Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Mychael, Ichsan, Fajrul (122140104, 122140117, 122140118)** Tugas Ke: **Tugas Besar**
Mata Kuliah: **Pengolahan Sinyal Digital (IF3024)** Tanggal: 31 Mei 2025
Dosen Pengampu: Martin Clinton Tosima Manullang, S.T., M.T

Pemantauan Sinyal rPPG dan Respirasi Real-Time Berbasis Webcam

1 Pendahuluan

Proyek ini bertujuan untuk mengimplementasikan sebuah sistem pemantauan sinyal biologis non-kontak secara *real-time*. Program yang dikembangkan mampu mengekstraksi dan menampilkan dua jenis sinyal vital dari input video webcam, yaitu sinyal pernapasan dan sinyal detak jantung (rPPG). Aplikasi ini dilengkapi dengan antarmuka pengguna grafis (GUI) yang interaktif untuk memvisualisasikan data dan hasil estimasi secara langsung.

2 Landasan Teori

Bagian ini menjelaskan dasar matematis dari dua teknik utama yang digunakan dalam pemrosesan sinyal pada proyek ini.

2.1 Filter Butterworth

Filter Butterworth adalah jenis filter *Infinite Impulse Response* (IIR) yang memiliki karakteristik respons magnitudo yang sangat datar pada *passband* [1]. Karakteristik ini ideal untuk aplikasi biomedis di mana preservasi bentuk gelombang asli sangat penting. Respon magnitudo kuadrat dari filter Butterworth low-pass orde-n didefinisikan sebagai berikut [2]:

$$|H(j\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}} \tag{1}$$

di mana ω_c adalah frekuensi cutoff dan n adalah orde filter. Proyek ini menggunakan filter Butterworth tipe *bandpass* untuk mengisolasi rentang frekuensi sinyal target.

2.2 Periodogram untuk Estimasi Spektrum Daya

Periodogram digunakan untuk mengestimasi *Power Spectral Density* (PSD) dari sebuah sinyal, yang menggambarkan bagaimana daya sinyal terdistribusi pada berbagai frekuensi [1]. Untuk sinyal waktu

diskrit $x[k]$ dengan panjang N, Periodogram didefinisikan sebagai kuadrat magnitudo dari *Discrete Fourier Transform* (DFT) sinyal tersebut:

$$P_{xx}(f) = \frac{1}{N} \left| \sum_{k=0}^{N-1} x[k] e^{-j2\pi fk} \right|^2 \tag{2}$$

Frekuensi yang sesuai dengan nilai puncak dari $P_{xx}(f)$ dianggap sebagai frekuensi fundamental dari sinyal.

3 Metodologi

Metodologi proyek ini menerapkan landasan teori yang ada ke dalam sebuah alur kerja praktis, mulai dari akuisisi data hingga ekstraksi informasi fisiologis.

3.1 Ekstraksi Sinyal Pernapasan

Sinyal pernapasan diekstraksi dengan melacak pergerakan vertikal bahu.

- **Deteksi Landmark:** Menggunakan pustaka `MediaPipe Pose` [3], program mendeteksi landmark untuk bahu kiri (`LEFT_SHOULDER`) dan kanan (`RIGHT_SHOULDER`).
- **Generasi Sinyal Mentah:** Sinyal pernapasan mentah dihasilkan dengan menghitung rata-rata koordinat vertikal (koordinat Y) dari kedua bahu pada setiap frame video.

3.2 Ekstraksi Sinyal Detak Jantung (rPPG)

Sinyal rPPG didasarkan pada prinsip penyerapan cahaya oleh hemoglobin dalam darah [4].

- **Deteksi Wajah dan ROI:** Menggunakan `MediaPipe FaceMesh` [5], program mendeteksi 468 landmark wajah dan menentukan sebuah *Region of Interest* (ROI) di area dahi.
- **Generasi Sinyal Mentah:** Pada setiap frame, dihitung nilai rata-rata dari kanal warna hijau di dalam ROI. Kanal hijau dipilih karena sensitivitasnya yang tinggi terhadap perubahan volume darah [4].

