Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Andry Herwansyah, Justin Halim,**
**Dandy Arkandhiya Putra (119140078, 120140185, 120140203)** Tugas Ke: Tugas Besar
Mata Kuliah: **Teknologi Multimedia (IF4021)** Tanggal: 24 Desember 2024

# Bab 1 Pendahuluan

## 1.1 Latar Belakang

Dalam era teknologi yang semakin berkembang, inovasi dalam bidang multimedia menjadi salah satu fokus utama pengembangan teknologi. Salah satu implementasi inovasi tersebut adalah pengolahan audio-visual secara real-time yang memungkinkan manipulasi suara dan wajah secara interaktif. Proyek ini bertujuan untuk menciptakan sebuah aplikasi berbasis Python dengan nama **I am Thanos**, yang dapat memodifikasi ekspresi wajah dan suara pengguna secara langsung melalui kamera dan mikrofon.

Proyek ini memanfaatkan teknologi seperti OpenCV untuk pengolahan citra, MediaPipe untuk deteksi fitur wajah, NumPy untuk manipulasi data, dan pyaudio untuk pengolahan audio. Dengan teknologi tersebut, aplikasi ini mampu mengubah ekspresi wajah marah pengguna menjadi menyerupai karakter Thanos, dengan warna kulit ungu, wajah lebih lebar, serta suara yang lebih berat. Proyek ini diharapkan dapat menjadi dasar untuk aplikasi hiburan dan edukasi yang lebih kompleks di masa depan.

## 1.2 Deskripsi

Tujuan dari proyek **I am Thanos** adalah:

1. Mengembangkan aplikasi real-time yang mampu memodifikasi wajah pengguna sehingga menyerupai karakter Thanos, dengan perubahan warna kulit menjadi ungu dan wajah yang lebih lebar.
2. Mengimplementasikan algoritma pengolahan audio untuk mengubah suara pengguna menjadi lebih berat secara real-time.
3. Mengintegrasikan teknologi OpenCV, MediaPipe, NumPy, dan pyaudio dalam satu aplikasi yang fungsional.

# Bab 2 - Manipulasi Suara

## Pendahuluan

Manipulasi suara merupakan proses pengubahan karakteristik suara asli menjadi bentuk suara yang berbeda. Dalam proyek ini, manipulasi dilakukan untuk membuat suara pengguna terdengar lebih berat, menyerupai suara karakter *Thanos*. Proses ini melibatkan teknik pemrosesan sinyal digital menggunakan pustaka seperti pyaudio untuk ekstraksi fitur audio dan pengubahan frekuensi.

## Implementasi

Berikut adalah langkah-langkah implementasi manipulasi suara:

1. Konfigurasi audio dengan format 16-bit PCM mono, sampling rate 44,100 Hz, dan ukuran buffer 2,048 sampel.
2. Menerapkan low-pass filter untuk menghilangkan frekuensi tinggi yang tidak diinginkan.
3. Melakukan pitch shifting untuk menurunkan frekuensi suara.
4. Normalisasi amplitudo untuk memastikan suara tetap dalam rentang yang dapat didengar.
5. Menggunakan noise gate untuk menghilangkan noise yang tidak diinginkan.

```python
import cv2
import numpy as np
import mediapipe as mp
import pyaudio
from scipy.signal import butter, lfilter
import threading

# Konfigurasi audio
CHUNK = 2048  # Ukuran buffer: 2,048 sampel
FORMAT = pyaudio.paInt16  # Format audio: 16-bit PCM
CHANNELS = 1  # Mono
RATE = 44100  # Sampling rate: 44,100 Hz
NOISE_THRESHOLD = 300  # Ambang batas noise lebih rendah untuk responsivitas lebih baik
TARGET_PEAK = 25000    # Disesuaikan untuk audio yang lebih jelas

# Fungsi low-pass filter
def low_pass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype="low", analog=False)
    return lfilter(b, a, data)

# Fungsi untuk pitch shifting dengan interpolasi
def pitch_shift(data, pitch_factor):
    indices = np.arange(0, len(data), pitch_factor)
    indices = indices[indices < len(data)]
    return np.interp(indices, np.arange(len(data)), data).astype(data.dtype)

# Normalisasi amplitudo
def normalize_audio(data, target_peak=TARGET_PEAK):
    max_amplitude = np.max(np.abs(data))
    if max_amplitude == 0:
        return data
    scaling_factor = min(1.0, target_peak / max_amplitude)
    return np.clip(data * scaling_factor, -32768, 32767).astype(np.int16)

