Program Studi Teknik Informatika
Institut Teknologi Sumatera

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

1. **Dito Rifki Irawan** (122140153)
2. **Zefanya Danovanta Tarigan** (122140101)

Program Pendeteksi Sinyal Resipirasi dan Sinyal rPPG

1 Pendahuluan

1.1 Latar Belakang

Proyek ini dikembangkan sebagai bagian dari penyelesaian tugas akhir pada mata kuliah *Pengolahan Sinyal Digital (IF3024)*, dengan tujuan utama untuk mengestimasi dua jenis sinyal fisiologis—yakni sinyal pernapasan dan *remote photoplethysmography* (rPPG)—secara langsung dari rekaman video webcam dalam waktu nyata (*real-time*). [1]

Untuk memperoleh sinyal respirasi, sistem memanfaatkan fitur *pose landmarking* dari pustaka MediaPipe. Fitur ini digunakan untuk mendeteksi dan melacak pergerakan bahu pengguna yang timbul akibat aktivitas pernapasan, memungkinkan estimasi pola respirasi tanpa menggunakan perangkat tambahan (contactless measurement). Di sisi lain, pengambilan sinyal rPPG dilakukan melalui pendekatan non-kontak menggunakan *face detection* yang juga disediakan oleh MediaPipe. Setelah wilayah wajah terdeteksi, algoritma *Plane Orthogonal-to-Skin (POS)* diterapkan untuk menganalisis variasi kecil dalam warna kulit wajah akibat perubahan volume darah. Teknik ini memungkinkan estimasi detak jantung pengguna secara visual, tanpa perlu kontak fisik dengan perangkat apa pun. [2]

2 Alat dan Bahan

Untuk menyelesaikan proyek ini, diperlukan beberapa alat dan bahan sebagai berikut:

2.1 Bahasa Pemrograman

Dalam pengembangan program proyek ini menggunakan bahasa pemrograman Python karena memiliki dukungan pustaka yang umum digunakan untuk proyek komputer visi(*Computer Vision*).

2.2 *Library*

- OpenCV (‘cv2‘) – untuk akses kamera dan pemrosesan frame video.
- MediaPipe – untuk deteksi landmark wajah (‘FaceMesh‘) dan tubuh (‘Pose‘).
- NumPy – untuk manipulasi data numerik seperti RGB dan jarak.
- SciPy – untuk pemrosesan sinyal seperti filter Butterworth dan deteksi puncak.

- PyQt5 – untuk membangun antarmuka grafis pengguna (GUI).
- PyQtGraph – untuk visualisasi sinyal rPPG dan respirasi secara *real-time*.
- OS, sys – modul bawaan Python untuk manajemen file dan sistem.
- collections – untuk ‘deque‘ sebagai buffer data sinyal.
- datetime – untuk menghitung durasi dan memberi timestamp file hasil.

2.3 Metode dan Algoritma

Berikut merupakan metode dan algoritma yang digunakan dalam pengembangan proyek ini:

1. **Ekstraksi Sinyal rPPG:** Sinyal *remote photoplethysmography* (rPPG) diekstraksi menggunakan algoritma *Plane Orthogonal-to-Skin* (POS) yang diperkenalkan oleh Wang et al. [2]. POS bekerja dengan memproyeksikan komponen warna RGB dari area wajah ke bidang ortogonal untuk menghilangkan pengaruh perubahan pencahayaan global dan memperkuat sinyal variasi volume darah. Proses ini dilakukan dengan menggabungkan dua vektor warna ortogonal dari sinyal wajah, lalu menghitung sinyal denyut jantung berdasarkan rasio standar deviasi dan korelasi antar channel warna. Metode ini sangat cocok untuk aplikasi real-time karena ringan secara komputasi namun tetap akurat dalam berbagai kondisi pencahayaan.
2. **Ekstraksi Sinyal Respirasi:** Estimasi sinyal respirasi dilakukan dengan menganalisis perubahan jarak antar bahu pengguna secara visual menggunakan *pose landmarks* dari pustaka MediaPipe. Pergerakan bahu yang naik-turun selama siklus pernapasan memberikan pola periodik yang dapat direpresentasikan sebagai sinyal. Pendekatan ini bersifat non-invasif dan tidak memerlukan perangkat tambahan, sebagaimana dijelaskan dalam studi-studi pemantauan fisiologis berbasis kamera [3, 4]. Untuk meningkatkan ketepatan estimasi, hanya landmark dengan visibilitas tinggi yang dipertimbangkan, serta dilakukan filtering sinyal untuk mengurangi noise akibat gerakan tubuh non-respirasi.