3.3 Rangkaian Pemrosesan dan Filtrasi Sinyal

Sinyal mentah yang diperoleh dari webcam umumnya mengandung derau dari berbagai sumber. Untuk mendapatkan sinyal yang bersih dan informatif, serangkaian proses filtrasi diterapkan sesuai dengan implementasi pada `signal_filter.py`:

1. **Median Filter:** Filter ini diterapkan pertama untuk menghilangkan noise impulsif atau lonjakan (spike) yang mungkin muncul pada sinyal akibat gangguan sesaat.
2. **Savitzky-Golay Filter:** Selanjutnya, filter Savitzky-Golay digunakan untuk melakukan penghalusan (smoothing) sinyal. Filter ini bekerja dengan mencocokkan potongan-potongan kecil data sinyal ke sebuah polinomial orde rendah, yang efektif mengurangi noise frekuensi tinggi sambil mempertahankan bentuk dan lebar puncak sinyal.
3. **Butterworth Bandpass Filter:** Tahap akhir filtrasi adalah filter Butterworth bandpass. Filter ini dirancang untuk melewatkan komponen frekuensi hanya dalam rentang yang spesifik dan relevan untuk masing-masing sinyal biologis, sekaligus menolak frekuensi di luar rentang tersebut (termasuk komponen DC dan noise frekuensi tinggi).

- Untuk sinyal rPPG (Detak Jantung): Rentang frekuensi yang dijaga adalah **0.7 Hz – 3.0 Hz** (setara dengan 42 – 180 BPM).
- Untuk sinyal Respirasi: Rentang frekuensi yang dijaga adalah **0.1 Hz – 0.5 Hz** (setara dengan 6 – 30 napas per menit).

Urutan filtrasi ini bertujuan untuk secara bertahap membersihkan dan mengkondisikan sinyal sebelum estimasi frekuensi.

3.4 Estimasi Laju Detak Jantung dan Pernapasan

Setelah sinyal difilter, estimasi laju dilakukan dengan menganalisis sinyal di domain frekuensi menggunakan metode Periodogram (seperti yang dijelaskan di Bab 2.2). Frekuensi yang memiliki daya tertinggi dalam rentang fisiologis yang valid dianggap sebagai frekuensi dominan dan dikonversi ke BPM atau napas per menit.

4 Implementasi Perangkat Lunak

Bab ini menjelaskan arsitektur perangkat lunak, pustaka yang digunakan, dan alur kerja aplikasi berdasarkan struktur kode yang telah dimodularisasi.

4.1 Pustaka dan Struktur Kode

Proyek ini dibangun menggunakan Python dengan pustaka utama yang tercantum dalam `requirements.txt`, meliputi `OpenCV` untuk manajemen video, `MediaPipe` untuk deteksi landmark, `SciPy` dan `NumPy` untuk pemrosesan sinyal, serta `Tkinter` dan `Matplotlib` untuk antarmuka pengguna grafis (GUI).

Kode program telah diorganisir ke dalam struktur direktori yang modular, dengan direktori utama `src_code/root/` berisi logika inti dan direktori `src_code/modules/` berisi komponen-komponen pendukung GUI dan pemrosesan. File `main.py` dalam `src_code/root/` berfungsi sebagai titik masuk utama yang akan menjalankan aplikasi. Kelas utama aplikasi, `RespirasiRPPGApp`, didefinisikan dalam `app.py` (di `src_code/root/`) dan bertanggung jawab untuk inisialisasi GUI serta mengelola state aplikasi. Logika inti pemrosesan video frame-demi-frame, termasuk ekstraksi sinyal dan pembaruan GUI, ditangani oleh modul `video_processing.py` yang berada di dalam direktori `modules`. Modul-modul lain di dalam `modules` seperti `layout.py`, `plotting.py`, dan `recording.py` mendukung fungsionalitas spesifik terkait tampilan dan interaksi pengguna. Sementara itu, logika untuk ekstraksi sinyal mentah (`rppg_signal.py`, `respirasi_signal.py`), filtrasi (`signal_filter.py`), dan utilitas estimasi (`utils.py`) berada di direktori `root` sebagai komponen inti.

