Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama:

1. **Freddy Harahap (122140018)**
2. **Dwi Arthur Revangga (122140144)**
3. **Siti Nur Aarifah (122450006)**

Tugas: **Tugas Besar**

Mata Kuliah: **Digital Signal Processing (IF3024)**

Tanggal: 30 Mei 2025

1 Pendahuluan

Perkembangan teknologi dalam bidang *Digital Signal Processing* (DSP) telah membuka banyak peluang untuk meningkatkan kualitas layanan kesehatan, khususnya dalam pemantauan kondisi fisiologis secara non-invasif dan *real-time*. Salah satu pendekatan yang kini semakin berkembang adalah pemanfaatan kamera, seperti webcam, untuk mengukur parameter vital tubuh tanpa perlu kontak langsung, dengan *remote photoplethysmography* (rPPG) untuk mendeteksi detak jantung dan pelacakan gerakan dada untuk laju pernapasan.

Pada Tugas Besar mata kuliah IF3024 - Pengolahan Sinyal Digital ini, dikembangkan sebuah sistem yang mampu mengekstraksi sinyal rPPG dan respirasi langsung dari video webcam. Sistem ini mengandalkan MediaPipe untuk mendeteksi pose tubuh dan wajah secara akurat, serta menerapkan algoritma *Plane-Orthogonal-to-Skin* (POS) untuk mengestimasi sinyal rPPG dari variasi warna mikro pada kulit wajah. POS merupakan metode yang terbukti efektif untuk mengekstraksi sinyal denyut jantung secara jarak jauh (*remote*) tanpa kontak langsung dengan kulit, sebagaimana dijelaskan oleh [1] bahwa metode ini mampu memproyeksikan sinyal warna kulit ke domain orthogonal untuk meningkatkan rasio sinyal terhadap gangguan cahaya dan gerakan kecil.Di sisi lain, untuk mendeteksi sinyal respirasi, digunakan pendekatan berbasis pelacakan gerakan vertikal bahu menggunakan *Lucas-Kanade Optical Flow* yang ditanamkan pada titik-titik hasil deteksi awal dari *pose landmark.*

Kualitas sinyal yang baik dipengaruhi oleh pencahayaan, posisi wajah, serta parameter filter yang digunakan, seperti *lowcut*, *highcut*, dan *order filter*. Oleh karena itu, pada sistem ini diimplementasikan algoritma *Cat Swarm Optimization* (CSO) untuk mengoptimasi parameter-parameter tersebut. CSO bekerja dengan prinsip pencarian terbaik di antara populasi kandidat, dan dalam konteks ini digunakan untuk memaksimalkan kualitas sinyal dengan cara mencari nilai *Signal-to-Noise Ratio* (SNR) tertinggi setelah *filtering*. Pendekatan ini diambil karena SNR merupakan indikator penting dalam menilai kejernihan sinyal fisiologis dari gangguan dan noise lingkungan CSO [2].

CSO sendiri merupakan algoritma metaheuristik yang terinspirasi dari perilaku kucing, memiliki dua mode eksplorasi yaitu *seeking* dan *tracking*, yang telah diperkenalkan oleh [2] sebagai alternatif yang efisien untuk masalah optimasi global nonliniercso. Hasil dari proses ini ditampilkan secara *real-time* dalam sebuah antarmuka GUI berbasis Tkinter, yang menyediakan estimasi denyut jantung (BPM), laju napas (BR), serta opsi untuk melakukan optimasi otomatis dan penyimpanan data ke dalam file CSV.

2 Alat dan Bahan

Untuk menyelesaikan proyek ini, diperlukan beberapa alat dan bahan untuk membuat program yang dapat memperoleh sinyal respirasi dan sinyal rPPG secara non-kontak. Berikut merupakan alat dan bahan yang digunakan:

2.1 Bahasa Pemrograman

Dalam pengembangan program proyek ini, digunakan bahasa pemrograman Python karena memiliki banyak *library* pendukung dalam bidang pemrosesan sinyal dan visi komputer, serta sangat cocok untuk membangun aplikasi prototipe yang kompleks dengan cepat.

2.2 *Library*

Berikut merupakan pustaka (*library*) yang digunakan pada pengembangan proyek ini:

