



Nama: Dias Morello Sembiring (120140167)
Moratua Putra Pardede (121140079)
Joanne Polama Putri Sembiring (121140128)
Mata Kuliah: Pengolahan Sinyal Digital (IF - 3024)

Tugas Ke: Project Tugas Besar
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:

```
1 import cv2 # OpenCV untuk pemrosesan gambar dan video
2 import numpy as np # NumPy untuk operasi numerik
3 import matplotlib.pyplot as plt # Matplotlib untuk visualisasi data
4 from scipy.signal import find_peaks, butter, filtfilt # SciPy untuk pemrosesan sinyal
5 from scipy.fft import fft, fftfreq # SciPy untuk analisis frekuensi
6 import time # Untuk pengukuran waktu dan timing
7
```

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

```
1 def detect_face(frame):
2     """
3     Args:
4         frame: Frame video dari webcam dalam format BGR
5
6     Returns:
7         tuple: (x, y, w, h) koordinat dan dimensi wajah, atau None jika tidak ada wajah
8     """
9
10    face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + '
11    haarcascade_frontalface_default.xml')
```

- Mengkonversi frame ke grayscale karena diperlukan untuk deteksi wajah

```
1 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
2
```

- Membuat fungsi untuk mendeteksi wajah dalam frame

```
1 faces = face_cascade.detectMultiScale(gray, 1.1, 4)
2
3 if len(faces) > 0:
4     # Jika ada wajah terdeteksi, pilih yang terbesar (diasumsikan terdekat)
5     areas = [w * h for (x, y, w, h) in faces]
6     max_area_idx = np.argmax(areas)
7     x, y, w, h = faces[max_area_idx]
8
9     # Perluas area ROI sebesar 10% untuk mencakup lebih banyak area wajah
10    x = max(0, x - int(0.1 * w)) # Geser ke kiri
11    y = max(0, y - int(0.1 * h)) # Geser ke atas
12    w = min(frame.shape[1] - x, int(1.2 * w)) # Perlebar
13    h = min(frame.shape[0] - y, int(1.2 * h)) # Perteinggi
14
15    return (x, y, w, h)
16    return None
17
18 def get_forehead_roi(face_roi):
19     """
20     Mengekstrak ROI dahi dari ROI wajah yang terdeteksi.
21     Dahi dipilih karena memiliki pembuluh darah yang dekat dengan permukaan
22     dan biasanya tidak tertutup rambut/makeup.
23
24     Args:
25         face_roi: tuple (x, y, w, h) koordinat wajah
26
27     Returns:
28         tuple: koordinat ROI dahi
29     """
30    x, y, w, h = face_roi
31    forehead_h = int(h * 0.3) # Ambil 30% bagian atas wajah sebagai dahi
32    return (x, y, w, forehead_h)
33
34 def process_signal(frame, roi=None):
35     """
36     Memproses frame video untuk mengekstrak sinyal vital.
37
38     Args:
39         frame: Frame video dari webcam
40         roi: Region of Interest (x, y, w, h)
41
```

```
42 Returns:
43     tuple: (nilai_respirasi, nilai_rppg, frame_yang_diproses)
44     """
45
```

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

```
1 processed_frame = frame.copy()
2
```

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

```
1 if roi is None:
2     roi = (0, 0, frame.shape[1], frame.shape[0])
3
```

- Mengekstrak koordinat ROI

```
1 x, y, w, h = roi
2     roi_frame = frame[y:y+h, x:x+w]
3
```

- Membuat gambar kotak ROI pada frame untuk menampilkan visualisasi

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

- Mengekstrak nilai rata-rata RGB dari ROI

```
1 b = np.mean(roi_frame[:, :, 0]) # Blue channel
2 g = np.mean(roi_frame[:, :, 1]) # Green channel
3 r = np.mean(roi_frame[:, :, 2]) # Red channel
4
```

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

```
1 S1 = g # Signal 1: green channel
2 S2 = r # Signal 2: red channel
3 alpha = S2/S1 # Rasio normalisasi
4
```

- Menghitung sinyal rRPG dan sinyal respirasi

```
1 rppg_value = S1 - alpha * S2
2
3 # Hitung sinyal respirasi dari pergerakan ROI dalam grayscale
4 gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
5 respiratory_value = np.mean(gray)
6
7 return respiratory_value, rppg_value, processed_frame
8
```

