Program Studi Teknik Informatika
Institut Teknologi Sumatera

Mata Kuliah: **Pengolahan Sinyal Digital (IF3024)** Tugas : **Tugas Besar**

Nama Anggota:

1. **Winnerson Laia (121140121)**
2. **Muhammad Qaessar Kartadilaga (121140119)**

Program Menghitung Denyut Jantung dan Pernapasan

1 Pendahuluan

1.1 Latar Belakang

Tanda-tanda vital seperti detak jantung dan laju pernapasan merupakan hal yang penting sehingga harus terus dipantau . Salah satu metode yang digunakan untuk memantau denyut jantung tanpa menyentuh adalah Remote Photoplethysmography (rPPG), yang memanfaatkan kamera RGB untuk menangkap perubahan warna kulit yang disebabkan oleh aliran darah [1]. Sinyal respirasi juga dapat diperoleh dengan memanfaatkan perubahan posisi area badan seperti bahu atau dada yang terekam kamera, yang berhubungan dengan aktivitas pernapasan [2]. Program ini mammpu mengolah video wajah dan tubuh, mengekstrak sinyal rPPG dan respirasi, lalu menghitung denyut jantung dan laju napas secara otomatis.

2 Alat dan Bahan

Berikut merupakan alat dan bahan yang diperlukan untuk menyelesaikan tugas besar ini :

2.1 Bahasa Pemrograman

Bahasa pemrograman yang digunakan dalam tugas besar ini adalah bahasa pemrograman python. Alasan memilih bahasa pemrograman ini karena bahasa pemrograman ini memiliki banyak library yang mendukung untuk pengolahan sinyal seperti sinyal jantung dan sinyal pernapasan.

2.2 Library

Adapun library yang dipakai untuk menyelasaikan tugas besar ini, antara lain :

1. **MediaPipe** : digunakan untuk mendeteksi wajah dan bahu
2. **OpenCV** : digunakan untuk memproses video dan gambar
3. **NumPy** : digunakan untuk operasi array
4. **Matplotlib** : digunakan untuk menampilkan sinyal yang didapat
5. **Scipy** : digunakan untuk proses filter sinyal yang didapat
6. **Os** : digunakan untuk dapat mengambil video dari sistem

2.3 Metode dan Algoritma

Adapaun metode dan algoritma yang digunakan dalam program ini, yakni :

1. ***Plane Orthogonal-to-Skin (POS)*** : metode ini digunakan untuk mengukur perubahan warna kulit akibat dari aliran darah.
2. ***Lucal-Kanade Optical Flow*** : metode ini digunakan untuk melacak perubahan posisi bagian tubuh seperti dada atau bahu.

3 Penjelasan Kode Program

3.1 Modul main.py

3.1.1 Import Library

```
1
2   import mediapipe as mp
3   from mediapipe.tasks import python
4   from mediapipe.tasks.python import vision
5   import cv2
6   import numpy as np
7   import matplotlib.pyplot as plt
8   import os
9   from fungsi import initialize_features
10  from fungsi import POS
11  from scipy.signal import butter, filtfilt, find_peaks
```

Kode 1: Import Library

Bagian ini memuat seluruh library yang diperlukan. MediaPipe digunakan untuk mendeteksi pose dan wajah. OpenCV (cv2) digunakan untuk membaca dan memproses video. NumPy dan Matplotlib digunakan untuk manipulasi data numerik dan visualisasi. initialize_features dan POS adalah fungsi khusus yang diambil dari file eksternal. Scipy digunakan untuk filtering dan pendeteksian puncak sinyal.

3.1.2 Inisialisasi Variabel dan Model

```
1
2   resp_signal = []
3   rppg_signal = []
4   r_signal, g_signal, b_signal = [], [], []
5
6   fps = 30
7   time_window = 60
8   frame_buffer_limit = time_window * fps
9   frame_buffer = []
10
11  features = None
12  lk_params = None
13  old_gray = None
14  left_x, top_y, right_x, bottom_y = None, None, None, None
15  STANDARD_SIZE = (640, 480)
16
17  margin_x = 10
18  scaling_factor = 0.8
19
20  model_path = "models/pose_landmarker.task"
21  BaseOptions = mp.tasks.BaseOptions
22
```

```
    PoseLandmarkerOptions = vision.PoseLandmarkerOptions
    VisionRunningMode = vision.RunningMode

    options_image = PoseLandmarkerOptions(...)
    pose_landmarker = vision.PoseLandmarker.create_from_options(options_image)

    base_model="models/blaze_face_short_range.tflite"
    base_options = python.BaseOptions(model_asset_path=base_model)
    FaceDetectorOptions = vision.FaceDetectorOptions
    options = FaceDetectorOptions(...)
    face_detector = vision.FaceDetector.create_from_options(options)
```

Kode 2: Import Library

Variabel-variabel penting diinisialisasi di sini termasuk buffer frame, ukuran standar frame, serta model deteksi wajah dan pose dari MediaPipe.