1. **Tkinter**
   Digunakan untuk membangun antarmuka pengguna grafis (GUI) berbasis Python seperti tampilan video, tombol kontrol, dan grafik sinyal secara *real-time*.
2. **OpenCV (cv2)**
   Digunakan untuk mengakses kamera, membaca *frame* video secara langsung, serta melakukan pemrosesan citra (misalnya *cropping ROI*, *resizing*, konversi warna).
3. **MediaPipe (mp)**
   Digunakan untuk mendeteksi *landmark* wajah dan pose tubuh, terutama pada area bahu dan wajah. Model *face detector* dan *pose landmarker* digunakan dalam mode *real-time*.
4. **NumPy**
   Digunakan untuk pemrosesan data numerik berbasis *array*, manipulasi sinyal, serta operasi statistik seperti rata-rata dan standar deviasi.
5. **SciPy**
   Digunakan untuk proses filtering sinyal menggunakan *Butterworth bandpass filter*, serta deteksi puncak sinyal dengan fungsi `find_peaks`.
6. **Matplotlib**
   Digunakan untuk menampilkan sinyal rPPG dan respirasi dalam bentuk grafik di dalam GUI. Modul *back-end* `FigureCanvasTkAgg` digunakan untuk integrasi dengan Tkinter.
7. **PIL (Pillow)**
   Digunakan untuk konversi frame OpenCV menjadi format yang bisa ditampilkan di Tkinter (`ImageTk.PhotoImage`).
8. **Threading**
   Digunakan untuk menjalankan proses perekaman dan pemrosesan secara paralel agar GUI tetap responsif.

9. **Time** dan **Datetime**
   Digunakan untuk menghitung durasi perekaman, mencatat timestamp, dan memberi nama file hasil rekaman berdasarkan waktu.
10. **os**
    Digunakan untuk membuat folder dan mengatur path file hasil rekaman CSV.
11. **Ctypes**
    Digunakan untuk mengatur *DPI awareness* agar tampilan GUI tidak buram pada layar dengan resolusi tinggi.
12. **Random**
    Digunakan dalam implementasi algoritma *Cat Swarm Optimization* untuk menginisialisasi populasi dan variasi kandidat parameter.
13. **Collections.deque**
    Digunakan untuk menyimpan buffer sinyal RGB dan respirasi secara efisien dengan batas waktu (*rolling buffer*).

2.3 Metode dan Algoritma

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

1. **Ekstraksi Sinyal rPPG**
   Dilakukan menggunakan metode *Plane-Orthogonal-to-Skin* (POS) dengan input sinyal RGB dari wajah. POS mengolah sinyal warna secara spasial untuk mengekstrak sinyal denyut jantung dari perubahan warna mikro kulit wajah [1].
2. **Ekstraksi Sinyal Respirasi**
   Menggunakan pelacakan gerakan vertikal titik bahu dengan algoritma *Lucas-Kanade Optical Flow*, yang diinialisasi berdasarkan deteksi awal menggunakan MediaPipe *PoseLandmarker* [3].
3. **Filtering Sinyal**
   Sinyal rPPG dan respirasi difilter menggunakan *Butterworth bandpass filter* orde n yang dirancang dengan pustaka `scipy.signal`.
4. **Optimasi Parameter Filter**
   Parameter filter (*lowcut*, *highcut*, dan *order*) dioptimasi menggunakan algoritma *Cat Swarm Optimization* (CSO) yang bertujuan memaksimalkan rasio sinyal terhadap noise (*Signal-to-Noise Ratio*) [2].

3 Penjelasan

Proyek ini dibagi menjadi lima modul utama yaitu:

1. `gui_app.py`: Antarmuka pengguna berbasis Tkinter untuk perekaman dan visualisasi sinyal rPPG dan respirasi secara real-time.
2. `filter_utils.py`: Berisi fungsi filter sinyal digital, khususnya *filter bandpass Butterworth*.

3. `resp_utils.py`: Berisi deteksi pose dan pelacakan gerakan bahu untuk sinyal respirasi menggunakan *Optical Flow*.
4. `rppg_utils.py`: Berisi algoritma POS (*Plane-Orthogonal-to-Skin*) untuk ekstraksi sinyal rPPG dari perubahan warna wajah.
5. `cso.py`: Berisi implementasi algoritma *Cat Swarm Optimization* untuk mencari parameter filter terbaik berdasarkan SNR.

3.1 Instalasi Library

Dengan asumsi bahwa Anda sudah memiliki *environment manager* seperti `conda`, langkah instalasi adalah sebagai berikut:

1. Buat environment baru:

   ```
   conda create -n respiration python
   ```

2. Aktifkan environment:

   ```
   conda activate respiration
   ```

3. Install semua *library* yang dibutuhkan (link github repo)

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

4. Jalankan program:

   ```
   python gui_app.py
   ```

3.2 Modul `gui_app.py`

Modul `gui_app.py` merupakan antarmuka utama dari aplikasi ini yang dibuat menggunakan pustaka `Tkinter`. Modul ini berfungsi untuk:

- Menampilkan video webcam secara *real-time*.
- Melakukan rekaman sinyal rPPG dan respirasi.
- Menampilkan grafik sinyal secara langsung.
- Melakukan optimasi parameter filter dengan algoritma *Cat Swarm Optimization*.
- Memberikan antarmuka pengguna yang interaktif.

