Program Studi Teknik Informatika
Institut Teknologi Sumatera

Anggota 1: **Wuri Wilatiningsih (121140167)**
Anggota 2: **Andrean Syahrezi (121140169)**
Mata Kuliah: **Sistem/Teknologi Multimedia (IF4021)**

Tugas Ke: **Tugas Besar**
Tanggal: **11-22-2024**

Karaoke Face Animation

1 Pendahuluan

Karaoke Face Animation adalah sebuah inovasi dalam teknologi augmented reality (AR) yang memadukan animasi wajah dengan elemen karaoke. Teknologi ini memungkinkan pengguna untuk merasakan pengalaman bernyanyi secara interaktif dengan memanfaatkan fitur pelacakan wajah dan sinkronisasi audio-visual. Efek ini sangat populer di platform media sosial seperti Instagram dan TikTok, di mana pengguna dapat mengekspresikan kreativitas mereka melalui filter-filter yang unik dan menarik.

Perkembangan teknologi AR telah membawa perubahan signifikan dalam cara orang berinteraksi dengan perangkat digital. Karaoke Face Animation memanfaatkan teknologi pelacakan wajah (face tracking) untuk mendeteksi elemen-elemen wajah pengguna secara real-time, seperti bibir, mata, dan ekspresi wajah. Selanjutnya, animasi wajah ditambahkan, seperti gerakan bibir yang seolah-olah mengikuti lirik lagu, serta teks karaoke yang bergerak sesuai dengan tempo musik. Hal ini menciptakan pengalaman menyanyi yang tidak hanya menyenangkan tetapi juga sangat interaktif.

Teknologi yang digunakan dalam pembuatan Karaoke Face Animation meliputi penggunaan perangkat lunak pengembang AR seperti Spark AR Studio atau Effect House, yang mendukung integrasi elemen visual, audio, dan scripting. Selain itu, pengolahan data audio dan teks secara sinkron menjadi salah satu aspek penting dalam memastikan filter ini berjalan dengan baik.

Proyek ini bertujuan untuk mengembangkan filter Karaoke Face Animation yang dapat memberikan pengalaman pengguna yang interaktif dan menghibur, sekaligus menunjukkan potensi teknologi AR dalam menciptakan konten kreatif. Dengan implementasi yang tepat, filter ini dapat menjadi salah satu alat hiburan digital yang inovatif dan bermanfaat, baik untuk individu maupun untuk keperluan promosi di industri kreatif.

Dalam laporan ini, akan dijelaskan secara rinci konsep, teknologi yang digunakan, proses pengembangan, serta evaluasi kinerja dari Karaoke Face Animation sebagai salah satu implementasi teknologi AR modern.

2 Alat dan Cara Kerja

Dalam pengembangan proyek ini menggunakan beberapa alat dan bahan utama untuk sistem karaoke face animation. Bahasa pemrograman yang digunakan adalah Python dengan dukungan library **pyaudio, numpy, cv2, pygame** untuk menginput audio, manipulasi array, pengelolaan gambar dan audio. Perangkat keras yang digunakan meliputi laptop dengan menggunakan camera bawaan laptop sebagai input visual. .

3 Penjelasan Kode Program

3.1 Import Library

Library yang diperlukan dalam program ini diantaranya cv2, mediapipe, dan numpy.

```

    import cv2
    import threading
    import wave
    import pyaudio
    import numpy as np
    from tkinter import Tk, Label, Button, Frame
    from PIL import Image, ImageTk
    from audio_recorder import AudioRecorder
    from music_player import MusicPlayer
    from face_animation import animate_cat_face, animate_angry_face, animate_girl_face

```

Kode 3.1: Library yang Digunakan

Penjelasan:

- cv2 (OpenCV): Digunakan untuk memproses citra/video secara real-time, termasuk menangkap gambar dari webcam dan menampilkan hasil.
- threading: Digunakan untuk menjalankan beberapa tugas secara paralel menggunakan thread
- wave: Digunakan untuk membaca dan menulis file audio dalam format WAV.
- pyaudio: Digunakan untuk merekam dan memutar audio secara realtime.
- numpy as np: Digunakan untuk manipulasiarray.
- tkinter import Tk, Label, Button, Frame: Digunakan untuk membuat antarmuka grafis.
- PIL import Image, ImageTk: Digunakan untuk manipulasi gambar dan menampilkannya di GUI Tkinter.
- audio.recorder import AudioRecorder: Digunakan untuk merekam audio.
- music.player import MusicPlayer: Digunakan untuk memutar musik
- face.animation import animate.cat.face, animate.angry.face, animate.girl.face: Untuk membuat animasi wajah(seperti animasi wajah kucing, wajah marah, wajah wanita).