- Membuat filter Butterworth bandpass

```
1 def butter_bandpass(lowcut, highcut, fs, order=5):
2     """
3
4     Args:
5         lowcut: Frekuensi cut-off bawah
6         highcut: Frekuensi cut-off atas
7         fs: Frekuensi sampling
8         order: Orde filter
9
10    Returns:
11        tuple: Koefisien filter (b, a)
12    """
13    nyq = 0.5 * fs # Frekuensi Nyquist
14    low = lowcut / nyq # Normalisasi frekuensi
15    high = highcut / nyq
16    b, a = butter(order, [low, high], btype='band')
17    return b, a
18
```

- Menerapkan filter bandpass pada data sinyal

```
1 def apply_bandpass_filter(data, lowcut, highcut, fs, order=5):
2     """
3
4     Args:
5         data: Array data input
6         lowcut: Frekuensi cut-off bawah
7         highcut: Frekuensi cut-off atas
8         fs: Frekuensi sampling
9         order: Orde filter
10
11    Returns:
12        array: Data yang telah difilter
13    """
14    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
15    y = filtfilt(b, a, data) # Forward-backward filtering
16    return y
17
```

- Menghitung heart rate atau respiratory rate dari sinyal

```
1 def calculate_rate(signal, fs, signal_type="rppg"):
2     """
3
4     Args:
5         signal: Array sinyal input
6         fs: Frekuensi sampling
7         signal_type: "rppg" untuk heart rate atau "respiratory" untuk respiratory rate
8
9     Returns:
10        float: Rate dalam beats/breaths per menit
11    """
12    if len(signal) < fs * 2: # Cek apakah ada cukup data (minimal 2 detik)
13        return 0
14
```

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

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

```
3 else:
4     lowcut, highcut = 0.1, 0.5 # 6-30 breaths/min untuk respirasi
5
```

- Menerapkan filter bandpass

```
1 filtered_signal = apply_bandpass_filter(signal, lowcut, highcut, fs)
2
```

- Menghitung FFT untuk analisis frekuensi

```
1 yf = fft(filtered_signal)
2 xf = fftfreq(len(filtered_signal), 1/fs)
3
```

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

```
1 valid_range = (xf >= lowcut) & (xf <= highcut)
2 max_freq_idx = np.argmax(np.abs(yf[valid_range]))
3 dominant_freq = xf[valid_range][max_freq_idx]
4
5 rate = dominant_freq * 60
6
7 return rate
8
```

- Membuat fungsi utama untuk menjalankan program deteksi vital signs

```
1 def main():
2
3     # Inisialisasi webcam
4     cap = cv2.VideoCapture(1) # Index 1 untuk webcam eksternal
5     if not cap.isOpened():
6         print("Error: Tidak dapat mengakses webcam.")
7         return
8
9     # Set parameter untuk akuisisi dan processing
10    window_size = 300 # Ukuran window untuk analisis (dalam frames)
11    fs = cap.get(cv2.CAP_PROP_FPS) # Dapatkan frame rate webcam
12
13    # Inisialisasi buffer untuk menyimpan data sinyal
14    respiration_signal = []
15    rppg_signal = []
16    time_points = []
17
18    # Setup visualisasi matplotlib
19    plt.ion() # Aktifkan mode interaktif
20    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
21
22    # Inisialisasi variabel tracking
23    frame_count = 0
24    start_time = time.time()
25    last_roi = None # Untuk menyimpan ROI terakhir yang valid
26
27    try:
28        while True:
29            # Baca frame dari webcam
30            ret, frame = cap.read()
31            if not ret:
32                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         else:
95
96             # Tampilkan pesan jika tidak ada wajah terdeteksi
97             cv2.putText(frame, "No face detected", (10, 30),
98                         cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
99             cv2.imshow('Face Detection & Vital Signs', frame)
100
101
```

- Membuat fungsi untuk keluar dari jendela program

```
1  if cv2.waitKey(1) & 0xFF == ord('q'):
2      break
3
4      frame_count += 1
5
6  except KeyboardInterrupt:
7      print("\nProgram dihentikan oleh pengguna.")
8
9  finally:
10     # Bersihkan resources
11     cap.release()
12     cv2.destroyAllWindows()
13     plt.ioff()
14     plt.close()
15
16 if __name__ == "__main__":
17     main()
18
```