4.2 Alur Kerja Aplikasi

Dengan struktur yang baru, alur kerja aplikasi saat pengguna menekan tombol "START" dapat dijelaskan sebagai berikut:

1. `main.py` dieksekusi, yang kemudian mengimpor dan membuat instance dari kelas `RespirasiRPPGApp` dari `app.py`.
2. Konstruktor (`__init__`) pada `app.py` menginisialisasi jendela utama Tkinter, variabel-variabel yang diperlukan (buffer, status, dll.), objek ekstraktor sinyal (`RPPGExtractor` dan `RespirasiExtractor`), dan memanggil fungsi dari `modules/layout.py` (diasumsikan) untuk membangun seluruh elemen visual GUI.
3. Ketika tombol "START" ditekan, fungsi `start_video` dari `modules/video_processing.py` dipanggil. Fungsi ini menginisialisasi penangkapan video menggunakan `OpenCV` dan memulai loop pemrosesan video.

4. Fungsi inti pemrosesan video, yaitu `update_video` dalam `modules/video_processing.py`, dipanggil secara rekursif (menggunakan `app.window.after`) untuk memproses setiap frame. Di dalam fungsi ini:
    - Frame ditangkap dari webcam.
    - Frame dikirim ke metode `extract` dari objek `RPPGExtractor` (dari `rppg_signal.py`) dan `RespirasiExtractor` (dari `respirasi_signal.py`) untuk mendapatkan nilai sinyal mentah.
    - Sinyal mentah ditambahkan ke buffer yang ada di objek `app`.
    - Jika buffer sudah memiliki cukup data, data tersebut dikirim ke fungsi `preprocess_signal` atau fungsi filtering terkait (dari `signal_filter.py`).
    - Sinyal yang telah difilter kemudian dikirim ke fungsi `estimate_heart_rate` atau `estimate_respiratio` (dari `utils.py`) untuk mendapatkan estimasi laju.
    - Hasil estimasi dan sinyal terfilter digunakan untuk memperbarui plot pada GUI melalui fungsi-fungsi di `modules/plotting.py` (diasumsikan).
5. Proses ini berlanjut hingga tombol "STOP" ditekan, yang akan memanggil fungsi `stop_video` dari `modules/video_processing.py` untuk menghentikan loop dan melepaskan sumber daya kamera.

4.3 Kebutuhan Sistem

Untuk menjalankan aplikasi ini, dibutuhkan spesifikasi minimum sebagai berikut:

- Python 3.10 atau versi lebih baru.
- Webcam yang berfungsi (internal atau eksternal).
- Sistem Operasi: Windows, Linux, atau macOS.

4.4 Tinjauan Kode Fungsi Kunci

Berikut adalah penjelasan dari beberapa fungsi kunci yang menjadi inti dari aplikasi ini, dengan merujuk pada struktur file yang baru.

4.4.1 Inisialisasi Aplikasi dan Loop Utama

Kelas `RespirasiRPPGApp` dalam `app.py` mengatur seluruh komponen aplikasi. Konstruktornya menginisialisasi semua variabel penting dan membangun GUI dengan memanggil `init_layout` (diasumsikan dari `modules/layout.py`).

```
# Di dalam src_code/root/app.py
class RespirasiRPPGApp:
    def __init__(self):
        self.hr_plot = None # Dan atribut plot lainnya
        self.window = tk.Tk()
        self.window.title("Realtime rPPG and Respiration Rate Tracker")
        # ... (inisialisasi jendela, variabel, buffer) ...
        try:
            self.respirasi_extractor = RespirasiExtractor()
            self.rppg_extractor = RPPGExtractor()
        except Exception as e:
            messagebox.showerror("Initialization Error", f"Failed to initialize extractors: {str(e)
    }")
            return
```

```
        # ... (inisialisasi buffer sinyal, variabel recording) ...
        init_layout(self) # Memanggil layout builder dari modules/layout.py
```