3.2.1 Inisialisasi *Library* dan Konstanta

Berikut adalah pustaka dan konstanta yang digunakan pada file `gui_app.py`:

```
import tkinter as tk
from tkinter import messagebox
from threading import Thread
import cv2
import numpy as np
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import os
import time
from datetime import datetime
from scipy.signal import find_peaks
from collections import deque
import ctypes

from rppg_utils import extract_rppg
from resp_utils import create_pose_landmarker, RespTracker
from filter_utils import bandpass_filter
from cso import cat_swarm_optimize, bandpass_and_eval

FPS = 30.0
DEFAULT_LOW_RPPG = 0.8
DEFAULT_HIGH_RPPG = 2.5
DEFAULT_ORDER = 4
LOW_RESP, HIGH_RESP = 0.1, 0.7
```

Kode 1: Import Library dan Konstanta

3.2.2 Kelas `GUIApp`

Kelas `GUIApp` adalah kelas utama yang mengatur seluruh antarmuka aplikasi. Dalam kelas ini, terdapat komponen untuk menampilkan video, grafik sinyal, dan kontrol tombol (mulai rekam, optimasi, keluar, dan sebagainya).

```
class GUIApp:
    """
    Kelas utama GUI berbasis Tkinter untuk pemrosesan sinyal rPPG dan respirasi secara real-time.
    Menggabungkan input webcam, deteksi wajah/bahu, filter, dan plotting sinyal.
    """
    def __init__(self, master):
        self.master = master
        self.master.title("DSP GUI - rPPG & Respirasi")
        ...
```

Kode 2: Kelas GUIApp untuk GUI aplikasi

Konstruktor `__init__()` mengatur tampilan jendela utama, menginisialisasi *video capture*, *buffer* sinyal RGB dan respirasi, serta menyiapkan plot *matplotlib* yang ditanam ke dalam GUI Tkinter.

3.2.3 Fungsi `start_recording()`

Fungsi ini digunakan untuk memulai proses perekaman sinyal. Proses diawali dengan *countdown* selama 5 detik, kemudian dilanjutkan dengan perekaman sinyal rPPG dan respirasi selama durasi yang

dimasukkan oleh pengguna.

```
def start_recording(self):
    """
    Melakukan perekaman sinyal rPPG dan respirasi dari webcam selama durasi tertentu.
    Hasil disimpan ke file CSV.
    """
    try:
        duration_sec = int(self.duration_entry.get())
        ...
```

Kode 3: Fungsi start_recording

Fungsi ini juga menangani pemanggilan *pose detector* untuk *tracking* bahu serta pengisian `deque` sinyal untuk digunakan dalam optimasi dan plotting.

3.2.4 Fungsi Optimasi Filter

Dua fungsi optimasi terpisah disediakan yaitu:

- `run_filter_optimization()`: Untuk optimasi parameter filter rPPG.
- `run_resp_optimization()`: Untuk optimasi parameter filter respirasi.

Kedua fungsi tersebut memanfaatkan algoritma CSO dari modul `cso.py`.

```
def run_filter_optimization(self):
    """
    Melakukan optimasi parameter filter rPPG menggunakan algoritma Cat Swarm Optimization (CSO).
    """
    rgb_arr = np.array(self.rgb_buffer).T
    ...
```

Kode 4: Fungsi Optimasi Filter rPPG

3.2.5 Fungsi Plot dan Visualisasi

Fungsi `update_realtime_plot()` akan menampilkan sinyal rPPG dan respirasi yang telah difilter ke dalam GUI. Selain itu, juga menghitung estimasi BPM dan BR berdasarkan puncak sinyal.

```
def update_realtime_plot(self):
    """
    Memperbarui grafik matplotlib dengan sinyal rPPG dan respirasi terbaru.
    Menampilkan titik puncak dan menghitung estimasi BPM dan BR.
    """
    ...
```

Kode 5: Fungsi update realtime plot sinyal

Fungsi ini dijalankan setiap beberapa detik selama perekaman berlangsung untuk memastikan sinyal yang ditampilkan selalu *up-to-date* dan informatif.

3.3 Modul `filter_utils.py`

Modul `filter_utils.py` digunakan untuk menangani proses filtering sinyal dengan metode digital. Filter yang digunakan pada proyek ini adalah *filter Butterworth* dengan tipe `bandpass`, yang bertujuan menyaring sinyal dalam frekuensi tertentu dan membuang noise di luar jangkauan tersebut.

