Program Studi Teknik Informatika
Institut Teknologi Sumatera

Mata Kuliah: Pengolahan Sinyal Digital (IF3024) Tugas: Tugas Besar

Nama Anggota:

1. A Kevin Sergian (121140125)

Program Pendeteksi Sinyal Resipirasi dan Sinyal rPPG

1 Pendahuluan

1.1 Latar Belakang

Proyek ini merupakan tugas akhir dari mata kuliah "Pengolahan Sinyal Digital IF(3024)" yang dapat digunakan untuk memperoleh sinyal respirasi dan sinyal *remote-photopletysmography* (rPPG) dari input video web-cam secara *real-time*.

Program ini memperoleh sinyal respirasi dengan menggunakan *pose-landmarker* dari MediaPipe untuk menghitung pergerakan bahu saat pengguna melakukan pernapasan. Sedangkan untuk sinyal rPPG, program ini menggunakan *face-detector* dari MediaPipe dan algoritma *Plane Orthogonal-toSkin* (POS) untuk menghitung detak jantung secara non-kontak dengan menganalisis perubahan warna pada wajah pengguna[1].

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 dukungan *library* yang umum digunakan untuk proyek komputer visi.

2.2 *Library*

Berikut merupakan *library* yang digunakan pada pengembangan proyek ini:

1. OpenCV: Digunakan untuk memproses gambar dan video.
2. MediaPipe: *face-detector* dan *pose-landmarker* dari MediaPipe digunakan untuk membuat landmark pada wajah dan bahu pengguna.

3. PyQt: Digunakan untuk membuat grafik antarmuka.
4. NumPy: Digunakan untuk pengolahan data numerik, seperti penghitungan sinyal dan transformasi data.
5. os: Digunakan untuk mengakses fungsi-fungsi sistem operasi
6. sys: Digunakan untuk mengakses variabel sistem
7. requests: Digunakan untuk mengirimkan permintaan HTTP
8. tqdm: Digunakan untuk membuat progress bar
9. platform: Digunakan untuk mengakses informasi sistem operasi dari pengguna
10. scipy: Membantu dalam analisis sinyal, termasuk ekstraksi fitur dari sinyal respirasi dan rPPG.

2.3 Metode dan Algoritma

Berikut merupakan Metode dan Algoritma yang digunakan pada pengembangan proyek ini:

1. Ekstraksi Sinyal rPPG: Menggunakan algoritma *Plane Orthogonal-to-Skin* (POS) berbasis perubahan intensitas warna pada area wajah untuk mendeteksi sinyal detak jantung [1].
2. Ekstraksi Sinyal Respirasi: Menggunakan analisis pergerakan bahu dan area wajah untuk mendapatkan pola respirasi pengguna [2].

3 Penjelasan

Program pada proyek ini dibagi menjadi 4 modul, yaitu main.py, check_gpu.py, download_model.py, dan heart_rate.py. Berikut adalah penjelasan mengenai alur kerja program yang dikembangkan pada proyek ini:

3.1 Instalasi Library

Agar dapat menjalankan program, pengguna dapat menambahkan beberapa library ke dalam python environment yang akan digunakan. Pengguna dapat mengunduh library yang digunakan menggunakan dua cara, yaitu:

3.1.1 Menggunakan requirements.txt

Kode 2: Install Library/Pustaka menggunakan requirements.txt

3.2 Modul main.py

3.2.1 Import Library

Library yang digunakan pada program ini adalah sebagai berikut: sys, NumPy, OpenCV, os, subprocess, requests, tqdm, MediaPipe, PyQt, PyQtgraph, platform, SciPy. Selain itu, fungsi cpu_POS.py, download_model_face_detection, download_model_pose_detection, dan check_gpu dipanggil sebagai modul dari folder utils.

```
3   import sys
4   import numpy as np
5   import cv2
6   import mediapipe as mp
7   from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QGridLayout
8   from PyQt5.QtCore import QTimer, Qt
9   from PyQt5.QtGui import QImage, QPixmap
10  import pyqtgraph as pg
11  from scipy.signal import find_peaks
12  from utils.download_model import download_model_face_detection, download_model_pose_detection
13  from utils.check_gpu import check_gpu
14  from utils.heart_rate import cpu_POS, bandpass_filter_signal, moving_average_filter, get_initial_roi
```

Kode 3: Import Library/Pustaka

3.2.2 Kelas HeartRateMonitor

Berikut merupakan penjelasan dari kelas HeartRateMonitor:

```
20  class HeartRateMonitor(QWidget):
```