Kode 1: Inisialisasi Aplikasi dari file `src_code/root/app.py`

Loop utama pemrosesan video ditangani oleh fungsi `update_video` dalam `modules/video_processing.py`. Fungsi ini secara terus-menerus mengambil frame, mengekstraksi sinyal, memfilter, mengestimasi, dan memperbarui GUI.

```
# Di dalam src_code/modules/video_processing.py
def update_video(app):
    if app.running and app.cap:
        try:
            ret, frame = app.cap.read()
            if not ret:
                raise Exception("Failed to read frame from camera")

            # === rPPG Processing ===
            green = app.rppg_extractor.extract(frame)

            # === Respirasi Processing ===
            y_res = app.respirasi_extractor.extract(frame)

            # === Buffer Update ===
            if green is not None:
                app.rppg_buffer.append(green)
                if len(app.rppg_buffer) > app.buffer_max: app.rppg_buffer.pop(0)
            if y_res is not None:
                app.respirasi_buffer.append(y_res)
                if len(app.respirasi_buffer) > app.buffer_max: app.respirasi_buffer.pop(0)

            # === Perekaman Data (30s) ===
            # ... (logika penyimpanan data jika app.recording_30s aktif) ...

            # === Tampilan Frame ke GUI ===
            # ... (konversi frame dan update app.video_label) ...

            # === Update Plot (memanggil fungsi dari modules/plotting.py atau langsung) ===
            update_hr_plot(app)
            update_rr_plot(app)

        except Exception as e:
            app.running = False

    if app.running:
        app.window.after(33, lambda: update_video(app))
```

Kode 2: Loop Pemrosesan Video Inti dari `modules/video_processing.py`

Penjelasan: Fungsi `update_video` adalah inti dari interaksi real-time. Ia mengambil objek `app` sebagai argumen untuk mengakses dan memodifikasi state aplikasi. Pemanggilan rekursif melalui `app.window.after` memastikan pemrosesan berjalan secara kontinu.

4.4.2 Ekstraksi Sinyal rPPG

Fungsi `extract` pada kelas `RPPGExtractor` (dari `src_code/root/rppg_signal.py`) bertanggung jawab untuk mengekstraksi nilai rata-rata kanal hijau dari ROI dahi.

```
# Di dalam src_code/root/rppg_signal.py
def extract(self, frame):
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = self.face_mesh.process(frame_rgb)
    if not results.multi_face_landmarks: return None
```

```
    h, w, _ = frame.shape
    lm = results.multi_face_landmarks[0].landmark
    xs, ys = [], []
    for idx in self.forehead_indices:
        x = int(lm[idx].x * w)
        y = int(lm[idx].y * h)
        xs.append(x); ys.append(y)
    x_min, x_max = max(min(xs),0), min(max(xs),w)
    y_min, y_max = max(min(ys),0), min(max(ys),h)
    roi = frame[y_min:y_max, x_min:x_max]
    if roi.size == 0: return None
    avg_color = np.mean(roi, axis=(0,1))
    return avg_color[1] / 255.0
```