3.2 Memuat File Audio

```
        class AudioRecorder:
    """Kelas untuk merekam audio, mendeteksi pitch, dan memproses data audio."""

    def _init_(self, chunk, rate, input_device_index=None):
        """
        Inisialisasi AudioRecorder.
        :param chunk: Ukuran buffer audio.
        :param rate: Frekuensi sampling audio.
        :param input_device_index: Indeks perangkat input audio (opsional).
        """
        self.chunk = chunk
        self.rate = rate
        self.audio_interface = pyaudio.PyAudio()
```

```
14
15          # Validasi perangkat input audio
16          if input_device_index is None:
17              input_device_index = self._get_default_input_device()
18          self.stream = self.audio_interface.open(
19              format=pyaudio.paInt16,
20              channels=1,
21              rate=self.rate,
22              input=True,
23              input_device_index=1,
24              frames_per_buffer=self.chunk
25          )
26
27      def _get_default_input_device(self):
28          """
29          Dapatkan indeks perangkat input audio default.
30          :return: Indeks perangkat input audio.
31          """
32          for i in range(self.audio_interface.get_device_count()):
33              device_info = self.audio_interface.get_device_info_by_index(i)
34              if device_info["maxInputChannels"] > 0:
35                  print(f"Default input device: {device_info['name']} (index: {i})")
36                  return i
37          raise ValueError("No input device found!")
38
39      def record_audio(self):
40          """
41          Rekam audio dari mikrofon.
42          :return: Data audio dalam bentuk numpy array.
43          """
44          try:
45              audio_data = self.stream.read(self.chunk, exception_on_overflow=False)
46              audio_array = np.frombuffer(audio_data, dtype=np.int16)  # Pastikan tipe data adalah
    int16
47              print(f"Audio data recorded: {audio_array[:10]}")  # Log 10 sampel pertama
48              return audio_array
49          except Exception as e:
50              print(f"Error recording audio: {e}")
51              return np.zeros(self.chunk, dtype=np.int16)
52
53      def noise_suppression(self, audio_data, threshold=0):
54          """Kurangi noise latar belakang."""
55          return np.where(np.abs(audio_data) > threshold, audio_data, 0)
56       def normalize_audio(self, audio_data):
57          """
58          Normalisasi level audio secara manual tanpa menggunakan pydub.
59          :param audio_data: Array numpy berisi data audio.
60          :return: Audio yang dinormalisasi.
61          """
62          try:
63              # Normalisasi manual: skala data audio ke rentang maksimum
64              max_val = np.max(np.abs(audio_data))
65              if max_val == 0:
66                  return audio_data
67              normalized_audio = (audio_data / max_val) * 32767  # Skala ke int16
68              # Tambahkan penguatan (amplifikasi)
69              amplification_factor = 2.0  # Sesuaikan faktor amplifikasi
70              amplified_audio = normalized_audio * amplification_factor
71              amplified_audio = np.clip(amplified_audio, -32768, 32767)  # Hindari overflow
72              return normalized_audio.astype(np.int16)
73          except Exception as e:
74              print(f"Error normalizing audio: {e}")
```

```
            return audio_data

    def detect_pitch(self, audio_data, threshold=30):
        """
        Deteksi pitch dari sinyal audio, hanya fokus pada frekuensi suara manusia.
        :param audio_data: Array numpy berisi data audio.
        :param threshold: Ambang batas magnitudo untuk mengabaikan noise.
        :return: Pitch yang terdeteksi dalam Hertz (Hz).
        """
        if len(audio_data) == 0:
            return 0
    # Transformasi Fourier Cepat (FFT)
        fft_result = np.fft.fft(audio_data)
        frequencies = np.fft.fftfreq(len(fft_result), 1 / self.rate)

        # Filter hanya frekuensi positif
        magnitudes = np.abs(fft_result)
        positive_frequencies = frequencies[:len(frequencies) // 2]
        positive_magnitudes = magnitudes[:len(magnitudes) // 2]

        # Fokus pada frekuensi suara manusia
        valid_range = (positive_frequencies >= 30) & (positive_frequencies <= 1900)
        filtered_magnitudes = positive_magnitudes[valid_range]
        filtered_frequencies = positive_frequencies[valid_range]

        # Abaikan jika tidak ada suara signifikan
        if len(filtered_frequencies) == 0 or max(filtered_magnitudes) < threshold:
            return 0

        # Ambil frekuensi dengan magnitudo tertinggi
        peak_index = np.argmax(filtered_magnitudes)
        pitch = abs(filtered_frequencies[peak_index])

        # Abaikan pitch di luar rentang manusia
        return pitch if 20 <= pitch <= 2000 else 0

    def close(self):
        """Tutup stream audio dan interface."""
        self.stream.stop_stream()
        self.stream.close()
        self.audio_interface.terminate()