Kode 4: Kelas HeartRateMonitor

Kelas HeartRateMonitor berfungsi untuk menampilkan *Graphic User Interface* (GUI) dan menghitung detak jantung dan pernapasan secara real-time. Selain itu, Kelas ini akan menyimpan nilai sinyal rppg dan nilai sinyal pernafasan dari pose detection. Berikut merupakan penjabaran dari kelas HeartRateMonitor:

```python
    def __init__(self):
        """
        Konstruktor kelas HeartRateMonitor.
        Menginisialisasi GUI, kamera, detektor MediaPipe, dan properti sinyal/plot.
        """
        super().__init__()
        self.initUI()

        # Platform Specific Camera Backend
        video_backend = cv2.CAP_DSHOW if sys.platform == 'win32' else cv2.CAP_AVFOUNDATION
        self.cap = cv2.VideoCapture(0, video_backend)
        if not self.cap.isOpened():
            print("Error: Could not open video stream. Please check webcam.")
            sys.exit(1)

        self.fps = self.cap.get(cv2.CAP_PROP_FPS)
        if self.fps == 0:
            print("Warning: FPS is 0, setting to default 30.")
            self.fps = 30

        # Properties for Storing values
        self.r_signal, self.g_signal, self.b_signal = [], [], []
        self.resp_signal = []

        # Initialize MediaPipe detectors
        self.face_detector = self.initialize_face_detector()
        self.pose_landmarker = self.initialize_pose_landmarker()

        # Inisialisasi properti untuk ROI pernapasan berbasis landmark
        self.resp_roi_center_y_history = []
        self.last_pose_landmarks = None
            self.left_x_resp = None
            self.top_y_resp = None
            self.right_x_resp = None
            self.bottom_y_resp = None

            # Setup QTimer for frame updates
            self.timer = QTimer()
            self.timer.timeout.connect(self.update_frame)
            self.timer.start(1000 // int(self.fps))
```

Kode 5: Metode Konstruktor dari Kelas HeartRateMonitor

Metode ini berguna sebagai konstruktor dari kelas HeartRateMonitor untuk menangani, antara lain:

1. Proses inisialiasi antarmuka
2. Pengaturan *backend* kamera (supaya kompatibel dengan sistem operasi pengguna)
3. Inisialisasi variabel penyimpanan
4. Inisialiasi model detector serta landmark dari MediaPipe
5. Pengaturan timer

```python
    def initUI(self):
        """
        Metode untuk mempersiapkan antarmuka pengguna grafis (GUI) menggunakan PyQt5.
        Menyiapkan layout, label video, label hasil HR/RR, dan widget plot.
        """
        self.setWindowTitle('Real-Time Heart Rate and Respiration Monitor')
        self.setGeometry(100, 100, 1200, 800)

        # Prepare GUI elements
        self.video_label = QLabel(self)
        self.hr_label = QLabel('Heart Rate: -- BPM (Beat Per Minute)', self)
        self.hr_label.setAlignment(Qt.AlignCenter)
        self.resp_label = QLabel('Respiration Rate: -- BPM (Breath Per Minute)', self)
        self.resp_label.setAlignment(Qt.AlignCenter)

        # Heart Rate Plot
        self.plot_widget_hr = pg.PlotWidget()
        self.plot_widget_hr.setYRange(-3, 3)
        self.plot_widget_hr.setTitle("Heart Rate Signal (rPPG)")
        self.plot_widget_hr.setLabel('left', 'Amplitude')
        self.plot_widget_hr.setLabel('bottom', 'Samples')
        self.plot_curve_hr = self.plot_widget_hr.plot(pen='r')

        # Respiration Rate Plot
        self.plot_widget_resp = pg.PlotWidget()
        self.plot_widget_resp.setYRange(-3, 3)
        self.plot_widget_resp.setTitle("Respiration Signal")
        self.plot_widget_resp.setLabel('left', 'Amplitude')
        self.plot_widget_resp.setLabel('bottom', 'Samples')
        self.plot_curve_resp = self.plot_widget_resp.plot(pen='b')
```

```python
        # Making Layout instance and insert the plot into the layout
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.hr_label)
        left_layout.addWidget(self.plot_widget_hr)
        left_layout.addWidget(self.resp_label)
        left_layout.addWidget(self.plot_widget_resp)

        right_layout = QVBoxLayout()
        right_layout.addWidget(self.video_label)

        main_layout = QGridLayout()
        main_layout.addLayout(left_layout, 0, 0)
        main_layout.addLayout(right_layout, 0, 1)
        main_layout.setColumnStretch(0, 1)
        main_layout.setColumnStretch(1, 2)

        self.setLayout(main_layout)
```

Kode 6: Metode initUI

Metode di atas ini digunakan untuk mempersiapkan kelas PyQT sebagai GUI untuk overlay dari feed live time video dan sinyal (rppg + resp) karena kelas merupakan anak dari QWidget.

```python
def initialize_face_detector(self):
    """
    Menginisialisasi objek Face Detector dari MediaPipe untuk proses rPPG.
    Model akan diunduh jika belum ada.

    Returns:
        mediapipe.tasks.vision.FaceDetector: Objek FaceDetector yang sudah terinisialisasi.
    """
    model_path = download_model_face_detection()
    BaseOptions = mp.tasks.BaseOptions
    gpu_checked = check_gpu()

    if sys.platform == 'win32':
        delegate = python.BaseOptions.Delegate.CPU
    else:
        delegate = python.BaseOptions.Delegate.GPU if gpu_checked == "NVIDIA" else python.BaseOptions.Delegate.CPU

    options = vision.FaceDetectorOptions(
        base_options=BaseOptions(
            model_asset_path=model_path,
            delegate=delegate
        )
    )
    return vision.FaceDetector.create_from_options(options)
```