3.1.3 Fungsi bandpass_filter()

```

    def bandpass_filter(signal, fs, lowcut, highcut, order=3):
        nyq = 0.5 * fs
        low = lowcut / nyq
        high = highcut / nyq
        b, a = butter(order, [low, high], btype='band')
        filtered = filtfilt(b, a, signal)
        return filtered
```

Kode 3: Import Library

Fungsi bandpass_filter digunakan untuk menyaring sinyal agar hanya frekuensi dalam rentang tertentu yang dilewatkan, sementara frekuensi di luar rentang tersebut ditekan. Fungsi ini menerima parameter sinyal input (signal), frekuensi sampling (fs), batas bawah (lowcut) dan atas (highcut) dari frekuensi yang ingin dipertahankan, serta urutan filter (order) yang mengatur ketajaman filter. Filter ini menggunakan metode Butterworth untuk mendesain koefisien filter (b dan a), dan fungsi filtfilt dari scipy.signal untuk menerapkan filter secara dua arah (forward dan backward) sehingga menghasilkan sinyal yang tidak mengalami fase delay. Fungsi ini sangat berguna untuk mengekstraksi komponen frekuensi tertentu seperti denyut jantung atau laju pernapasan dari sinyal biologis.

3.1.4 Ekstraksi Sinyal Respirasi

```

    cap = cv2.VideoCapture(file_path)

    if not cap.isOpened():
        print("Error: Could not open video.")
        exit()

    x = 0

    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    while len(frame_buffer) < frame_buffer_limit:
        x = x + 1
        ret, frame = cap.read()
        if not ret:
            print("Error: Could not read frame.")
            break

        frame = cv2.resize(frame, STANDARD_SIZE)
```

```
        frame_buffer.append(frame)

        if features is None:
            features, lk_params, old_gray, left_x, top_y, right_x, bottom_y = initialize_features(
    frame, pose_landmarker, STANDARD_SIZE)

        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        if len(features) > 10:
            new_features, status, error = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, features,
    None, **lk_params)
            good_old = features[status == 1]
            good_new = new_features[status == 1]
            mask = np.zeros_like(frame)
            for i, (new, old) in enumerate(zip(good_new, good_old)):
                a, b = new.ravel()
                c, d = old.ravel()
                mask = cv2.line(mask, (int(a), int(b)), (int(c), int(d)), (0, 255, 0), 2)
                frame = cv2.circle(frame, (int(a), int(b)), 3, (0, 255, 0), -1)
            frame = cv2.add(frame, mask)
            if len(good_new) > 0:
                avg_y = np.mean(good_new[:, 1])
                resp_signal.append(avg_y)
                features = good_new.reshape(-1, 1, 2)
            old_gray = frame_gray.copy()
        else:
            initialize_features(frame, pose_landmarker, STANDARD_SIZE) ##inih

        cv2.rectangle(frame, (int(left_x), int(top_y)), (int(right_x), int(bottom_y)), (0, 255, 0),
     2)


        print(f"Processed frame {x}/{total_frames}")
```

Kode 4: Import Library

Bagian ini bertanggung jawab untuk membaca frame video dan mengekstraksi sinyal respirasi berdasarkan pergerakan vertikal fitur tubuh menggunakan Optical Flow.

3.1.5 Ekstrasi Sinyal RPPG

```

    while len(frame_buffer) < frame_buffer_limit:
        x = x + 1

        ret, frame = cap.read()
        if not ret:
            print("Error: Could not read frame.")
            break

        frame_buffer.append(frame)

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        mp_image = mp.Image(
            image_format=mp.ImageFormat.SRGB,
            data=rgb_frame
        )

        result = face_detector.detect(mp_image)

        if result.detections:
            for detection in result.detections:
```

```
                bboxC = detection.bounding_box
                x, y, w, h = bboxC.origin_x, bboxC.origin_y, bboxC.width, bboxC.height

                new_x = int(x + margin_x)
                new_w = int(w * scaling_factor)
                new_h = int(h * scaling_factor)

                face_roi = rgb_frame[y:y+new_h, new_x:new_x+new_w]

                mean_rgb = cv2.mean(face_roi)[:3]

                r_signal.append(mean_rgb[0])
                g_signal.append(mean_rgb[1])
                b_signal.append(mean_rgb[2])

        print(f"Processed frame {x}/{total_frames}")
```

Kode 5: Import Library

Setelah sinyal respirasi, bagian ini mengekstraksi sinyal rPPG dari wajah yang terdeteksi menggunakan face detector dan algoritma POS. Rata-rata RGB dari ROI wajah dihitung untuk digunakan dalam ekstraksi sinyal rPPG.