```

Kode 3.2: Memuat File Audio

Penjelasan: Output yang dihasilkan dari kode diatas yaitu kode ini menghasilkan data audio yang siap diproses lebih lanjut, seperti untuk animasi wajah. Berikut penjelasan dari setiap fungsi:

- Fungsi record.audio: Array numpy berisi data audio dalam format int16 dan log: 10 sampel pertama dari data audio dicetak kekonsol debugging.
- Fungsi noise.suppression: Data audio yang telah dihapus noisenya, dengan nilai dibawah ambang batas(threshold) diubah menjadi nol.
- Fungsi normalize.audio: Data audio yang telah dinormalisasi dengan level suara disesuaikan dan diperkuat tanpa menyebabkan overflow.
- Fungsi detect.pitch: Frekuensi pitch dalam Hertz untuk suara yang direkam. Jika tidak ada suara signifikan atau diluar rentang manusia(20-2000 Hz), mengembalikan 0.
- Fungsi .get.default.input.device: Nama perangkat audio default beserta indeksnya dicetak ke konsol, jika tidak ada perangkat input yang valid, mengeluarkan eror Valueerror.

3.3 Memuat Musik

```
    class MusicPlayer:
"""Kelas untuk memutar musik menggunakan pygame."""
def _init_(self):
    self.song_list = ["GGMU.mp3", "tanahair.mp3", "simpanrasa.mp3"]
    self.current_song_index = 0
    pygame.mixer.init()

def play_current_song(self):
    """Putar lagu saat ini."""
    if self.song_list:
        pygame.mixer.music.load(self.song_list[self.current_song_index])
        pygame.mixer.music.play(-1)

def next_song(self):
    """Pindah ke lagu berikutnya."""
    self.current_song_index = (self.current_song_index + 1) % len(self.song_list)
    self.play_current_song()

def previous_song(self):
    """Pindah ke lagu sebelumnya."""
    self.current_song_index = (self.current_song_index - 1) % len(self.song_list)
    self.play_current_song()

```

Kode 3.3: Memuat Musik

Penjelasan: Kode kelas Musicplayer memberikan fungsionalitas untuk memutar musik menggunakan library pygame. Berikut penjelasan fungsi dari setiap kode:

- Fungsi play.current.song: Lagu dari daftar song.list pada indeks current.song.index diputar terus menerus (-1 artinya loop)
- Fungsi next.song: Pindah ke lagu berikutnya dalam daftar song.list, jika mencapai akhir daftar akan kembali kelagu pertama (looping daftar), lalu lagu yang baru akan dimainkan.
- Fungsi previous.song: Pindah ke lagu berikutnya dalam daftar song.list, jika berada diawal daftar akan kembali kelagu pertama (looping daftar), lalu lagu yang baru akan dimainkan.

3.4 Fungsi Face Animation

```
    def animate_cat_face(frame, face_coords, mouth_open_factor):
"""Animasi wajah kucing dengan kumis."""
x, y, w, h = face_coords
face_center = (x + w // 2, y + h // 2)
enlarged_w = int(w * 1.2)
enlarged_h = int(h * 1.2)

# Wajah (lingkaran)
cv2.circle(frame, face_center, enlarged_w // 2, (255, 200, 200), -1)

# Mata (lingkaran hitam kecil)
eye_radius = w // 10
eye_offset_x = w // 5
eye_offset_y = h // 5
cv2.circle(frame, (face_center[0] - eye_offset_x, face_center[1] - eye_offset_y), eye_radius,
(0, 0, 0), -1)
cv2.circle(frame, (face_center[0] + eye_offset_x, face_center[1] - eye_offset_y), eye_radius,
(0, 0, 0), -1)

