Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama Anggota: **Rayhan Fatih Gunawan, Muhammad Nelwan Fakhri, Raditya Erza Farandi**
Tugas Ke: **Final Project**
Mata Kuliah: **Sistem / Teknologi Multimedia (IF25-40305)** Tanggal: 11/12/2025
c

1 Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi multimedia dan *computer vision* telah membuka berbagai peluang inovatif dalam analisis citra serta interaksi visual. Salah satu cabang yang mengalami kemajuan pesat adalah *human pose estimation*, yaitu proses mengidentifikasi posisi dan orientasi bagian tubuh manusia dari gambar atau video secara otomatis. Teknologi ini berperan penting dalam berbagai aplikasi seperti pengenalan aksi, pelatihan olahraga, animasi digital, dan sistem interaktif berbasis gerakan. Penelitian terbaru menunjukkan bahwa estimasi pose menjadi fondasi bagi pengembangan sistem interaktif yang responsif terhadap input gerakan manusia [1].

Sejalan dengan perkembangan tersebut, berbagai framework modern seperti MediaPipe memungkinkan deteksi *landmark* tubuh secara efisien dan real-time. Framework ini telah banyak diterapkan dalam aplikasi analisis gerak dan sistem pelatihan interaktif karena akurasi dan performanya yang stabil [2].

TwinBros merupakan sebuah filter atau aplikasi interaktif yang dikembangkan untuk memungkinkan pengguna merekonstruksi momen ikonik dari pasangan karakter dalam budaya populer. Istilah "Twin" merujuk pada istilah slang yang digunakan untuk menggambarkan dua individu yang memiliki kemiripan gaya, pose, atau ekspresi dalam cara yang kompak dan menyenangkan, sering kali muncul dalam budaya internet sebagai ekspresi kedekatan atau *chemistry* antar individu.

Dalam penggunaan TwinBros, pengguna hanya perlu menirukan pose dari sebuah momen ikonik duo karakter. Sistem kemudian melakukan deteksi pose kedua pengguna melalui pendekatan *pose estimation*, lalu menyesuaikan elemen visual seperti latar belakang, atribut, dan warna agar menyerupai pose referensi. Dengan demikian, pengguna dapat memperoleh pengalaman interaktif seolah sedang merekonstruksi momen tersebut secara digital melalui pemrosesan citra real-time.

1.2 Tujuan Pembuatan Filter

Tujuan pembuatan filter TwinBros adalah sebagai berikut:

1. Mengimplementasikan teknologi augmented reality berbasis pendeteksian pose untuk menghasilkan efek visual yang interaktif.
2. Memberikan media hiburan yang memungkinkan pengguna merekonstruksi momen ikonik duo pop culture bersama teman.

3. Meningkatkan kreativitas dan interaksi sosial pengguna melalui konsep visual yang fun dan kompak.
4. Memenuhi tugas mata kuliah multimedia melalui perancangan dan implementasi filter interaktif berbasis AR.

2 TwinBros

TwinBros adalah sebuah filter yang memungkinkan pengguna merekonstruksi momen ikonik dari pasangan karakter pop culture bersama teman (bros). Nama "Twin" diambil dari istilah slang, yang berarti "match", "mirroring each other", atau "berpenampilan/bergaya sama dengan orang lain dalam cara yang fun dan kompak". Istilah ini banyak digunakan di budaya internet untuk menunjukkan kebersamaan atau chemistry antar dua orang.

Cara kerjanya adalah pengguna cukup meniru pose dari sebuah iconic pop culture duo moment, kemudian filter akan mendeteksi pose kedua pengguna dan menyesuaikan visualnya (background, atribut, warna, dll.) agar menyerupai referensi aslinya, sehingga terlihat seperti recreation dari momen ikonik tersebut.