Kode 7: Metode initialize_face_detector

Metode di atas ini digunakan untuk menginisialisasi fungsi face_detector dari mediapipe untuk proses RPPG.

```python
def initialize_pose_landmarker(self):
    """
    Menginisialisasi objek Pose Landmarker dari MediaPipe untuk proses ekstraksi
    sinyal respirasi. Model akan diunduh jika belum ada.

    Returns:
        mediapipe.tasks.vision.PoseLandmarker: Objek PoseLandmarker yang sudah terinisialisasi.
    """
    model_path = download_model_pose_detection()
    BaseOptions = mp.tasks.BaseOptions
    PoseLandmarkerOptions = mp.tasks.vision.PoseLandmarkerOptions
    VisionRunningMode = mp.tasks.vision.RunningMode
    gpu_checked = check_gpu()

    if sys.platform == 'win32':
        delegate = BaseOptions.Delegate.CPU
    else:
        delegate = BaseOptions.Delegate.GPU if gpu_checked == "NVIDIA" else BaseOptions.Delegate.CPU

    options_image = PoseLandmarkerOptions(
        base_options=BaseOptions(
            model_asset_path=model_path,
            delegate = delegate
        ),
        running_mode=VisionRunningMode.IMAGE,
        num_poses=1,
        min_pose_detection_confidence=0.5,
        min_pose_presence_confidence=0.5,
        min_tracking_confidence=0.5,
        output_segmentation_masks=False
    )

    return mp.tasks.vision.PoseLandmarker.create_from_options(options_image)
```

