Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: Muhammad Kaisar Teddy, Reynaldi Cristian Simamora, Eichal Elphindo Ginting (122140058, 122140116, 122140165)
Laporan Akhir Proyek
Mata Kuliah: **Pengolahan Sinyal Digital (IF3024)**
Tanggal: **31 Mei 2025**

Laporan Pengembangan Aplikasi Monitoring Fisiologis Real-time: rPPG dan Estimasi Pernapasan

1 Pendahuluan

Laporan ini merinci pengembangan sebuah aplikasi perangkat lunak yang bertujuan untuk melakukan monitoring parameter fisiologis dasar manusia secara non-invasif dan real-time menggunakan input video dari kamera. Dua parameter utama yang menjadi fokus adalah estimasi detak jantung menggunakan teknik remote photoplethysmography (rPPG) dan estimasi laju pernapasan berdasarkan analisis gerakan tubuh. Aplikasi ini dikembangkan menggunakan bahasa pemrograman Python dengan memanfaatkan berbagai library populer seperti OpenCV untuk pemrosesan video, MediaPipe untuk deteksi wajah dan pose tubuh, NumPy dan SciPy untuk pemrosesan sinyal, serta Tkinter dan Matplotlib untuk antarmuka pengguna grafis (GUI) dan visualisasi data.

Tujuan utama dari proyek ini adalah untuk menciptakan sebuah sistem yang mampu memberikan estimasi parameter fisiologis secara berkelanjutan dan mudah diakses, yang berpotensi berguna dalam berbagai skenario seperti pemantauan kesehatan pribadi, aplikasi kebugaran, atau penelitian awal. Laporan ini akan membahas arsitektur sistem, metode implementasi untuk setiap fitur utama, visualisasi data yang dihasilkan, serta panduan penggunaan dan konfigurasi aplikasi.

2 Dasar Teori

Pengembangan aplikasi ini didasarkan pada beberapa konsep dan teknik fundamental dalam pemrosesan sinyal, Computer Vision, dan analisis fisiologis non-invasif.

2.1 Remote Photoplethysmography (rPPG)

Photoplethysmography (PPG) adalah sebuah teknik optik sederhana yang digunakan untuk mendeteksi perubahan volume darah dalam sirkulasi mikro pada jaringan [1]. Prinsip dasarnya adalah iluminasi kulit dan pengukuran perubahan intensitas cahaya yang diserap atau dipantulkan. Hemoglobin dalam darah memiliki karakteristik penyerapan cahaya yang berbeda tergantung pada saturasi oksigen dan volume darah. Saat jantung memompa darah, terjadi perubahan volume darah di pembuluh darah perifer, yang menyebabkan variasi periodik pada intensitas cahaya yang terdeteksi.

Remote PPG (rPPG) mengadaptasi prinsip ini untuk pengukuran tanpa kontak menggunakan kamera digital dan sumber cahaya sekitar. Kamera menangkap perubahan warna subtil pada permukaan kulit wajah yang tidak terlihat oleh mata telanjang. Perubahan warna ini berkorelasi dengan perubahan volume darah di bawah kulit. Channel warna hijau (green channel) dari sensor kamera seringkali menunjukkan sinyal PPG dengan rasio sinyal terhadap noise (SNR) yang paling tinggi karena puncak serapan hemoglobin berada pada spektrum ini [2].

Untuk meningkatkan robustnes sinyal rPPG terhadap variasi iluminasi dan artefak gerakan, berbagai algoritma telah dikembangkan. Salah satunya adalah metode **POS (Plane-Orthogonal to Skin)** [3]. Metode POS memproyeksikan sinyal RGB dari kulit ke sebuah bidang yang ortogonal terhadap variasi warna kulit akibat perubahan iluminasi, sehingga komponen sinyal yang terkait dengan aliran darah (PPG) dapat diekstraksi dengan lebih baik.

2.2 Estimasi Laju Pernapasan dari Video

Laju pernapasan dapat diestimasi dari video melalui beberapa pendekatan. Metode yang umum digunakan berfokus pada pendeteksian perubahan fisik atau fisiologis yang disebabkan oleh siklus inspirasi dan ekspirasi.

2.2.1 Analisis Gerakan Tubuh

Pendekatan ini mengandalkan deteksi gerakan periodik pada bagian tubuh tertentu yang terkait dengan pernapasan, seperti dada, bahu, atau perut.

- **Penentuan Region of Interest (ROI)**: Area spesifik pada tubuh dipilih sebagai ROI. Penentuan ROI ini bisa dilakukan secara manual, semi-otomatis, atau otomatis menggunakan teknik deteksi objek atau estimasi pose tubuh. Dalam pengembangan ini, MediaPipe Pose Landmarker digunakan untuk mengidentifikasi posisi bahu yang kemudian menjadi dasar penentuan ROI secara dinamis.
- **Pelacakan Gerakan**: Setelah ROI ditentukan, gerakan di dalamnya dianalisis. Teknik seperti **Optical Flow** (misalnya, Lucas-Kanade yang digunakan dalam pengembangan ini melalui pelacakan `cv2.goodFeaturesToTrack`) dapat digunakan untuk mengestimasi vektor gerakan dari sekumpulan titik fitur di dalam ROI. Perubahan posisi vertikal rata-rata dari fitur-fitur ini dari waktu ke waktu dapat membentuk sinyal mentah pernapasan.

2.2.2 Modulasi Sinyal Fisiologis Lain (Pendekatan Alternatif)

Pernapasan juga diketahui memodulasi sinyal fisiologis lain. Contohnya adalah Respiratory Sinus Arrhythmia (RSA), di mana laju detak jantung sedikit berfluktuasi sinkron dengan siklus pernapasan. Beberapa metode rPPG lanjutan juga mencoba mengekstrak informasi pernapasan dari modulasi amplitudo atau frekuensi pada sinyal rPPG itu sendiri. Namun, pendekatan utama dalam aplikasi ini adalah analisis gerakan.

2.3 Pemrosesan Sinyal Digital untuk Sinyal Fisiologis

Sinyal fisiologis mentah yang diekstrak dari video (baik rPPG maupun gerakan pernapasan) seringkali mengandung noise dan komponen yang tidak diinginkan. Oleh karena itu, beberapa tahapan pemrosesan sinyal digital diperlukan:

- **Detrending (Penghilangan Tren)**: Sinyal mentah dapat memiliki pergeseran baseline (baseline wander) atau tren frekuensi rendah akibat gerakan subjek yang lambat, perubahan pencahayaan bertahap, atau sifat intrinsik dari metode akuisisi. Detrending bertujuan untuk menghilangkan komponen tren ini. Metode yang umum digunakan adalah pengurangan rata-rata bergerak (moving average subtraction) dari sinyal, seperti yang diimplementasikan menggunakan `scipy.ndimage.uniform_filter1d`.
- **Filtering (Penyaringan)**: Filter digital digunakan untuk mengisolasi komponen frekuensi yang relevan dan meredam noise. **Filter Bandpass Butterworth** adalah pilihan umum untuk sinyal fisiologis karena respons frekuensinya yang datar (maximally flat) di passband dan roll-off yang cukup baik. Batas frekuensi cut-off untuk filter ini ditentukan berdasarkan rentang frekuensi

fisiologis yang diharapkan (misalnya, 0.75-4 Hz untuk detak jantung, dan 0.1-0.8 Hz untuk pernapasan). Penggunaan `filtfilt` memastikan filtering tanpa pergeseran fasa (zero-phase filtering).