3 Implementasi Teknis

3.1 Platform dan Tools yang Digunakan

Implementasi filter TwinBros dikembangkan menggunakan bahasa pemrograman *Python* dengan dukungan beberapa library, yaitu:

- **ultralytics**
- **opencv-python**
- **numpy**
- **pygame**
- **librosa**
- **soundfile**

Library tersebut digunakan untuk mendukung proses pendeteksian pose, pengolahan citra, pemrosesan audio, serta visualisasi hasil secara real-time.

3.2 Metode Deteksi Pose

Metode pendeteksian pose pada filter TwinBros menggunakan model **YOLO11n**. Model ini digunakan untuk mendeteksi pose dua pengguna secara real-time sebagai dasar dalam mencocokkan pose dengan referensi momen ikonik.

3.3 Alur Kerja Sistem

Alur kerja sistem pada filter TwinBros dibagi ke dalam beberapa modul utama yang saling terintegrasi, yaitu sebagai berikut:

1. **main.py**
   Berfungsi sebagai program utama yang mengatur seluruh alur eksekusi sistem, mulai dari inisialisasi kamera, pemanggilan modul deteksi, pencocokan pose, hingga penampilan hasil visual.

2. **pose detection.py**
   Modul ini bertanggung jawab untuk melakukan pendeteksian pose pengguna menggunakan model YOLO11n melalui library ultralytics dan opencv.

3. **pose matching.py**
   Modul ini digunakan untuk mencocokkan hasil pose yang terdeteksi dengan pose referensi dari momen ikonik yang telah disiapkan.

4. **pose preprocessing.py**
   Modul ini berfungsi untuk melakukan proses *preprocessing* pose dari gambar referensi yang digunakan sebagai acuan pencocokan pada sistem TwinBros.

5. **audio.py**
   Modul ini menangani pemrosesan dan pemutaran audio menggunakan library pygame dan os, karena seluruh audio telah disediakan dan ditentukan oleh kami, sehingga sistem hanya perlu memanggil dan memutar file audio tersebut melalui *directory manager*.

6. **audio chop.py**
   Modul ini bekerja dengan cara membaca file audio yang kamu berikan, lalu memotongnya berdasarkan waktu mulai dan waktu akhir yang kamu tentukan, kemudian menyimpan hasil potongan itu sebagai file audio baru dalam format **WAV** di folder yang sama.

3.4 Pengolahan Visual Hasil

Pada tahap ini, hasil dari pendeteksian pose diproses untuk menyesuaikan elemen visual agar menyerupai referensi momen ikonik, seperti latar belakang, atribut, warna, dan efek tambahan lainnya.

4 Code

Bagian ini berisi potongan kode utama yang digunakan dalam pembuatan filter TwinBros beserta penjelasan singkat fungsinya.

4.1 Struktur File Program

Struktur file program yang digunakan dalam pengembangan sistem adalah sebagai berikut:

- main.py
- pose detection.py
- pose matching.py
- pose preprocessing.py
- audio.py
- audio chop.py