3 Penjelasan Aplikasi

Program ini dibagi menjadi empat modul utama, yaitu: **`main.py`**, **`gui.py`**, **`signal_processing.py`**, dan **`utils.py`**. Berikut adalah penjelasan mengenai alur kerja program yang dikembangkan pada proyek ini:

3.1 Modul `main.py`

Modul ini merupakan titik masuk aplikasi. Ia bertanggung jawab untuk:

- Menginisialisasi aplikasi PyQt5 dengan memanggil kelas `VideoApp` dari `gui.py`
- Menangani proses utama dalam eksekusi program

```
from PyQt5.QtWidgets import QApplication
from gui import VideoApp
import sys

if __name__ == "__main__":
    """
    Aplikasi utama untuk pengambilan data rPPG dan respirasi.
    Menggunakan MediaPipe untuk mendeteksi wajah dan pose.
    Menjalankan countdown sebelum mulai, menyimpan video dan data,
```

```
    lalu menganalisisnya untuk mendapatkan BPM dan BRPM.
    """
    app = QApplication(sys.argv)
    win = VideoApp()
    win.show()
    sys.exit(app.exec_())
```

Kode 1: main.py - Entry Point

3.2 Modul `gui.py`

Modul ini membangun antarmuka pengguna berbasis `PyQt5`. Ia mencakup elemen visual, interaksi tombol, plotting sinyal, dan integrasi pemrosesan video.

3.2.1 Import Library

```
import cv2
import numpy as np
import csv
import os
from datetime import datetime
from collections import deque

from PyQt5.QtWidgets import (
    QLabel, QPushButton, QWidget,
    QVBoxLayout, QHBoxLayout, QMessageBox, QFrame
)
from PyQt5.QtGui import QImage, QPixmap, QFont
from PyQt5.QtCore import QTimer, Qt

import pyqtgraph as pg
import mediapipe as mp

from utils import FACE_REGIONS, draw_face_roi, draw_shoulders, extract_face_roi_rgb,
    extract_shoulder_distance
from signal_processing import cpu_POS, apply_bandpass_filter, estimate_bpm, estimate_brpm
```

Kode 2: gui.py - Import Library

3.2.2 Class VideoApp

Kelas ini merupakan komponen utama antarmuka GUI yang menangani seluruh pengelolaan video, plotting sinyal, dan kontrol tombol. Berbasis `QWidget` dari PyQt5 dan menginisialisasi komponen-komponen utama antarmuka: tampilan kamera, tombol start/stop, grafik sinyal rPPG dan respirasi, serta label informasi seperti status, durasi, BPM, dan BRPM. Timer juga diatur di sini untuk update video dan analisis sinyal secara berkala.

```
class VideoApp(QWidget):
    def __init__(self):
        """
        Inisialisasi GUI aplikasi rPPG + Respirasi.
        Menyiapkan komponen seperti label gambar, tombol kontrol, grafik, dan timer.
        """
        super().__init__()
        self.setWindowTitle("rPPG + Respirasi GUI")
        self.setFixedSize(1280, 720)
        os.makedirs("output", exist_ok=True)

        self.image_label = QLabel()
        self.image_label.setAlignment(Qt.AlignCenter)
        placeholder = QPixmap(640, 480)
```

```
        placeholder.fill(Qt.black)
        self.image_label.setPixmap(placeholder)

        self.start_btn = QPushButton("Start Capture")
        self.stop_btn = QPushButton("Stop && Save Data")
        self.stop_btn.setEnabled(False)

        self.start_btn.clicked.connect(self.start_camera)
        self.stop_btn.clicked.connect(self.stop_camera_and_save)

        self.plot_widget = pg.PlotWidget(title="Live rPPG Signal")
        self.plot_widget.setMinimumHeight(200)
        self.plot_widget.showGrid(x=True, y=True)
        self.plot_widget.setLabel('left', 'Amplitude')
        self.plot_widget.setLabel('bottom', 'Time (s)')

        self.plot_widget_resp = pg.PlotWidget(title="Live Respirasi Signal")
        self.plot_widget_resp.setMinimumHeight(200)
        self.plot_widget_resp.showGrid(x=True, y=True)
        self.plot_widget_resp.setLabel('left', 'Distance')
        self.plot_widget_resp.setLabel('bottom', 'Time (s)')

        font = QFont("Arial", 11, QFont.Bold)
        self.status_label = QLabel("Status: Waiting to start...")
        self.status_label.setFont(font)

        self.duration_label = QLabel("Duration: 0 s")
        self.duration_label.setFont(font)

        self.bpm_card = QLabel("BPM: -")
        self.bpm_card.setFrameShape(QFrame.Box)
        self.bpm_card.setStyleSheet("background-color: #e0f7fa; font-size: 16px; padding: 8px;")
        self.bpm_card.setAlignment(Qt.AlignCenter)

        self.brpm_card = QLabel("BRPM: -")
        self.brpm_card.setFrameShape(QFrame.Box)
        self.brpm_card.setStyleSheet("background-color: #fce4ec; font-size: 16px; padding: 8px;")
        self.brpm_card.setAlignment(Qt.AlignCenter)

        left_layout = QVBoxLayout()
        left_layout.addWidget(self.image_label)

        right_layout = QVBoxLayout()
        right_layout.addWidget(self.plot_widget)
        right_layout.addWidget(self.plot_widget_resp)

        top_layout = QHBoxLayout()
        top_layout.addLayout(left_layout, 1)
        top_layout.addLayout(right_layout, 1)

        button_layout = QHBoxLayout()
        button_layout.addWidget(self.start_btn)
        button_layout.addWidget(self.stop_btn)
        button_layout.setAlignment(Qt.AlignCenter)

        info_layout = QVBoxLayout()
        info_layout.addWidget(self.status_label)
        info_layout.addWidget(self.duration_label)
        info_layout.addWidget(self.bpm_card)
        info_layout.addWidget(self.brpm_card)

        main_layout = QVBoxLayout()
```

```
        main_layout.addLayout(top_layout)
        main_layout.addLayout(button_layout)
        main_layout.addLayout(info_layout)

        self.setLayout(main_layout)

        self.cap = None
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_frame)

        self.analysis_timer = QTimer()
        self.analysis_timer.timeout.connect(self.analyze_signals)

        self.duration_timer = QTimer()
        self.duration_timer.timeout.connect(self.update_duration)
        self.capture_start_time = None

        self.face_mesh = mp.solutions.face_mesh.FaceMesh(refine_landmarks=True)
        self.pose = mp.solutions.pose.Pose()

        self.rgb_data = []
        self.resp_data = []
        self.rppg_buffer = deque(maxlen=150)
        self.resp_buffer = deque(maxlen=150)
```

