Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: Dias Morello Sembiring (120140167)
Moratua Putra Pardede (121140079)
Joanne Polama Putri Sembiring (121140128)

Tugas Ke: Project Tugas Besar

Mata Kuliah: Pengolahan Sinyal Digital (IF - 3024)

Tanggal: 24 Desember 2024

1 Deskripsi Project

Sinyal respirasi adalah data atau informasi yang diperoleh dari proses pernapasan seseorang. Sinyal ini mencerminkan pola, ritme, dan karakteristik pernapasan, yang digunakan untuk memantau fungsi respirasi atau mendeteksi gangguan pada sistem pernapasan. Sinyal rRPG (Respiratory-induced Respiratory Plethysmographic) adalah sinyal yang dihasilkan dari metode respiratory plethysmography, yang digunakan untuk memantau pergerakan respirasi berdasarkan perubahan volume atau aliran udara dalam tubuh.

2 Alat dan Bahan

Dalam pengembangan proyek ini diperlukan alat dan bahan sebagai berikut:

Perangkat Lunak:

1. Python 3.10.0
2. OpenCV
3. Visual Studio Code

Perangkat Keras:

1. Laptop dengan kamera terintegrasi

3 Langkah-langkah Pengerjaan

- Mengimpor library yang dibutuhkan:

```
import cv2  # OpenCV untuk pemrosesan gambar dan video
import numpy as np  # NumPy untuk operasi numerik
import matplotlib.pyplot as plt  # Matplotlib untuk visualisasi data
from scipy.signal import find_peaks, butter, filtfilt  # SciPy untuk pemrosesan sinyal
from scipy.fft import fft, fftfreq  # SciPy untuk analisis frekuensi
import time  # Untuk pengukuran waktu dan timing

```

- Membuat fungsi untuk mendeteksi wajah dalam frame menggunakan Haar Cascade Classifier

```
def detect_face(frame):
    """
    Args:
        frame: Frame video dari webcam dalam format BGR

    Returns:
        tuple: (x, y, w, h) koordinat dan dimensi wajah, atau None jika tidak ada wajah
    """

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + '
    haarcascade_frontalface_default.xml')

```

- Mengkonversi frame ke grayscale karena diperlukan untuk deteksi wajah

```
 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

```

- Membuat fungsi untuk mendeteksi wajah dalam frame

```
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    if len(faces) > 0:
        # Jika ada wajah terdeteksi, pilih yang terbesar (diasumsikan terdekat)
        areas = [w * h for (x, y, w, h) in faces]
        max_area_idx = np.argmax(areas)
        x, y, w, h = faces[max_area_idx]

        # Perluas area ROI sebesar 10% untuk mencakup lebih banyak area wajah
        x = max(0, x - int(0.1 * w))  # Geser ke kiri
        y = max(0, y - int(0.1 * h))  # Geser ke atas
        w = min(frame.shape[1] - x, int(1.2 * w))  # Perlebar
        h = min(frame.shape[0] - y, int(1.2 * h))  # Pertinggi

return (x, y, w, h)
    return None

def get_forehead_roi(face_roi):
    """
    Mengekstrak ROI dahi dari ROI wajah yang terdeteksi.
    Dahi dipilih karena memiliki pembuluh darah yang dekat dengan permukaan
    dan biasanya tidak tertutup rambut/makeup.

    Args:
        face_roi: tuple (x, y, w, h) koordinat wajah

    Returns:
        tuple: koordinat ROI dahi
    """
    x, y, w, h = face_roi
    forehead_h = int(h * 0.3)  # Ambil 30% bagian atas wajah sebagai dahi
    return (x, y, w, forehead_h)

def process_signal(frame, roi=None):
    """
    Memproses frame video untuk mengekstrak sinyal vital.

    Args:
        frame: Frame video dari webcam
        roi: Region of Interest (x, y, w, h)

```

```
    Returns:
        tuple: (nilai_respirasi, nilai_rppg, frame_yang_diproses)
    """

```

- Membuat salinan frame untuk menggambar visualisasi sinyal respirasi dan sinyal rRPG

```
processed_frame = frame.copy()

```

- Membuat fungsi 'if' untuk mengantisipasi jika ROI tidak ada/tidak terdeteksi, maka sistem akan secara otomatis membaca seluruh frame

```
 if roi is None:
        roi = (0, 0, frame.shape[1], frame.shape[0])