4.2 Kode Program main.py

```
1  import cv2
2  import numpy as np
3  import time
4  import os
5  from camera import Camera
```

```
    from ui import UI
    from transition import Transition
    from pose_detection import PoseDetector
    from pose_matching import PoseMatcher
    from audio import AudioManager
    from video_recorder import VideoRecorder

    def main():
        cam = Camera()
        ui = UI()
        trans = Transition(ui)

        detector = PoseDetector()          # load YOLO once
        matcher = PoseMatcher()            # load reference poses once

        # Initialize Audio and Video
        base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        sound_dir = os.path.join(base_dir, "assets", "sounds")
        output_dir = os.path.join(base_dir, "output")

        audio_manager = AudioManager(sound_dir)
        video_recorder = VideoRecorder(output_dir)

        print("=== TwinBros Pose Matching App ===")
        print("[INFO] Tekan 's' untuk mulai sesi, 'q' untuk keluar.")
        print("[INFO] Gunakan Panah Kiri/Kanan (atau A/D) untuk memilih musik.")

        SESSION_DURATION = 15  # seconds
        POSE_INTERVAL = 3      # seconds

        # Pre-load iconic images mapping
        iconic_img_dir = os.path.join(base_dir, "data", "iconic_images")

        while True:
            frame, keypoints = cam.get_frame()
            if frame is None:
                print("[ERROR] Kamera tidak tersedia.")
                break

            # Draw YOLO keypoints
            if keypoints is not None:
                for person in keypoints:
                    for (x, y, conf) in person:
                        if conf > 0.5:
                            cv2.circle(frame, (int(x), int(y)), 3, (0, 255, 0), -1)

            ui.overlay_text(frame, "Press 'S' to Start | 'Q' to Quit", (20, 460))
            ui.overlay_text(frame, f"Music (< A/D >): {audio_manager.get_current_track_name()}",
    (20, 430), scale=0.6)

            cv2.imshow("TwinBros", frame)

            key = cv2.waitKeyEx(1)
            if key == ord('q'):
                break

            # Arrow Keys (Windows) or A/D
            # Left Arrow: 2424832, Right Arrow: 2555904
            elif key == 2424832 or key == ord('a'): # Left
                audio_manager.prev_track()
            elif key == 2555904 or key == ord('d'): # Right
                audio_manager.next_track()
```

```
67
68          elif key == ord('s'):
69              print("[INFO] Starting session...")
70
71              # Start Music and Recording
72              audio_manager.play()
73              video_path = video_recorder.start(frame.shape[1], frame.shape[0])
74
75              start_time = time.time()
76              next_capture_time = start_time + POSE_INTERVAL
77
78              while (time.time() - start_time) < SESSION_DURATION:
79                  frame, keypoints = cam.get_frame()
80                  if frame is None: break
81
82                  # Record frame
83                  video_recorder.write(frame)
84
85                  remaining_time = int(SESSION_DURATION - (time.time() - start_time))
86                  ui.overlay_text(frame, f"Session Time: {remaining_time}s", (20, 50), color
    =(255, 255, 255))
87
88                  # Show countdown to next capture
89                  time_to_capture = next_capture_time - time.time()
90                  if time_to_capture > 0:
91                       ui.overlay_text(frame, f"Next Pose in: {int(time_to_capture)+1}", (20,
    100), color=(255, 255, 255))
92
93                  # Draw keypoints
94                  if keypoints is not None:
95                      for person in keypoints:
96                          for (x, y, conf) in person:
97                              if conf > 0.5:
98                                  cv2.circle(frame, (int(x), int(y)), 3, (0, 255, 0), -1)
99
100                 cv2.imshow("TwinBros", frame)
101                 key = cv2.waitKey(1)
102
103                 if time.time() >= next_capture_time:
104                     # Capture!
105                     trans.flash_effect(video_recorder)
106                     print("[INFO] Capturing pose...")
107
108                     if keypoints and len(keypoints) > 0:
109                         person = keypoints[0]
110                         coords = np.array(person)[:, :2]
111                         norm_pose = detector.normalize_pose(coords)
112
113                         best_name, best_score = matcher.match(norm_pose)
114                         print(f"[OK] Match found: {best_name} (Score: {best_score:.2f})")
115
116                         # Find iconic image
117                         base_name = best_name.split("_person")[0]
118                         image_path = None
119                         for f in os.listdir(iconic_img_dir):
120                             if f.startswith(base_name):
121                                 image_path = os.path.join(iconic_img_dir, f)
122                                 break
123
124                         # Pause recording briefly if we want to show the result in the video?
125                         # Or just show it on screen. The recorder captures frames from the loop
    .
```

```
                            # If show_match_result uses cv2.imshow and waitKey, it might NOT be recorded unless we explicitly write frames there.
                            # For now, let's keep the simple flow. The user sees the result, but the video might skip it or show it frozen if we don't handle it.
                            # Actually show_match_result blocks the loop with waitKey.
                            # To record the result overlay, we would need to modify show_match_result to return the image or handle recording internally.
                            # For now, let's accept that the result display is a "pause" in the session flow.

                            ui.show_match_result(image_path, base_name, best_score, duration=2000, video_recorder=video_recorder)

                            next_capture_time = time.time() + POSE_INTERVAL

                        else:
                            print("[INFO] Tidak ada pose terdeteksi.")
                            ui.transition_message("No Pose Detected", 1000, video_recorder)
                            next_capture_time = time.time() + POSE_INTERVAL

                print("[INFO] Session ended.")

                # Stop Music and Recording
                audio_manager.stop()
                video_recorder.stop()

                ui.transition_message("Session Ended", 2000, video_recorder)

                # Drive Upload Message
                drive_link = "https://drive.google.com/drive/folders/1Knv2PaaFLN57YNlExWAAgRxrw-eAakg5?usp=sharing"
                print(f"\n[IMPORTANT] Video saved to: {video_path}")
                print(f"[IMPORTANT] Please upload the video to Google Drive: {drive_link}\n")

                # Show message on UI
                blank = np.zeros((480, 640, 3), dtype=np.uint8)
                cv2.putText(blank, "Video Saved!", (200, 200), ui.font, 1, (255, 255, 255), 2)
                cv2.putText(blank, "Upload to Drive", (180, 250), ui.font, 0.8, (255, 255, 255), 1)
                cv2.imshow("TwinBros", blank)
                cv2.waitKey(3000)

        cam.release()
        cv2.destroyAllWindows()

    if __name__ == "__main__":
        main()