Kode 3: Ekstraksi sinyal rPPG dari file `src_code/root/rppg_signal.py`

4.4.3 Rangkaian Filtrasi Sinyal

Fungsi `preprocess_signal` dari `src_code/root/signal_filter.py` adalah fungsi utama yang dipanggil untuk membersihkan sinyal mentah melalui beberapa tahap filter.

```
# Di dalam src_code/root/signal_filter.py
def preprocess_signal(data, fs, signal_type="rPPG", apply_median=True, apply_savgol=True):
    if not isinstance(data, np.ndarray):
        data = np.array(data)
    if data is None or len(data) < 30:
        return data

    processed_data = data.copy()
    try:
        if apply_median:
            processed_data = apply_median_filter(processed_data, kernel_size=5)
        if apply_savgol:
            window_len = max(5, min(11, len(processed_data) // 3))
            if window_len % 2 == 0: window_len -= 1
            if window_len >= 5:
                processed_data = apply_savgol_filter(processed_data,
                                                     window_length=window_len,
                                                     poly_order=3)

        if signal_type.lower() == "rppg":
            processed_data = apply_bandpass_filter(processed_data, 0.7, 3.0, fs)
        elif signal_type.lower() == "respirasi":
            processed_data = apply_bandpass_filter(processed_data, 0.1, 0.5, fs)
        return processed_data
    except Exception as e:
        print(f"Preprocessing error: {e}. Returning original data.")
        return data
```

Kode 4: Pipeline preprocessing sinyal dari `src_code/root/signal_filter.py`

Fungsi ini secara berurutan menerapkan filter median, Savitzky-Golay (jika diaktifkan), dan akhirnya filter Butterworth bandpass sesuai dengan jenis sinyalnya.

4.4.4 Estimasi Detak Jantung

Fungsi `estimate_heart_rate` dari `src_code/root/utils.py` melakukan estimasi detak jantung dari sinyal terfilter menggunakan periodogram.

```
# Di dalam src_code/root/utils.py
def estimate_heart_rate(signal, fs):
    f, Pxx = periodogram(signal, fs)
```

```
    f_range = (f >= 0.7) & (f <= 3.0)
    if not np.any(f_range): return 0
    peak_freq = f[f_range][np.argmax(Pxx[f_range])]
    return int(peak_freq * 60)
```

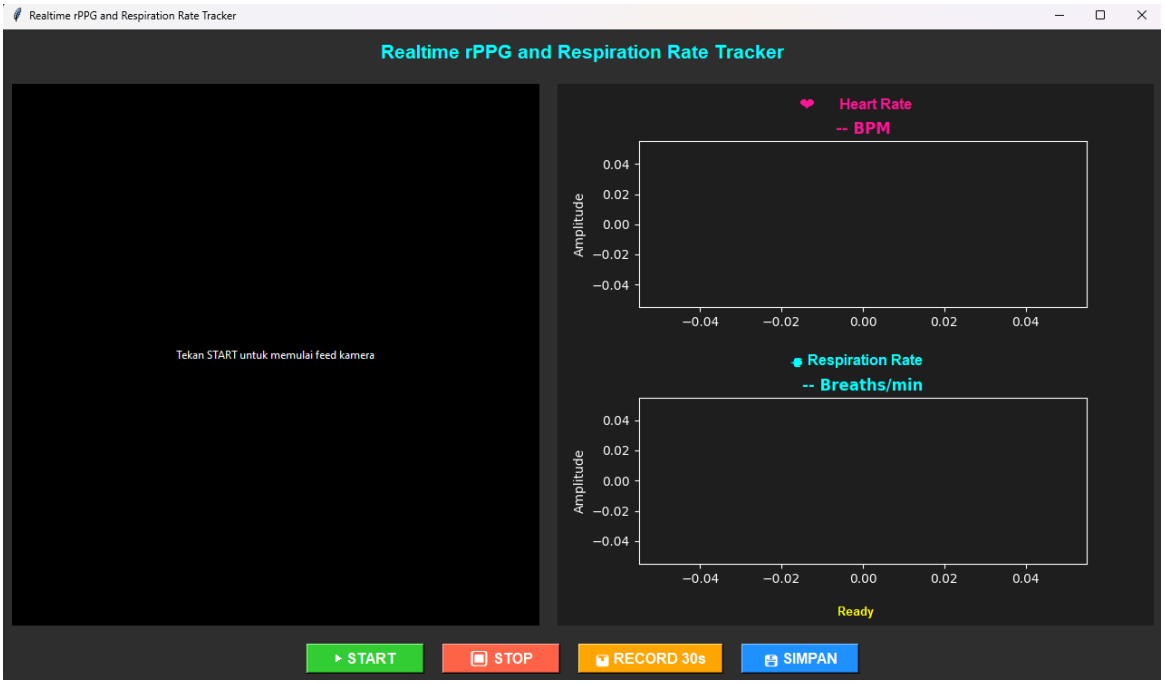
Kode 5: Estimasi detak jantung dari file `src_code/root/utils.py`