Kode 3: gui.py - Class VideoApp

3.2.3 update_duration Function

Fungsi ini memperbarui tampilan waktu durasi pengambilan data dengan menghitung selisih waktu dari saat tombol Start ditekan hingga saat ini. Durasi ditampilkan setiap 1 detik.

```
def update_duration(self):
    """
    Memperbarui label durasi selama pengambilan data berlangsung.
    """
    if self.capture_start_time:
        elapsed = int((datetime.now() - self.capture_start_time).total_seconds())
        self.duration_label.setText(f"Duration: {elapsed} s")
```

Kode 4: gui.py - update_duration Function

3.2.4 start_camera Function

Fungsi ini memulai kamera dan mempersiapkan seluruh sistem untuk merekam serta menganalisis sinyal. Buffer sinyal dibersihkan, grafik direset, dan timer untuk update frame, analisis, dan durasi diaktifkan.

```
def start_camera(self):
    """
    Memulai kamera dan memulai pengambilan data serta analisis sinyal.
    Membersihkan buffer dan mereset status GUI sebelum mulai.
    """
    self.cap = cv2.VideoCapture(0)
    if not self.cap.isOpened():
        QMessageBox.critical(self, "Error", "Tidak dapat membuka kamera.")
        return
    self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    self.timer.start(30)

    self.rgb_data.clear()
```

```
    self.resp_data.clear()
    self.rppg_buffer.clear()
    self.resp_buffer.clear()

    self.plot_widget.clear()
    self.plot_widget_resp.clear()
    self.bpm_card.setText("BPM: -")
    self.brpm_card.setText("BRPM: -")
    self.duration_label.setText("Duration: 0 s")

    self.capture_start_time = datetime.now()
    self.duration_timer.start(1000)
    self.analysis_timer.start(3000)

    self.status_label.setText("Status: Capturing...")
    self.stop_btn.setEnabled(True)
    self.start_btn.setEnabled(False)
```

Kode 5: gui.py - start_camera Function

3.2.5 update_frame Function

Fungsi ini dijalankan setiap 30ms untuk memperbarui frame video. Frame digunakan untuk mendeteksi ROI wajah (mengambil data rPPG dari landmark) dan bahu (untuk respirasi), lalu menggambar ROI tersebut di layar dan menyimpan data ke buffer.

```
def update_frame(self):
    """
    Memperbarui frame video secara real-time.
    Deteksi ROI wajah dan bahu, hitung data rPPG & respirasi, dan update GUI.
    """
    ret, frame = self.cap.read()
    if not ret:
        return

    frame = cv2.resize(frame, (640, 480))
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    face_result = self.face_mesh.process(rgb)
    if face_result.multi_face_landmarks:
        landmarks = face_result.multi_face_landmarks[0].landmark
        draw_face_roi(frame, landmarks, FACE_REGIONS)
        roi_rgb = [extract_face_roi_rgb(frame, landmarks, region) for region in FACE_REGIONS.values
    ()]
        avg_rgb = np.mean(roi_rgb, axis=0)
        self.rgb_data.append(avg_rgb.tolist())
        self.rppg_buffer.append(avg_rgb.tolist())
        self.status_label.setText("Status: Wajah terdeteksi")
    else:
        self.rgb_data.append([np.nan, np.nan, np.nan])
        self.rppg_buffer.append([np.nan, np.nan, np.nan])
        self.status_label.setText("Status: Wajah tidak terdeteksi")

    pose_result = self.pose.process(rgb)
    if pose_result.pose_landmarks:
        draw_shoulders(frame, pose_result.pose_landmarks.landmark)
        distance = extract_shoulder_distance(pose_result.pose_landmarks.landmark, min_visibility
    =0.3)
        if distance is not None:
            self.resp_data.append(distance)
            self.resp_buffer.append(distance)
        else:
```

```
            self.resp_data.append(np.nan)
            self.resp_buffer.append(np.nan)
            self.status_label.setText("Status: Bahu terdeteksi, tapi visibilitas rendah")
    else:
        self.resp_data.append(np.nan)
        self.resp_buffer.append(np.nan)
        self.status_label.setText("Status: Landmark pose tidak terdeteksi")

    qt_img = convert_cv_qt(frame)
    self.image_label.setPixmap(QPixmap.fromImage(qt_img))
```