3.3.1 Fungsi `bandpass_filter`

Fungsi utama dari modul ini adalah `bandpass_filter`, yang menggunakan pustaka `scipy.signal` untuk membangun dan menerapkan filter pada sinyal input.

```python
import numpy as np
from scipy import signal

def bandpass_filter(data: np.ndarray, lowcut: float, highcut: float,
                    fs: float, order: int = 5) -> np.ndarray:
    """
    Menerapkan filter band-pass Butterworth pada sinyal.

    Parameter:
    - data: array 1D dari sinyal masukan
    - lowcut: frekuensi batas bawah (Hz)
    - highcut: frekuensi batas atas (Hz)
    - fs: frekuensi sampling (Hz)
    - order: orde filter (default = 5)

    Return:
    - filtered_data: sinyal hasil filtering
    """
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = signal.butter(order, [low, high], btype='band')
    return signal.filtfilt(b, a, data)
```

Kode 6: Fungsi bandpass_filter pada filter_utils.py

3.3.2 Penjelasan Proses

Langkah-langkah yang dilakukan oleh fungsi `bandpass_filter` adalah sebagai berikut:

1. Menentukan frekuensi Nyquist (`nyq`), yaitu setengah dari frekuensi sampling.
2. Mengubah batas bawah dan atas dari domain Hz ke domain normalisasi (0–1).
3. Mendesain koefisien *filter Butterworth* menggunakan `signal.butter`.
4. Menerapkan filter dua arah (*zero-phase*) dengan `signal.filtfilt` untuk menghindari distorsi fasa.

3.3.3 Contoh Penggunaan dalam Program Utama

Fungsi ini digunakan di dalam `gui_app.py` saat proses plotting sinyal rPPG dan respirasi:

```
rppg = extract_rppg(rgb_arr, fps=FPS, lowcut=low_rppg, highcut=high_rppg)
resp = bandpass_filter(np.array(self.resp_buffer), LOW_RESP, HIGH_RESP, fs=FPS)
```

Kode 7: Contoh pemanggilan bandpass_filter di gui_app.py

Melalui pendekatan ini, pengguna dapat mengatur parameter ***lowcut***, ***highcut***, dan ***order*** untuk menyesuaikan filter dengan sinyal yang dihasilkan, baik melalui input manual atau melalui proses optimasi.

3.4 Modul `resp_utils.py`

Modul `resp_utils.py` bertanggung jawab terhadap proses ekstraksi sinyal respirasi dari video webcam. Modul ini menggabungkan kemampuan deteksi pose dari MediaPipe dan pelacakan gerakan bahu menggunakan *Optical Flow*.

3.4.1 Fungsi `create_pose_landmarker`

Fungsi ini digunakan untuk memuat dan mengonfigurasi model `pose_landmarker` dari MediaPipe.

```
def create_pose_landmarker(model_path: str, use_gpu: bool=False):
    from mediapipe.tasks.python.core.base_options import BaseOptions
    from mediapipe.tasks.python.vision import PoseLandmarker, PoseLandmarkerOptions, RunningMode

    with open(model_path, "rb") as f:
        model_content = f.read()

    base_options = BaseOptions(
        model_asset_buffer=model_content,
        delegate=BaseOptions.Delegate.GPU if use_gpu else BaseOptions.Delegate.CPU
    )

    options = PoseLandmarkerOptions(
        base_options=base_options,
        running_mode=RunningMode.VIDEO,
        num_poses=1,
        min_pose_detection_confidence=0.5,
        min_pose_presence_confidence=0.5,
        min_tracking_confidence=0.5
    )

    print(f"[DEBUG] Loaded model from buffer: {model_path}")
    return PoseLandmarker.create_from_options(options)
```

Kode 8: Fungsi create_pose_landmarker

Fungsi ini mengembalikan objek `PoseLandmarker` yang siap digunakan untuk mendeteksi *landmark pose* tubuh dari *frame* video.

3.4.2 Kelas `RespTracker`

Kelas ini bertugas melakukan pelacakan pergerakan vertikal bahu menggunakan algoritma *Lucas-Kanade Optical Flow* setelah titik bahu terdeteksi pada *frame* awal.

```python
class RespTracker:
    def __init__(self, landmarker, x_size=100, y_size=100, shift_x=0, shift_y=0):
        ...
        self.lk_params = dict(
            winSize=(15, 15), maxLevel=2,
            criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03)
        )
```

Kode 9: Kelas RespTracker

Metode `initialize()`: Bertugas mendeteksi posisi awal bahu kiri dan kanan dari *landmark pose*, kemudian mengekstrak ROI di sekitar bahu untuk proses pelacakan titik fitur.

```python
def initialize(self, frame: np.ndarray, timestamp_ms: int):
    h, w = frame.shape[:2]
    img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    mp_img = mp.Image(image_format=mp.ImageFormat.SRGB, data=img_rgb)
    res = self.landmarker.detect_for_video(mp_img, timestamp_ms=timestamp_ms)
    ...
```

Kode 10: Metode initialize pada RespTracker

Metode `update()`: Digunakan untuk menghitung rata-rata posisi vertikal titik-titik fitur yang dilacak dari *frame* ke *frame*.

```python
def update(self, frame: np.ndarray) -> float:
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    new_pts, status, _ = cv2.calcOpticalFlowPyrLK(self.old_gray, gray, self.features, None, **self.
    lk_params)
    good_new = new_pts[status == 1].reshape(-1, 2)
    self.features = good_new.reshape(-1, 1, 2)
    self.old_gray = gray
    return float(np.mean(good_new[:, 1]))
```

Kode 11: Metode update pada RespTracker

Output dari metode ini adalah nilai posisi vertikal rata-rata yang kemudian disimpan dalam buffer dan digunakan sebagai sinyal respirasi.