Fungsi ini bekerja dengan pertama-tama menghitung Power Spectral Density (PSD) menggunakan `periodogram`. Kemudian, dibuat sebuah `f_range` untuk membatasi pencarian hanya pada frekuensi yang masuk akal untuk detak jantung manusia (0.7 Hz hingga 3.0 Hz). `np.argmax(Pxx[f_range])` menemukan indeks dari daya tertinggi dalam rentang tersebut. Frekuensi yang bersesuaian dengan puncak daya ini (`peak_freq`) kemudian dikalikan 60 untuk mengubah satuannya dari Hz (siklus per detik) menjadi BPM (detak per menit).

5 Hasil dan Analisis

5.1 Antarmuka Pengguna (GUI) Interaktif

Antarmuka Pengguna Grafis (GUI) aplikasi, seperti yang terlihat pada Gambar 1, dirancang dengan gaya gelap (*dark mode*) yang responsif dan memberikan pengalaman pengguna yang intuitif. Pada sisi kiri, terdapat panel utama yang menampilkan feed video langsung dari webcam. Sisi kanan GUI didedikasikan untuk visualisasi sinyal secara *real-time* melalui dua plot terpisah untuk detak jantung dan pernapasan. Di bagian bawah jendela aplikasi, terdapat tombol-tombol kontrol utama.



Gambar 1: Tampilan Antarmuka Pengguna (GUI) Aplikasi.

5.2 Fungsionalitas Penyimpanan Data

Aplikasi menyediakan dua mode untuk menyimpan data, yang sangat berguna untuk analisis offline dan validasi hasil.

- **Simpan Cepat (Tombol 'SIMPAN'):** Menyimpan data buffer sinyal (sekitar 10 detik terakhir).

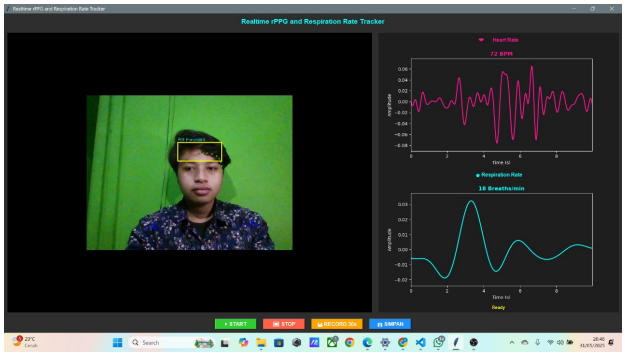
- **Rekaman 30 Detik (Tombol 'RECORD 30s'):** Merekam data selama 30 detik penuh, termasuk sinyal mentah, sinyal terfilter, dan timestamp. Setelah selesai, aplikasi secara otomatis menghasilkan plot perbandingan dalam format `.png` seperti yang dicontohkan pada Gambar 3 dan menyimpan data numerik ke file `.txt`.