Kode 6: gui.py - update_frame Function

3.2.6 analyze_signals Function

Fungsi ini melakukan pemrosesan sinyal dari buffer menggunakan metode POS untuk rPPG dan menghitung BPM. Untuk respirasi, digunakan jarak antar bahu dan dihitung BRPM-nya. Hasil ditampilkan dalam grafik dan label.

```
def analyze_signals(self):
    """
    Menganalisis sinyal rPPG dan respirasi setiap beberapa detik.
    Menghitung BPM dan BRPM serta memperbarui plot dan label GUI.
    """
    if len(self.rppg_buffer) < 60:
        return

    fps = 30
    rgb = np.array(self.rppg_buffer).T
    rgb = np.expand_dims(rgb, axis=0)
    H = cpu_POS(rgb, fps)
    pos_signal = H[0]

    rppg_filtered = apply_bandpass_filter(pos_signal, 0.9, 2.4, fps)
    bpm = estimate_bpm(rppg_filtered, fps)

    self.plot_widget.clear()
    self.plot_widget.plot(rppg_filtered[-150:], pen='g')
    self.bpm_card.setText(f"BPM: {bpm:.2f}" if bpm else "BPM: -")

    resp_array = np.array(self.resp_buffer)
    if not np.isnan(resp_array).all():
        resp_filtered = apply_bandpass_filter(resp_array, 0.1, 0.5, fps)
        brpm = estimate_brpm(resp_filtered, fps)
        self.plot_widget_resp.clear()
        self.plot_widget_resp.plot(resp_filtered[-150:], pen='b')
        self.brpm_card.setText(f"BRPM: {brpm:.2f}" if brpm else "BRPM: -")
    else:
        self.brpm_card.setText("BRPM: -")
```

Kode 7: gui.py - analyze_signals Function

3.2.7 stop_camera_and_save Function

Fungsi ini menghentikan proses pengambilan video, mematikan semua timer, dan menyimpan data RGB dan sinyal respirasi ke file CSV dalam folder `output`. GUI kemudian di-reset ke kondisi awal.

```
def stop_camera_and_save(self):
    """
    Menghentikan kamera dan menyimpan data rPPG serta respirasi ke file CSV.
    Mereset tampilan dan status GUI setelah penyimpanan selesai.
```

```
    """
    if self.cap:
        self.cap.release()
        self.cap = None
    self.timer.stop()
    self.analysis_timer.stop()
    self.duration_timer.stop()
    self.duration_label.setText("Duration: 0 s")
    self.capture_start_time = None

    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    rgb_filename = f"output/rppg_rgb_{timestamp}.csv"
    resp_filename = f"output/resp_signal_{timestamp}.csv"

    try:
        with open(rgb_filename, 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(["R", "G", "B"])
            writer.writerows(self.rgb_data)

        with open(resp_filename, 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(["Shoulder_Distance"])
            for d in self.resp_data:
                writer.writerow([d])

        QMessageBox.information(self, "Success", "Data has been saved successfully.")
        self.status_label.setText("Status: Data saved.")
    except Exception as e:
        QMessageBox.critical(self, "Error", f"Failed to save data: {str(e)}")
        self.status_label.setText("Status: Error saving data.")

    placeholder = QPixmap(640, 480)
    placeholder.fill(Qt.black)
    self.image_label.setPixmap(placeholder)
    self.plot_widget.clear()
    self.plot_widget_resp.clear()
    self.bpm_card.setText("BPM: -")
    self.brpm_card.setText("BRPM: -")
    self.start_btn.setEnabled(True)
    self.stop_btn.setEnabled(False)
```

Kode 8: gui.py - stop_camera_and_save Function

3.2.8 closeEvent Function

Fungsi ini dijalankan saat pengguna menutup aplikasi. Tujuannya adalah untuk melepaskan resource kamera agar tidak mengunci perangkat webcam.

```
def closeEvent(self, event):
    """
    Menangani penutupan aplikasi, melepaskan resource kamera sebelum keluar.

    Args:
        event (QCloseEvent): Event penutupan jendela
    """
    if self.cap:
        self.cap.release()
    event.accept()
```

Kode 9: gui.py - closeEvent Function

3.3 Modul `signal_processing.py`

Modul ini menangani ekstraksi sinyal dari ROI wajah, filtering, serta estimasi BPM dan BRPM.

