Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama Anggota:
1. Arkan Hariz Chandrawinata Liem (122140038)
2. Bezalel Samuel Manik (122140140)
Tugas: **Tugas Besar**
Mata Kuliah: **Pengolahan Sinyal Digital (IF3024)**
Tanggal: **May 31, 2025**

1 Pendahuluan

1.1 Latar Belakang

Kesehatan merupakan hak asasi manusia yang fundamental dan menjadi salah satu unsur penting dalam mewujudkan kesejahteraan hidup. Dalam konteks global, pemahaman mengenai kesehatan telah berkembang menjadi konsep yang holistik, meliputi kesehatan fisik, mental, sosial, dan spiritual. Salah satu pendekatan terbaru dalam upaya menjaga kesehatan adalah paradigma 24-jam pergerakan, yang menyatakan bahwa perilaku bergerak sepanjang hari—termasuk aktivitas fisik, perilaku sedentari, dan tidur—berkontribusi secara signifikan terhadap berbagai indikator kesehatan pada semua kelompok usia. Studi menunjukkan bahwa kombinasi waktu yang ideal dalam satu hari untuk aktivitas tersebut berkorelasi positif dengan kualitas hidup, kebugaran jasmani, serta kesehatan mental dan sosial seseorang [1]. Oleh karena itu, pemahaman menyeluruh tentang bagaimana komposisi perilaku harian memengaruhi kesehatan menjadi sangat penting dalam mendukung kesehatan masyarakat.

Di sisi lain, kesehatan masyarakat juga sangat dipengaruhi oleh perilaku hidup sehat yang dilakukan secara konsisten. Faktor utama yang menentukan status kesehatan individu dan masyarakat adalah faktor genetik, lingkungan, pelayanan kesehatan, dan perilaku hidup sehat, dengan perilaku dan lingkungan menjadi faktor paling dominan. Perilaku seperti merokok, kurang konsumsi buah dan sayur, serta kurangnya aktivitas fisik telah terbukti secara statistik berkaitan dengan rendahnya status kesehatan [2]. Meskipun faktor genetik dan lingkungan turut berperan, namun perubahan perilaku sehat merupakan langkah konkret yang dapat dilakukan secara individual untuk memperbaiki derajat kesehatan masyarakat.

Dalam bidang kesehatan kini sudah diterapkannya teknologi yang sudah maju dalam perkembangannya. Perkembangan teknologi telah membawa dampak besar dalam bidang kesehatan, khususnya dalam peningkatan layanan diagnosis, pemantauan, dan pengobatan pasien. Teknologi seperti Internet of Things (IoT), kecerdasan buatan (AI), dan big data kini digunakan untuk mengembangkan sistem kesehatan cerdas yang mampu memantau kondisi pasien secara real-time dan jarak jauh. Contohnya, sistem berbasis IoT memungkinkan tenaga medis memantau tanda vital pasien seperti detak jantung dan suhu tubuh melalui perangkat wearable yang terhubung ke internet. Selain itu, kecerdasan buatan telah digunakan untuk menganalisis citra medis seperti MRI dan CT scan guna membantu diagnosis penyakit secara lebih cepat dan akurat. Teknologi ini juga dimanfaatkan untuk mengembangkan chatbot medis dan asisten virtual yang dapat memberikan konsultasi kesehatan dasar kepada masyarakat. Dengan pemanfaatan teknologi yang terus berkembang, sistem layanan kesehatan menjadi lebih efisien, personal, dan mudah diakses oleh semua lapisan masyarakat [3].

Dalam proyek ini, Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ullamcorper, tortor vitae porta dapibus, arcu lacus fringilla ante, eget elementum neque ante et lorem. Quisque non pretium ligula. Cras tempus ultrices ultricies. Mauris ac felis mattis, hendrerit ante at, rutrum dui. Class aptent taciti sociosqu ad litora torquent per conubia.