```

- Mengekstrak koordinat ROI

```
x, y, w, h = roi
    roi_frame = frame[y:y+h, x:x+w]

```

- Membuat gambar kotak ROI pada frame untuk menampilkan visualisasi

```
cv2.rectangle(processed_frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

```

- Mengekstrak nilai rata-rata RGB dari ROI

```
b = np.mean(roi_frame[:, :, 0])  # Blue channel
g = np.mean(roi_frame[:, :, 1])  # Green channel
r = np.mean(roi_frame[:, :, 2])  # Red channel

```

- Mengimplementasikan metode POS (Plane-Orthogonal-to-Skin)

```
S1 = g  # Signal 1: green channel
S2 = r  # Signal 2: red channel
alpha = S2/S1  # Rasio normalisasi

```

- Menghitung sinyal rRPG dan sinyal respirasi

```
rppg_value = S1 - alpha * S2

# Hitung sinyal respirasi dari pergerakan ROI dalam grayscale
gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
respiratory_value = np.mean(gray)

return respiratory_value, rppg_value, processed_frame

```

- Membuat filter Butterworth bandpass

```
def butter_bandpass(lowcut, highcut, fs, order=5):
    """

    Args:
        lowcut: Frekuensi cut-off bawah
        highcut: Frekuensi cut-off atas
        fs: Frekuensi sampling
        order: Orde filter

    Returns:
        tuple: Koefisien filter (b, a)
    """
    nyq = 0.5 * fs  # Frekuensi Nyquist
    low = lowcut / nyq  # Normalisasi frekuensi
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

```

- Menerapkan filter bandpass pada data sinyal

```
def apply_bandpass_filter(data, lowcut, highcut, fs, order=5):
    """

    Args:
        data: Array data input
        lowcut: Frekuensi cut-off bawah
        highcut: Frekuensi cut-off atas
        fs: Frekuensi sampling
        order: Orde filter

    Returns:
        array: Data yang telah difilter
    """
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = filtfilt(b, a, data)  # Forward-backward filtering
    return y

```

- Menghitung heart rate atau respiratory rate dari sinyal

```
def calculate_rate(signal, fs, signal_type="rppg"):
    """

    Args:
        signal: Array sinyal input
        fs: Frekuensi sampling
        signal_type: "rppg" untuk heart rate atau "respiratory" untuk respiratory rate

    Returns:
        float: Rate dalam beats/breaths per menit
    """
    if len(signal) < fs * 2:  # Cek apakah ada cukup data (minimal 2 detik)
        return 0

```

- Membuat fungsi 'if' untuk set range frekuensi berdasarkan tipe sinyal

```
if signal_type == "rppg":
    lowcut, highcut = 0.7, 4.0  # 42-240 BPM untuk heart rate
```

```
else:
    lowcut, highcut = 0.1, 0.5  # 6-30 breaths/min untuk respirasi

```

- Menerapkan filter bandpass

```
filtered_signal = apply_bandpass_filter(signal, lowcut, highcut, fs)

```

- Menghitung FFT untuk analisis frekuensi

```
yf = fft(filtered_signal)
xf = fftfreq(len(filtered_signal), 1/fs)

```

- Mencari frekuensi dominan dalam range yang valid dan mengkonversi frekuensi ke rate per menit

```
valid_range = (xf >= lowcut) & (xf <= highcut)
max_freq_idx = np.argmax(np.abs(yf[valid_range]))
dominant_freq = xf[valid_range][max_freq_idx]

rate = dominant_freq * 60

return rate