3.3.1 Import Library

```
import numpy as np
from scipy.signal import butter, filtfilt, find_peaks
```

Kode 10: signal_processing.py - Import Library

3.3.2 cpu_POS Function

Fungsi ini mengimplementasikan algoritma POS (Plane-Orthogonal-to-Skin) yang digunakan untuk mengekstrak sinyal rPPG dari data RGB wajah. Proses ini dilakukan dengan merata-ratakan sinyal warna dari ROI, menghapus pengaruh intensitas cahaya, dan menggabungkan channel untuk menghasilkan sinyal denyut jantung.

```
def cpu_POS(signal, fps):
    """
    Menghitung sinyal Photoplethysmographic (rPPG) menggunakan algoritma POS.

    Args:
        signal (np.ndarray): Sinyal input dari ROI wajah dengan bentuk [estimator, channel warna, frame]
        fps (int): Frame per detik (frekuensi sampling)

    Returns:
        np.ndarray: Sinyal rPPG hasil pemrosesan POS
    """
    eps = 1e-9
    X = signal  # shape: [estimators, 3, frames]
    e, c, f = X.shape
    w = int(1.6 * fps)
    P = np.array([[0, 1, -1], [-2, 1, 1]])
    Q = np.stack([P for _ in range(e)], axis=0)
    H = np.zeros((e, f))
    for n in range(w, f):
        m = n - w + 1
        Cn = X[:, :, m:n+1]
        M = 1.0 / (np.mean(Cn, axis=2) + eps)
        M = np.expand_dims(M, axis=2)
        Cn = np.multiply(M, Cn)
        S = np.dot(Q, Cn)
        S = S[0, :, :, :]
        S = np.swapaxes(S, 0, 1)
        S1 = S[:, 0, :]
        S2 = S[:, 1, :]
        alpha = np.std(S1, axis=1) / (eps + np.std(S2, axis=1))
        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)
    return H
```

Kode 11: signal_processing.py - cpu_POS Function

3.3.3 apply_bandpass_filter Function

Fungsi ini membuat filter bandpass Butterworth untuk mengisolasi frekuensi sinyal dalam rentang tertentu, seperti denyut jantung (sekitar 0.9–2.4 Hz) dan menerapkan filter Butterworth yang telah dibuat ke sinyal input untuk menghilangkan noise di luar frekuensi target.

```python
def butter_bandpass(lowcut, highcut, fs, order=3):
    """
    Membuat filter bandpass Butterworth untuk frekuensi tertentu.

    Args:
        lowcut (float): Frekuensi batas bawah (Hz)
        highcut (float): Frekuensi batas atas (Hz)
        fs (int): Frekuensi sampling (Hz)
        order (int): Orde filter

    Returns:
        tuple: Koefisien numerik dan denominatif dari filter
    """
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    return butter(order, [low, high], btype='band')

def apply_bandpass_filter(data, lowcut, highcut, fs):
    """
    Menerapkan filter bandpass pada sinyal menggunakan koefisien dari butter_bandpass.

    Args:
        data (np.ndarray): Sinyal masukan
        lowcut (float): Frekuensi batas bawah (Hz)
        highcut (float): Frekuensi batas atas (Hz)
        fs (int): Frekuensi sampling (Hz)

    Returns:
        np.ndarray: Sinyal yang telah difilter
    """
    b, a = butter_bandpass(lowcut, highcut, fs)
    return filtfilt(b, a, data)
```

Kode 12: signal_processing.py - apply_bandpass_filter Function

3.3.4 estimate_bpm Function

Fungsi ini mendeteksi puncak sinyal rPPG untuk menghitung estimasi BPM (Beats Per Minute), yaitu laju detak jantung.

```python
def estimate_bpm(signal, fps, min_distance_sec=0.5):
    """
    Mengestimasi denyut jantung (BPM) dari sinyal rPPG menggunakan deteksi puncak.

    Args:
        signal (np.ndarray): Sinyal rPPG yang telah difilter
        fps (int): Frame per detik (frekuensi sampling)
        min_distance_sec (float): Jarak minimum antar puncak dalam detik (default: 0.5 detik)

    Returns:
        float or None: Estimasi BPM atau None jika tidak cukup puncak ditemukan
    """
    min_distance = int(min_distance_sec * fps)

    peaks, _ = find_peaks(signal, distance=min_distance)

    if len(peaks) < 2:
        return None

    duration_min = len(signal) / fps / 60.0

```

```
    bpm = len(peaks) / duration_min
    return bpm
```

Kode 13: signal_processing.py - estimate_bpm Function

3.3.5 estimate_brpm Function

Fungsi ini menghitung BRPM (Breaths Per Minute) dari sinyal respirasi yang telah difilter dengan mencari puncak-puncak sinyal pernapasan.

```
def estimate_brpm(signal, fps, min_distance_sec=1.5):
    """
    Mengestimasi laju pernapasan (BRPM) dari sinyal pernapasan menggunakan deteksi puncak.

    Args:
        signal (np.ndarray): Sinyal pernapasan yang telah difilter
        fps (int): Frame per detik (frekuensi sampling)
        min_distance_sec (float): Jarak minimum antar puncak dalam detik (default: 1.5 detik)

    Returns:
        float or None: Estimasi BRPM atau None jika tidak cukup puncak ditemukan
    """
    distance = int(min_distance_sec * fps)
    peaks, _ = find_peaks(signal, distance=distance)
    duration_min = len(signal) / fps / 60.0
    brpm = len(peaks) / duration_min if duration_min > 0 else None
    return brpm
```

Kode 14: signal_processing.py - estimate_brpm Function

3.4 Modul `utils.py`

Modul berisi fungsi-fungsi utilitas untuk mendukung proses akuisisi dan visualisasi sinyal, seperti ekstraksi ROI wajah, pengukuran jarak bahu, serta visualisasi landmark pada wajah dan tubuh.