3.5 Modul `rppg_utils.py`

Modul `rppg_utils.py` bertugas melakukan proses ekstraksi sinyal *remote-photoplethysmography* (rPPG) dari data warna (RGB) wajah. Metode yang digunakan adalah ***Plane-Orthogonal-to-Skin (POS)***, yaitu salah satu pendekatan berbasis proyeksi spasial untuk menghasilkan sinyal denyut jantung dari variasi warna kulit yang direkam oleh kamera.

3.5.1 Fungsi `cpu_POS`

Fungsi `cpu_POS` adalah implementasi algoritma POS pada CPU menggunakan `NumPy`. Fungsi ini menerima input dalam bentuk *array* RGB berukuran $(e, 3, f)$, di mana:

- e = jumlah *estimator* (biasanya 1),

- 3 = kanal warna R, G, B,
- f = jumlah *frame*.

```python
def cpu_POS(X: np.ndarray, fps: float) -> np.ndarray:
    eps = 1e-9
    e, c, f = X.shape
    w = int(1.6 * fps)
    P = np.array([[0, 1, -1], [-2, 1, 1]])
    Q = np.stack([P]*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)
        Cn = Cn * M[:, :, None]
        S = np.tensordot(Q, Cn, axes=([2],[1]))
        Hnm = np.zeros((e, w))
        for i in range(e):
            S1, S2 = S[i, :, :]
            alpha = np.std(S1) / (np.std(S2) + eps)
            Hn = S1 + alpha * S2
            Hnm[i] = Hn - np.mean(Hn)
        H[:, m:n+1] += Hnm
    return H
```

Kode 12: Fungsi cpu_POS

Proses utama yang dilakukan meliputi:

- Normalisasi temporal pada *window RGB*.
- Proyeksi RGB ke dua vektor orthogonal.
- Penyesuaian *(tuning)* dan kombinasi sinyal.
- Penjumlahan sinyal pada sliding window secara bertumpuk (*overlap-adding*).

3.5.2 Fungsi `extract_rppg`

Fungsi `extract_rppg` bertindak sebagai wrapper untuk mengeksekusi `cpu_POS` dan sekaligus menerapkan *filter bandpass* untuk membersihkan sinyal dari *noise*.

```python
def extract_rppg(rgb_buffer: np.ndarray, fps: float,
                 lowcut: float = 0.8, highcut: float = 2.5,
                 filter_order: int = 5) -> np.ndarray:
    sig = rgb_buffer[np.newaxis, ...]
    raw = cpu_POS(sig, fps=fps).flatten()
    return bandpass_filter(raw, lowcut, highcut, fs=fps, order=filter_order)
```

Kode 13: Fungsi extract_rppg

Fungsi ini dipanggil pada saat *plotting* sinyal rPPG di GUI. *Buffer* RGB akan diproses melalui POS, lalu difilter dengan `bandpass_filter` dari `filter_utils.py`, dan dikembalikan sebagai sinyal satu dimensi.

3.5.3 Contoh Penggunaan

Berikut adalah contoh pemanggilan fungsi ini di dalam aplikasi GUI:

```
rppg = extract_rppg(rgb_arr, fps=FPS,
                    lowcut=low_rppg, highcut=high_rppg,
                    filter_order=order)
```

Kode 14: Contoh penggunaan extract_rppg di gui_app.py

Melalui implementasi ini, sinyal rPPG dapat dihasilkan dan dibersihkan secara efisien dari input video webcam pada sistem *real-time*.

3.6 Modul `cso.py`

Modul `cso.py` berfungsi sebagai implementasi algoritma *Cat Swarm Optimization* (CSO) yang digunakan dalam proyek ini untuk mencari parameter optimal dari *filter bandpass* baik untuk sinyal rPPG maupun respirasi.

3.6.1 Bagaimana Cara Kerja CSO?

CSO adalah algoritma optimasi yang terinspirasi dari perilaku kucing. Dalam dunia nyata, kucing memiliki dua kebiasaan utama: diam mengamati *(seeking)* dan aktif berburu atau mengejar *(tracking)*. CSO meniru dua perilaku ini untuk mencari solusi terbaik dari suatu masalah.

Secara sederhana, cara kerja algoritma ini ialah sebagai berikut:

1. Algoritma memulai dengan sekelompok *kucing virtual*, yang masing-masing mewakili kombinasi parameter (misalnya batas frekuensi filter).
2. Beberapa kucing menjalankan ***seeking mode***, yaitu mereka mencoba beberapa kemungkinan di sekitar posisi mereka sekarang dan memilih yang terbaik.
3. Kucing lainnya masuk ke ***tracking mode***, yaitu bergerak mendekati kucing terbaik yang pernah ditemukan.
4. Proses ini diulang berkali-kali (*iterasi*) hingga ditemukan kombinasi parameter yang menghasilkan kualitas sinyal terbaik, berdasarkan nilai ***Signal-to-Noise Ratio*** (SNR).