Kode 8: Metode initialize_pose_landmarker

Metode di atas ini digunakan untuk menginisialisasi fungsi pose_detection dari mediapipe untuk proses pendeteksian sinyal respirasi pengguna.

```
175     def update_frame(self):
176         """
177         Metode utama yang dipanggil secara periodik oleh QTimer untuk memproses setiap frame.
178         Melakukan:
179         1. Pembacaan frame dari webcam.
180         2. Deteksi wajah dan ekstraksi ROI dahi untuk sinyal rPPG.
181         3. Deteksi pose dan ekstraksi sinyal respirasi berbasis landmark.
182         4. Pemrosesan sinyal (filtering, normalisasi, smoothing).
183         5. Perhitungan detak jantung dan laju pernapasan.
184         6. Pembaruan tampilan GUI (video feed, plot sinyal, label hasil).
185         """
186         ret, frame = self.cap.read()
187         if not ret:
188             print("Failed to grab frame.")
189             self.timer.stop()
190             return
191 
192         h, w, _ = frame.shape
193 
194         # --- rPPG Signal Extraction (Forehead ROI) ---
195         frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
196         mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame_rgb)
197         detection_result_face = self.face_detector.detect(mp_image)
198 
199         if detection_result_face.detections:
200             detection = detection_result_face.detections[0]
201             bbox = detection.bounding_box
202 
203             # --- PENYESUAIAN KOTAK HIJAU UNTUK RPPG: KIRA-KIRA SEBESAR MUKA ---
204             # Menggunakan lebar penuh wajah dan bagian atas wajah untuk ROI dahi/rPPG
205             forehead_x = int(bbox.origin_x)
206             forehead_y = int(bbox.origin_y)
207             forehead_width = int(bbox.width)
208             forehead_height = int(bbox.height * 0.4)
```

```
            forehead_x = max(0, forehead_x)
            forehead_y = max(0, forehead_y)
            forehead_width = min(forehead_width, w - forehead_x)
            forehead_height = min(forehead_height, h - forehead_y)

            if forehead_width > 0 and forehead_height > 0:
                roi_forehead = frame[forehead_y : forehead_y + forehead_height,
                                     forehead_x : forehead_x + forehead_width]

                if roi_forehead.size > 0:
                    self.r_signal.append(np.mean(roi_forehead[:, :, 2]))
                    self.g_signal.append(np.mean(roi_forehead[:, :, 1]))
                    self.b_signal.append(np.mean(roi_forehead[:, :, 0]))

                    cv2.rectangle(frame, (forehead_x, forehead_y),
                                  (forehead_x + forehead_width, forehead_y + forehead_height),
                                  (0, 255, 0), 2) # Green rectangle for forehead

        # --- Respiration Signal Extraction (Landmark-based) ---
        detection_result_pose = self.pose_landmarker.detect(mp_image)

        if detection_result_pose.pose_landmarks:
            current_pose_landmarks = detection_result_pose.pose_landmarks[0]

            # Menggunakan indeks numerik standar untuk landmark bahu
            # LEFT_SHOULDER: 11, RIGHT_SHOULDER: 12
            left_shoulder = current_pose_landmarks[11]
            right_shoulder = current_pose_landmarks[12]

            # Konversi koordinat normalized (0-1) ke piksel
            shoulder_y_avg_px = int(((left_shoulder.y + right_shoulder.y) / 2) * h)

            # --- PENYESUAIAN SINYAL RESPIRASI: FOKUS PADA BAHU ---
            self.resp_signal.append(shoulder_y_avg_px) # Sinyal pernapasan hanya dari posisi Y bahu
```

```
        # --- Gambar Kotak Merah (ROI Pernapasan) yang Mengikuti ---
        # Lebar kotak berdasarkan jarak bahu
        shoulder_x_min_px = int(min(left_shoulder.x, right_shoulder.x) * w)
        shoulder_x_max_px = int(max(left_shoulder.x, right_shoulder.x) * w)

        box_height_resp = 20 # Tinggi kotak, dibuat lebih tipis untuk efek "garis"

        # Posisi Y kotak merah, berpusat pada rata-rata Y bahu
        self.top_y_resp = int(shoulder_y_avg_px - (box_height_resp / 2))
        self.bottom_y_resp = self.top_y_resp + box_height_resp

        # Lebar kotak, sedikit dilebihkan dari lebar bahu
        padding_x_resp = int((shoulder_x_max_px - shoulder_x_min_px) * 0.1) # 10% padding
        self.left_x_resp = max(0, shoulder_x_min_px - padding_x_resp)
        self.right_x_resp = min(w, shoulder_x_max_px + padding_x_resp)

        # Pastikan koordinat tetap dalam batas frame
        self.top_y_resp = max(0, self.top_y_resp)
        self.left_x_resp = max(0, self.left_x_resp)
        self.bottom_y_resp = min(h, self.bottom_y_resp)
        self.right_x_resp = min(w, self.right_x_resp)

        # Gambar kotak merah
        if all(v is not None for v in [self.left_x_resp, self.top_y_resp, self.right_x_resp, self.bottom_y_resp]):
            cv2.rectangle(frame, (self.left_x_resp, self.top_y_resp), (self.right_x_resp, self.bottom_y_resp), (0, 0, 255), 2) # Merah
```

```python
        # --- Signal Processing and Rate Calculation ---
        # rPPG processing
        if len(self.g_signal) >= self.fps * 10:
            rgb_signals = np.array([self.r_signal, self.g_signal, self.b_signal])
            rgb_signals_reshaped = rgb_signals.reshape(1, 3, -1)

            rppg_signal_raw = cpu_POS(rgb_signals_reshaped, fps=self.fps)
            rppg_signal = rppg_signal_raw.reshape(-1)

            filtered_signal = bandpass_filter_signal(rppg_signal, 0.75, 3.0, self.fps, order=5)
            if filtered_signal.size > 0:
                normalized_signal = (filtered_signal - np.mean(filtered_signal)) / (np.std(filtered_signal) + 1e-6)
                smoothed_signal = moving_average_filter(normalized_signal, window_size=int(self.fps/2))

                if smoothed_signal.size > 0:
                    peaks_hr, _ = find_peaks(smoothed_signal, distance=self.fps / 3.0)
                    peak_intervals_hr = np.diff(peaks_hr) / self.fps
                    heart_rate = 60.0 / np.mean(peak_intervals_hr) if len(peak_intervals_hr) > 0 else 0
                else:
                    heart_rate = 0.0

                self.hr_label.setText(f'Heart Rate: {heart_rate:.2f} BPM (Beat Per Minute)')
                self.plot_curve_hr.setData(smoothed_signal)
            else:
                self.hr_label.setText(f'Heart Rate: -- BPM (Beat Per Minute)')
                self.plot_curve_hr.setData([])

            self.r_signal, self.g_signal, self.b_signal = [], [], []
```

```python
        # Respiration processing
        if len(self.resp_signal) >= self.fps * 10:
            resp_signal_raw = np.array(self.resp_signal)

            filtered_resp_signal = bandpass_filter_signal(resp_signal_raw, 0.1, 0.5, self.fps, order=5)
            if filtered_resp_signal.size > 0:
                normalized_resp_signal = (filtered_resp_signal - np.mean(filtered_resp_signal)) / (np.std(filtered_resp_signal) + 1e-6)
                smoothed_resp_signal = moving_average_filter(normalized_resp_signal, window_size=int(self.fps/2))

                if smoothed_resp_signal.size > 0:
                    resp_peaks, _ = find_peaks(smoothed_resp_signal, distance=self.fps / 0.5)
                    resp_intervals = np.diff(resp_peaks) / self.fps
                    respiration_rate = 60.0 / np.mean(resp_intervals) if len(resp_intervals) > 0 else 0
                else:
                    respiration_rate = 0.0

                self.resp_label.setText(f'Respiration Rate: {respiration_rate:.2f} BPM (Breath Per Minute)')
                self.plot_curve_resp.setData(smoothed_resp_signal)
            else:
                self.resp_label.setText(f'Respiration Rate: -- BPM (Breath Per Minute)')
                self.plot_curve_resp.setData([])

            self.resp_signal = []

        # Convert frame to RGB for displaying in video_label
        frame_rgb_display = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = QImage(frame_rgb_display.data, frame_rgb_display.shape[1], frame_rgb_display.shape[0], QImage.Format_RGB888)
        self.video_label.setPixmap(QPixmap.fromImage(image.scaled(self.video_label.size(), Qt.KeepAspectRatio, Qt.SmoothTransformation)))
```