```

- Membuat fungsi utama untuk menjalankan program deteksi vital signs

```
def main():

    # Inisialisasi webcam
    cap = cv2.VideoCapture(1)  # Index 1 untuk webcam eksternal
    if not cap.isOpened():
        print("Error: Tidak dapat mengakses webcam.")
        return

    # Set parameter untuk akuisisi dan processing
    window_size = 300  # Ukuran window untuk analisis (dalam frames)
    fs = cap.get(cv2.CAP_PROP_FPS)  # Dapatkan frame rate webcam

    # Inisialisasi buffer untuk menyimpan data sinyal
    respiration_signal = []
    rppg_signal = []
    time_points = []

    # Setup visualisasi matplotlib
    plt.ion()  # Aktifkan mode interaktif
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # Inisialisasi variabel tracking
    frame_count = 0
    start_time = time.time()
    last_roi = None  # Untuk menyimpan ROI terakhir yang valid

try:
        while True:
            # Baca frame dari webcam
            ret, frame = cap.read()
            if not ret:
                break
```

```
33
34              # Resize frame untuk konsistensi
35              frame = cv2.resize(frame, (640, 480))
36
37              # Deteksi wajah
38              face_roi = detect_face(frame)
39              if face_roi is not None:
40                  last_roi = face_roi
41              elif last_roi is not None:
42                  face_roi = last_roi
43
44              # Proses frame jika wajah terdeteksi
45              if face_roi is not None:
46                  # Dapatkan ROI dahi
47                  forehead_roi = get_forehead_roi(face_roi)
48
49                  # Proses sinyal
50                  resp_val, rppg_val, processed_frame = process_signal(frame, forehead_roi)
51
52                  # Update buffer data
53                  current_time = time.time() - start_time
54                  respiration_signal.append(resp_val)
55                  rppg_signal.append(rppg_val)
56                  time_points.append(current_time)
57
58                  # Jaga ukuran buffer tetap konstan
59                  if len(respiration_signal) > window_size:
60                      respiration_signal.pop(0)
61                      rppg_signal.pop(0)
62                      time_points.pop(0)
63
64                  # Hitung dan tampilkan rate jika ada cukup data
65                  if len(respiration_signal) >= window_size:
66                      # Hitung respiratory rate dan heart rate
67                      resp_rate = calculate_rate(np.array(respiration_signal), fs, "respiratory
    ")
68                      heart_rate = calculate_rate(np.array(rppg_signal), fs, "rppg")
69
70                      # Tampilkan rate pada frame
71                      cv2.putText(processed_frame, f"Heart Rate: {heart_rate:.1f} BPM",
72                                  (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
73                      cv2.putText(processed_frame, f"Resp Rate: {resp_rate:.1f} breaths/min",
74                                  (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
75
76                  # Update plot sinyal
77                      ax1.cla()
78                      ax1.plot(time_points, respiration_signal)
79                      ax1.set_title(f'Respiratory Signal (Rate: {resp_rate:.1f} breaths/min)')
80                      ax1.set_xlabel('Time (s)')
81                      ax1.set_ylabel('Amplitude')
82
83                      ax2.cla()
84                      ax2.plot(time_points, rppg_signal)
85                      ax2.set_title(f'rPPG Signal (Heart Rate: {heart_rate:.1f} BPM)')
86                      ax2.set_xlabel('Time (s)')
87                      ax2.set_ylabel('Amplitude')
88
89                      plt.tight_layout()
90                      plt.pause(0.001)
91
92                  # Tampilkan frame yang telah diproses
93                  cv2.imshow('Face Detection & Vital Signs', processed_frame)
```

```
94
95              else:
96
97                  # Tampilkan pesan jika tidak ada wajah terdeteksi
98                  cv2.putText(frame, "No face detected", (10, 30),
99                              cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
100                 cv2.imshow('Face Detection & Vital Signs', frame)
101
```

- Membuat fungsi untuk keluar dari jendela program

```
 if cv2.waitKey(1) & 0xFF == ord('q'):
                break

            frame_count += 1

    except KeyboardInterrupt:
        print("\nProgram dihentikan oleh pengguna.")

    finally:
        # Bersihkan resources
        cap.release()
        cv2.destroyAllWindows()
        plt.ioff()
        plt.close()

if __name__ == "__main__":
    main()