5.3 Hasil Pemantauan Sinyal

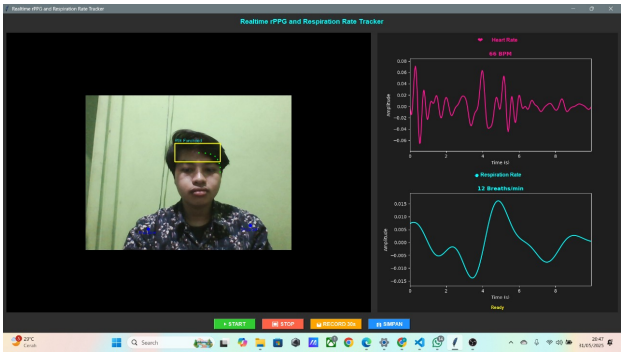
Bagian ini menunjukkan contoh penggunaan aplikasi secara langsung pada subjek. Pengujian dilakukan dalam kondisi ideal untuk memaksimalkan akurasi deteksi, yaitu dengan pencahayaan yang cukup dan subjek yang relatif diam menghadap kamera.

Gambar 2 menampilkan dua tangkapan layar dari sesi pemantauan. Gambar pertama (2a) menunjukkan aplikasi mendeteksi detak jantung sekitar 72 BPM dan laju pernapasan 18 napas/menit. Terlihat dengan jelas kotak kuning ROI (Region of Interest) yang secara otomatis ditempatkan di area dahi subjek untuk ekstraksi sinyal rPPG, beserta titik-titik hijau yang menandakan landmark wajah yang digunakan. Selain itu, titik-titik biru pada area bahu menunjukkan deteksi landmark untuk sinyal pernapasan. Grafik di sisi kanan menampilkan bentuk gelombang sinyal rPPG (atas, berwarna magenta) dan sinyal pernapasan (bawah, berwarna cyan) yang telah difilter secara *real-time*.

Gambar 2b menunjukkan kondisi beberapa saat kemudian, di mana estimasi detak jantung adalah 66 BPM dan laju pernapasan 12 napas/menit. Perbedaan nilai estimasi ini wajar terjadi karena variabilitas alami dari sinyal fisiologis dan sensitivitas metode terhadap perubahan kecil pada subjek atau lingkungan. Kedua gambar ini mengilustrasikan kemampuan aplikasi untuk secara kontinu melacak landmark, mengekstraksi sinyal, dan memberikan estimasi laju secara visual dan numerik. Keberhasilan deteksi ROI dan landmark yang stabil, seperti yang ditunjukkan, merupakan prasyarat penting untuk akurasi ekstraksi sinyal.



(a) Pemantauan sinyal dengan estimasi 72 BPM dan 18 napas/menit.



(b) Pemantauan sinyal beberapa saat kemudian dengan estimasi 66 BPM dan 12 napas/menit.

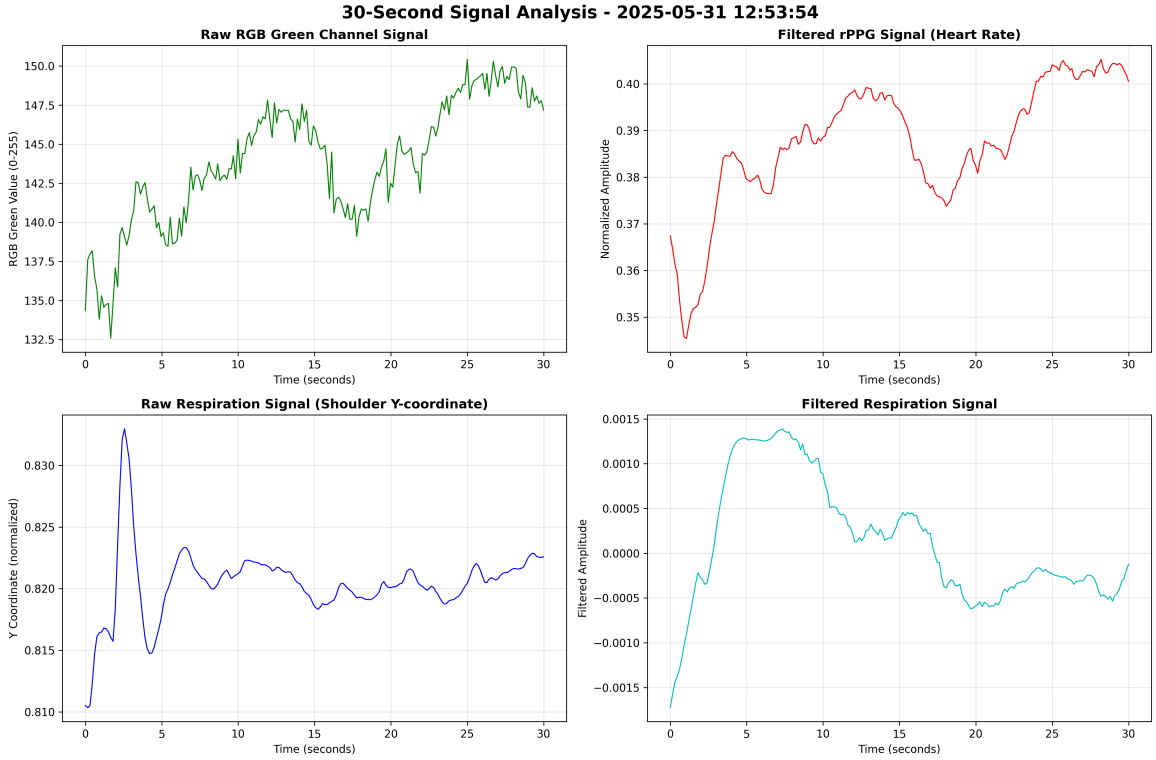
Gambar 2: Contoh hasil pemantauan sinyal pada subjek pengguna pada dua waktu berbeda.