```

Kode 1: Captionnya tulis di sini class

4.3 Kode Program pose detection.py

```
    import cv2
import numpy as np
from ultralytics import YOLO

class PoseDetector:
    def __init__(self, model_path="yolo11n-pose.pt", conf=0.5):
        self.model = YOLO(model_path)
        self.conf = conf

```

```
    def normalize_pose(self, keypoints):
        coords = keypoints[:, :2]  # hanya ambil x, y
        coords -= coords.mean(axis=0)  # pusatkan ke tengah
        norm = np.linalg.norm(coords)
        if norm > 0:
            coords /= norm
        return coords

    def detect_poses(self, frame):
        results = self.model(frame, conf=self.conf)
        poses = []

        for result in results:
            if result.keypoints is not None and len(result.keypoints) > 0:
                for kp in result.keypoints:
                    coords = kp.xy.cpu().numpy().squeeze()
                    norm_pose = self.normalize_pose(coords)
                    poses.append(norm_pose)
        return poses

    def draw_poses(self, frame, results):
        annotated = frame.copy()
        for result in results:
            if result.keypoints is not None:
                annotated = result.plot()
        return annotated


if __name__ == "__main__":
    detector = PoseDetector()
    cap = cv2.VideoCapture(0)

    print("[INFO] Kamera aktif. Tekan 'q' untuk keluar.")
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        results = detector.model(frame)
        annotated = detector.draw_poses(frame, results)

        poses = detector.detect_poses(frame)
        cv2.putText(annotated, f"Detected poses: {len(poses)}",
                    (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)

        cv2.imshow("Pose Detection - YOLO11", annotated)

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

    cap.release()
    cv2.destroyAllWindows()
```

Kode 2: Captionnya tulis di sini class

4.4 Kode Program pose matching.py

```
    import os
    import json
    import numpy as np

```

```python
class PoseMatcher:

    def __init__(self, base_dir=None):
      if base_dir is None:
        base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
        self.reference_dir = os.path.join(base_dir, "data", "reference_poses")
        self.references = self._load_reference_poses()


    def _load_reference_poses(self):
        references = {}
        for file in os.listdir(self.reference_dir):
            if file.endswith(".json"):
                path = os.path.join(self.reference_dir, file)
                with open(path, "r") as f:
                    data = np.array(json.load(f))
                    references[file] = data
        print(f"[INFO] Loaded {len(references)} reference poses from {self.reference_dir}")
        return references

    def _cosine_similarity(self, a, b):
        """Hitung kesamaan antara dua pose dengan cosine similarity."""
        a = a.flatten()
        b = b.flatten()
        a /= np.linalg.norm(a) + 1e-8
        b /= np.linalg.norm(b) + 1e-8
        return np.dot(a, b)


    def match(self, user_pose):

        if not self.references:
            raise RuntimeError("Tidak ada pose referensi yang dimuat!")

        best_name, best_score = None, -1

        for name, ref_pose in self.references.items():
            n = min(len(user_pose), len(ref_pose))
            score = self._cosine_similarity(np.array(user_pose[:n]), ref_pose[:n])
            if score > best_score:
                best_name, best_score = name, score

        return best_name, best_score