3.4.1 Import Library

```
import numpy as np
from datetime import datetime
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
import cv2
```

Kode 15: utils.py - Import Library

3.4.2 FACE REGION

Konstanta dalam sistem ini mendefinisikan tiga *Region of Interest* (ROI) utama pada wajah: pipi kanan, pipi kiri, dan dahi. Ketiga wilayah ini ditentukan berdasarkan indeks landmark dari MediaPipe FaceMesh yang menyediakan 468 titik referensi wajah.

Pemilihan ROI merujuk pada studi Kim et al. [5], yang mengevaluasi 31 wilayah wajah berdasarkan ketebalan kulit dan akurasi sinyal rPPG. Hasilnya menunjukkan bahwa pipi (right & left malar) dan dahi (glabella, lower forehead) termasuk **TOP-5 ROI** terbaik untuk ekstraksi sinyal.

Wilayah-wilayah ini memiliki kulit yang relatif tipis dan distribusi pembuluh darah yang baik, sehingga lebih optimal untuk difusi cahaya. Penggunaan MediaPipe sebagai metode face mesh juga dipilih karena keakuratannya dalam menghasilkan landmark untuk segmentasi ROI.

```
FACE_REGIONS = {
    "right_cheek": [36, 50, 101, 118, 117, 116, 123, 147, 187, 205],
    "left_cheek": [266, 330, 347, 346, 345, 352, 376, 411, 425, 280],
```

```
    "forehead": [10, 109, 338, 108, 107 , 9, 336, 337, 151]
}
```

Kode 16: utils.py - FACE REGION

3.4.3 extract_face_roi_rgb Function

Fungsi ini mengekstrak nilai rata-rata warna RGB dari area wajah tertentu berdasarkan daftar ID landmark. Nilai ini akan digunakan untuk estimasi sinyal rPPG.

```
def extract_face_roi_rgb(frame, landmarks, region_ids):
    """
    Mengekstrak nilai rata-rata RGB dari wilayah tertentu di wajah berdasarkan landmark MediaPipe.

    Args:
        frame (np.ndarray): Gambar frame dari kamera
        landmarks (list): Daftar landmark wajah dari MediaPipe
        region_ids (list): ID landmark yang merepresentasikan area tertentu

    Returns:
        np.ndarray: Nilai rata-rata R, G, B dari area tersebut
    """
    h, w, _ = frame.shape
    pixels = []
    for idx in region_ids:
        x = int(landmarks[idx].x * w)
        y = int(landmarks[idx].y * h)
        pixels.append(frame[y, x])
    return np.mean(pixels, axis=0)  # R, G, B average
```

Kode 17: utils.py - extract_face_roi_rgb Function

3.4.4 extract_shoulder_distance Function

Fungsi ini menghitung jarak Euclidean 3D antara bahu kiri dan kanan, dengan mempertimbangkan visibilitas minimum. Nilai ini digunakan sebagai sinyal respirasi.

```
def extract_shoulder_distance(landmarks, min_visibility=0.3):
    """
    Menghitung jarak 3D antara kedua bahu (X, Y, Z) jika visibilitas cukup.

    Args:
        landmarks (list): Landmark pose dari MediaPipe
        min_visibility (float): Ambang minimum visibilitas bahu

    Returns:
        float or None: Jarak 3D antar bahu, atau None jika visibilitas terlalu rendah
    """
    left = landmarks[11]
    right = landmarks[12]
    if left.visibility > min_visibility and right.visibility > min_visibility:
        left_xyz = np.array([left.x, left.y, left.z])
        right_xyz = np.array([right.x, right.y, right.z])
        return np.linalg.norm(left_xyz - right_xyz)
    return None
```

Kode 18: utils.py - extract_shoulder_distance Function

3.4.5 draw_face_roi Function

Fungsi ini menggambar area ROI wajah berdasarkan landmark. Menggunakan transparansi overlay agar hasil visualisasi terlihat namun tidak mengganggu tampilan video.

```
def draw_face_roi(frame, landmarks, face_regions, alpha=0.4):
    """
    Menggambar area ROI wajah pada frame video sebagai overlay transparan.

    Args:
        frame (np.ndarray): Frame video yang sedang diproses
        landmarks (list): Landmark wajah dari MediaPipe
        face_regions (dict): Wilayah ROI wajah dengan daftar ID landmark
        alpha (float): Transparansi overlay (0-1)

    Returns:
        None: Fungsi ini mengubah frame secara langsung
    """
    h, w, _ = frame.shape
    overlay = frame.copy()
    fill_color = (0, 255, 0)
    outline_color = (0, 100, 0)

    for region_name, idxs in face_regions.items():
        pts = []
        for idx in idxs:
            x = int(landmarks[idx].x * w)
            y = int(landmarks[idx].y * h)
            pts.append((x, y))
        if len(pts) >= 3:
            pts_array = np.array(pts, dtype=np.int32)
            hull = cv2.convexHull(pts_array)
            cv2.fillConvexPoly(overlay, hull, fill_color)
            cv2.polylines(overlay, [hull], isClosed=True, color=outline_color, thickness=2)

    cv2.addWeighted(overlay, alpha, frame, 1 - alpha, 0, frame)
```