```

3.1 Kode Lengkap

Berikut ini adalah kode lengkap pengembangan Sistem Pengukuran Sinyal Respirasi dan rRPG

```
# Import library yang diperlukan
import cv2  # OpenCV untuk pemrosesan gambar dan video
import numpy as np  # NumPy untuk operasi numerik
import matplotlib.pyplot as plt  # Matplotlib untuk visualisasi data
from scipy.signal import find_peaks, butter, filtfilt  # SciPy untuk pemrosesan sinyal
from scipy.fft import fft, fftfreq  # SciPy untuk analisis frekuensi
import time  # Untuk pengukuran waktu dan timing

def detect_face(frame):
    """
    Mendeteksi wajah dalam frame menggunakan Haar Cascade Classifier.

    Args:
        frame: Frame video dari webcam dalam format BGR

    Returns:
        tuple: (x, y, w, h) koordinat dan dimensi wajah, atau None jika tidak ada wajah
    """
    # Memuat Haar Cascade Classifier untuk deteksi wajah
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + '
    haarcascade_frontalface_default.xml')

    # Konversi frame ke grayscale karena diperlukan untuk deteksi wajah
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Deteksi wajah dalam frame
```

```
    # scaleFactor=1.1: seberapa banyak ukuran gambar dikurangi pada setiap skala gambar
    # minNeighbors=4: berapa banyak tetangga yang harus dimiliki setiap kandidat persegi panjang untuk mempertahankannya
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    if len(faces) > 0:
        # Jika ada wajah terdeteksi, pilih yang terbesar (diasumsikan terdekat)
        areas = [w * h for (x, y, w, h) in faces]
        max_area_idx = np.argmax(areas)
        x, y, w, h = faces[max_area_idx]

        # Perluas area ROI sebesar 10% untuk mencakup lebih banyak area wajah
        x = max(0, x - int(0.1 * w))  # Geser ke kiri
        y = max(0, y - int(0.1 * h))  # Geser ke atas
        w = min(frame.shape[1] - x, int(1.2 * w))  # Perlebar
        h = min(frame.shape[0] - y, int(1.2 * h))  # Pertinggi

        return (x, y, w, h)
    return None

def get_forehead_roi(face_roi):
    """
    Mengekstrak ROI dahi dari ROI wajah yang terdeteksi.
    Dahi dipilih karena memiliki pembuluh darah yang dekat dengan permukaan
    dan biasanya tidak tertutup rambut/makeup.

    Args:
        face_roi: tuple (x, y, w, h) koordinat wajah

    Returns:
        tuple: koordinat ROI dahi
    """
    x, y, w, h = face_roi
    forehead_h = int(h * 0.3)  # Ambil 30% bagian atas wajah sebagai dahi
    return (x, y, w, forehead_h)

def process_signal(frame, roi=None):
    """
    Memproses frame video untuk mengekstrak sinyal vital.

    Args:
        frame: Frame video dari webcam
        roi: Region of Interest (x, y, w, h)

    Returns:
        tuple: (nilai_respirasi, nilai_rppg, frame_yang_diproses)
    """
    # Buat salinan frame untuk menggambar visualisasi
    processed_frame = frame.copy()

    # Jika tidak ada ROI, gunakan seluruh frame
    if roi is None:
        roi = (0, 0, frame.shape[1], frame.shape[0])

    # Ekstrak koordinat ROI
    x, y, w, h = roi
    roi_frame = frame[y:y+h, x:x+w]

    # Gambar kotak ROI pada frame untuk visualisasi
    cv2.rectangle(processed_frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Ekstrak nilai rata-rata RGB dari ROI
```

```
    b = np.mean(roi_frame[:, :, 0])  # Blue channel
    g = np.mean(roi_frame[:, :, 1])  # Green channel
    r = np.mean(roi_frame[:, :, 2])  # Red channel

    # Implementasi metode POS (Plane-Orthogonal-to-Skin)
    # Referensi: Wang et al. "Algorithmic Principles of Remote PPG"
    S1 = g  # Signal 1: green channel
    S2 = r  # Signal 2: red channel
    alpha = S2/S1  # Rasio normalisasi

    # Hitung sinyal rPPG
    rppg_value = S1 - alpha * S2

    # Hitung sinyal respirasi dari pergerakan ROI dalam grayscale
    gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
    respiratory_value = np.mean(gray)

    return respiratory_value, rppg_value, processed_frame

def butter_bandpass(lowcut, highcut, fs, order=5):
    """
    Membuat filter Butterworth bandpass.

    Args:
        lowcut: Frekuensi cut-off bawah
        highcut: Frekuensi cut-off atas
        fs: Frekuensi sampling
        order: Orde filter

    Returns:
        tuple: Koefisien filter (b, a)
    """
    nyq = 0.5 * fs  # Frekuensi Nyquist
    low = lowcut / nyq  # Normalisasi frekuensi
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

def apply_bandpass_filter(data, lowcut, highcut, fs, order=5):
    """
    Menerapkan filter bandpass pada data sinyal.

    Args:
        data: Array data input
        lowcut: Frekuensi cut-off bawah
        highcut: Frekuensi cut-off atas
        fs: Frekuensi sampling
        order: Orde filter

    Returns:
        array: Data yang telah difilter
    """
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = filtfilt(b, a, data)  # Forward-backward filtering
    return y

def calculate_rate(signal, fs, signal_type="rppg"):
    """
    Menghitung heart rate atau respiratory rate dari sinyal.

    Args:
        signal: Array sinyal input
```

```
        fs: Frekuensi sampling
        signal_type: "rppg" untuk heart rate atau "respiratory" untuk respiratory rate

    Returns:
        float: Rate dalam beats/breaths per menit
    """
    if len(signal) < fs * 2:  # Cek apakah ada cukup data (minimal 2 detik)
        return 0

    # Set range frekuensi berdasarkan tipe sinyal
    if signal_type == "rppg":
        lowcut, highcut = 0.7, 4.0  # 42-240 BPM untuk heart rate
    else:
        lowcut, highcut = 0.1, 0.5  # 6-30 breaths/min untuk respirasi

    # Terapkan filter bandpass
    filtered_signal = apply_bandpass_filter(signal, lowcut, highcut, fs)

    # Hitung FFT untuk analisis frekuensi
    yf = fft(filtered_signal)
    xf = fftfreq(len(filtered_signal), 1/fs)

    # Cari frekuensi dominan dalam range yang valid
    valid_range = (xf >= lowcut) & (xf <= highcut)
    max_freq_idx = np.argmax(np.abs(yf[valid_range]))
    dominant_freq = xf[valid_range][max_freq_idx]

    # Konversi frekuensi ke rate per menit
    rate = dominant_freq * 60

    return rate

def main():
    """
    Fungsi utama untuk menjalankan program deteksi vital signs.
    """
    # Inisialisasi webcam
    cap = cv2.VideoCapture(1)  # Index 1 untuk webcam eksternal
    if not cap.isOpened():
        print("Error: Tidak dapat mengakses webcam.")
        return

    # Set parameter untuk akuisisi dan processing
    window_size = 300  # Ukuran window untuk analisis (dalam frames)
    fs = cap.get(cv2.CAP_PROP_FPS)  # Dapatkan frame rate webcam

    # Inisialisasi buffer untuk menyimpan data sinyal
    respiration_signal = []
    rppg_signal = []
    time_points = []

    # Setup visualisasi matplotlib
    plt.ion()  # Aktifkan mode interaktif
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # Inisialisasi variabel tracking
    frame_count = 0
    start_time = time.time()
    last_roi = None  # Untuk menyimpan ROI terakhir yang valid

    try:
        while True:
```