```

Kode 3: Captionnya tulis di sini class

4.5 Kode Program pose preprocessing.py

```python
import os
import cv2
import json
import numpy as np
from ultralytics import YOLO

# === PATH SETUP ===
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
ICONIC_DIR = os.path.join(BASE_DIR, "data", "iconic_images")
OUTPUT_DIR = os.path.join(BASE_DIR, "data", "reference_poses")
VISUAL_DIR = os.path.join(BASE_DIR, "data", "pose_visualizations")

```

```
    os.makedirs(OUTPUT_DIR, exist_ok=True)
    os.makedirs(VISUAL_DIR, exist_ok=True)

    # === YOLOv8 Pose Model ===
    model = YOLO("yolo11n-pose.pt")  # Updated to YOLO11


    def resize_and_pad(image, target_size=(640, 480)):
        """Resize gambar tanpa merusak rasio aslinya dan beri padding hitam."""
        h, w = image.shape[:2]
        target_w, target_h = target_size
        scale = min(target_w / w, target_h / h)
        new_w, new_h = int(w * scale), int(h * scale)
        resized = cv2.resize(image, (new_w, new_h))

        canvas = np.zeros((target_h, target_w, 3), dtype=np.uint8)
        x_offset = (target_w - new_w) // 2
        y_offset = (target_h - new_h) // 2
        canvas[y_offset:y_offset + new_h, x_offset:x_offset + new_w] = resized
        return canvas


    def normalize_pose(keypoints):
        """Normalisasi koordinat pose agar konsisten antar gambar."""
        coords = np.array(keypoints)
        coords = coords[~np.all(coords == 0, axis=1)]  # hapus titik kosong

        if len(coords) == 0:
            return None

        coords -= coords.mean(axis=0)  # pusatkan
        body_height = coords[:, 1].max() - coords[:, 1].min()
        if body_height > 0:
            coords /= body_height

        norm = np.linalg.norm(coords)
        if norm != 0:
            coords /= norm

        return coords.tolist()


    def extract_pose_from_image(image_path):
        """Deteksi semua pose manusia pada gambar dan kembalikan (poses, vis_image)."""
        image = cv2.imread(image_path)
        if image is None:
            print(f"[WARNING] Gagal membaca gambar: {image_path}")
            return [], None

        image = resize_and_pad(image)
        results = model(image, verbose=False)
        vis = image.copy()
        all_poses = []

        for result in results:
            if result.keypoints is None:
                continue

            for pid, keypoints in enumerate(result.keypoints.xy):
                keypoints = keypoints.cpu().numpy()
                norm_pose = normalize_pose(keypoints)