3.1.6 Filtering dan Pendeteksian Puncak

```

    sinyal_respirasi_filter = bandpass_filter(resp_signal, 30, 0.1, 0.5)
    sinyal_rppg_filter = bandpass_filter(rppg_signal, 30, 0.7, 2.5)

    puncak_respirasi, _ = find_peaks(sinyal_respirasi_filter)
    puncak_rppg, _ = find_peaks(sinyal_rppg_filter)

    breaths_per_minute = len(puncak_respirasi) / (len(sinyal_respirasi_filter) / 30) * 60
    heart_rate = 60 * len(puncak_rppg) / (len(sinyal_rppg_filter) / 30)
```

Kode 6: Import Library

Sinyal yang telah diperoleh kemudian difilter menggunakan bandpass filter untuk menghilangkan noise. Puncak sinyal yang telah difilter dideteksi sebagai siklus napas dan detak jantung. Berdasarkan jumlah puncak dan durasi waktu, dihitung laju pernapasan dan denyut jantung per menit.

3.1.7 Visualisasi

```

    fig, ax = plt.subplots(2, 1, figsize=(40, 5))
    ax[0].plot(resp_signal, color='blue')
    ax[0].set_title('Sinyal respirasi')
    ax[1].plot(rppg_signal, color='red')
    ax[1].set_title('Sinyal rppg')
    plt.tight_layout()
    plt.show()


    fig, ax = plt.subplots(2, 1, figsize=(40, 5))
    ax[0].plot(sinyal_respirasi_filter, color='blue')
    ax[0].set_title(f'Sinyal Respirasi, BPM: {breaths_per_minute:.2f})')
    ax[0].plot(puncak_respirasi, sinyal_respirasi_filter[puncak_respirasi], 'rx', label='Detected
    Breaths')
    ax[1].plot(sinyal_rppg_filter, color='red')
    ax[1].set_title(f'Sinyal RPPG, Denyut Jantung : {heart_rate:.2f})')
    ax[1].plot(puncak_rppg, sinyal_rppg_filter[puncak_rppg], 'rx', label='Detected Breaths')
```

```
    plt.tight_layout()
    plt.show()
```

Kode 7: Import Library

Terakhir, sinyal mentah dan sinyal hasil filtering ditampilkan menggunakan Matplotlib. Grafik dilengkapi dengan anotasi puncak untuk menunjukkan titik-titik siklus napas dan detak jantung yang terdeteksi.

3.2 Modul pemrosesan_respirasi.py

3.2.1 Fungsi get_initial_roi()

```

    def get_initial_roi(image, landmarker, x_size=100, y_size=100, shift_x=0, shift_y=0):
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        height, width = image.shape[:2]

        from mediapipe import Image, ImageFormat
        mp_image = Image(image_format=ImageFormat.SRGB, data=image_rgb)
        detection_result = landmarker.detect(mp_image)

        if not detection_result.pose_landmarks:
            raise ValueError("No pose detected in the frame")

        landmarks = detection_result.pose_landmarks[0]
        left_shoulder = landmarks[11]
        right_shoulder = landmarks[12]

        center_x = int((left_shoulder.x + right_shoulder.x) * width / 2)
        center_y = int((left_shoulder.y + right_shoulder.y) * height / 2)

        center_x += shift_x
        center_y += shift_y

        left_x = max(0, center_x - x_size)
        right_x = min(width, center_x + x_size)
        top_y = max(0, center_y - y_size)
        bottom_y = min(height, center_y)

        if (right_x - left_x) <= 0 or (bottom_y - top_y) <= 0:
            raise ValueError("Invalid ROI dimensions")

        return (left_x, top_y, right_x, bottom_y)
```

Kode 8: Import Library

Fungsi ini bertujuan untuk menentukan Region of Interest (ROI), yaitu area pada bagian dada berdasarkan posisi bahu pengguna yang terdeteksi dari citra input. Gambar diubah menjadi format RGB dan dikirim ke pose landmarker dari MediaPipe. Dari hasil deteksi, diambil koordinat bahu kiri dan kanan (landmark 11 dan 12). Titik tengah dari dua bahu ini dihitung sebagai pusat ROI. Selanjutnya, ROI didefinisikan sebagai sebuah persegi panjang dengan lebar dan tinggi tertentu di sekitar titik tersebut, serta dapat disesuaikan dengan parameter shift_x dan shift_y. Validasi dilakukan untuk memastikan bahwa ROI valid secara spasial.