2 Alat dan Bahan

untuk menyelesaikan Project akhir Pendeteksi Sinyal Resipirasi dan Sinyal rPPG berikut dibutuhkan alat dan bahan sebagai berikut :

2.1 Bahasa Pemrograman

bahasa pemrograman yang digunakan untuk menyelesaikan project akhir ini adalah bahasa pemrograman Python. hal tersebut dikarenkan library pada python yang sangat membantu proyek akhir ini

2.2 Library

berikut library yang digunakan pada proyek akhir ini :

2.2.1 PyQt5

Fungsi: Framework GUI berbasis Qt untuk Python yang mendukung pembuatan antarmuka interaktif.
Kegunaan di proyek: Digunakan untuk membuat tampilan antarmuka pengguna seperti tombol dan grafik real-time.

2.2.2 opencv-python

Fungsi: Komputer visi dan pengolahan citra.
Kegunaan di proyek: Digunakan untuk menangkap video dari kamera dan mendeteksi area wajah secara real-time.

2.2.3 matplotlib

Fungsi: Visualisasi data dalam bentuk grafik atau plot.
Kegunaan di proyek: Menampilkan grafik detak jantung (HR) dan laju pernapasan (RR) secara real-time dalam GUI.

2.2.4 numpy

Fungsi: Perhitungan numerik dan manipulasi array.
Kegunaan di proyek: Digunakan untuk pemrosesan sinyal seperti pengolahan array frame video dan perhitungan statistik sederhana.

2.2.5 scipy

Fungsi: Library ilmiah yang menyediakan fungsi statistik, sinyal, FFT, dan lainnya.
Kegunaan di proyek: Memanfaatkan `scipy.signal` untuk filtering sinyal dan deteksi puncak, serta fungsi statistik lainnya.

2.2.6 mediapipe

Fungsi: Framework machine learning untuk pelacakan pose, tangan, dan wajah secara real-time.
Kegunaan di proyek: Digunakan untuk mendeteksi landmark wajah dan bahu guna mengekstraksi sinyal detak jantung (rPPG) dan respirasi.

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

2. Ekstraksi Sinyal Respirasi

Menggunakan analisis pergerakan bahu untuk mendapatkan pola respirasi pengguna

3 Penjelasan

Program pada proyek ini dibagi menjadi 5 modul, yaitu **main.py**, **GUI.py**, **respirasi.py**, **rPPG.py**, dan **signal_processing.py**. Berikut adalah penjelasan mengenai alur kerja program yang dikembangkan pada proyek ini:

3.1 Penjelasan GUI.py

3.1.1 Import Library

Berikut adalah penjelasan import library pada GUI.py:

```
    import sys
    import threading
    import numpy as np
    from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QLabel,
    QHBoxLayout
    from PyQt5.QtGui import QImage, QPixmap, QFont
    from PyQt5.QtCore import QTimer
    import cv2
    import matplotlib.pyplot as plt
    from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
    from signal_processing import estimate_bpm
    import main

```

Kode 1: Import library pada GUI.py

Pada GUI.py, meng-import beberapa library yaitu PyQt5, cv2, numpy, threading, matplotlib, signal_processing dan main. PyQt5 bertanggung jawab untuk membangun antarmuka grafis (GUI) dengan elemen seperti tombol, label, dan tata letak. OpenCV digunakan untuk menangkap dan memproses input video dari kamera, sementara Matplotlib menampilkan grafik sinyal respirasi dan rPPG di dalam GUI. Untuk manipulasi data numerik yang krusial dalam pemrosesan sinyal, digunakan NumPy. Modul threading memungkinkan pemrosesan paralel, menjaga antarmuka tetap responsif. QTimer berperan dalam memperbarui tampilan aplikasi secara berkala. Selain itu, fungsi estimate_bpm dari modul signal_processing digunakan untuk menghitung detak jantung atau laju pernapasan dari sinyal video, dan modul main kemungkinan besar berisi logika utama aplikasi.