```

```
                if norm_pose is not None:
                    all_poses.append(norm_pose)

                    # Gambar landmark di visualisasi
                    for x, y in keypoints:
                        if x > 0 and y > 0:
                            cv2.circle(vis, (int(x), int(y)), 3, (0, 255, 0), -1)
                    cv2.putText(vis, f"P{pid+1}", (int(keypoints[0][0]), int(keypoints[0][1]) - 10)
    ,
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 1)

        return all_poses, vis


    def main():
        print("=== Pose Preprocessing (YOLO11-Pose) ===")
        files = [f for f in os.listdir(ICONIC_DIR) if f.lower().endswith((".jpg", ".png", ".jpeg"))
    ]

        if not files:
            print("[INFO] Tidak ada gambar di folder 'iconic_images/'.")
            return

        for file in files:
            path = os.path.join(ICONIC_DIR, file)
            poses, vis = extract_pose_from_image(path)

            if not poses:
                print(f"[WARNING] Tidak ada pose terdeteksi di {file}")
                continue

            # Simpan semua pose ke JSON
            for i, pose_data in enumerate(poses, start=1):
                out_path = os.path.join(OUTPUT_DIR, f"{os.path.splitext(file)[0]}_person{i}.json")
                with open(out_path, "w") as f:
                    json.dump(pose_data, f, indent=2)
                print(f"[OK] Pose {i} disimpan: {out_path}")

            # Simpan visualisasi ke file
            if vis is not None:
                vis_path = os.path.join(VISUAL_DIR, f"{os.path.splitext(file)[0]}_vis.jpg")
                cv2.imwrite(vis_path, vis)
                print(f"[IMG] Visualisasi disimpan: {vis_path}")

            # (opsional) tampilkan sebentar
            cv2.imshow("Pose Preview", vis)
            cv2.waitKey(500)

        print("=== Selesai! Semua pose referensi & visualisasi telah diekstrak. ===")
        cv2.destroyAllWindows()


    if __name__ == "__main__":
        main()

```

Kode 4: Captionnya tulis di sini class

4.6 Kode Program audio.py

```
    import pygame
```

```
import os

class AudioManager:
    def __init__(self, sound_dir):
        self.sound_dir = sound_dir
        self.tracks = []
        self.current_index = 0

        # Initialize pygame mixer
        pygame.mixer.init()

        # Load tracks
        if os.path.exists(sound_dir):
            for f in os.listdir(sound_dir):
                if f.lower().endswith('.mp3') or f.lower().endswith('.wav'):
                    self.tracks.append(os.path.join(sound_dir, f))

        if not self.tracks:
            print(f"[WARNING] No music files found in {sound_dir}")

    def play(self):
        if not self.tracks: return
        try:
            pygame.mixer.music.load(self.tracks[self.current_index])
            pygame.mixer.music.play(-1) # Loop indefinitely
        except Exception as e:
            print(f"[ERROR] Failed to play music: {e}")

    def stop(self):
        pygame.mixer.music.stop()

    def next_track(self):
        if not self.tracks: return
        self.current_index = (self.current_index + 1) % len(self.tracks)

    def prev_track(self):
        if not self.tracks: return
        self.current_index = (self.current_index - 1) % len(self.tracks)

    def get_current_track_name(self):
        if not self.tracks: return "No Music Found"
        return os.path.basename(self.tracks[self.current_index])
```

Kode 5: Captionnya tulis di sini class

4.7 Kode Program audio chop.py

```
    import os
    import sys
    import argparse

    try:
        import librosa
        import soundfile as sf
    except ImportError:
        print("[ERROR] Required libraries not found.")
        print("Please run: pip install librosa soundfile")
        sys.exit(1)

    def chop_audio(file_path, start_sec, end_sec=None):
        if not os.path.exists(file_path):
```