3.2.2 Fungsi initialize_feature()

```

    def initialize_features(frame, pose_landmarker, STANDARD_SIZE):
        roi_coords = get_initial_roi(frame, pose_landmarker)
        left_x, top_y, right_x, bottom_y = roi_coords
```

```
5
6       frame = cv2.resize(frame, STANDARD_SIZE)
7       old_frame = frame.copy()
8       old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
9       roi_chest = old_gray[top_y:bottom_y, left_x:right_x]
10      features = cv2.goodFeaturesToTrack(roi_chest, maxCorners=50, qualityLevel=0.2, minDistance
    =5, blockSize=3)
11
12      if features is None:
13          raise ValueError("No features found to track!")
14      features = np.float32(features)
15      features[:,:,0] += left_x
16      features[:,:,1] += top_y
17
18      lk_params = dict(winSize=(15, 15), maxLevel=2,
19                       criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
20
21      return features, lk_params, old_gray, left_x, top_y, right_x, bottom_y
```

Kode 9: Import Library

Fungsi ini bertugas menginisialisasi fitur-fitur visual pada area dada yang akan digunakan dalam pelacakan pergerakan (optical flow). Setelah ROI didapatkan dari fungsi sebelumnya, frame diubah ukurannya dan dikonversi ke grayscale. Kemudian, fitur-fitur yang bisa dilacak (misalnya sudut atau tepi) diidentifikasi dalam area ROI menggunakan fungsi cv2.goodFeaturesToTrack. Koordinat fitur dikalibrasi ulang terhadap posisi sebenarnya dalam gambar. Parameter untuk algoritma Lucas-Kanade Optical Flow juga disiapkan dalam bentuk dictionary lk_params. Fungsi ini mengembalikan fitur-fitur awal yang dapat dilacak, parameter pelacakan, citra grayscale awal, serta koordinat ROI sebagai output.

3.3 Modul pemrosesan_rppg.py

3.3.1 Fungsi POS()

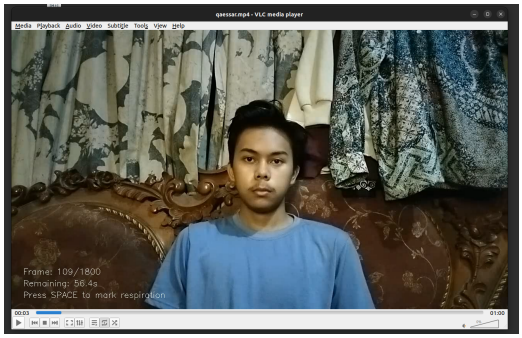
```
1
2   def POS(signal, **kargs):
3       eps = 10**-9
4       X = signal
5       e, c, f = X.shape
6       w = int(1.6 * kargs['fps'])
7
8
9       P = np.array([[0, 1, -1], [-2, 1, 1]])
10      Q = np.stack([P for _ in range(e)], axis = 0)
11
12      H = np.zeros((e, f))
13      for n in np.arange(w, f):
14          m = n - w + 1
15
16          Cn = X[:, :, m:(n+1)]
17          M = 1.0 / (np.mean(Cn, axis = 2) + eps)
18          M = np.expand_dims(M, axis=2) # shape [e, c, w]
19          Cn = np.multiply(Cn, M)
20
21          S = np.dot(Q, Cn)
22          S = S[0, :, :, :]
23          S = np.swapaxes(S, 0, 1)
24
25          S1 = S[:, 0, :]
26          S2 = S[:, 1, :]
27          alpha = np.std(S1, axis=1) / (eps + np.std(S2, axis=1))
28          alpha - np.expand_dims(alpha, axis=1)
```

```
            Hn = np.add(S1, alpha * S2)
            Hnm = Hn - np.expand_dims(np.mean(Hn, axis=1), axis=1)

            H[:, m:(n + 1)] = np.add(H[:, m:(n + 1)], Hnm)  # Add the tuned signal to the output matrix

    return H
```