- **Analisis Spektral (Estimasi Frekuensi)**: Setelah sinyal dibersihkan, **Fast Fourier Transform (FFT)** diterapkan untuk mengubah sinyal dari domain waktu ke domain frekuensi. Puncak (peak) dominan dalam spektrum frekuensi pada rentang yang diharapkan kemudian diidentifikasi. Posisi puncak ini menunjukkan frekuensi dasar dari sinyal fisiologis (detak jantung atau laju pernapasan), yang kemudian dapat dikonversi menjadi satuan per menit (BPM atau RPM).

Kombinasi teknik-teknik ini memungkinkan ekstraksi dan kuantifikasi parameter fisiologis dari data video dengan tingkat akurasi dan stabilitas tertentu.

3 Arsitektur dan Komponen Aplikasi

Aplikasi ini dibangun dengan struktur modular untuk memisahkan fungsionalitas utama ke dalam beberapa file Python. Berikut adalah komponen-komponen utama beserta perannya:

- **`main.py`**: Skrip utama yang berfungsi sebagai titik masuk (entry point) untuk menjalankan aplikasi. Skrip ini menginisialisasi dan memulai antarmuka pengguna grafis.
- **`gui.py`**: Bertanggung jawab untuk membangun dan mengelola seluruh antarmuka pengguna grafis (GUI) menggunakan Tkinter. Ini mencakup tampilan feed video, plot sinyal, label data (BPM, RPM), tombol kontrol, dan status bar.
- **`video_capture.py`**: Bertanggung jawab untuk menangkap video dari kamera secara real-time sebagai input visual bagi sistem.
- **`utils.py`**: Berisi fungsi-fungsi utilitas, termasuk kelas `FaceDetectorMediaPipeTasks` yang menggunakan model BlazeFace dari MediaPipe untuk mendeteksi wajah dan mengekstrak ROI serta rata-rata nilai RGB dari wajah untuk analisis rPPG.
- **`pose_respiration_tracker.py`** (sebelumnya `pose_respiration_tracker.py`): Berisi kelas `OpticalFlowRespirationTracker` yang menggunakan MediaPipe Pose Landmarker untuk menentukan ROI awal pada area dada/bahu, kemudian menggunakan Optical Flow (Lucas-Kanade) untuk melacak fitur di dalam ROI tersebut dan menghasilkan sinyal gerakan mentah untuk estimasi pernapasan.
- **`motion_tracker.py`** (sebelumnya `pose_respiration_tracker.py`): Memantau perpindahan titik-titik fitur pada area tertentu (ROI) di video untuk mengestimasi sinyal pernapasan berdasarkan perubahan posisi vertikal gerakan tubuh. Bila titik fitur terlalu sedikit atau tidak dapat terlacak dengan baik, fitur akan dideteksi ulang untuk menjaga kestabilan sinyal. Teknik optical flow membantu mendeteksi pergerakan halus dari frame ke frame, sehingga bisa digunakan untuk memantau pola gerakan pernapasan secara non-invasif.
- **`signal_processing.py`**: Penerapan filter bandpass Butterworth untuk memisahkan frekuensi relevan (detak jantung 0.75–4 Hz, respirasi 0.1–0.8 Hz), menghilangkan tren (detrending) menggunakan moving average agar sinyal lebih stabil dan bebas drift, dan perhitungan spektrum frekuensi dengan FFT untuk menemukan frekuensi dominan yang kemudian dikonversi ke BPM (detak jantung) atau RPM (pernapasan), menyimpan buffer sinyal mentah untuk analisis berkelanjutan.
- **`visualization.py`**: Mengelola pembuatan dan pembaruan plot sinyal rPPG dan respirasi secara real-time menggunakan Matplotlib, yang kemudian di-embed ke dalam GUI. Untuk tampilan visualisasi yang ditampilkan adalah grafik yang ditampilkan adalah sinyal awal dari sinyal rPPG dan sinyal respirasi, dan juga sinyal setelah filter dari sinyal rPPG dan sinyal respirasi.

Interaksi antar komponen ini memungkinkan aplikasi untuk mengambil input video, mendeteksi wajah dan pose, mengekstrak sinyal fisiologis mentah, memproses sinyal tersebut untuk mengurangi noise dan mengestimasi laju, serta menampilkannya kepada pengguna melalui GUI yang interaktif.

4 Implementasi Fitur Utama

4.1 Estimasi Detak Jantung (rPPG)

Estimasi detak jantung dilakukan melalui beberapa tahapan:

1. **Pengumpulan Sinyal Mentah (Buffer)**: Sinyal mentah (misalnya, rata-rata nilai hijau dari ROI wajah untuk rPPG, atau sinyal gerakan bahu/ROI untuk pernapasan) dikumpulkan secara berkelanjutan dan disimpan dalam buffer dengan ukuran tertentu.

2. **Penghilangan Tren (Detrending)**: Sinyal yang sudah di-detrend kemudian melewati filter bandpass Butterworth. Filter ini dirancang untuk hanya meloloskan frekuensi dalam rentang yang relevan dengan detak jantung (misalnya, 0.75 Hz - 4.0 Hz) atau pernapasan (misalnya, 0.1 Hz - 0.8 Hz), sekaligus menghilangkan noise dan frekuensi lain yang tidak diinginkan.

3. **Penyaringan (Filtering)**: Sinyal RGB mentah yang telah dikumpulkan dalam buffer kemudian diproses menggunakan algoritma POS. Metode ini bertujuan untuk mengekstrak sinyal photoplethysmographic (PPG) dengan memproyeksikan sinyal RGB ke sebuah bidang yang ortogonal terhadap variasi warna kulit akibat iluminasi, sehingga lebih sensitif terhadap perubahan volume darah. Kode Program 4 menunjukkan bagian penting dari implementasi POS.

4. **Analisis Domain Frekuensi (Fast Fourier Transform - FFT)**: SFFT diterapkan pada sinyal yang telah difilter. FFT mengubah sinyal dari domain waktu ke domain frekuensi, memungkinkan identifikasi komponen frekuensi yang ada dalam sinyal.

5. **Estimasi Frekuensi Dominan**: Dalam spektrum frekuensi yang dihasilkan oleh FFT, frekuensi dominan (yaitu, frekuensi dengan amplitudo atau "kekuatan" tertinggi) dalam rentang yang relevan (detak jantung atau pernapasan) diidentifikasi.