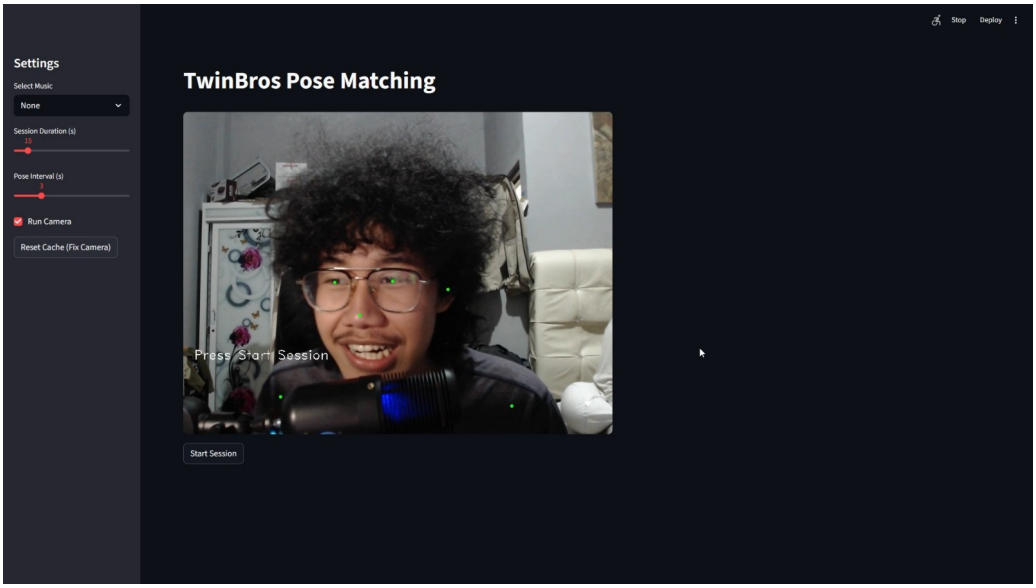
```
            print(f"[ERROR] File not found: {file_path}")
            return

        try:
            print(f"[INFO] Loading: {file_path}")
            # Load audio with original sampling rate
            y, sr = librosa.load(file_path, sr=None)

            # Calculate start and end samples
            start_sample = int(start_sec * sr)

            if end_sec:
                end_sample = int(end_sec * sr)
                # Ensure end_sample doesn't exceed length
                end_sample = min(end_sample, len(y))
            else:
                end_sample = len(y)

            if start_sample >= end_sample:
                print("[ERROR] Start time must be less than end time (or total duration).")
                return

            # Chop
            y_chopped = y[start_sample:end_sample]

            # Save
            dir_name = os.path.dirname(file_path)
            base_name = os.path.basename(file_path)
            name, ext = os.path.splitext(base_name)

            # Default to .wav for output if input is mp3, as soundfile has limited mp3 write
    support
            output_ext = ".wav"
            output_filename = f"{name}_chopped{output_ext}"
            output_path = os.path.join(dir_name, output_filename)

            sf.write(output_path, y_chopped, sr)
            print(f"[SUCCESS] Saved to: {output_path}")
            print(f"          Duration: {(end_sample - start_sample) / sr:.2f} seconds")

        except Exception as e:
            print(f"[ERROR] Failed to process file: {e}")

    if __name__ == "__main__":
        parser = argparse.ArgumentParser(description="Chop audio file using librosa.")
        parser.add_argument("file", help="Path to the audio file")
        parser.add_argument("--start", type=float, default=0, help="Start time in seconds (default:
     0)")
        parser.add_argument("--end", type=float, help="End time in seconds (optional, defaults to
    end of file)")

        args = parser.parse_args()

        chop_audio(args.file, args.start, args.end)
```

Kode 6: Captionnya tulis di sini class

5 Hasil

Bagian ini menjelaskan hasil pengujian dari filter TwinBros berdasarkan implementasi yang telah dilakukan.