Kode 9: Metode update_frame

Metode di atas ini digunakan untuk mengambil frame pengguna lalu ditentukan beberapa proses seperti deteksi wajah dengan mediapipe dan pose detection, lalu menentukan bounding box untuk proses selanjutnya. Untuk sinyal RPPG ditetapkan daerah dahi sebagai ROI, dan untuk sinyal respirasi ditetapkan

daerah sekitar baru sebagai ROI. Untuk Pose Detection akan diterapkan Optical Flow untuk mengurangi kinerja beban tracking setiap frame.

```
330     def closeEvent(self, event):
331         """
332         Metode turunan dari kelas QWidget yang dipanggil ketika jendela ditutup.
333         Digunakan untuk melepaskan sumber daya kamera OpenCV.
334 
335         Args:
336             event (QCloseEvent): Objek event penutupan jendela.
337         """
338         self.cap.release()
339         cv2.destroyAllWindows()
340         print("Application closed, camera released.")
341         event.accept()
```

Kode 10: Metode closeEvent

Metode turunan dari kelas QWidget untuk melepaskan resource yang tidak dibutuhkan lagi.

3.2.4Eksekusi Program dengan Fungsi Main

Modul main.py akan dieksusi sebagai program utama pada proyek tugas besar ini.

```
343 if __name__ == '__main__':
344     import os
345     if not os.path.exists("models"):
346         os.makedirs("models")
347 
348     app = QApplication(sys.argv)
349     ex = HeartRateMonitor()
350     ex.show()
351     sys.exit(app.exec_())
```

Kode 11: Fungsi Main

Berikut merupakan penjelasan dari fungsi di atas ini:

1. app = QApplication(sys.argv) digunakan untuk menginstansiasi QApplication yang baru. Pengelolaan aliran kontrol dan pengaturan utama applikasi GUI juga diatur pada baris kode ini.
2. ex = HeartRateMonitor() digunakan untuk menginstansiasi jendela applikasi utama.
3. ex.show() digunakan untuk membuat jendela terlihat pada layar pengguna.
4. sys.exit(app.exec_()) digunakan untuk memulai perulangan utama dari program dan memastikan bahwa program dapat dihentikan sewaktu ditutup oleh pengguna.

3.3 Modul check_gpu.py

Pada Program ini, modul check_gpu.py digunakan untuk memeriksa apabila pengguna sedang menggunakan *Graphical Processing Unit* (GPU) atau MLX (Arsitektur Apple Silicon). Jika tersedia, modul akan memerintahkan program untuk menggunakan GPU atau MLX sebagai unit pemrosesan.

3.3.1 Import Library

Berikut merupakan penjelasan dari Import Library pada modul check_gpu.py.

```
2   import subprocess
3   import os
```

Kode 12: Import Library untuk check_gpu.py

Modul check_gpu.py hanya menggunakan dua pustaka yaitu: platform dan subprocess. Kedua pustaka ini berguna untuk mengakses informasi terkait sistem operasi yang digunakan pengguna.

3.3.2 Fungsi check_gpu

Berikut merupakan penjelasan dari fungsi check_gpu.

```
5   def check_gpu():
6       """
7       Checks for the presence of an NVIDIA GPU using nvidia-smi.
8
9       Returns:
10          str: "NVIDIA" if an NVIDIA GPU is detected, otherwise "CPU".
11      """
12      try:
13          # Try to run nvidia-smi command
14          # Capture stdout and stderr
15          result = subprocess.run(['nvidia-smi'], capture_output=True, text=True, check=True)
16          # If the command runs without error, it means nvidia-smi is available and thus an NVIDIA GPU is likely present
17          if "NVIDIA-SMI" in result.stdout:
18              print("NVIDIA GPU detected.")
19              return "NVIDIA"
20      except (subprocess.CalledProcessError, FileNotFoundError):
21          # If nvidia-smi command fails or is not found, assume no NVIDIA GPU
22          print("No NVIDIA GPU detected or nvidia-smi not found. Using CPU.")
23      return "CPU"
```

Kode 13: Fungsi check_gpu

Fungsi di atas ini berguna untuk memeriksa ketersediaan GPU pada sistem. Fungsi akan mengembalikan nilai *string*;

1. "NVIDIA" apabila pengguna menggunakan sistem dengan *Graphical Processing Unit* (GPU)
2. "MLX" apabila pengguna menggunakan sistem dengan Apple Silicon
3. "CPU" sebagai nilai *default* dimana pengguna tidak menggunakan baik GPU ataupun Apple Silicon.