3.1.2 Class RPPGApp dan fungsi def __init__

Berikut adalah penjelasan class RPPGApp dan fungsi def __init__:

```
    class RPPGApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Real-time Medical Monitor Style")
        self.setStyleSheet("background-color: black; color: white;")
        self.resize(1400, 800)

        self.fps = 30
        self.main_layout = QHBoxLayout()
        self.setLayout(self.main_layout)

        # --- Left panel: signals and labels ---
        self.left_panel = QVBoxLayout()

        self.fig, self.axs = plt.subplots(2, 1, figsize=(8, 4), dpi=100)
        for ax in self.axs:
            ax.set_facecolor('black')
            ax.tick_params(colors='white')
            for spine in ax.spines.values():
                spine.set_color('white')
            ax.grid(True, linestyle='--', alpha=0.3, color='gray')
        self.fig.patch.set_facecolor('black')
        self.canvas = FigureCanvas(self.fig)
        self.left_panel.addWidget(self.canvas)

        stat_layout = QHBoxLayout()
        self.hr_label = QLabel("HR: -- bpm")
        self.hr_label.setStyleSheet("color: lime;")
        self.hr_label.setFont(QFont("Consolas", 28))

        self.rr_label = QLabel("RR: -- bpm")
        self.rr_label.setStyleSheet("color: cyan;")
        self.rr_label.setFont(QFont("Consolas", 28))

        stat_layout.addWidget(self.hr_label)
        stat_layout.addStretch()
        stat_layout.addWidget(self.rr_label)
        self.left_panel.addLayout(stat_layout)

        control_layout = QHBoxLayout()
        self.start_btn = QPushButton("Start")
        self.stop_btn = QPushButton("Stop")
        self.stop_btn.setEnabled(False)
        control_layout.addWidget(self.start_btn)
        control_layout.addWidget(self.stop_btn)
        self.left_panel.addLayout(control_layout)

        # --- Right panel: video feed ---
        self.right_panel = QVBoxLayout()
        self.video_label = QLabel()
        self.video_label.setFixedSize(640, 480)
        self.right_panel.addWidget(self.video_label)

        # Combine both panels
        self.main_layout.addLayout(self.left_panel, 2)
        self.main_layout.addLayout(self.right_panel, 1)

        self.start_btn.clicked.connect(self.start_monitoring)
        self.stop_btn.clicked.connect(self.stop_monitoring)
```

```
60
61        self.timer = QTimer()
62        self.timer.timeout.connect(self.update_ui)
63
```

Kode 2: Class RPPGApp dan fungsi def __init__

Terdapat class RPPGApp, Kelas ini mewarisi QWidget dan dirancang dengan tampilan jendela berlatar belakang hitam, teks putih, serta ukuran 1400x800 piksel. Antarmuka aplikasi dibagi menjadi dua panel utama yang disusun secara horizontal menggunakan QHBoxLayout: panel kiri didedikasikan untuk tampilan grafik sinyal dan informasi medis, sementara panel kanan menampilkan feed video dari kamera. Panel kiri memiliki dua grafik Matplotlib yang ditempatkan dalam subplot terpisah untuk menampilkan sinyal respirasi dan rPPG. Grafik ini menggunakan gaya visual bernuansa medis dengan latar belakang hitam, garis putih, dan grid abu-abu untuk kemudahan pembacaan. Di bawah grafik, terdapat label-label yang menampilkan nilai detak jantung (HR) dan laju pernapasan (RR), diatur dengan font "Consolas" berukuran besar dan berwarna mencolok (lime untuk HR dan cyan untuk RR) agar mudah terlihat. Selain itu, dua tombol kontrol, "Start" dan "Stop", memungkinkan pengguna untuk memulai atau menghentikan proses pemantauan.
Sementara itu, panel kanan dirancang lebih sederhana, hanya berisi sebuah QLabel yang berfungsi untuk menampilkan video dengan ukuran tetap 640x480 piksel. Seluruh panel ini kemudian digabungkan ke dalam main_layout, di mana panel kiri diberikan proporsi ruang yang lebih besar (2 banding 1) untuk mengakomodasi tampilan grafik yang lebih mendetail. Fungsi start_monitoring dan stop_monitoring dihubungkan ke tombol masing-masing, memungkinkan interaksi pengguna yang intuitif. Selain itu, QTimer digunakan untuk memperbarui antarmuka secara berkala melalui metode update_ui, yang memastikan bahwa tampilan data dan video berjalan secara real-time dan sinkron, memberikan pengalaman pemantauan yang lancar dan responsif.