```

```python
# Noise Gate untuk menghilangkan noise
def noise_gate(data, threshold=NOISE_THRESHOLD):
    if np.max(np.abs(data)) < threshold:
        return np.zeros_like(data)
    return data

# Efek suara Thanos
def thanos_effect(audio_chunk, pitch_factor=0.85, resonance_factor=1.2, cutoff_freq=1000):
    gated_audio = noise_gate(audio_chunk)
    pitched_audio = pitch_shift(gated_audio, pitch_factor)
    filtered_audio = low_pass_filter(pitched_audio, cutoff=cutoff_freq, fs=RATE)
    resonant_audio = filtered_audio * resonance_factor
    return normalize_audio(resonant_audio)

# Fungsi utama untuk filter suara real-time
def realtime_voice_filter():
    audio_interface = pyaudio.PyAudio()
    stream_input = audio_interface.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True,
    frames_per_buffer=CHUNK)
    stream_output = audio_interface.open(format=FORMAT, channels=CHANNELS, rate=RATE, output=True,
    frames_per_buffer=CHUNK)

    print("Filter suara Thanos aktif. Silahkan bicara ke mikrofon...")
    try:
        while True:
            input_data = stream_input.read(CHUNK, exception_on_overflow=False)
            audio_chunk = np.frombuffer(input_data, dtype=np.int16)
            modified_audio = thanos_effect(audio_chunk)
            stream_output.write(modified_audio.tobytes())
    except KeyboardInterrupt:
        pass
    finally:
        stream_input.stop_stream()
        stream_input.close()
        stream_output.stop_stream()
        stream_output.close()
        audio_interface.terminate()
```

Kode 1: Kode Manipulasi Suara

# Bab 3 - Manipulasi Bentuk Wajah

## Pendahuluan

Manipulasi bentuk wajah dilakukan untuk mengubah proporsi wajah pengguna agar menyerupai karakter *Thanos*. Teknik ini menggunakan pustaka `MediaPipe` untuk pendeteksian fitur wajah, kemudian dilakukan transformasi geometri untuk memperlebar wajah.

## Implementasi

Langkah-langkah implementasi manipulasi bentuk wajah adalah:

1. Inisialisasi MediaPipe Face Mesh untuk mendeteksi fitur wajah.
2. Deteksi titik-titik fitur wajah dan buat mask untuk area wajah.
3. Terapkan warna ungu pada wajah dan tambahkan garis tepi wajah.

```python
# Inisialisasi MediaPipe Face Mesh
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(max_num_faces=1, min_detection_confidence=0.7,
    min_tracking_confidence=0.7, refine_landmarks=True)

# Fungsi untuk mengubah warna wajah menjadi ungu (warna khas Thanos)
def apply_thanos_color(image, mask):
    purple_tint = image.copy()
    purple_tint[:, :, 0] = np.clip(purple_tint[:, :, 0] * 1.5, 0, 255)  # Biru
    purple_tint[:, :, 1] = np.clip(purple_tint[:, :, 1] * 0.6, 0, 255)  # Hijau
    purple_tint[:, :, 2] = np.clip(purple_tint[:, :, 2] * 1.8, 0, 255)  # Merah
    return cv2.addWeighted(image, 0.3, purple_tint, 0.7, 0, dtype=cv2.CV_8U)

# Fungsi untuk mendeteksi wajah dan menerapkan filter Thanos
def process_frame(image):
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(rgb_image)

    if results.multi_face_landmarks:
        for face_landmarks in results.multi_face_landmarks:
            h, w, _ = image.shape
            face_points = [(int(landmark.x * w), int(landmark.y * h)) for landmark in
    face_landmarks.landmark]

            # Buat mask untuk area wajah
            mask = np.zeros((h, w), dtype=np.uint8)
            cv2.fillConvexPoly(mask, np.array(face_points, dtype=np.int32), 255)
            mask_3d = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)

            # Terapkan warna ungu pada wajah
            image = np.where(mask_3d > 0, apply_thanos_color(image, mask_3d), image)

            # Tambahkan garis tepi wajah
            hull = cv2.convexHull(np.array(face_points, dtype=np.int32))
            cv2.polylines(image, [hull], True, (145, 0, 145), 2)

    return image
```

Kode 2: Kode Manipulasi Bentuk Wajah

# Bab 4 - Implementasi Real-Time

## Pendahuluan

Implementasi real-time dilakukan untuk menggabungkan manipulasi suara dan wajah secara langsung melalui kamera dan mikrofon. Teknik ini menggunakan pustaka `OpenCV` untuk pengolahan citra dan `pyaudio` untuk pengolahan audio.

## Implementasi

Langkah-langkah implementasi real-time adalah:

1. Inisialisasi kamera dan mikrofon.
2. Jalankan filter suara secara paralel menggunakan threading.
3. Proses frame dari kamera untuk mendeteksi dan memodifikasi wajah.
4. Tampilkan hasil modifikasi wajah dan suara secara real-time.

```python
# Fungsi utama untuk filter wajah dan suara secara real-time
def main():
    threading.Thread(target=realtime_voice_filter, daemon=True).start()

    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("Error: Tidak dapat membuka kamera.")
        return

    print("Filter wajah dan suara Thanos aktif.")
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            print("Error: Tidak dapat membaca frame dari kamera.")
            break

        # Proses frame untuk filter wajah
        frame = process_frame(frame)

        # Tampilkan hasil
        cv2.imshow("Thanos Face Effect", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

Kode 3: Kode Implementasi Real-Time

## References