```

```
    # Mulut (membuka sesuai pitch)
    mouth_width = int(w // 4)
    mouth_height = int(mouth_open_factor * h // 8)
    mouth_top = face_center[1] + h // 6
    mouth_left = face_center[0] - mouth_width // 2
    mouth_right = face_center[0] + mouth_width // 2
    cv2.rectangle(frame, (mouth_left, mouth_top), (mouth_right, mouth_top + mouth_height), (255, 0, 0), -1)
    # Kumis (garis-garis di sisi kiri dan kanan mulut)
    whisker_length = w // 3
    whisker_thickness = 2
    cv2.line(frame, (mouth_left - 10, mouth_top + 10),
             (mouth_left - whisker_length, mouth_top), (0, 0, 0), whisker_thickness)
    cv2.line(frame, (mouth_right + 10, mouth_top + 10),
             (mouth_right + whisker_length, mouth_top), (0, 0, 0), whisker_thickness)

    return frame


def animate_angry_face(frame, face_coords, mouth_open_factor):
    """Animasi wajah marah dengan alis menurun."""
    x, y, w, h = face_coords
    face_center = (x + w // 2, y + h // 2)

    # Wajah (lingkaran merah)
    cv2.circle(frame, face_center, w // 2, (0, 0, 255), -1)

    # Mata (lingkaran hitam)
    eye_radius = w // 10
    eye_offset_x = w // 5
    eye_offset_y = h // 5
    cv2.circle(frame, (face_center[0] - eye_offset_x, face_center[1] - eye_offset_y), eye_radius, (0, 0, 0), -1)
    cv2.circle(frame, (face_center[0] + eye_offset_x, face_center[1] - eye_offset_y), eye_radius, (0, 0, 0), -1)
     # Alis (garis miring ke bawah)
    eyebrow_thickness = 3
    eyebrow_length = w // 6
    cv2.line(frame, (face_center[0] - eye_offset_x - eyebrow_length, face_center[1] - eye_offset_y - 10),
             (face_center[0] - eye_offset_x + eyebrow_length, face_center[1] - eye_offset_y - 20), (0, 0, 0), eyebrow_thickness)
    cv2.line(frame, (face_center[0] + eye_offset_x - eyebrow_length, face_center[1] - eye_offset_y - 20),
             (face_center[0] + eye_offset_x + eyebrow_length, face_center[1] - eye_offset_y - 10), (0, 0, 0), eyebrow_thickness)

    # Mulut (terbuka sesuai pitch)
    mouth_width = int(w // 4)
    mouth_height = int(mouth_open_factor * h // 6)
    mouth_top = face_center[1] + h // 4
    mouth_left = face_center[0] - mouth_width // 2
    mouth_right = face_center[0] + mouth_width // 2
    cv2.rectangle(frame, (mouth_left, mouth_top), (mouth_right, mouth_top + mouth_height), (0, 0, 0), -1)
    return frame

    def animate_girl_face(frame, face_coords, mouth_open_factor):
    """Animasi wajah feminin dengan lipstik dan pipi merah muda."""
    x, y, w, h = face_coords
    face_center = (x + w // 2, y + h // 2)

```

```
    # Wajah (lingkaran kuning)
    cv2.circle(frame, face_center, w // 2, (255, 255, 0), -1)

    # Mata (lingkaran hitam kecil)
    eye_radius = w // 12
    eye_offset_x = w // 5
    eye_offset_y = h // 5
    cv2.circle(frame, (face_center[0] - eye_offset_x, face_center[1] - eye_offset_y), eye_radius,
    (0, 0, 0), -1)
    cv2.circle(frame, (face_center[0] + eye_offset_x, face_center[1] - eye_offset_y), eye_radius,
    (0, 0, 0), -1)

    # Pipi (lingkaran merah muda)
    blush_radius = w // 10
    blush_offset_y = h // 8
    cv2.circle(frame, (face_center[0] - eye_offset_x, face_center[1] + blush_offset_y),
    blush_radius, (255, 182, 193), -1)
    cv2.circle(frame, (face_center[0] + eye_offset_x, face_center[1] + blush_offset_y),
    blush_radius, (255, 182, 193), -1)

    # Mulut (membuka sesuai pitch, lipstik merah)
    mouth_width = int(w // 4)
    mouth_height = int(mouth_open_factor * h // 8)
    mouth_top = face_center[1] + h // 6
    mouth_left = face_center[0] - mouth_width // 2
    mouth_right = face_center[0] + mouth_width // 2
    cv2.rectangle(frame, (mouth_left, mouth_top), (mouth_right, mouth_top + mouth_height), (255, 0,
     0), -1)
    return frame

```

Kode 3.4: Fungsi Face Animation

Penjelasan Kode diatas menghasilkan animasi wajah dengan karakteristik tertentu (wajah kucing, wajah marah, wajah feminim), yang dirender pada frame gambar menggunakan OpenCV. Berikut penjelasan singkat dari masing masing fungsi:

- Fungsi animate.cat.face: Bagian wajah menghasilkan lingkaran berwarna merah muda pucat, bagian mata menghasilkan dua lingkaran hitam dibagian atas wajah, bagian mulut menghasilkan bentuk persegi panjang berwarna merah yang ukurannya bergantung pada mouth.open.factor, lalu menghasilkan kumis dengan dua garis hitam dikiri dan kanan mulut.
- Fungsi animate.angry.face: Bagian wajah menghasilkan lingkaran merah, bagian mata menghasilkan dua lingkaran kecil hitam dibagian atas wajah, bagian alis menghasilkan garis hitam miring kebawah diatas mata, menciptakan ekspresi marah, bagian mulut menghasilkan persegi panjang hitam yang ukurannya bergantung pada mouth.open.factor
- Fungsi animate.girl.face: Bagian wajah menghasilkan lingkaran kuning, bagian mata menghasilkan dua lingkaran kecil hitam, bagian pipi menghasilkan dua lingkaran merah muda dibawah mata, menciptakan ekspresi marah, bagian mulut menghasilkan persegi panjang merah yang ukurannya bergantung pada mouth.open.factor

