dan jumlah frame konfirmasi.

7 Pembahasan

Sistem membuktikan bahwa sinyal sederhana (EAR) cukup untuk membuat interaksi musik yang intuitif. Keterbatasan yang ditemui antara lain:

- Sensitivitas terhadap sudut wajah: sudut tidak frontal menurunkan akurasi landmark.
- Ketergantungan pada pencahayaan: ruangan gelap menurunkan performa deteksi.
- Delay yang muncul karena FPS kamera dan proses per-frame (walau biasanya masih terasa real-time untuk aplikasi interaktif ringan).

8 Kesimpulan

Blink Beat berhasil mengubah kedipan mata menjadi kontrol drum realtime menggunakan kombinasi MediaPipe, OpenCV, dan Pygame. EAR terbukti efektif sebagai indikator kedipan selama threshold dan mekanisme konfirmasi frame diatur dengan tepat.

9 Saran

Untuk pengembangan selanjutnya:

- Tambahkan kalibrasi EAR otomatis saat inisialisasi, agar pengguna tidak perlu mengubah threshold manual.
- Tambahkan opsi mapping suara (misal tukar kanan/kiri) dan mode drum kit yang berbeda.
- Optimalkan rendering partikel menggunakan GPU atau library khusus agar visual lebih halus pada perangkat dengan spesifikasi rendah.

10 Lampiran

10.1 Struktur Folder Proyek (disarankan)

```
blink-beat-main/
 main.py
 requirements.txt
 assets/
    sounds/
```

```
        kick.wav
        snare.wav
        cymbal.wav
src/
    blink_detector.py
    audio_manager.py
    particle_system.py
```

10.2 Contoh Perintah Instalasi

```
python -m venv env
env\Scripts\activate         # Windows
pip install -r requirements.txt
# requirements.txt minimal:
# opencv-python
# mediapipe
# pygame
# numpy
```

11 Referensi dan Daftar Pustaka

1. T. Soukupova and J. Cech, "Real-Time Eye Blink Detection using Facial Landmarks," in *Proc. Computer Vision Winter Workshop*, 2016.
2. Google, "MediaPipe Face Mesh," https://developers.google.com/mediapipe/solutions/vision/face_mesh.
3. Pygame Documentation, https://www.pygame.org/docs/.
4. OpenCV Documentation, https://docs.opencv.org/.
5. Freesound.org — sumber sampel audio gratis.

References