6. **Konversi ke BPM/RPM**: Frekuensi dominan yang ditemukan (dalam Hertz) kemudian dikonversi menjadi unit yang lebih mudah dipahami: Untuk detak jantung: BPM=Frekuensi Dominan (Hz)×60 Untuk pernapasan: RPM=Frekuensi Dominan (Hz)×60

```
class SignalProcessor:
    def __init__(self, fs, buffer_size=300): # Menggunakan nilai default contoh
        self.fs = fs if fs > 0 else 30.0
        self.buffer_size = buffer_size
        self.rppg_raw_signal = []
        self.resp_raw_signal = []

    def _apply_filter(self, data, lowcut, highcut, order):
        if len(data) < order * 3: return np.array([])
        nyq = 0.5 * self.fs
        low, high = lowcut / nyq, highcut / nyq
        if not (0 < low < 1 and 0 < high < 1 and low < high): return np.array(data)
        try:
            b, a = butter(order, [low, high], btype='band')
            return filtfilt(b, a, data)
        except ValueError:
            return np.array(data)

    def _detrend(self, signal_segment, window_seconds):
        if len(signal_segment) == 0: return np.array([])
```

```
        window_samples = max(3, int(self.fs * window_seconds))
        if len(signal_segment) >= window_samples:
            moving_avg = uniform_filter1d(signal_segment, size=window_samples, mode='reflect')
            return signal_segment - moving_avg
        return signal_segment - np.mean(signal_segment)

    def _estimate_frequency(self, filtered_signal, lowcut, highcut):
        N = len(filtered_signal)
        if N < self.fs: return 0.0
        yf = fft(filtered_signal)
        xf = np.fft.fftfreq(N, 1.0/self.fs)[:N//2]

        valid_freq_indices = np.where((xf >= lowcut) & (xf <= highcut))[0]
        if len(valid_freq_indices) == 0: return 0.0

        abs_yf = np.abs(yf[valid_freq_indices])
        if len(abs_yf) == 0: return 0.0

        dominant_freq_index = np.argmax(abs_yf)
        return xf[valid_freq_indices[dominant_freq_index]]

    def process_rppg(self, green_mean):
        self.rppg_raw_signal.append(green_mean)
        if len(self.rppg_raw_signal) > self.buffer_size: self.rppg_raw_signal.pop(0)
        if len(self.rppg_raw_signal) < self.buffer_size: return np.array([]), 0.0

        segment = np.array(self.rppg_raw_signal)
        detrended = self._detrend(segment, window_seconds=2.0)
        filtered = self._apply_filter(detrended, RPPG_LOWCUT, RPPG_HIGHCUT, RPPG_FILTER_ORDER)
        if len(filtered) == 0: return segment, 0.0

        dominant_freq = self._estimate_frequency(filtered, RPPG_LOWCUT, RPPG_HIGHCUT)
        bpm = round(dominant_freq * 60, 1)
        return filtered, bpm

    def process_respiration(self, motion_value):
        self.resp_raw_signal.append(motion_value)
        if len(self.resp_raw_signal) > self.buffer_size: self.resp_raw_signal.pop(0)
        if len(self.resp_raw_signal) < self.buffer_size: return np.array([]), 0.0

        segment = np.array(self.resp_raw_signal)
        detrended = self._detrend(segment, window_seconds=10.0)
        filtered = self._apply_filter(detrended, RESP_LOWCUT, RESP_HIGHCUT, RESP_FILTER_ORDER)
        if len(filtered) == 0: return segment, 0.0

        dominant_freq = self._estimate_frequency(filtered, RESP_LOWCUT, RESP_HIGHCUT)
        rpm = round(dominant_freq * 60, 1)
        return filtered, rpm

    def get_raw_rppg_signal_for_plot(self): return np.array(self.rppg_raw_signal)
    def get_raw_resp_signal_for_plot(self): return np.array(self.resp_raw_signal)
```

Kode Program 1: Implementasi inti algoritma Analisis Domain Frekuensi menggunakan Fast Fourier Transform (FFT). (bagian dari `signal_processing.py`)

4.2 Estimasi Laju Pernapasan

Estimasi laju pernapasan dilakukan dengan menganalisis gerakan tubuh, khususnya pada area dada atau bahu, melalui dua pendekatan utama yang saling melengkapi:

1. **Pelacakan Respirasi Berbasis Pose (`PoseRespirationTracker`)**:

- **Deteksi Pose dan Identifikasi Landmark Bahu**: Menggunakan model *MediaPipe Pose* untuk mendeteksi pose manusia pada setiap *frame*. Fungsi `get_respiration_signal_and_draw_landm` secara spesifik mengidentifikasi landmark bahu kiri dan kanan. Ini merupakan langkah fundamental untuk menangkap posisi tubuh yang relevan terhadap aktivitas pernapasan.
- **Ekstraksi Sinyal Gerakan Bahu**: Perubahan posisi vertikal (`dy`) dari titik tengah bahu antara *frame* saat ini dan sebelumnya dihitung. Gerakan naik-turun bahu ini berkorelasi langsung dengan aktivitas inspirasi dan ekspirasi.
- **Penyempurnaan Sinyal**: Sinyal `dy` mentah kemudian diproses melalui metode *smoothing* menggunakan rata-rata bergerak untuk mengurangi *noise*. Sinyal yang telah dihaluskan kemudian diskalakan untuk menghasilkan sinyal pernapasan mentah yang lebih bersih.

2. **Pelacakan Gerakan dalam ROI (`RespirationMotionTracker`)**:

   - **Penentuan dan Pemotongan ROI Dinamis**: Komponen ini mengambil Region of Interest (ROI) dari *frame* video, misalnya area dada atau perut. ROI dapat ditentukan secara dinamis berdasarkan landmark pose yang terdeteksi, memastikan area yang relevan tetap terpantau meskipun terjadi pergerakan subjek.
   - **Inisialisasi dan Pelacakan Fitur Optik**: Di dalam ROI, titik-titik fitur yang layak dilacak diidentifikasi menggunakan fungsi `cv2.goodFeaturesToTrack` pada *frame* awal, atau ketika fitur perlu dideteksi ulang. Fitur-fitur ini dilacak antar *frame* menggunakan algoritma *Lucas-Kanade Optical Flow* (`cv2.calcOpticalFlowPyrLK`).
   - **Ekstraksi Sinyal Gerakan ROI**: Rata-rata perpindahan vertikal (koordinat-Y) dari fitur-fitur yang berhasil dilacak digunakan sebagai sinyal gerakan mentah. Sinyal ini merepresentasikan pergerakan halus dalam ROI akibat aktivitas pernapasan.

3. **Filtering Sinyal Pernapasan Gabungan**: Sinyal gerakan mentah dari salah satu atau kombinasi pendekatan di atas kemudian difilter menggunakan *Butterworth bandpass filter* (biasanya 0,1 Hz–0,8 Hz), untuk mengisolasi frekuensi yang relevan dengan pernapasan serta menghilangkan *noise* dan artefak gerakan lainnya.

4. **Estimasi RPM (Respiration Per Minute)**: Transformasi Fourier Cepat (FFT) diterapkan pada sinyal pernapasan yang telah difilter untuk mengidentifikasi frekuensi dominan. Frekuensi tersebut kemudian dikonversi menjadi laju pernapasan per menit (RPM), memberikan estimasi pernapasan yang akurat secara non-invasif.