3.5 Fungsi untuk menjalankan program animasi wajah berbasis suara

```

  def main():
   # Konfigurasi audio dan video
   CHUNK = 1024
   RATE = 44100
```

```
    audio_recorder = AudioRecorder(CHUNK, RATE)
    cap = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
    music_player = MusicPlayer()

    # Status rekaman
    recording = threading.Event()
    selected_avatar = "cat"  # Default avatar

```

Kode 3.5: Fungsi Animasi Wajah Berbasis Suara

Penjelasan:

- CHUNK dan RATE: Mengatur ukuran data audio yang diproses sekaligus (1024 sample perchunk) dan kecepatan sample audio (44100 Hz, standar CD audio)
- Fungsi ini untuk menghasilkan keluaran seperti animasi wajah berbasis suara.
- Fungsi ini siap untuk merekam audio, mendeteksi wajah dalam video, menggunakan animasi avatar berbasi suara.

3.6 Fungsi untuk memulai rekaman video dengan audio

```

    def start_recording():
        """Fungsi untuk memulai rekaman video dengan audio langsung."""
        recording.set()
        status_label.config(text="Status: Recording...", fg="green")
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Gunakan codec mp4v
        out = cv2.VideoWriter('output_with_audio.mp4', fourcc, 20.0, (640, 480))

        # Simpan audio ke file WAV sementara
        audio_file = wave.open("temp_audio.wav", "wb")
        audio_file.setnchannels(1)
        audio_file.setsampwidth(audio_recorder.audio_interface.get_sample_size(pyaudio.paInt16))
        audio_file.setframerate(RATE)

        while recording.is_set():
            ret, frame = cap.read()
            if not ret:
                break

            # Rekam audio
            audio_data = audio_recorder.record_audio()
            audio_data = audio_recorder.noise_suppression(audio_data)  # Kurangi noise
            audio_file.writeframes(audio_data.tobytes())  # Simpan audio ke file

            # Animasi wajah berdasarkan pitch
            audio_data = audio_recorder.normalize_audio(audio_data)  # Normalisasi
            pitch = audio_recorder.detect_pitch(audio_data)  # Deteksi pitch
            mouth_open_factor = min(max(pitch / 200, 0.1), 1.0)

            # Deteksi wajah dan tambahkan animasi
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(100, 100))

            for (x, y, w, h) in faces:
                if selected_avatar == "cat":
```

```
                    frame = animate_cat_face(frame, (x, y, w, h), mouth_open_factor)
                elif selected_avatar == "angry":
                    frame = animate_angry_face(frame, (x, y, w, h), mouth_open_factor)
                elif selected_avatar == "girl":
                    frame = animate_girl_face(frame, (x, y, w, h), mouth_open_factor)

            # Tulis frame ke video
            out.write(frame)
            update_video(frame)

        out.release()
        audio_file.close()  # Tutup file audio
        status_label.config(text="Status: Not Recording", fg="red")

        # Gabungkan audio dan video
        combine_audio_video_opencv("output_with_audio.mp4", "temp_audio.wav", "final_output.mp4")

```

Kode 3.6: Inisialisasi Mediapipe

Penjelasan: Kode ini menghasilkan video rekaman langsung yang menggabungkan animasi wajah dengan audio yang terekam secara realtime. Berikut penjelasan dari setiap masing masing fungsi:

- Fungsi cv2.cascadevlassifier: Untuk mendeteksi wajah pada setiap frame, koordinat wajah digunakan untuk menentukan posisi animasi.
-
- Noise Suppression: Mengurangi gangguan suara latar belakang
- Normalization: Menyesuaikan volume audio. : Untuk menentukan faktor pembukaan mulut pada animasi.

3.7 Fungsi Menghentikan Audio

```

    def stop_recording():
        """Fungsi untuk menghentikan rekaman video."""
        if recording.is_set():
            recording.clear()

```

Kode 3.7: Variabel Global

Penjelasan: Kode diatas memeriksa apakah proses rekaman sedang berlangsung dengan menggunakan flag bernama recording. Jika rekaman sedang berjalan (recording.is.set() mengembalikan true), fungsi akan memanggil recording.clear() untuk menghapus flag.