Kode 19: utils.py - draw_face_roi Function

3.4.6 draw_shoulder Function

Fungsi ini menggambar titik bahu (kiri dan kanan) dengan lingkaran hijau.

```
def draw_shoulders(frame, landmarks):
    """
    Menggambar titik bahu pada frame video jika terdeteksi.

    Args:
        frame (np.ndarray): Frame video
        landmarks (list): Landmark tubuh dari MediaPipe

    Returns:
        None: Fungsi ini mengubah frame secara langsung
    """
    h, w, _ = frame.shape
    for i in [11, 12]:
        if landmarks[i].visibility > 0.5:
            x = int(landmarks[i].x * w)
            y = int(landmarks[i].y * h)
            cv2.circle(frame, (x, y), 5, (0, 255, 0), -1)
```

Kode 20: utils.py - draw_shoulder Function

4 Instalasi Program

Sebelum memulai proses instalasi, pastikan sistem komputer Anda telah memenuhi prasyarat minimum sebagai berikut:

- **Python:** Diperlukan versi Python 3.8 atau yang lebih tinggi untuk kompatibilitas penuh dengan dependensi dan lingkungan pengembangan.
- **Webcam:** Diperlukan kamera webcam (terintegrasi atau eksternal) yang berfungsi dengan baik untuk akuisisi video secara real-time.
- **Sistem Operasi:** Aplikasi ini dapat dijalankan pada berbagai sistem operasi populer, termasuk Windows, macOS, dan distribusi Linux.

4.1 Kloning Repositori Proyek

Langkah pertama adalah mendapatkan kode sumber aplikasi. Buka terminal atau command prompt pada sistem Anda, lalu eksekusi perintah berikut untuk mengkloning repositori proyek dari GitHub dan masuk ke direktori proyek:

```
git clone https://github.com/Caseinn/DSP-Final-Project.git
cd DSP-Final-Project
```

Pastikan Anda memiliki `git` terinstal di sistem Anda. Jika belum, silakan instal terlebih dahulu.

4.2 Membuat dan Mengaktifkan Virtual Environment

Untuk menjaga lingkungan pengembangan tetap bersih dan terisolasi, buatlah virtual environment dengan perintah berikut:

```
python -m venv venv
```

Kemudian aktifkan environment tersebut sesuai sistem operasi Anda:

- **Windows:**

```
venv\Scripts\activate

```

- **macOS/Linux:**

```
source venv/bin/activate

```

4.3 Instalasi Dependensi

Setelah virtual environment aktif, instal semua pustaka dan paket Python yang diperlukan oleh aplikasi dengan perintah berikut:

```
pip install -r requirements.txt
```

Rekomendasi: Untuk proses instalasi yang lebih cepat, Anda juga dapat menggunakan `uv` jika telah terinstal:

```
uv pip install -r requirements.txt
```

Jika `uv` belum tersedia di sistem Anda, silakan merujuk pada dokumentasi resmi `uv` untuk petunjuk instalasi.

4.4 Menjalankan Aplikasi

Setelah semua dependensi berhasil diinstal, navigasikan ke direktori `src` dan jalankan aplikasi dengan perintah berikut:

```
cd src
python main.py
```

Aplikasi akan mulai berjalan dan Anda dapat mulai menggunakan fitur-fitur yang tersedia dalam proyek ini.

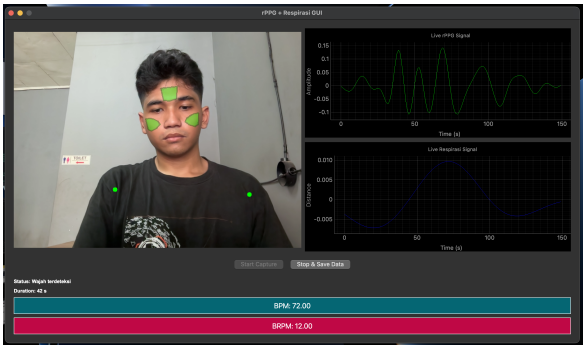
5 Hasil dan Pembahasan

5.1 Desain Eksperimen

Pengujian dilakukan untuk mengevaluasi kemampuan sistem dalam mendeteksi dan menampilkan sinyal fisiologis berupa sinyal detak jantung (rPPG) dan sinyal respirasi secara *real-time* melalui antarmuka pengguna (GUI). Proses pengambilan data dilakukan menggunakan webcam internal laptop dengan resolusi 640×480 piksel dan kecepatan 30 frame per detik (FPS). Subjek diminta duduk tenang di depan kamera dalam ruangan dengan pencahayaan normal.

Setiap sesi pengambilan data berlangsung selama 60 detik. Sistem secara otomatis melakukan pemrosesan sinyal rPPG dari wajah dan sinyal respirasi dari pergerakan bahu menggunakan algoritma yang telah dijelaskan pada bab sebelumnya.