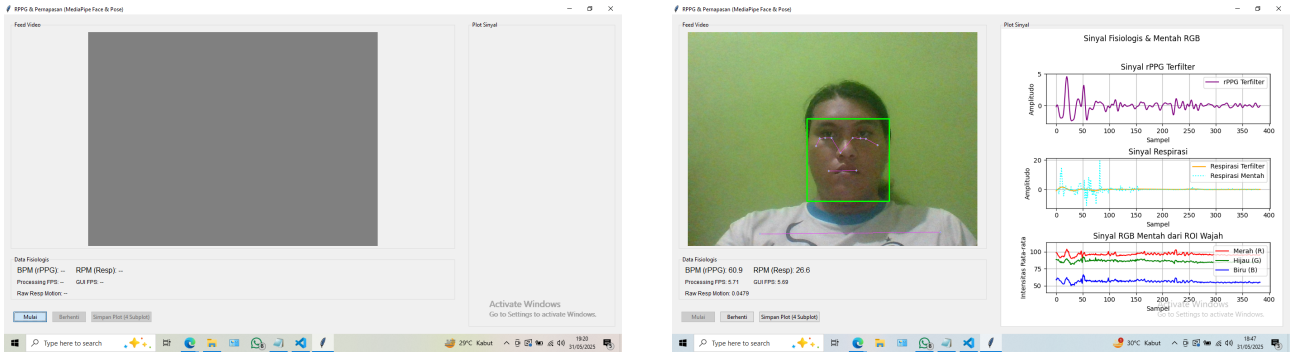
```python
class RespirationMotionTracker:
    def __init__(self, max_features=30, quality_level=0.2, min_distance=5, block_size=5):
        """
        Inisialisasi pelacak gerakan untuk estimasi sinyal pernapasan.
        """
        self.prev_gray_roi = None
        self.prev_points_roi = None

        self.feature_params = dict(
            maxCorners=max_features,
            qualityLevel=quality_level,
            minDistance=min_distance,
            blockSize=block_size
        )

        self.lk_params = dict(
            winSize=(15, 15),
            maxLevel=2,
            criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03)
```

```
        )

        self.retrack_threshold = max_features // 3

    def get_motion_signal(self, frame_bgr, roi_coords_dict):
        """
        Hitung sinyal gerakan vertikal rata-rata dari ROI pada frame input.
        """
        x, y, w, h = roi_coords_dict['x'], roi_coords_dict['y'], roi_coords_dict['w'],
    roi_coords_dict['h']

        frame_h_orig, frame_w_orig = frame_bgr.shape[:2]
        x = max(0, min(x, frame_w_orig - 1))
        y = max(0, min(y, frame_h_orig - 1))
        w = max(1, min(w, frame_w_orig - x))
        h = max(1, min(h, frame_h_orig - y))

        if w <= 0 or h <= 0:
            self.prev_gray_roi = None
            self.prev_points_roi = None
            return 0.0

        roi_frame = frame_bgr[y:y+h, x:x+w]
        current_gray_roi = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
        motion_signal = 0.0

        if self.prev_points_roi is None or len(self.prev_points_roi) < self.retrack_threshold:
            self.prev_points_roi = cv2.goodFeaturesToTrack(current_gray_roi, mask=None, **self.
    feature_params)
            self.prev_gray_roi = current_gray_roi.copy()
            if self.prev_points_roi is None:
                return 0.0
            return 0.0

        if self.prev_gray_roi is not None and self.prev_points_roi is not None:
            p1_roi, st, err = cv2.calcOpticalFlowPyrLK(
                self.prev_gray_roi, current_gray_roi, self.prev_points_roi, None, **self.lk_params
            )

            if p1_roi is not None and st is not None:
                good_new_roi = p1_roi[st == 1]
                good_old_roi = self.prev_points_roi[st == 1]

                if len(good_new_roi) >= self.retrack_threshold // 2 and len(good_new_roi) > 3:
                    dy = good_new_roi[:, 1] - good_old_roi[:, 1]
                    motion_signal = np.mean(dy)
                    self.prev_points_roi = good_new_roi.reshape(-1, 1, 2)
                else:
                    self.prev_points_roi = None
            else:
                self.prev_points_roi = None

        self.prev_gray_roi = current_gray_roi.copy()
        return motion_signal if not np.isnan(motion_signal) else 0.0
```

Kode Program 2: Implementasi pelacak gerakan berbasis Optical Flow untuk estimasi sinyal respirasi. (bagian dari `motion_tracker.py`)

5 Visualisasi Hasil

Bagian ini akan menampilkan contoh visual dari antarmuka aplikasi dan sinyal yang dihasilkan. *(Catatan: Ganti path file gambar di bawah ini dengan path gambar hasil screenshot aplikasi Anda.)*



(a) Tampilan Antarmuka Awal Program.

(b) Tampilan Antarmuka Setelah Tombol Mulai.

Gambar 1: Visualisasi Antarmuka dan Sinyal

Gambar 1a menunjukkan layout GUI secara keseluruhan, termasuk area video, panel data, dan plot sinyal. Terdapat beberapa tombol Mulai, Berhenti, Simpan Plot (4 plot). Selain tombol pada frame atau panel, terdapat 3 tombol pada fram tkinter yaitu minimize, RestoreDown, dan Close. Tombol mulai untuk memulai menangkap video secara realtime, dan tombol berhenti untuk menghentikan penangkapan video, sedangkan tombol Simpan Plot untuk menyimpan plot dalam 4 grafik yaitu (sinyal rPPG awal, rPPG filter, respirasi awal, dan respirasi filter). Sementara itu, Gambar 1b merupakan tambil aksi dari tombol Mulai. Lebih detail dari sinyal rPPG dan respirasi yang telah difilter dan divisualisasikan oleh aplikasi. Sinyal rPPG awal (plot atas) digunakan untuk estimasi BPM, dan sinyal respirasi (plot tengah , yang telah difilter dan awal) dan sinyal rPPG yang telah difilter(plot bawah). Untuk tampilan di bawah video real-time terdapat BPM dan RPM. Hasil kalkulasi BPM dan RPM yang dicapture akan muncul setelah 40-50 detik karena pada frame yang diambil untuk plot di panel kanan GUI memiliki frame 400. Sedangkan fps video yang ditangkap kamera adalah sekitar 8-10 fps.

6 Cara Penggunaan dan Konfigurasi

6.1 Persiapan Lingkungan

Sebelum menjalankan aplikasi, perlu untuk masuk ke environment menggunakan conda. Untuk conda kami menggunakan miniconda, dan telah menginstallnya serta memberikan pathnya di environment variabel.

```
conda create -n myenv python=3.10.6
conda activate myenv
```

Kemudian instal dependensi menggunakan pip:Untuk pastikan semua dependensi Python telah terinstal. Buat file `requirements.txt` dengan isi sebagai berikut:

```
opencv-python==4.9.0.80
numpy==1.26.4
scipy==1.13.0
matplotlib==3.8.4
Pillow==10.3.0
mediapipe==0.10.11
```

Kode Program 3: Isi file `requirements.txt`

Kemudian instal dependensi menggunakan pip:

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

Pastikan juga file model MediaPipe (`blaze_face_short_range.tflite` dan `pose_landmarker.task`) berada di dalam subdirektori `models/` dalam direktori proyek Anda.

6.2 Menjalankan Aplikasi

Untuk menjalankan aplikasi, navigasi ke direktori utama proyek melalui terminal dan eksekusi skrip `main.py`:

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

GUI aplikasi akan muncul. Tekan tombol "Mulai" untuk memulai pengambilan video dan analisis sinyal. Tekan "Berhenti" untuk menghentikan proses dan Simpan Plot RGB untuk menyimpan plot yang telah ada di frame kanan pada GUI setelah melakukan Tombol mulai atau capture sinyal.

6.3 Desain Filter dan Justifikasi Matematis

Untuk mengekstraksi sinyal fisiologis seperti rPPG (remote photoplethysmography) dan respirasi dari video, digunakan pendekatan pemrosesan sinyal berbasis filter bandpass Butterworth. Pemilihan jenis filter ini didasarkan pada karakteristiknya yang memiliki respon frekuensi halus dan tidak menghasilkan distorsi fasa jika diterapkan secara *zero-phase* menggunakan metode `filtfilt`.