3.1.3 Fungsi def start_monitoring dan stop_monitoring

Berikut adalah penjelasan Fungsi def start_monitoring dan stop_monitoring:

```
1     def start_monitoring(self):
2         main.monitoring_active = True
3         self.start_btn.setEnabled(False)
4         self.stop_btn.setEnabled(True)
5         self.timer.start(1000)
6         threading.Thread(target=main.run_main, daemon=True).start()
7
8     def stop_monitoring(self):
9         main.monitoring_active = False
10        self.timer.stop()
11        self.start_btn.setEnabled(True)
12        self.stop_btn.setEnabled(False)
13        self.show_final_plot()
14
```

Kode 3: Fungsi start_monitoring dan stop_monitoring

Terdapat 2 fungsi yaitu start_monitoring dan stop_monitoring yang digunakan untuk pengelolaan GUI nya. Saat pengguna menekan tombol "Start", fungsi start_monitoring akan bekerja. Ini akan mengatur status monitoring_active di modul main menjadi True, menonaktifkan tombol "Start", dan mengaktifkan tombol "Stop" untuk menghindari kesalahan klik. Bersamaan dengan itu, QTimer akan mulai beroperasi dengan interval 1000 milidetik (1 detik), secara berkala memanggil fungsi update_ui untuk memperbarui tampilan. Yang terpenting, main.run_main akan dijalankan di thread terpisah menggunakan modul threading, memastikan bahwa proses utama seperti pengambilan video dan pemrosesan sinyal berjalan mulus tanpa mengganggu responsivitas antarmuka pengguna.

Sebaliknya, ketika tombol "Stop" ditekan, fungsi stop_monitoring akan menghentikan proses pemantauan. Ini dilakukan dengan mengatur monitoring_active menjadi False, menghentikan QTimer, dan mengembalikan status tombol ke kondisi awal. Setelah itu, fungsi show_final_plot akan dipanggil untuk menampilkan grafik ringkasan dari sinyal yang telah dipantau, memberikan pengguna tinjauan data dari sesi tersebut.

3.1.4 Fungsi update_ui

Berikut adalah penjelasan dari fungsi update_ui:

```
    def update_ui(self):
        # Video frame
        if main.frame_display is not None:
            frame = main.frame_display
            rgb_image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            h, w, ch = rgb_image.shape
            bytes_per_line = ch * w
            qt_image = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format_RGB888)
            self.video_label.setPixmap(QPixmap.fromImage(qt_image))

        # Signal plotting
        if len(main.rgb_buffer) > 100:
            rgb_np = np.array(main.rgb_buffer)
            h_signal = main.apply_pos(rgb_np)
            hr_bpm, hr_filt = estimate_bpm(h_signal, fs=main.fps, lowcut=0.7, highcut=3.0)
            rr_bpm, rr_filt = estimate_bpm(list(main.resp_signal), fs=main.fps, lowcut=0.1, highcut
    =0.7)

            self.axs[0].cla()
            self.axs[0].plot(rr_filt[-300:], color='cyan')
            self.axs[0].set_title("Respiratory Signal", color='white')
            self.axs[0].set_facecolor('black')

            self.axs[1].cla()
            self.axs[1].plot(hr_filt[-300:], color='lime')
            self.axs[1].set_title("rPPG Signal", color='white')
            self.axs[1].set_facecolor('black')

            self.canvas.draw()

            self.hr_label.setText(f"HR: {hr_bpm:.1f} bpm")
            self.rr_label.setText(f"RR: {rr_bpm:.1f} bpm")

```

Kode 4: Fungsi update_ui

Fungsi update_ui adalah jantung pembaruan real-time, mengonversi dan menampilkan frame video, memproses sinyal warna menjadi rPPG dan sinyal pernapasan, serta menghitung dan memperbarui nilai HR dan RR. Semua grafik dan nilai ditampilkan secara real-time dan disinkronkan, memastikan pengalaman pemantauan yang lancar dan informatif.