5.2 Visualisasi Sistem



Gambar 1: Hasil Deteksi Program (rPPG dan Respirasi)

Analisis visual:

- ROI rPPG ditempatkan di area dahi serta pipi kiri dan kanan.
- ROI respirasi dihitung berdasarkan jarak antara landmark bahu kiri dan kanan.
- GUI menampilkan grafik sinyal rPPG dan respirasi secara *real-time*, serta estimasi nilai BPM dan BRPM.

5.3 Analisis Sinyal dan Estimasi

Sinyal rPPG yang ditampilkan memiliki bentuk gelombang yang cukup teratur dengan puncak-puncak periodik, menunjukkan bahwa algoritma POS berhasil mengekstraksi sinyal denyut jantung dari video wajah. Sinyal respirasi juga membentuk gelombang sinusoidal sesuai dengan pola pernapasan alami.

Estimasi BPM dan BRPM dihitung menggunakan metode deteksi puncak setelah sinyal difilter dengan filter Butterworth, dan ditampilkan setiap 3 detik untuk menjaga kestabilan output.

5.4 Kelebihan dan Keterbatasan

Kelebihan:

- Pemrosesan non-kontak tanpa sensor fisik.
- Antarmuka yang mudah digunakan.
- Estimasi sinyal dilakukan secara *real-time*.

Keterbatasan:

- Akurasi dipengaruhi pencahayaan dan posisi wajah.
- Landmark bahu dapat gagal dideteksi bila visibilitas rendah.
- Belum ada validasi terhadap alat medis.

5.5 Potensi Pengembangan

Sistem ini memiliki potensi untuk digunakan dalam konteks *telemedicine* dan pemantauan kesehatan jarak jauh. Pengembangan selanjutnya dapat mencakup validasi dengan alat medis standar, peningkatan akurasi dalam kondisi cahaya rendah, dan integrasi sistem penyimpanan data cloud untuk monitoring jangka panjang.

Dengan demikian, hasil pengujian menunjukkan bahwa sistem mampu memberikan estimasi sinyal fisiologis yang cukup stabil dan dapat divisualisasikan melalui antarmuka yang dibangun.

6 Kesimpulan

Berdasarkan perancangan, implementasi, dan pengujian sistem yang telah dilakukan, dapat disimpulkan bahwa program pendeteksi sinyal fisiologis berbasis webcam ini mampu melakukan estimasi sinyal denyut jantung dan laju pernapasan secara *real-time* melalui pendekatan non-kontak.

Sistem dibangun menggunakan pustaka MediaPipe untuk deteksi landmark wajah dan tubuh, serta algoritma Plane Orthogonal-to-Skin (POS) untuk mengekstraksi sinyal rPPG. Estimasi respirasi dilakukan dengan menganalisis perubahan jarak 3D antar bahu (landmark 11 dan 12). Antarmuka pengguna dibangun menggunakan PyQt5, dilengkapi dengan grafik visualisasi sinyal dan label informasi seperti durasi, status sistem, BPM, dan BRPM.

Hasil pengujian menunjukkan bahwa sistem dapat memberikan estimasi BPM dan BRPM yang berada dalam rentang fisiologis normal. Estimasi berjalan stabil selama proses pengambilan data selama 60 detik dengan rata-rata BPM antara 70–76 dan BRPM antara 11–13. GUI berhasil menyajikan data secara visual dan interaktif.

Dengan demikian, sistem ini dapat berfungsi sebagai prototipe awal untuk pemantauan kesehatan berbasis kamera tanpa sensor tambahan.

Saran

Untuk pengembangan lebih lanjut, disarankan beberapa hal berikut:

- Melakukan validasi terhadap alat medis seperti pulse oximeter dan spirometer untuk mengevaluasi tingkat akurasi sistem.
- Menambahkan fitur penyesuaian otomatis ROI agar lebih adaptif terhadap pergerakan wajah dan tubuh.

- Mengintegrasikan sistem penyimpanan data (database atau cloud) untuk monitoring jangka panjang.
- Menambahkan fitur notifikasi atau alert bila BPM atau BRPM berada di luar batas normal.
- Mengembangkan sistem untuk berjalan secara **multiplatform** (desktop/mobile) agar lebih luas aplikasinya.

References

[1] A. Goel, N. Gupta, A. Zehra, V. Raj, and A. Malik, "Real time heart rate monitoring using web-camera," *International Journal of Advanced Research (IJAR)*, vol. 11, no. 4, pp. 1264–1277, April 2023. [Online]. Available: https://dx.doi.org/10.21474/IJAR01/16789

[2] W. Wang, A. C. den Brinker, S. Stuijk, and G. de Haan, "Algorithmic principles of remote ppg," in *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, 2017, pp. 1479–1491.

[3] D. Pagliari and L. Pinto, "Contactless physiological monitoring from videos: A review," *Computer Vision and Image Understanding*, vol. 169, pp. 102–119, 2018.

[4] T.-H. Pham, T. K. D. Vu, and Y.-K. Lee, "Deep learning-based non-contact vital sign monitoring: A review," *IEEE Access*, vol. 10, pp. 71 530–71 552, 2022.

[5] D.-Y. Kim, K. Lee, and C.-B. Sohn, "Assessment of roi selection for facial video-based rppg," *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/7923

- GitHub Repository
- Chat With AI