3.1 Kode Lengkap

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

```
1  # Import library yang diperlukan
2  import cv2 # OpenCV untuk pemrosesan gambar dan video
3  import numpy as np # NumPy untuk operasi numerik
4  import matplotlib.pyplot as plt # Matplotlib untuk visualisasi data
5  from scipy.signal import find_peaks, butter, filtfilt # SciPy untuk pemrosesan sinyal
6  from scipy.fft import fft, fftfreq # SciPy untuk analisis frekuensi
7  import time # Untuk pengukuran waktu dan timing
8
9  def detect_face(frame):
10     """
11     Mendeteksi wajah dalam frame menggunakan Haar Cascade Classifier.
12
13     Args:
14         frame: Frame video dari webcam dalam format BGR
15
16     Returns:
17         tuple: (x, y, w, h) koordinat dan dimensi wajah, atau None jika tidak ada wajah
18     """
19     # Memuat Haar Cascade Classifier untuk deteksi wajah
20     face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + '
21     haarcascade_frontalface_default.xml')
22
23     # Konversi frame ke grayscale karena diperlukan untuk deteksi wajah
24     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
25
26     # Deteksi wajah dalam frame
```

```
26 # scaleFactor=1.1: seberapa banyak ukuran gambar dikurangi pada setiap skala gambar
27 # minNeighbors=4: berapa banyak tetangga yang harus dimiliki setiap kandidat persegi
   panjang untuk mempertahankannya
28 faces = face_cascade.detectMultiScale(gray, 1.1, 4)
29
30 if len(faces) > 0:
31     # Jika ada wajah terdeteksi, pilih yang terbesar (diasumsikan terdekat)
32     areas = [w * h for (x, y, w, h) in faces]
33     max_area_idx = np.argmax(areas)
34     x, y, w, h = faces[max_area_idx]
35
36     # Perluas area ROI sebesar 10% untuk mencakup lebih banyak area wajah
37     x = max(0, x - int(0.1 * w)) # Geser ke kiri
38     y = max(0, y - int(0.1 * h)) # Geser ke atas
39     w = min(frame.shape[1] - x, int(1.2 * w)) # Perlebar
40     h = min(frame.shape[0] - y, int(1.2 * h)) # Perteinggi
41
42     return (x, y, w, h)
43 return None
44
45 def get_forehead_roi(face_roi):
46     """
47     Mengekstrak ROI dahi dari ROI wajah yang terdeteksi.
48     Dahi dipilih karena memiliki pembuluh darah yang dekat dengan permukaan
49     dan biasanya tidak tertutup rambut/makeup.
50
51     Args:
52         face_roi: tuple (x, y, w, h) koordinat wajah
53
54     Returns:
55         tuple: koordinat ROI dahi
56     """
57     x, y, w, h = face_roi
58     forehead_h = int(h * 0.3) # Ambil 30% bagian atas wajah sebagai dahi
59     return (x, y, w, forehead_h)
60
61 def process_signal(frame, roi=None):
62     """
63     Memproses frame video untuk mengekstrak sinyal vital.
64
65     Args:
66         frame: Frame video dari webcam
67         roi: Region of Interest (x, y, w, h)
68
69     Returns:
70         tuple: (nilai_respirasi, nilai_rppg, frame_yang_diproses)
71     """
72     # Buat salinan frame untuk menggambar visualisasi
73     processed_frame = frame.copy()
74
75     # Jika tidak ada ROI, gunakan seluruh frame
76     if roi is None:
77         roi = (0, 0, frame.shape[1], frame.shape[0])
78
79     # Ekstrak koordinat ROI
80     x, y, w, h = roi
81     roi_frame = frame[y:y+h, x:x+w]
82
83     # Gambar kotak ROI pada frame untuk visualisasi
84     cv2.rectangle(processed_frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
85
86     # Ekstrak nilai rata-rata RGB dari ROI
```



```
87     b = np.mean(roi_frame[:, :, 0]) # Blue channel
88     g = np.mean(roi_frame[:, :, 1]) # Green channel
89     r = np.mean(roi_frame[:, :, 2]) # Red channel
90
91     # Implementasi metode POS (Plane-Orthogonal-to-Skin)
92     # Referensi: Wang et al. "Algorithmic Principles of Remote PPG"
93     S1 = g # Signal 1: green channel
94     S2 = r # Signal 2: red channel
95     alpha = S2/S1 # Rasio normalisasi