5.1 Tampilan dari Sistem

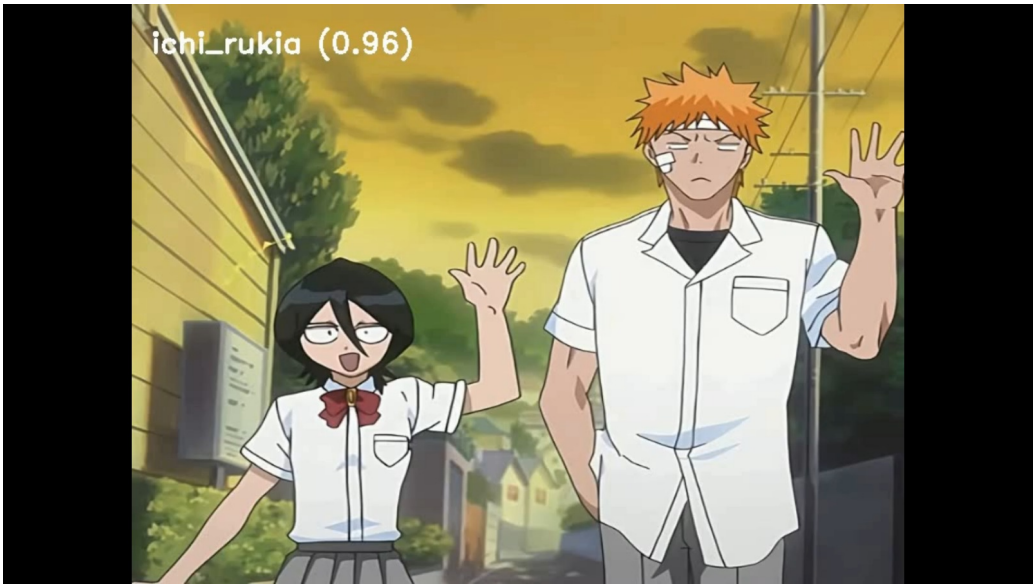


Gambar 1: Tampilan dari sistem

5.2 Respon Sistem terhadap Pose Pengguna

Gambar 2: Contoh Pose yang diperagakan

5.3 Hasil Visual yang Dihasilkan



Gambar 3: Contoh Pose yang diperagakan

6 Kesimpulan dan Evaluasi

6.1 Kesimpulan

Filter yang dikembangkan mampu mendeteksi beberapa pose-pose yang diterapkan. Namun, terdapat beberapa pose yang tidak berhasil terdeteksi oleh sistem. Hal ini disebabkan oleh kualitas gambar referensi yang digunakan sebagai acuan, khususnya pada bagian penandaan (marking) pose yang kurang jelas, sehingga memengaruhi akurasi proses deteksi.

6.2 Evaluasi Sistem

Evaluasi sistem menunjukkan bahwa filter yang dikembangkan mampu mendeteksi beberapa pose ikonik dengan tingkat akurasi yang cukup baik, namun belum sepenuhnya optimal untuk seluruh pose yang diuji. Ketidakberhasilan pada sebagian pose dipengaruhi oleh kualitas marking pada gambar referensi yang kurang jelas, serta sensitivitas sistem terhadap variasi sudut pengambilan gambar, posisi tubuh, dan skala pose. Selain itu, hasil deteksi belum sepenuhnya konsisten, khususnya pada pose yang memiliki kemiripan bentuk antar anggota tubuh, serta masih terbatasnya jumlah dan variasi data referensi yang digunakan.

6.3 Saran Pengembangan

Untuk pengembangan selanjutnya, disarankan agar kualitas gambar referensi ditingkatkan, khususnya pada kejelasan marking pose, guna meningkatkan akurasi deteksi sistem. Selain itu, penambahan jumlah dan variasi pose referensi dengan berbagai sudut pandang dan skala tubuh juga perlu dilakukan agar sistem lebih robust terhadap perbedaan pose pengguna. Penggunaan model pose yang lebih akurat atau penyesuaian parameter deteksi juga dapat dipertimbangkan untuk meningkatkan konsistensi hasil. Dengan pengembangan tersebut, diharapkan performa sistem dalam mendeteksi pose ikonik dapat menjadi lebih optimal.

References

[1] L. Song, G. Yu, and J. Yuan, "Human pose estimation and its application to action recognition: A survey," *Journal of Visual Communication and Image Representation*, vol. 76, p. 103055, 2021. [Online]. Available: https://www.researchgate.net/publication/349853529_Human_pose_estimation_and_its_application_to_action_recognition_A_survey

[2] U. Gunadarma, "Penerapan teknologi pose estimation menggunakan mediapipe dalam aplikasi pelatihan olahraga," https://lembaga.gunadarma.ac.id/repository/penerapan-teknologi-pose-estimation-menggunakan-mediapipe-dalam-aplikasi-pelatihan-olahraga-ssm, 2018, [Online]. Accessed: 2022-01-06.