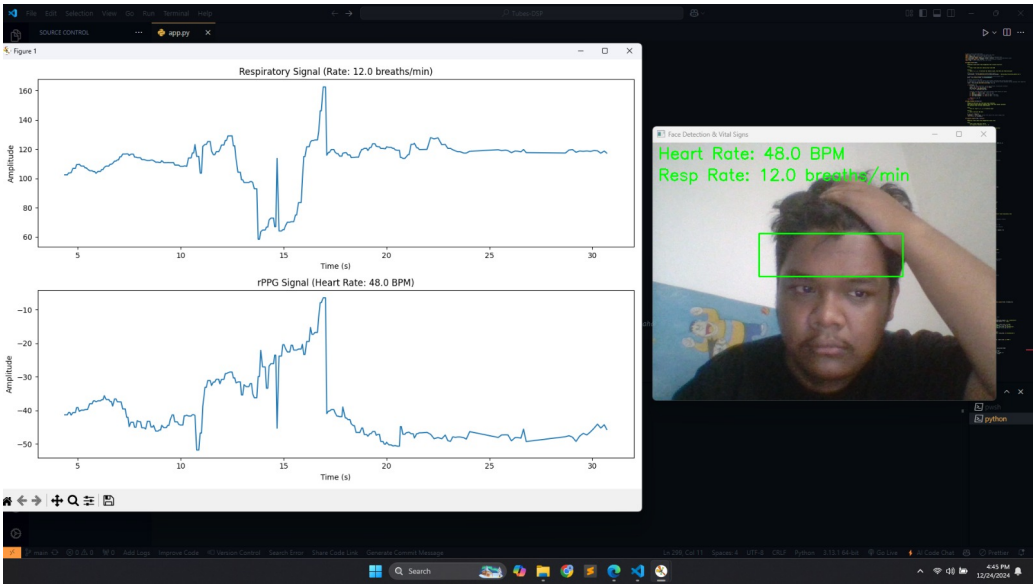
```
            # Baca frame dari webcam
            ret, frame = cap.read()
            if not ret:
                break

            # Resize frame untuk konsistensi
            frame = cv2.resize(frame, (640, 480))

            # Deteksi wajah
            face_roi = detect_face(frame)
            if face_roi is not None:
                last_roi = face_roi
            elif last_roi is not None:
                face_roi = last_roi

            # Proses frame jika wajah terdeteksi
            if face_roi is not None:
                # Dapatkan ROI dahi
                forehead_roi = get_forehead_roi(face_roi)

                # Proses sinyal
                resp_val, rppg_val, processed_frame = process_signal(frame, forehead_roi)

                # Update buffer data
                current_time = time.time() - start_time
                respiration_signal.append(resp_val)
                rppg_signal.append(rppg_val)
                time_points.append(current_time)

                # Jaga ukuran buffer tetap konstan
                if len(respiration_signal) > window_size:
                    respiration_signal.pop(0)
                    rppg_signal.pop(0)
                    time_points.pop(0)

                # Hitung dan tampilkan rate jika ada cukup data
                if len(respiration_signal) >= window_size:
                    # Hitung respiratory rate dan heart rate
                    resp_rate = calculate_rate(np.array(respiration_signal), fs, "respiratory
    ")
                    heart_rate = calculate_rate(np.array(rppg_signal), fs, "rppg")

                    # Tampilkan rate pada frame
                    cv2.putText(processed_frame, f"Heart Rate: {heart_rate:.1f} BPM",
                                (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
                    cv2.putText(processed_frame, f"Resp Rate: {resp_rate:.1f} breaths/min",
                                (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

                    # Update plot sinyal
                    ax1.cla()
                    ax1.plot(time_points, respiration_signal)
                    ax1.set_title(f'Respiratory Signal (Rate: {resp_rate:.1f} breaths/min)')
                    ax1.set_xlabel('Time (s)')
                    ax1.set_ylabel('Amplitude')

                    ax2.cla()
                    ax2.plot(time_points, rppg_signal)
                    ax2.set_title(f'rPPG Signal (Heart Rate: {heart_rate:.1f} BPM)')
                    ax2.set_xlabel('Time (s)')
                    ax2.set_ylabel('Amplitude')

                    plt.tight_layout()
```

```
                    plt.pause(0.001)

                # Tampilkan frame yang telah diproses
                cv2.imshow('Face Detection & Vital Signs', processed_frame)
            else:
                # Tampilkan pesan jika tidak ada wajah terdeteksi
                cv2.putText(frame, "No face detected", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                cv2.imshow('Face Detection & Vital Signs', frame)

            # Cek untuk keluar dari program
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

            frame_count += 1

    except KeyboardInterrupt:
        print("\nProgram dihentikan oleh pengguna.")

    finally:
        # Bersihkan resources
        cap.release()
        cv2.destroyAllWindows()
        plt.ioff()
        plt.close()

if __name__ == "__main__":
    main()
```