3.4 Modul download_model.py

3.4.1 Import Library

Kode 18: Import Library pada modul download_model.py

Modul download_model.py hanya menggunakan 3 pustaka, yaitu: os, requests, dan tqdm.

3.4.2 Fungsi download_model_face_detection

```
3   import os
4   import requests
5   from tqdm import tqdm
```

Kode 14: fungsi download_model_face_detection

Fungsi di atas ini berguna untuk mengunduh model face_detector dari MediaPipe. Model ini digunakan untuk mendeteksi muka pengguna dalam video.

3.4.3 Fungsi download_model_pose_detection

Berikut merupakan penjelasan dari fungsi pada download_model_pse_detection:

```
12  def download_file(url, dest_folder):
13
14      if not os.path.exists(dest_folder):
15          os.makedirs(dest_folder)
16
17      file_name = url.split('/')[-1]
18      file_path = os.path.join(dest_folder, file_name)
19
20      if os.path.exists(file_path):
21          print(f"Model '{file_name}' already exists at '{file_path}'. Skipping download.")
22          return file_path
23
24      print(f"Downloading {file_name} from {url}...")
25      try:
26          response = requests.get(url, stream=True)
27          response.raise_for_status() # Raise an exception for HTTP errors
28
29          total_size_in_bytes = int(response.headers.get('content-length', 0))
30          block_size = 1024 # 1 Kibibyte
31          progress_bar = tqdm(total=total_size_in_bytes, unit='iB', unit_scale=True)
32
33          with open(file_path, 'wb') as file:
34              for data in response.iter_content(block_size):
35                  progress_bar.update(len(data))
36                  file.write(data)
37          progress_bar.close()
38
39          if total_size_in_bytes != 0 and progress_bar.n != total_size_in_bytes:
40              print("ERROR, something went wrong during download.")
41              raise IOError("Download incomplete.")
42
43          print(f"Successfully downloaded '{file_name}' to '{file_path}'.")
44          return file_path
45      except requests.exceptions.RequestException as e:
46          print(f"Error downloading {file_name}: {e}")
```

```
48              if os.path.exists(file_path):
49                  os.remove(file_path)
50              raise
```

Kode 15: Fungsi download_model_pose_detection

Fungsi di atas ini berguna untuk mengunduh model pose landmarker dari MediaPipe. Model ini digunakan untuk mendeteksi pose tubuh pengguna dalam video (khususnya dalam bagian bahu / torso atas).

3.5 Modul heart_rate.py

3.5.1 Import Library

```
3   import numpy as np
4   from scipy.signal import butter, filtfilt
5   import cv2
6   import mediapipe as mp # Diperlukan untuk get_initial_roi
```

Kode 16: Import Library pada modul heart_rate.py

Modul heart_rate.py hanya menggunakan 3 pustaka yaitu: numpy, OpenCV, dan SciPy. 3 pustaka ini berguna dalam perhitungan algoritma *Plane Orthogonal-to-Skin* (POS).

3.5.2 Fungsi cpu_POS

Berikut merupakan penjelasan dari fungsi cpu_POS pada modul heart_rate.py:

```
8   def cpu_POS(input_video, fps):
9
10      # Reshape input_video to (3, N)
11      C = input_video[0] # Assuming input_video is (1, 3, N), take the first (and only) sample
12
13      H = np.zeros(C.shape[1]) # Initialize output signal
14
15      # Normalize the RGB channels
16      # Menghindari pembagian oleh nol jika norm(C, axis=0) menghasilkan nol
17      norm_C = C / (np.linalg.norm(C, axis=0) + 1e-6) # Add small epsilon for stability
18
19      # Calculate alpha and beta
20      alpha = np.std(norm_C[0]) / (np.std(norm_C[1]) + 1e-6)
21      beta = np.std(norm_C[0]) / (np.std(norm_C[2]) + 1e-6)
22
23      # Construct orthogonal projection plane
24      S = alpha * norm_C[0] + norm_C[1]
25      P = beta * norm_C[0] + norm_C[2]
26
27      # Calculate rPPG signal (H)
28      for t in range(C.shape[1]):
29          H[t] = S[t] - P[t]
30
31      return H
```

Kode 17: Fungsi cpu_POS

Fungsi di atas ini menggunakan algoritma *Plane Orthogonal-to-Skin* (POS) untuk menghitung sinyal rPPG atau jumlah detak jantung pengguna.