6.3.1 Filter untuk Sinyal rPPG (Denyut Jantung)

Rentang frekuensi fisiologis normal untuk detak jantung manusia berada antara 45 BPM hingga 240 BPM, yang ekuivalen dengan:

$$\text{Rentang frekuensi rPPG} = \left[\frac{45}{60}, \frac{240}{60}\right] = [0{,}75, 4{,}0] \text{ Hz}$$

Untuk mengekstrak komponen ini, digunakan filter bandpass Butterworth orde-5 dengan frekuensi cutoff:

$$f_{\text{low}} = 0{,}75 \text{ Hz}, \quad f_{\text{high}} = 4{,}0 \text{ Hz}$$

Filter ini dinyatakan secara matematis sebagai:

$$H(s) = \frac{1}{\sqrt{1 + \left(\frac{s}{\omega_c}\right)^{2n}}}$$

dimana n adalah orde filter, dan ω_c adalah frekuensi cutoff normalisasi. Setelah diskretisasi dan normalisasi terhadap frekuensi Nyquist:

$$f_{\text{norm}} = \frac{f}{f_s/2}$$

Dengan f_s adalah frekuensi sampling (frame rate kamera), desain filter dilakukan dalam bentuk digital:

$$[b, a] = \text{butter}(n, [f_{\text{low}}^{\text{norm}}, f_{\text{high}}^{\text{norm}}], \texttt{btype} =' band')$$

$$y[n] = \text{filtfilt}(b, a, x[n])$$

Alasan penggunaan: Filter Butterworth orde tinggi digunakan untuk memastikan transisi yang tajam di band edge, karena noise gerakan cukup kuat dan sinyal fisiologis cukup lemah. Diterapkan secara zero-phase untuk menghindari distorsi fasa pada sinyal rPPG yang sensitif terhadap bentuk gelombang.

6.3.2 Filter untuk Sinyal Respirasi

Rentang respirasi normal manusia berkisar antara 6 hingga 48 RPM (respirasi per menit), yaitu:

$$\text{Rentang frekuensi respirasi} = \left[\frac{6}{60}, \frac{48}{60}\right] = [0{,}1, 0{,}8] \text{ Hz}$$

Filter bandpass Butterworth orde-2 digunakan dengan cutoff yang lebih rendah:

$$f_{\text{low}} = 0{,}1 \text{ Hz}, \quad f_{\text{high}} = 0{,}8 \text{ Hz}$$

Alasan penggunaan: Orde filter yang lebih rendah digunakan karena sinyal respirasi memiliki komponen frekuensi yang lebih lambat dan dominan. Selain itu, noise yang muncul pada sinyal ini cenderung lebih lambat (misalnya drift posisi tubuh), sehingga filter yang lebih halus sudah memadai.

6.3.3 Detrending dengan Rata-Rata Bergerak (Moving Average)

Sebelum diterapkan filter bandpass, sinyal mentah dilakukan *detrending* untuk menghilangkan komponen tren lambat akibat pergeseran pencahayaan atau postur. Teknik detrending menggunakan rata-rata bergerak sebagai berikut:

$$x_{\text{detrend}}[n] = x[n] - \frac{1}{W} \sum_{k=n-\lfloor W/2 \rfloor}^{n+\lfloor W/2 \rfloor} x[k]$$

dengan $W = f_s \cdot T$ adalah ukuran jendela rata-rata, di mana: - Untuk rPPG digunakan $T = 2$ detik, - Untuk respirasi digunakan $T = 10$ detik.

Alasan penggunaan: Metode ini lebih ringan secara komputasi dibanding detrending berbasis regresi polinomial dan cocok untuk pemrosesan real-time.

6.3.4 Estimasi Frekuensi Dominan (FFT)

Setelah sinyal difilter, estimasi denyut jantung dan laju napas dilakukan melalui Transformasi Fourier Diskrit:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1$$

Dominan frekuensi f_{dominan} ditentukan dari magnitudo maksimum dalam rentang valid. Kemudian dikonversi ke BPM atau RPM:

$$\text{BPM} = 60 \cdot f_{\text{dominan}} \quad \text{atau} \quad \text{RPM} = 60 \cdot f_{\text{dominan}}$$

6.3.5 Kesimpulan Desain Filter

Desain sistem filtering dan detrending ini ditujukan untuk menyeimbangkan antara sensitivitas terhadap sinyal fisiologis dan ketahanan terhadap artefak gerakan dan noise pencahayaan. Kombinasi filter bandpass Butterworth dan rata-rata bergerak terbukti efektif dalam mengekstraksi sinyal denyut jantung dan pernapasan secara real-time, terutama ketika dikombinasikan dengan pendekatan zero-phase dan analisis spektral berbasis FFT.

6.4 Hasil Dan Analisis

Kami melakukan 3 studi kasus, dikarenakan pada penerapan program kami mengalami kendala di kamera, yang hanya memiliki konfigurasi dibawah 10 FPS. Namun pada penerapan kode kami menggunakan dimana :

6.5 Hasil dan Analisis

Kami melakukan tiga studi kasus. Pada penerapan program kami mengalami kendala pada kamera yang hanya memiliki konfigurasi di bawah 10 FPS. Namun dalam pengujian, kami menggunakan pendekatan sebagai berikut:

1. **Penggunaan 10 FPS dengan kondisi badan dan wajah tidak stabil**

   Pada kondisi ini, sinyal menunjukkan banyak gangguan akibat pergerakan wajah. Sinyal rPPG terfilter menunjukkan fluktuasi ekstrem dengan puncak amplitudo yang tajam dan tidak konsisten, mengindikasikan adanya artefak gerakan yang kuat. Sinyal respirasi juga menunjukkan fluktuasi yang tidak teratur dan noise tinggi. Sinyal mentah dari kanal RGB terlihat tidak sinkron dan sangat terpengaruh oleh noise gerakan. Pada GUI menampilkan BPM 57.9 dan RPM 12.6. Ini angka yang tidak normal sebenarnya jika dibandingkan dengan denyut nadi dewasa laki-laki yaitu 60-110.

   Analisis: Resolusi temporal yang rendah (10 fps), ditambah ketidakstabilan posisi subjek, menyebabkan penurunan signifikan dalam kualitas sinyal. Sistem sangat rentan terhadap noise dan tidak mampu mengekstrak sinyal fisiologis secara akurat.

```
# self.processing_thread = None
# self.is_processing = False
# self.effective_fps = 30.0
# print(f"Target effective FPS set to: {self.effective_fps}")

```

Kode Program 4: Implementasi inti algoritma POS (bagian dari `signal_processing.py`)

2. **Penggunaan 10 FPS dengan kondisi badan dan wajah stabil**

   Dengan kondisi subjek yang tetap diam, kualitas sinyal meningkat meskipun frame rate tetap rendah. Sinyal rPPG terfilter menunjukkan pola periodik dengan amplitudo yang kecil namun cukup stabil. Sinyal respirasi terfilter mulai menunjukkan pola sinusoidal yang dapat ditafsirkan sebagai ritme pernapasan. Sinyal RGB mentah juga menunjukkan kestabilan antar kanal warna. GUI menampilkan objek memiliki BPM 114.1 dan RPM 17.6. Ini mengindikasikan angka denyut jantung dan respirasi per menit normal.