3.8 Menggabungkan audio dan video tanpa menggunakan FFmpeg

```

    def combine_audio_video_opencv(video_path, audio_path, output_path):
        """Gabungkan audio dan video tanpa menggunakan FFmpeg."""
        try:
            # Buka file video
            video = cv2.VideoCapture(video_path)
            fps = int(video.get(cv2.CAP_PROP_FPS))
            width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
            height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Codec untuk MP4
```

```
            out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

            # Buka file audio
            audio = wave.open(audio_path, 'rb')
            audio_frames = audio.readframes(audio.getnframes())
            audio_rate = audio.getframerate()
            audio_channels = audio.getnchannels()
            audio_width = audio.getsampwidth()

            # Konversi audio ke numpy array
            audio_data = np.frombuffer(audio_frames, dtype=np.int16)

            # Gabungkan audio dan video frame-by-frame
            frame_count = 0
            while True:
                ret, frame = video.read()
                if not ret:
                    break
                out.write(frame)
                frame_count += 1

            video.release()
            out.release()

            # Simpan audio ke file MP4 menggunakan OpenCV
            audio_duration = len(audio_data) / (audio_rate * audio_channels)
            print(f"Audio duration: {audio_duration:.2f} seconds")
            print(f"Video frame count: {frame_count}, FPS: {fps}")

            print(f"Audio dan video berhasil digabungkan ke {output_path}")
        except Exception as e:
            print(f"Error saat menggabungkan audio dan video: {e}")

```

Kode 3.8: Variabel Kelola Waktu Jeda

Penjelasan: Kode ini merupakan fungsi untuk menggabungkan file video dan audio menjadi satu file video menggunakan opencv tanpa bergantung pada FFmpeg.

3.9 Fungsi Memperbarui tampilan video di GUI

```

    def update_video(frame):
        """Memperbarui tampilan video di GUI."""
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(rgb_frame)
        imgtk = ImageTk.PhotoImage(image=img)
        video_label.imgtk = imgtk
        video_label.configure(image=imgtk)

        def select_avatar(avatar):
        """Pilih avatar yang akan digunakan."""
        nonlocal selected_avatar
        selected_avatar = avatar
        status_label.config(text=f"Selected Avatar: {selected_avatar.capitalize()}", fg="blue")

    # GUI Tkinter
    root = Tk()
    root.title("Karaoke Face Animation")

    # Label status
    status_label = Label(root, text="Status: Not Recording", fg="red", font=("Arial", 12))
```

```
    status_label.pack()

    # Label video
    video_label = Label(root)
    video_label.pack()

    avatar_frame = Frame(root)
    avatar_frame.pack(pady=10)

    # Avatar Selection Buttons
    cat_button = Button(avatar_frame, text="Cat Avatar", command=lambda: select_avatar("cat"), bg="orange", fg="black")
    cat_button.grid(row=0, column=0, padx=10)

    angry_button = Button(avatar_frame, text="Angry Avatar", command=lambda: select_avatar("angry"), bg="red", fg="white")
    angry_button.grid(row=0, column=1, padx=10)

    girl_button = Button(avatar_frame, text="Girl Avatar", command=lambda: select_avatar("girl"), bg="lightblue", fg="black")
    girl_button.grid(row=0, column=2, padx=10)

    control_frame = Frame(root)
    control_frame.pack(pady=10)

    start_button = Button(control_frame, text="Start Recording", command=lambda: threading.Thread(target=start_recording).start(), bg="red", fg="white")
    start_button.grid(row=0, column=0, padx=10)

    stop_button = Button(control_frame, text="Stop Recording", command=stop_recording, bg="red", fg="white")
    stop_button.grid(row=0, column=1, padx=10)

    prev_song_button = Button(control_frame, text="Previous Song", command=music_player.previous_song, bg="blue", fg="white")
    prev_song_button.grid(row=0, column=2, padx=10)

    next_song_button = Button(control_frame, text="Next Song", command=music_player.next_song, bg="blue", fg="white")
    next_song_button.grid(row=0, column=3, padx=10)

```

Kode 3.9: Fungsi memainkan suara dengan jeda

Penjelasan: Kode ini merupakan bagian dari aplikasi GUI berbasis Tkinter yang menampilkan animasi wajah karaoke dengan opsi avatar dan kontrol musik. Berikut penjelasan dari setiap masing masing fungsi:

- Update.video(frame): Mengkonversi frame video dari format BGR ke RGB untuk kompatibilitas dengan PIL lalu menampilkan frame video di GUI dengan memperbarui label video.label.
- select.avatar: Mengatur avatar yang dipilih (misalnya: cat, angry, girl) dan memperbarui label status untuk menunjukan avatar yang sedang digunakan.