3.5.3 def initial ROI

```python
def get_initial_roi(image, landmarker, x_size=100, y_size=30, shift_x=0, shift_y=-30):
    """
    Mengambil ROI awal dari frame webcam untuk mendeteksi sinyal respirasi
    berdasarkan pergerakan posisi bahu pasien.

    Args:
        image (np.ndarray): Frame dari webcam dalam format BGR.
        landmarker (mediapipe.tasks.vision.PoseLandmarker): Objek MediaPipe pose detector.
        x_size (int): Setengah lebar ROI pada sumbu x. Total lebar ROI adalah 2 * x_size.
        y_size (int): Setengah tinggi ROI pada sumbu y. Total tinggi ROI adalah 2 * y_size.
        shift_x (int): Pergeseran ROI pada sumbu x (nilai positif menggeser ke kanan).
        shift_y (int): Pergeseran ROI pada sumbu y (nilai positif menggeser ke bawah).

    Returns:
        tuple: Koordinat ROI (left_x, top_y, right_x, bottom_y).

    Raises:
        ValueError: Jika tidak ada pose yang terdeteksi atau dimensi ROI tidak valid.
    """
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Mengubah warna BGR ke RGB
    height, width = image.shape[:2] # Mengambil dimensi frame webcam

    # Membuat gambar MediaPipe dari frame webcam
    mp_image = mp.Image(
        image_format=mp.ImageFormat.SRGB,
        data=image_rgb
    )

    # Mendeteksi pose dari frame webcam
    detection_result = landmarker.detect(mp_image)
```

Kode 18: def initial ROI

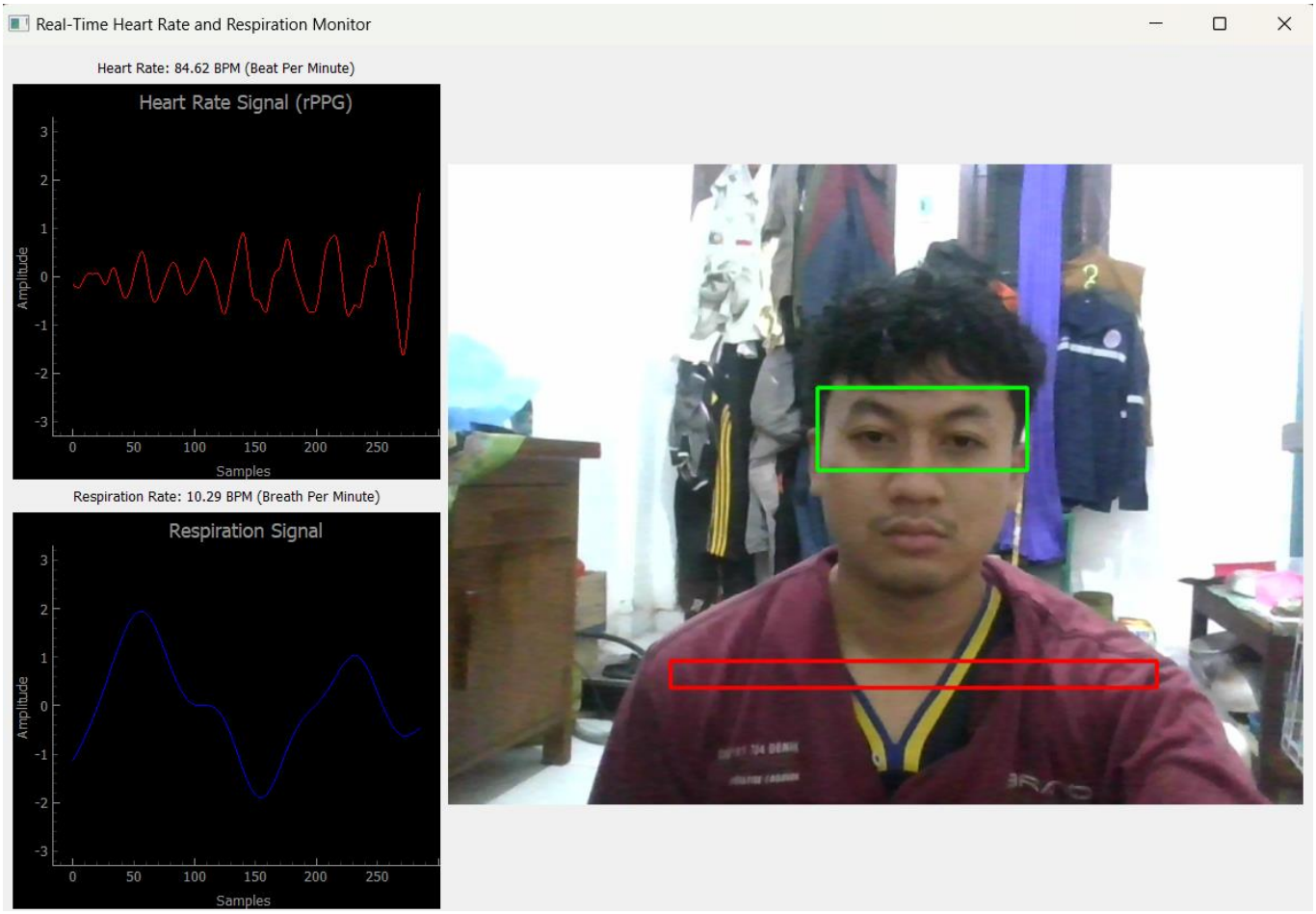
3.5.6 fungsi ROI

```python
    if not detection_result.pose_landmarks:
        raise ValueError("No pose detected in frame for respiration ROI initialization! Make sure your upper body is visible.")

    # Mendeteksi tubuh pengguna dari landmark pertama
    landmarks = detection_result.pose_landmarks[0]

    # Mengambil landmark bahu kiri dan kanan (indeks 11 dan 12)
    left_shoulder = landmarks[mp.solutions.pose.PoseLandmark.LEFT_SHOULDER]
    right_shoulder = landmarks[mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER]

    # Menghitung posisi tengah dari bahu kiri dan bahu kanan
    # Koordinat landmark adalah normalisasi (0-1), jadi kalikan dengan dimensi frame
    center_x = int(((left_shoulder.x + right_shoulder.x) / 2) * width)
    center_y = int(((left_shoulder.y + right_shoulder.y) / 2) * height)

    # Mengaplikasikan shift terhadap titik tengah
    center_x += shift_x
    center_y += shift_y

    # Menghitung batasan ROI berdasarkan posisi tengah dan ukuran ROI
    left_x = max(0, center_x - x_size)
    right_x = min(width, center_x + x_size)
    top_y = max(0, center_y - y_size)
    bottom_y = min(height, center_y + y_size)

    # Mevalidasi ukuran ROI
    if (right_x - left_x) <= 0 or (bottom_y - top_y) <= 0:
        raise ValueError(f"Invalid ROI dimensions: [{left_x}:{right_x}, {top_y}:{bottom_y}]. Adjust x_size, y_size, or shift parameters.")

    return (left_x, top_y, right_x, bottom_y)
```