96
97     # Hitung sinyal rPPG
98     rppg_value = S1 - alpha * S2
99
100    # Hitung sinyal respirasi dari pergerakan ROI dalam grayscale
101    gray = cv2.cvtColor(roi_frame, cv2.COLOR_BGR2GRAY)
102    respiratory_value = np.mean(gray)
103
104    return respiratory_value, rppg_value, processed_frame
105
106 def butter_bandpass(lowcut, highcut, fs, order=5):
107     """
108     Membuat filter Butterworth bandpass.
109
110     Args:
111         lowcut: Frekuensi cut-off bawah
112         highcut: Frekuensi cut-off atas
113         fs: Frekuensi sampling
114         order: Orde filter
115
116     Returns:
117         tuple: Koefisien filter (b, a)
118     """
119     nyq = 0.5 * fs # Frekuensi Nyquist
120     low = lowcut / nyq # Normalisasi frekuensi
121     high = highcut / nyq
122     b, a = butter(order, [low, high], btype='band')
123     return b, a
124
125 def apply_bandpass_filter(data, lowcut, highcut, fs, order=5):
126     """
127     Menerapkan filter bandpass pada data sinyal.
128
129     Args:
130         data: Array data input
131         lowcut: Frekuensi cut-off bawah
132         highcut: Frekuensi cut-off atas
133         fs: Frekuensi sampling
134         order: Orde filter
135
136     Returns:
137         array: Data yang telah difilter
138     """
139     b, a = butter_bandpass(lowcut, highcut, fs, order=order)
140     y = filtfilt(b, a, data) # Forward-backward filtering
141     return y
142
143 def calculate_rate(signal, fs, signal_type="rppg"):
144     """
145     Menghitung heart rate atau respiratory rate dari sinyal.
146
147     Args:
148         signal: Array sinyal input
```

```
149         fs: Frekuensi sampling
150         signal_type: "rppg" untuk heart rate atau "respiratory" untuk respiratory rate
151
152     Returns:
153         float: Rate dalam beats/breaths per menit
154     """
155     if len(signal) < fs * 2: # Cek apakah ada cukup data (minimal 2 detik)
156         return 0
157
158     # Set range frekuensi berdasarkan tipe sinyal
159     if signal_type == "rppg":
160         lowcut, highcut = 0.7, 4.0 # 42-240 BPM untuk heart rate
161     else:
162         lowcut, highcut = 0.1, 0.5 # 6-30 breaths/min untuk respirasi
163
164     # Terapkan filter bandpass
165     filtered_signal = apply_bandpass_filter(signal, lowcut, highcut, fs)
166
167     # Hitung FFT untuk analisis frekuensi
168     yf = fft(filtered_signal)
169     xf = fftfreq(len(filtered_signal), 1/fs)
170
171     # Cari frekuensi dominan dalam range yang valid
172     valid_range = (xf >= lowcut) & (xf <= highcut)
173     max_freq_idx = np.argmax(np.abs(yf[valid_range]))
174     dominant_freq = xf[valid_range][max_freq_idx]
175
176     # Konversi frekuensi ke rate per menit
177     rate = dominant_freq * 60
178
179     return rate
180
181 def main():
182     """
183     Fungsi utama untuk menjalankan program deteksi vital signs.
184     """
185     # Inisialisasi webcam
186     cap = cv2.VideoCapture(1) # Index 1 untuk webcam eksternal
187     if not cap.isOpened():
188         print("Error: Tidak dapat mengakses webcam.")
189         return
190
191     # Set parameter untuk akuisisi dan processing
192     window_size = 300 # Ukuran window untuk analisis (dalam frames)
193     fs = cap.get(cv2.CAP_PROP_FPS) # Dapatkan frame rate webcam
194
195     # Inisialisasi buffer untuk menyimpan data sinyal
196     respiration_signal = []
197     rppg_signal = []
198     time_points = []
199
200     # Setup visualisasi matplotlib
201     plt.ion() # Aktifkan mode interaktif
202     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
203
204     # Inisialisasi variabel tracking
205     frame_count = 0
206     start_time = time.time()
207     last_roi = None # Untuk menyimpan ROI terakhir yang valid
208
209     try:
210         while True:
```