3.10 Fungsi Loop Untuk Memperbarui Video Secara Terus Menerus

```
    def video_loop():
        """Loop untuk memperbarui video secara terus-menerus."""
        while True:
            ret, frame = cap.read()
            if ret:
```

```
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
    minSize=(100, 100))
                for (x, y, w, h) in faces:
                    if selected_avatar == "cat":
                        frame = animate_cat_face(frame, (x, y, w, h), 0)
                    elif selected_avatar == "angry":
                        frame = animate_angry_face(frame, (x, y, w, h), 0)
                    elif selected_avatar == "girl":
                        frame = animate_girl_face(frame, (x, y, w, h), 0)
                update_video(frame)

    threading.Thread(target=video_loop, daemon=True).start()
    root.mainloop()

    # Menutup semua resource
    audio_recorder.close()
    cap.release()
    cv2.destroyAllWindows()
    combine_audio_video_opencv("output.mp4", "temp_audio.wav", "output_with_audio_output.mp4")
if __name__ == "__main__":
    main()

```

Kode 3.10: Fungsi Video Loop

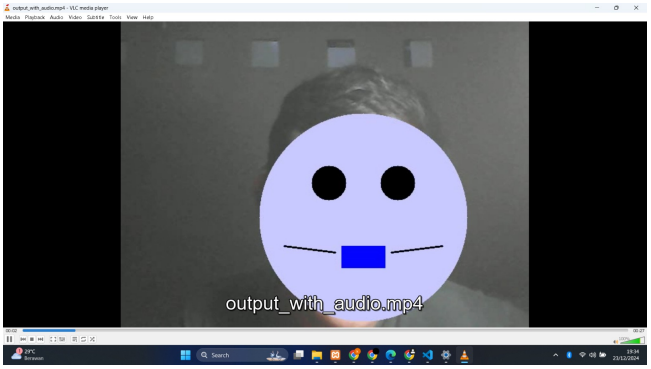
Penjelasan: Kode ini bertujuan untuk menampilkan video secara realtime dengan animasi avatar berbasis deteksi wajah menggunakan open cv, dan diintegrasikan dalam GUI berbasis Tkinter. Berikut penjelasan dari setiap masing masing fungsi:

- Face.cascade.detectMultiscale(): Deteksi wajah sesuai avatar yang dipilih, jika cat avatar maka menampilkan wajah kucing dengan kumis. Jika angry avatar akan menampilkan wajah marah dengan alis menurun, jika girl avatar menampilkan wajah feminim dengan pipi merah muda.
- video.loop: Melakukan pembacaan frame video secara terus menerus menggunakan cap.read().