   Analisis: Stabilitas posisi subjek mampu mengurangi artefak gerakan dan meningkatkan kualitas sinyal meskipun frame rate rendah.

3. **Penggunaan 30 FPS dengan kondisi stabil**

   Pada kondisi ini, sinyal rPPG dan respirasi menunjukkan kualitas yang sangat baik. Sinyal rPPG terfilter memiliki bentuk gelombang periodik yang jelas dan amplitudo yang seimbang. Sinyal respirasi juga sangat halus dan konsisten. Ketiga kanal warna RGB mentah menunjukkan kestabilan yang tinggi. Pada GUI menampilkan BPM 78.1 dan RPM, angka ini cukup normal.

   Analisis: Dengan peningkatan frame rate menjadi 30 fps dan posisi subjek yang stabil, kualitas sinyal meningkat secara signifikan. Sinyal yang diperoleh lebih akurat dan bersih, sangat layak untuk estimasi denyut jantung dan laju pernapasan secara real-time.

7 Kesimpulan dan Saran

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis terhadap tiga konfigurasi yang berbeda dalam sistem pengukuran sinyal fisiologis menggunakan metode rPPG dan estimasi respirasi, diperoleh beberapa kesimpulan sebagai berikut:

- Penggunaan konfigurasi 10 FPS dengan kondisi badan dan wajah tidak stabil menghasilkan sinyal dengan kualitas yang sangat buruk. Hal ini disebabkan oleh rendahnya resolusi temporal yang dikombinasikan dengan artefak gerakan, yang secara signifikan merusak kestabilan sinyal rPPG maupun sinyal respirasi.
- Dalam konfigurasi 10 FPS dengan kondisi badan dan wajah stabil, kualitas sinyal membaik secara signifikan. Artefak gerakan dapat diminimalkan dengan stabilitas posisi subjek, meskipun frame rate tetap rendah. Hal ini menunjukkan bahwa kestabilan subjek berperan besar dalam meningkatkan akurasi estimasi sinyal fisiologis meskipun dengan keterbatasan perangkat keras.
- Konfigurasi 30 FPS memberikan hasil terbaik secara keseluruhan. Sinyal rPPG dan respirasi yang dihasilkan bersih, periodik, dan memiliki amplitudo yang stabil. Hal ini menunjukkan bahwa peningkatan frame rate sangat krusial untuk akurasi dan reliabilitas sistem estimasi sinyal fisiologis berbasis kamera.

Secara umum, eksperimen ini menunjukkan bahwa kombinasi antara frame rate yang memadai dan kestabilan posisi subjek merupakan faktor utama dalam keberhasilan sistem estimasi sinyal fisiologis secara non-kontak menggunakan kamera RGB.

7.2 Saran

Untuk pengembangan sistem selanjutnya, beberapa saran yang dapat dipertimbangkan antara lain:

- **Penggunaan Kamera dengan Frame Rate Lebih Tinggi:** Disarankan untuk menggunakan kamera dengan frame rate minimal 30 FPS secara konsisten agar dapat menangkap sinyal fisiologis dengan lebih akurat dan responsif terhadap dinamika perubahan fisiologis.
- **Penambahan Algoritma Pelacak Wajah dan Kompensasi Gerakan:** Implementasi face tracking dan motion compensation dapat membantu menjaga kestabilan sinyal bahkan saat subjek bergerak, sehingga memperluas penerapan sistem ke situasi dunia nyata yang lebih dinamis.
- **Integrasi Filter Adaptif dan Peningkatan Prosesing Sinyal:** Penggunaan filter adaptif (misal Kalman filter atau adaptive bandpass) dapat meningkatkan kualitas sinyal rPPG dan respirasi pada kondisi noise tinggi atau cahaya rendah.
- **Evaluasi Klinis dan Dataset Lebih Luas:** Untuk validasi lebih lanjut, perlu dilakukan pengujian pada subjek dengan karakteristik berbeda-beda serta dibandingkan dengan ground truth dari alat medis standar, seperti pulse oximeter dan spirometer.
- **Peningkatan Antarmuka Pengguna (GUI):** GUI dapat dikembangkan lebih lanjut agar lebih interaktif dan menampilkan metrik real-time seperti HRV (Heart Rate Variability) dan RR (Respiratory Rate) secara numerik dan visual.

References

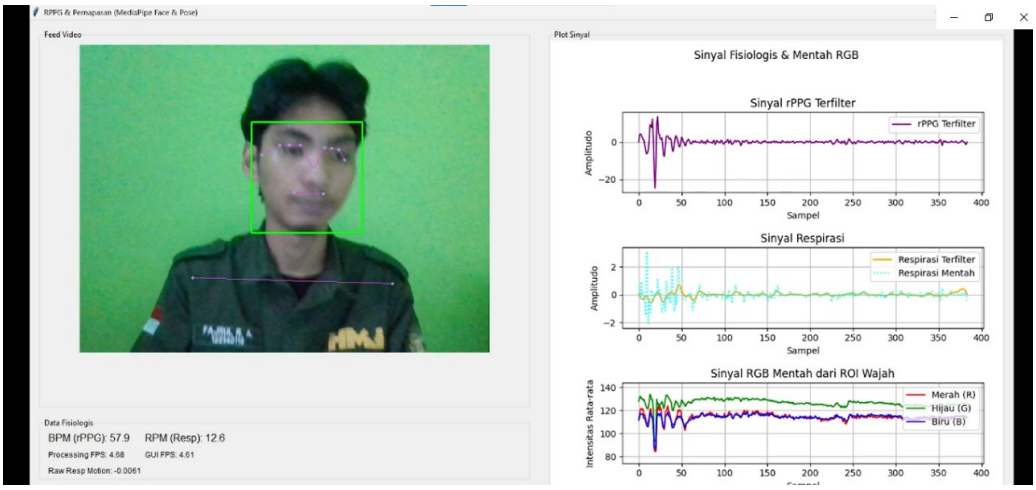
[1] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *Physiological Measurement*, vol. 28, no. 3, pp. R1–R39, 2007.

[2] M.-Z. Poh, D. J. McDuff, and R. W. Picard, "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation," *Optics Express*, vol. 18, no. 10, pp. 10 762–10 774, 2010.