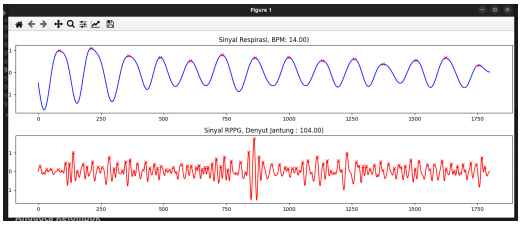
Kode 10: Import Library

Fungsi POS (Plane-Orthogonal-to-Skin) merupakan algoritma untuk mengekstraksi sinyal photoplethysmography (rPPG) dari video RGB wajah secara non-kontak. Fungsi ini bekerja dengan membagi sinyal RGB menjadi jendela waktu tertentu (berdasarkan frame rate), lalu menormalkan nilai warnanya untuk mengurangi efek pencahayaan. Selanjutnya, sinyal diproyeksikan ke ruang warna tertentu menggunakan matriks transformasi agar dapat memaksimalkan komponen yang mengandung informasi detak jantung. Dua sinyal hasil proyeksi digabung dengan bobot penyesuaian (alpha), lalu dikurangi rata-ratanya untuk menghilangkan offset. Proses ini diulang untuk tiap jendela waktu, dan sinyal gabungan disimpan dalam matriks output yang kemudian dikembalikan sebagai sinyal rPPG akhir.

4 Hasil



(a) Video yang Dipakai



(b) Sinyal yang didapatkan

Gambar 1: Augmentation Samples

Dari hasil pemrosesan video menggunakan metode optical flow untuk ekstraksi sinyal respirasi dan metode POS (Plane-Orthogonal-to-Skin) untuk ekstraksi sinyal rPPG (remote photoplethysmography), diperoleh dua sinyal fisiologis utama, yaitu sinyal respirasi dan sinyal denyut jantung. Sinyal respirasi dihasilkan dengan melacak pergerakan titik-titik fitur pada area dada menggunakan algoritma Lucas-Kanade optical flow. Sinyal ini menunjukkan pola gelombang yang periodik dan stabil setelah melalui proses bandpass filter dengan rentang frekuensi 0.1–0.5 Hz. Sinyal ini dianalisis lebih lanjut dengan metode pencarian puncak (peak detection) untuk menghitung frekuensi napas, yang menghasilkan estimasi laju respirasi sebesar 14 napas per menit (BPM).

Sementara itu, sinyal rPPG diperoleh dari perubahan nilai rata-rata intensitas kanal warna RGB di area wajah yang dideteksi secara otomatis menggunakan model BlazeFace. Nilai RGB tersebut diolah menggunakan algoritma POS dan difilter menggunakan bandpass filter pada frekuensi 0.7–2.5 Hz. Hasilnya menunjukkan pola denyut yang bervariasi namun tetap menunjukkan kehadiran puncak-puncak yang konsisten. Setelah dilakukan deteksi puncak, diperoleh estimasi denyut jantung sebesar 104 denyut per menit (BPM).

5 Kesimpulan

Berdasarkan hasil yang diperoleh, sistem ini mampu melakukan ekstraksi sinyal fisiologis dari video secara non-kontak dengan akurasi estimasi yang baik. Deteksi sinyal respirasi melalui pergerakan dada menunjukkan kestabilan dan bentuk gelombang yang mendekati ideal, sehingga cocok digunakan untuk pemantauan pernapasan. Sementara itu, meskipun sinyal rPPG lebih dipengaruhi oleh noise dan fluktuasi intensitas cahaya, metode POS masih mampu mengekstrak informasi denyut jantung dengan cukup baik. Kedua pendekatan ini membuktikan bahwa kombinasi antara pengolahan citra berbasis landmark pose dan analisis intensitas warna wajah dapat dimanfaatkan secara efektif dalam sistem pemantauan kesehatan berbasis video. Ke depan, peningkatan akurasi dapat dicapai melalui optimalisasi preprocessing, peningkatan kualitas deteksi wajah, dan penerapan metode filtering yang lebih adaptif terhadap noise lingkungan.

References

[1] S. Premkumar and D. J. Hemanth, "Intelligent remote photoplethysmography-based methods for heart rate estimation from face videos: A survey," *MDPI*, vol. 9, no. 3, p. 57, 2022, accessed: 2024-12-24. [Online]. Available: https://www.mdpi.com/2227-9709/9/3/57

[2] F. Guede-Fernández, M. Fernández-Chimeno, J. Ramos-Castro, and M. A. García-González, "Driver drowsiness detection based on respiratory signal analysis," in *IEEE Xplore*, 2019, accessed: 2024-12-24. [Online]. Available: https://xplorestaging.ieee.org/ielx7/6287639/8600701/08744224.pdf?arnumber=8744224