Dengan demikian, algoritma CSO mampu menyeimbangkan eksplorasi (mencari kemungkinan baru) dan eksploitasi (mengikuti solusi terbaik) dalam proses optimasi filter rPPG dan respirasi secara otomatis.

3.6.2 Fungsi `fitness_snr`

Fungsi ini menghitung kualitas sinyal berdasarkan nilai *Signal-to-Noise Ratio* (SNR), yang kemudian digunakan sebagai nilai objektif *(fitness)*. Perlu diketahui sebelumnya bahwa algoritma CSO bertujuan untuk meminimalkan nilai *fitness*. Akan tetapi, kualitas sinyal yang semakin bagus berdasarkan SNR ialah nilai yang semakin besar. Oleh karena itu, cara menggunakan fungsi ini ialah menggunakan negatif dari SNR untuk menghasilkan nilai yang semakin kecil *(minimum)*.

```python
def fitness\_snr(signal):
    snr = np.mean(signal)**2 / (np.std(signal)**2 + 1e-8)
    return -snr
```

Kode 15: Fungsi fitness_snr

3.6.3 Fungsi `bandpass_and_eval`

Fungsi ini bertanggung jawab untuk menerapkan filter ke sinyal berdasarkan parameter yang diberikan. Lalu, mengevaluasi hasilnya menggunakan `fitness_snr`. Ini digunakan sebagai fungsi objektif *(fitness)* untuk CSO.

```python
def bandpass_and_eval(signal, fs, apply_filter, param_set):
    lowcut, highcut, order = param_set
    if lowcut >= highcut or order < 2 or order > 8:
        return 1e9
    if len(signal) < 3 * fs:
        return 1e9
    try:
        filtered = apply_filter(signal, lowcut, highcut, fs, order=int(order))
        fitness_value = fitness_snr(filtered)
        return fitness_value if np.isfinite(fitness_value) else 1e9
    except Exception:
        return 1e9
```

Kode 16: Fungsi bandpass_and_eval

3.6.4 Fungsi `cat_swarm_optimize`

Fungsi ini merupakan implementasi utama dari algoritma *Cat Swarm Optimization*. CSO meniru dua perilaku utama kucing: ***`seeking`*** (eksplorasi solusi) dan ***`tracking`*** (eksploitasi solusi). Populasi kandidat disebut **kucing** dievaluasi dan diperbarui berdasarkan parameter *fitness*.

```python
def cat_swarm_optimize(
    objective_func, bounds, n_cats=10, max_iter=30,
    mixture_ratio=0.5, srd=0.2, smp=5):
    ...
    for it in range(max_iter):
        for i in range(n_cats):
            if random.random() < mixture_ratio:
                # SEEKING MODE
                ...
            else:
                # TRACKING MODE
                ...
    return best_cat, best_score
```

Kode 17: Fungsi cat_swarm_optimize

Penjelasan Parameter:

- `bounds`: batas bawah dan atas untuk setiap parameter yang akan dioptimasi.
- `n_cats`: jumlah kandidat dalam populasi.

- `max_iter`: jumlah iterasi maksimum.
- `mixture_ratio`: rasio antara mode *seeking* dan *tracking*.
- `srd`, `smp`: parameter eksplorasi dalam mode *seeking*.

3.6.5 Contoh Pemanggilan di GUI

Fungsi `cat_swarm_optimize` dipanggil untuk mengoptimasi parameter filter rPPG dan respirasi dalam antarmuka GUI:

```
best_param, best_score = cat_swarm_optimize(
    objective_func=obj,
    bounds=bounds,
    n_cats=12,
    max_iter=25
)
```