Kode 19: Fungsi ROI

4 Hasil

Berdasarkan hasil implementasi, program berhasil mendeteksi wajah dan bahu dengan menggunakan MediaPipe.



Gambar 1: Hasil Deteksi Program

Bounding box berfungsi sebagai panduan visual bagi pengguna untuk menyesuaikan posisi tubuh agar dapat dikenali dengan baik oleh sistem. Elemen ini memberikan umpan balik secara langsung, yang membantu meningkatkan akurasi deteksi serta mempermudah interaksi pengguna.

Penggunaan **optical flow** dalam proses deteksi respirasi memberikan efisiensi tinggi karena tidak memerlukan pendeteksian ulang pada setiap frame. Hal ini membuat pemrosesan video menjadi lebih ringan dan responsif.

Penentuan **Region of Interest (ROI)** untuk masing-masing fungsi juga memainkan peran penting dalam peningkatan performa sistem. Untuk **rPPG (Remote Photoplethysmography)**, ROI ditempatkan di area dahi, yang terbukti efektif dalam menangkap sinyal detak jantung. Sedangkan untuk deteksi sinyal pernapasan, ROI difokuskan pada area di antara bahu, tepatnya di titik tengah antara landmark 11 dan 12 dari MediaPipe. Area ini dipilih karena gerakan dada sebagai indikator utama pernapasan dapat terdeteksi dengan optimal di sana.

Lebih lanjut, integrasi dengan **antarmuka grafis (GUI)** berbasis PyQt5 turut meningkatkan pengalaman pengguna. GUI ini menampilkan tampilan langsung dari kamera, visualisasi sinyal, serta estimasi BPM (baik napas maupun detak jantung) secara real-time. Penyajian informasi ini dirancang agar mudah dipahami, menjadikan aplikasi ini lebih informatif dan ramah bagi pengguna.

5 Kesimpulan

Program yang dikembangkan mampu mendeteksi detak jantung (Heart Rate/HR) dan sinyal pernapasan dengan tingkat akurasi yang tinggi, berkat penerapan teknik **Remote Photoplethysmography (rPPG)** dan **MediaPipe**. MediaPipe digunakan untuk mendeteksi secara andal area wajah dan bahu, yang kemudian dijadikan acuan untuk analisis lebih lanjut. Pemilihan **Region of Interest (ROI)** — di dahi untuk HR dan di antara landmark bahu untuk pernapasan — terbukti efektif dalam menangkap sinyal yang dibutuhkan.

Penggunaan **optical flow** dalam analisis sinyal pernapasan memberikan keunggulan efisiensi dengan mengurangi kebutuhan untuk melakukan deteksi ulang pada setiap frame. Hal ini secara signifikan meningkatkan performa dan responsivitas program.

Tak hanya itu, antarmuka pengguna berbasis **PyQt5 GUI** menambah kenyamanan penggunaan dengan menampilkan tangkapan kamera secara langsung, visualisasi sinyal, serta estimasi **BPM (Breaths dan Beats per Minute)** secara real-time. Seluruh informasi disajikan secara intuitif dan mudah dipahami.

Dengan menggabungkan pendekatan visual berbasis landmark serta teknik pemrosesan sinyal yang efisien, program ini menawarkan solusi yang praktis, akurat, dan ramah pengguna untuk pemantauan HR dan respirasi berbasis gerakan tubuh.

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] C. Massaroni, D. Lo Presti, D. Formica, S. Silvestri, and E. Schena, “Non-contact monitoring of breathing pattern and respiratory rate via rgb signal measurement,” *Sensors*, vol. 19, no. 12, p. 2758, 2019.