5.4 Analisis Kualitatif dan Batasan

Secara kualitatif, aplikasi berhasil menunjukkan sinyal periodik untuk detak jantung dan pernapasan dalam kondisi ideal. Gambar 3 menunjukkan contoh sinyal rPPG terfilter (kanan atas) dan pernapasan terfilter (kanan bawah) yang menampilkan osilasi periodik. Sinyal mentah (kiri atas dan kiri bawah) terlihat jauh lebih bising.

Namun, akurasi metode ini dipengaruhi oleh faktor eksternal:

- **Gerakan Subjek:** Dapat menimbulkan motion artifact.
- **Kondisi Pencahayaan:** Dapat menurunkan kualitas sinyal rPPG.



Gambar 3: Contoh Plot Analisis Sinyal 30 Detik.

- **Kualitas Kamera:** Mempengaruhi kebisingan sinyal.

Untuk hasil terbaik, pengguna disarankan untuk tetap relatif diam dan berada di lingkungan dengan pencahayaan yang cukup dan stabil.

6 Kesimpulan

Proyek ini berhasil mengimplementasikan sistem non-kontak untuk memantau laju detak jantung dan pernapasan menggunakan webcam standar. Dengan memanfaatkan teknik dari pengolahan sinyal digital, terutama rangkaian filter Median, Savitzky-Golay, dan Butterworth bandpass, serta analisis spektral menggunakan periodogram, program mampu mengekstraksi informasi fisiologis yang bermakna dari sinyal video yang bising. Antarmuka pengguna yang interaktif dan fungsionalitas penyimpanan data mendukung kemudahan penggunaan dan analisis lebih lanjut.

7 Referensi

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

[2] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2010.

[3] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," 2020.

[4] W. Verkruysse, L. O. Svaasand, and J. S. Nelson, "Remote plethysmographic imaging using ambient light," *Optics Express*, vol. 16, no. 26, pp. 21 434–21 445, 2008.

[5] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile gpus," 2019.

A Lampiran

- **Tautan Video Demonstrasi:** https://youtu.be/0mAjfjLR54w
- **Tautan Bukti Bantuan AI:** https://drive.google.com/drive/folders/1NLZuOKd4hD_8J0Wj6zFGVSoJ1z81RhXt?usp=sharing
- **Tautan Projek Overleaf:** https://www.overleaf.com/read/dznkcrwrkbrt#87a7ec
- Kami juga menggunakan bantuan github copilot pada VSCode