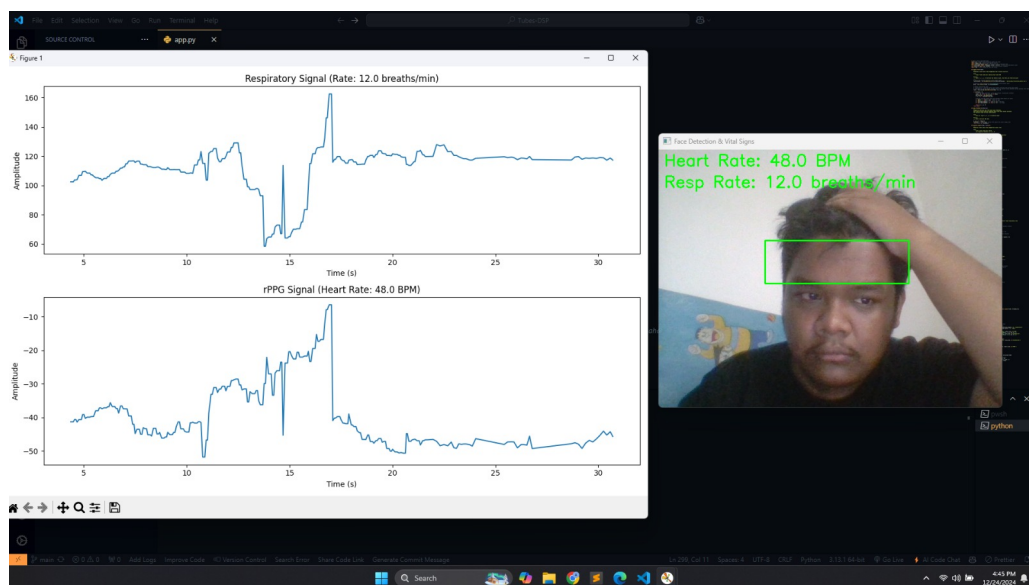
```
211     # Baca frame dari webcam
212     ret, frame = cap.read()
213     if not ret:
214         break
215
216     # Resize frame untuk konsistensi
217     frame = cv2.resize(frame, (640, 480))
218
219     # Deteksi wajah
220     face_roi = detect_face(frame)
221     if face_roi is not None:
222         last_roi = face_roi
223     elif last_roi is not None:
224         face_roi = last_roi
225
226     # Proses frame jika wajah terdeteksi
227     if face_roi is not None:
228         # Dapatkan ROI dahi
229         forehead_roi = get_forehead_roi(face_roi)
230
231         # Proses sinyal
232         resp_val, rppg_val, processed_frame = process_signal(frame, forehead_roi)
233
234         # Update buffer data
235         current_time = time.time() - start_time
236         respiration_signal.append(resp_val)
237         rppg_signal.append(rppg_val)
238         time_points.append(current_time)
239
240         # Jaga ukuran buffer tetap konstan
241         if len(respiration_signal) > window_size:
242             respiration_signal.pop(0)
243             rppg_signal.pop(0)
244             time_points.pop(0)
245
246         # Hitung dan tampilkan rate jika ada cukup data
247         if len(respiration_signal) >= window_size:
248             # Hitung respiratory rate dan heart rate
249             resp_rate = calculate_rate(np.array(respiration_signal), fs, "respiratory
250
251             heart_rate = calculate_rate(np.array(rppg_signal), fs, "rppg")
252
253         # Tampilkan rate pada frame
254         cv2.putText(processed_frame, f"Heart Rate: {heart_rate:.1f} BPM",
255                     (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
256         cv2.putText(processed_frame, f"Resp Rate: {resp_rate:.1f} breaths/min",
257                     (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
258
259         # Update plot sinyal
260         ax1.cla()
261         ax1.plot(time_points, respiration_signal)
262         ax1.set_title(f'Respiratory Signal (Rate: {resp_rate:.1f} breaths/min)')
263         ax1.set_xlabel('Time (s)')
264         ax1.set_ylabel('Amplitude')
265
266         ax2.cla()
267         ax2.plot(time_points, rppg_signal)
268         ax2.set_title(f'rPPG Signal (Heart Rate: {heart_rate:.1f} BPM)')
269         ax2.set_xlabel('Time (s)')
270         ax2.set_ylabel('Amplitude')
271
272     plt.tight_layout()
```

```
272         plt.pause(0.001)
273
274         # Tampilkan frame yang telah diproses
275         cv2.imshow('Face Detection & Vital Signs', processed_frame)
276     else:
277         # Tampilkan pesan jika tidak ada wajah terdeteksi
278         cv2.putText(frame, "No face detected", (10, 30),
279                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
280         cv2.imshow('Face Detection & Vital Signs', frame)
281
282         # Cek untuk keluar dari program
283         if cv2.waitKey(1) & 0xFF == ord('q'):
284             break
285
286         frame_count += 1
287
288 except KeyboardInterrupt:
289     print("\nProgram dihentikan oleh pengguna.")
290
291 finally:
292     # Bersihkan resources
293     cap.release()
294     cv2.destroyAllWindows()
295     plt.ioff()
296     plt.close()
297
298 if __name__ == "__main__":
299     main()
300
```

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>