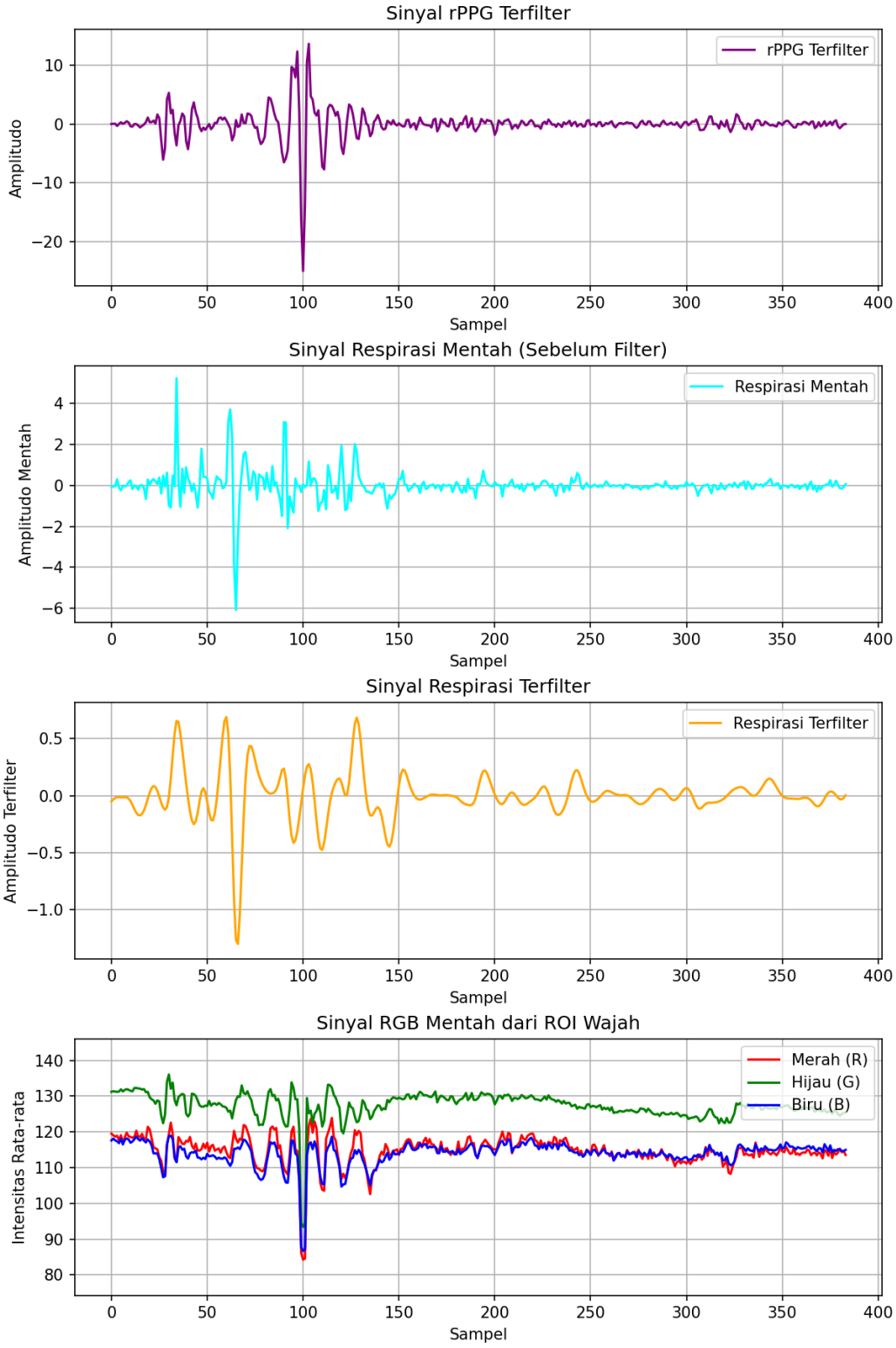
[3] W. Wang, A. C. den Brinker, S. Stuijk, and G. de Haan, "Algorithmic principles of remote ppg," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, pp. 1479–1491, 2016.

Lampiran

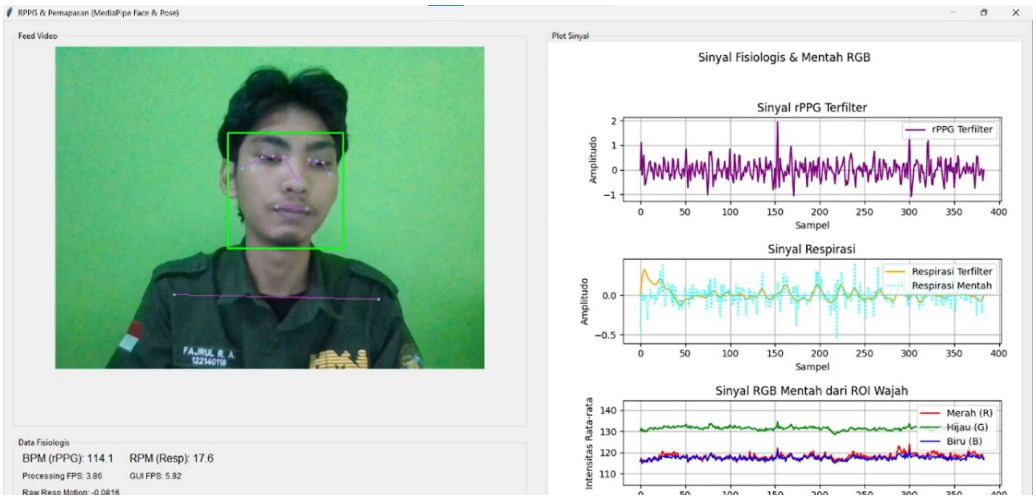
- **Demonstrasi dan Presentasi**: Tautan Google Drive
  Tautan Youtube
- **AI Chat**: Drive Bukti
  chatgpt
  Link Claude.ai
- **GitHub Repositori**: Github Repositori
- **Link Drive**: Tautan Link Drive



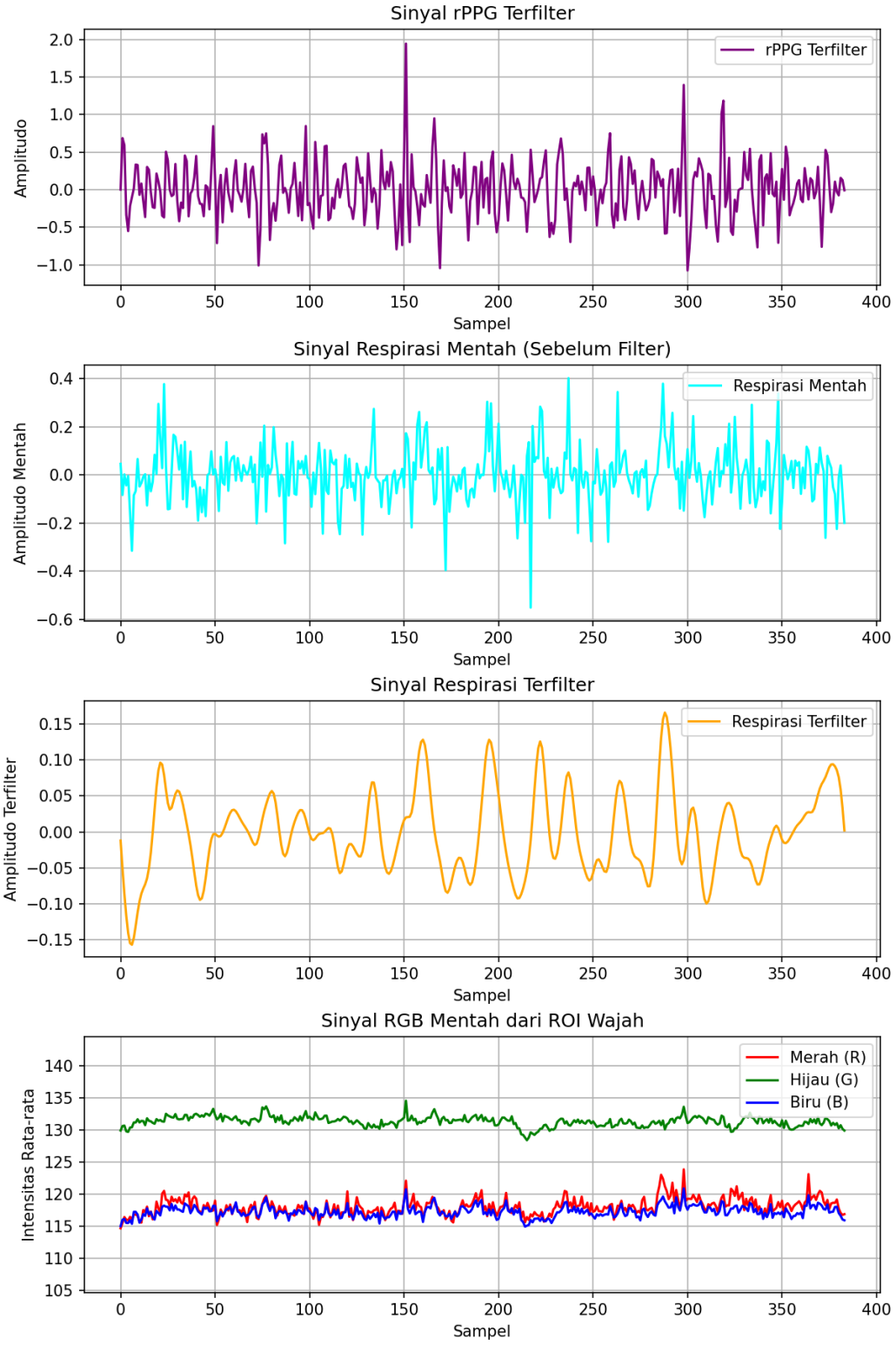
Gambar 2: Tampilan GUI 10 fps dengan kondisi tidak stabil