Kode 1: Kode lengkap Sistem Pengukuran Sinyal Respirasi dan rRPG

4 Hasil

Berikut ini adalah hasil pengukuran sinyal respirasi dan rRPG yang sudah dibuat:



Gambar 1: Hasil Pengukuran Sinyal Respirasi dan rRPG

Kode ini mampu untuk mendeteksi detak jantung (Heart Rate) dan laju pernapasan (Respira-

tory Rate) dengan menggunakan kamera (webcam) untuk mengambil video wajah, kemudian menganalisis perubahan cahaya pada wajah (khususnya di dahi) untuk mengukur sinyal vital tersebut.

- **rPPG Signal (Remote Photoplethysmography):** Menggunakan channel red dan green dari gambar untuk menghitung perubahan cahaya yang terjadi akibat detak jantung.
- **Respiratory Signal:** Menggunakan pergerakan intensitas cahaya pada gambar grayscale untuk menghitung laju pernapasan.

5 Kesimpulan

Sistem ini mampu mendeteksi detak jantung dan laju pernapasan hanya dengan menggunakan kamera dan analisis video. Teknik yang digunakan antara lain deteksi wajah, ekstraksi sinyal dari warna dan intensitas cahaya di area wajah (terutama dahi), serta pemrosesan sinyal menggunakan filter bandpass untuk menghitung laju detak jantung dan pernapasan.

6 Daftar Pustaka

https://chatgpt.com/share/676aa919-7d84-8002-b68f-798eee03bfd5