Kode 18: Contoh penggunaan cat_swarm_optimize

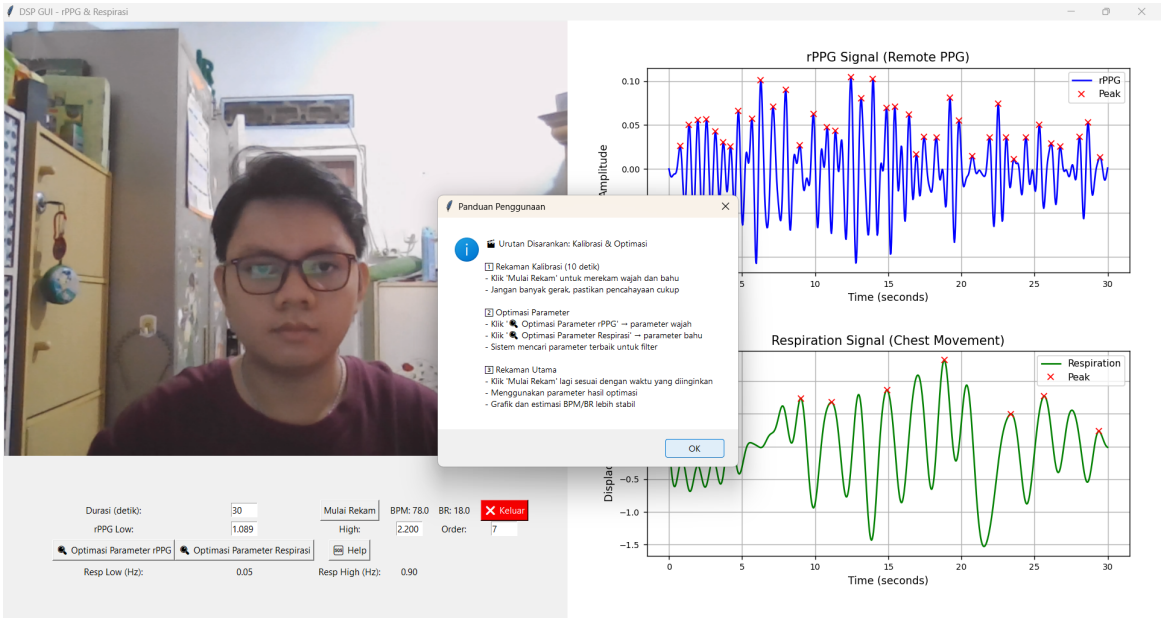
Melalui penerapan ini, program bisa secara otomatis menyesuaikan filter guna menghasilkan sinyal yang lebih jernih dengan cara memaksimalkan SNR dari hasil *filtering*.

4 Hasil

Pada bagian ini disajikan hasil pengujian sistem secara langsung melalui antarmuka GUI. Sistem diuji pada dua pengguna yang berbeda dengan kondisi pencahayaan dan latar belakang yang juga berbeda. Proses perekaman dilakukan dengan durasi yang berbeda: 10 detik, 30 detik, dan 60 detik. Sinyal rPPG dan respirasi kemudian divisualisasikan secara *real-time*.

4.1 Tampilan GUI dan Panduan

Gambar 1 menunjukkan tampilan antarmuka utama aplikasi. Tampilan GUI dibagi menjadi dua: kanan dan kiri. Pada bagian kiri terdapat tampilan video dari webcam, sementara pada bagian kanan ditampilkan grafik sinyal rPPG dan respirasi yang diperbarui secara dinamis. Di bagian bawah tampilan video dari webcam, terdapat berbagai input parameter, tombol rekam, tombol optimasi, serta informasi estimasi BPM dan BR.



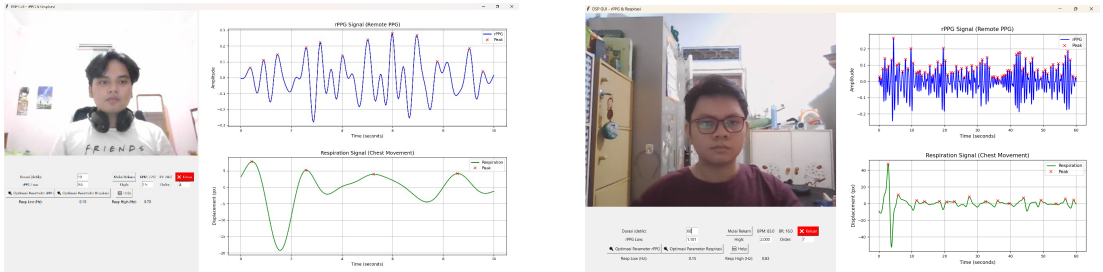
Gambar 1: Tampilan GUI aplikasi dengan fitur rekam, optimasi, dan visualisasi sinyal

Sistem juga menyediakan tombol bantuan (`Help`) yang menampilkan panduan langkah-langkah penggunaan secara jelas. Langkah yang disarankan adalah: (1) rekaman kalibrasi, (2) optimasi parameter, dan (3) rekaman utama.

4.2 Hasil Ekstraksi Sinyal rPPG dan Respirasi

Pada pengujian terhadap pengguna pertama yang bisa dilihat pada 1, sinyal rPPG dan respirasi berhasil diekstraksi secara stabil selama 30 detik. Estimasi detak jantung mencapai sekitar 78 BPM, dan laju napas 18 BR. Optimasi filter menghasilkan parameter `Lowcut = 1.089 Hz`, `Highcut = 2.200 Hz`, dan `Order = 7`. Sinyal menunjukkan bentuk gelombang periodik dengan puncak-puncak yang terdeteksi dengan baik, seperti ditunjukkan pada Gambar 1 yang ditampilkan sebelumnya.