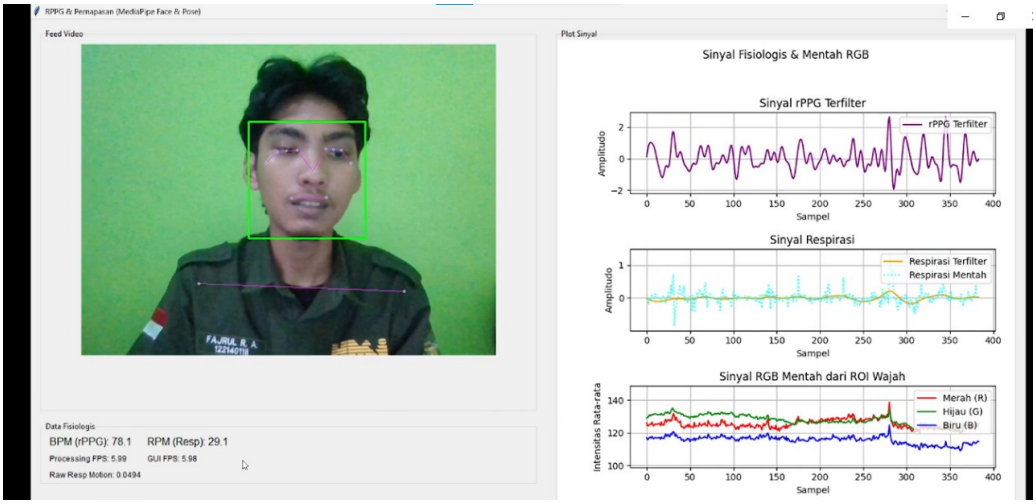
Gambar 3: Plot sinyal pada 10 fps (tidak stabil)



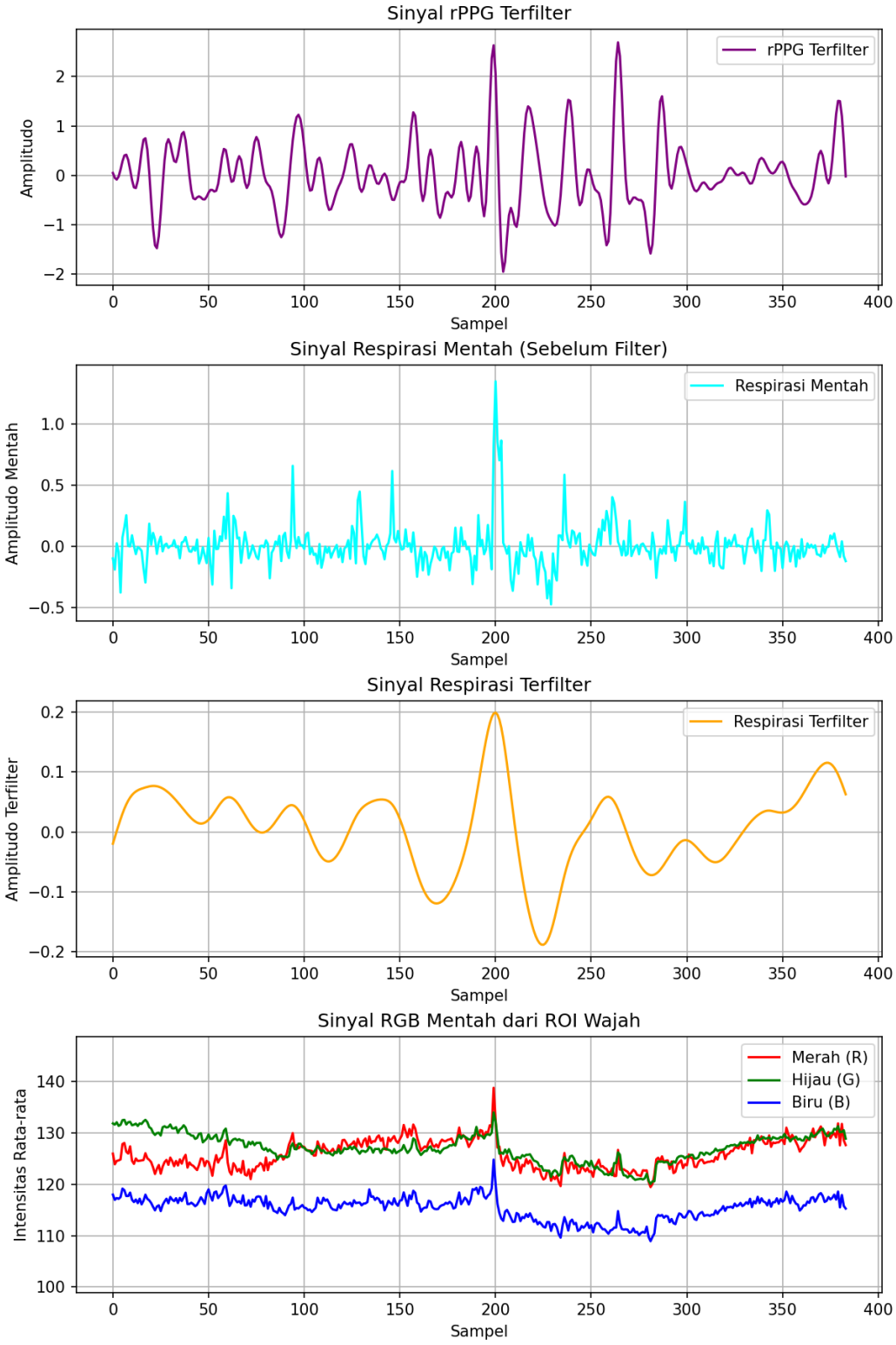
Gambar 4: Tampilan GUI 10 fps stabil



Gambar 5: Plot sinyal pada 10 fps (stabil)



Gambar 6: Tampilan GUI 30 fps stabil



Gambar 7: Plot sinyal pada 30 fps