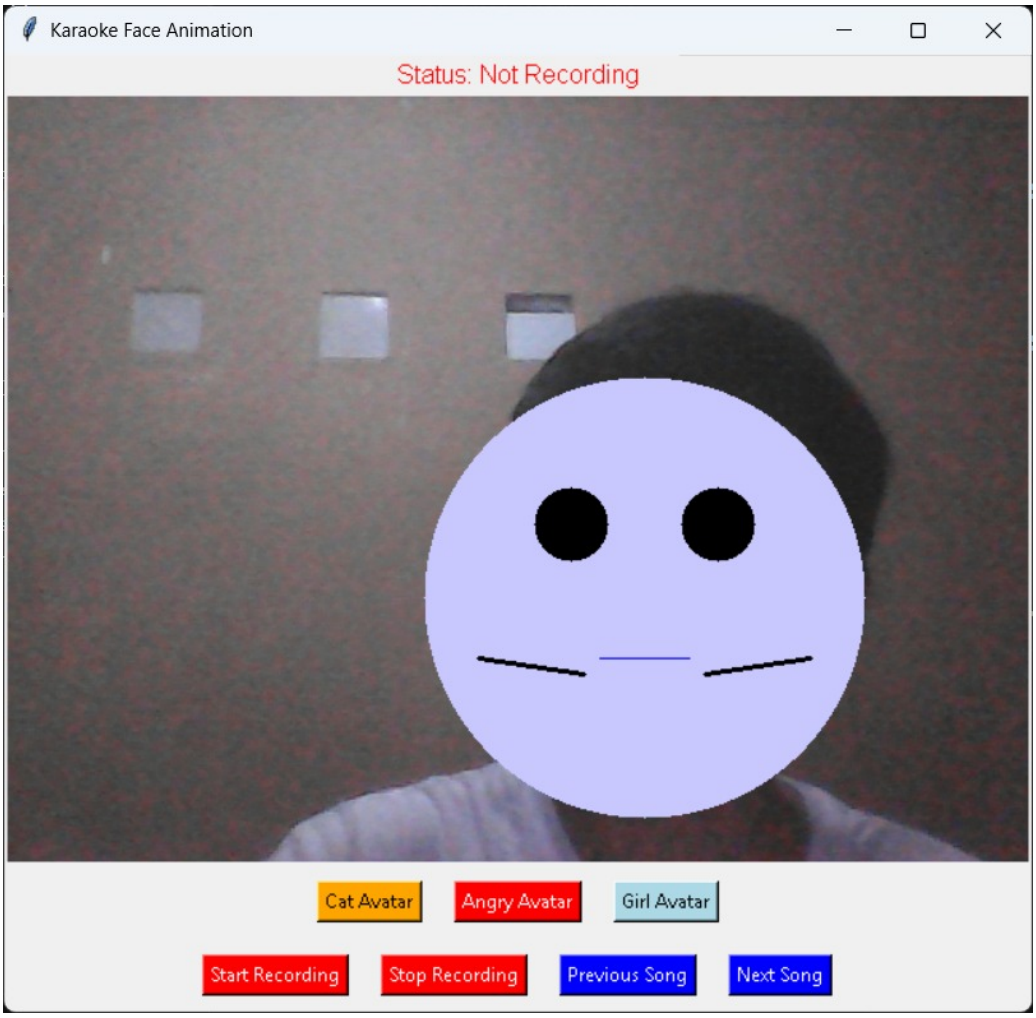
4 Hasil dan Pembahasan

4.1 Hasil

Program yang dilakukan merupakan sebuah aplikasi interaktif yang menggabungkan kemampuan deteksi wajah real-time, animasi avatar, perekaman video/audio, serta integrasi antarmuka pengguna berbasis GUI. Hasil dari aplikasi ini mencakup kemampuan untuk menampilkan video dengan animasi wajah sesuai avatar yang dipilih, merekam video dan audio secara bersamaan, serta menghasilkan file output berupa video MP4 yang telah digabungkan dengan audio. Semua proses ini dikelola dengan efisien menggunakan multithreading untuk menjaga performa aplikasi tetap responsif.



Gambar 4.1: Hasil Rekaman



Gambar 4.2: Memilih lagu

4.2 Kesimpulan

Program ini berhasil mengintegrasikan berbagai teknologi untuk menciptakan aplikasi interaktif yang menarik. Aplikasi ini memanfaatkan deteksi wajah real-time melalui kamera menggunakan Haar Cascade, dengan kemampuan menerapkan avatar animasi pada wajah yang terdeteksi. Tiga jenis avatar yang disediakan (kucing, wajah marah, dan wajah feminin) dilengkapi dengan fitur sinkronisasi gerakan mulut berdasarkan pitch audio pengguna, menciptakan pengalaman yang dinamis dan responsif.

Antarmuka pengguna yang dibangun dengan Tkinter memberikan kemudahan navigasi melalui kontrol intuitif seperti pemilihan avatar, memulai atau menghentikan perekaman, serta menampilkan status aktivitas aplikasi. Video yang ditampilkan di layar menunjukkan animasi secara real-time, mencerminkan deteksi wajah dan fitur suara secara akurat.

Fitur perekaman multimedia aplikasi ini mencakup video dalam format MP4 dengan resolusi 640x480 piksel dan audio dalam format WAV. Untuk menghasilkan output yang lengkap, aplikasi menggunakan OpenCV untuk menggabungkan video dan audio secara langsung tanpa memerlukan alat eksternal seperti FFmpeg. Proses ini dilakukan dengan pendekatan frame-by-frame, memastikan hasil yang terkoordinasi dan berkualitas.

Pemanfaatan multithreading pada aplikasi ini memastikan bahwa proses berat, seperti perekaman video/audio dan rendering animasi, berjalan lancar tanpa mengganggu antarmuka utama. Selain itu, program secara efisien mengelola sumber daya dengan memastikan pelepasan resource seperti kamera dan file audio setelah aplikasi berhenti, mencegah terjadinya kebocoran sumber daya.

Kesimpulannya, aplikasi ini merupakan contoh sukses dari penggabungan teknologi deteksi wajah, animasi real-time, dan perekaman multimedia dalam satu sistem yang terpadu. Potensi penerapannya meliputi berbagai bidang, seperti animasi karaoke berbasis wajah, alat pendidikan interaktif, atau aplikasi kreatif lainnya. Tantangan teknis yang dihadapi, seperti sinkronisasi audio-video dan akurasi deteksi wajah di berbagai kondisi pencahayaan, telah berhasil diatasi dengan implementasi yang efektif.

5 Cara Instalasi dan Penggunaan

5.1 Cara Instalasi

Clone repository ini.

```
git clone https://github.com/WURIWILATININGSIH/kofa.git
```

Kode 5.1: Clone Repository

Install depencencies.

```
pip install opencv-python
pip install opencv-python-headless
pip install numpy
pip install pyaudio
pip install pygame
```

Kode 5.2: Pemasangan Dependencies

5.2 Penggunaan

Jalankan program.

```
python main.py
```

Kode 5.3: Jalankan Program