Lalu, dilakukan dua pengujian lagi dengan durasi 10 detik dan 60 detik. Hasilnya dapat dilihat sebagai berikut:



(a) Hasil 10 detik (b) Hasil 60 detik

Gambar 2: Hasil percobaan 10 detik dan 60 detik pada pengguna yang berbeda

Pada pengujian terhadap pengguna kedua, durasi rekaman diubah menjadi 10 detik. Estimasi detak jantung adalah 72 BPM dan laju napas sekitar 24 BR. Parameter filter rPPG yang digunakan adalah `Lowcut = 0.8 Hz`, `Highcut = 2.5 Hz`, dengan `Order = 4`. Sinyal rPPG menunjukkan fluktuasi yang

jelas dan konsisten, sedangkan sinyal respirasi memiliki bentuk gelombang yang lebih lebar dan teratur bisa dilihat pada Gambar 2a.

Lalu, pengujian dilakukan lagi oleh pengguna sebelumnya dengan durasi rekaman diubah menjadi 60 detik. Hasilnya menunjukkan estimasi detak jantung adlaah 83 BPM dan laju napas sekitar 16 BR. Parameter filter rPPG dari hasil optimasi ialah `Lowcut = 1.101 Hz`, `Highcut = 2 Hz`, dengan `Order = 7`. Sinyal rPPG menunjukkan fluktuasi yang jelas dan konsisten seperti pada Gambar 2a, sedangkan sinyal respirasi terlihat tidak teratur dan inkonsisten, dilihat pada Gambar 2b. Hal ini bisa disebabkan oleh pengguna tidak menampilkan area dada dengan jelas sehingga tidak dapat ditangkap dengan baik oleh *pose landmark*.

4.3 Analisis

Berdasarkan pengujian yang dilakukan terhadap dua pengguna dengan durasi yang bervariasi (10, 30, dan 60 detik), sistem mampu mengekstraksi sinyal rPPG dan respirasi secara *real-time* dengan cukup baik dalam kondisi yang optimal. Sinyal rPPG pada ketiga durasi menunjukkan hasil yang stabil dan periodik, terutama setelah proses optimasi parameter filter menggunakan algoritma CSO. Hal ini dapat dilihat dari bentuk gelombang yang menunjukkan puncak-puncak yang terdeteksi dengan jelas serta estimasi BPM yang sesuai dengan nilai fisiologis normal.

Pada sinyal respirasi, hasil terbaik tampak saat durasi rekaman 10 dan 30 detik, dengan bentuk gelombang naik-turun yang teratur dan estimasi BR yang realistis. Akan tetapi, pada durasi 60 detik (Gambar 2b), terjadi penurunan kualitas sinyal respirasi yang ditandai oleh bentuk gelombang yang tidak konsisten. Hal ini kemungkinan besar disebabkan oleh area dada pengguna tidak terlihat sepenuhnya dalam frame video, sehingga titik bahu yang dideteksi oleh *pose landmark* tidak dapat dilacak secara optimal oleh algoritma *Optical Flow*.

Kinerja sistem sangat bergantung pada faktor-faktor seperti pencahayaan, kestabilan posisi wajah dan bahu, serta kejelasan area tubuh yang terekam kamera. Oleh karena itu, panduan pada aplikasi (melalui tombol `Help`) sangat penting agar pengguna dapat menghasilkan sinyal yang lebih bersih dan akurat. Dengan demikian, sistem bekerja dengan baik dalam menggabungkan deteksi fisiologis berbasis kamera dengan visualisasi sinyal yang informatif, serta mendemonstrasikan efektivitas pemrosesan sinyal digital dalam konteks non-kontak.

5 Kesimpulan

Proyek ini berhasil mengembangkan sebuah sistem untuk mengekstraksi sinyal rPPG dan respirasi secara *real-time* dari video webcam tanpa alat kontak. Melalui pemanfaatan *MediaPipe* untuk deteksi wajah dan bahu, algoritma POS untuk rPPG, serta *Lucas-Kanade Optical Flow* untuk respirasi, sistem mampu menampilkan grafik sinyal secara langsung dan menghitung estimasi BPM dan BR dengan cukup akurat.

Proses optimasi parameter filter menggunakan algoritma Cat Swarm Optimization terbukti meningkatkan kualitas sinyal melalui pemilihan parameter terbaik secara otomatis. Antarmuka GUI yang dikembangkan memudahkan pengguna untuk merekam, mengoptimasi, dan memvisualisasikan sinyal dengan interaktif. Hasil pengujian menunjukkan bahwa sistem bekerja efektif, tetapi tetap bergantung pada posisi tubuh dan pencahayaan saat perekaman.

References

[1] 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.

[2] S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimization," in *Lecture Notes in Computer Science (LNCS)*. Springer, 2006, vol. 4099, pp. 854–858.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981, proceedings of Imaging Understanding Workshop.