3.1.5 Fungsi show_final_plot

Berikut adalah penjelasan dari fungsi show_final_plot:

```
    def show_final_plot(self):
        if len(main.rgb_full) == 0 or len(main.resp_full) == 0:
            return

        rgb_np = np.array(main.rgb_full)
        h_signal = main.apply_pos(rgb_np)
```

```
        hr_bpm, hr_filt = estimate_bpm(h_signal, fs=main.fps, lowcut=0.7, highcut=3.0)
        rr_bpm, rr_filt = estimate_bpm(main.resp_full, fs=main.fps, lowcut=0.1, highcut=0.7)

        plt.figure("Final Full Signal", figsize=(10, 4))
        plt.subplot(2, 1, 1)
        plt.plot(rr_filt, color='cyan')
        plt.title(f"Respiration Signal (RR: {rr_bpm:.1f} bpm)")

        plt.subplot(2, 1, 2)
        plt.plot(hr_filt, color='lime')
        plt.title(f"rPPG Signal (HR: {hr_bpm:.1f} bpm)")

        plt.tight_layout()
        plt.show()

```

Kode 5: Fungsi show_final_plot

Fungsi show_final_plot bertugas menampilkan ringkasan grafik dari seluruh sesi pemantauan setelah proses berhenti. Pertama, fungsi ini memastikan bahwa data sinyal warna (main.rgb_full) dan sinyal pernapasan (main.resp_full) telah terkumpul. Jika salah satu kosong, fungsi akan berhenti. Sinyal RGB akan dikonversi menjadi array NumPy dan diproses oleh main.apply_pos untuk menghasilkan sinyal rPPG. Kedua sinyal, yaitu rPPG dan pernapasan, kemudian dianalisis menggunakan estimate_bpm. Proses ini menghitung rata-rata detak jantung (HR) dan laju pernapasan (RR) dalam satuan bpm, sekaligus memfilter sinyal untuk mengurangi noise.

3.1.6 Eksekusi Program Utama

Berikut penjelasan untuk eksekusi program utama:

```
    if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = RPPGApp()
    window.show()
    sys.exit(app.exec_())

```

Kode 6: Eksekusi program utama

Bagian ini untuk mengeksekusi program utama/main. if __name__ == '__main__': yang membuat QApplication, menampilkan jendela RPPGApp, dan menjalankan event loop aplikasi hingga ditutup.

3.2 Penjelasan `rPPG.py`

3.2.1 Import Library

```
import numpy as np
import cv2
from mediapipe.tasks import python
from mediapipe.tasks.python import vision
```

Kode 7: Import library

Penjelasan:

- `numpy`: Untuk manipulasi array numerik (contohnya saat mengelola data piksel).
- `cv2` (OpenCV): Untuk menangani operasi pemrosesan citra seperti konversi warna.
- `mediapipe.tasks.python`: Untuk menggunakan model deteksi wajah `BlazeFace` dari MediaPipe.

3.2.2 Inisialisasi Model Deteksi Wajah

```
base_model = "Model/blaze_face_short_range.tflite"
base_options = python.BaseOptions(model_asset_path=base_model)
FaceDetectorOptions = vision.FaceDetectorOptions
VisionRunningMode = vision.RunningMode

options = FaceDetectorOptions(
    base_options=base_options,
    running_mode=VisionRunningMode.IMAGE,
)
face_detector = vision.FaceDetector.create_from_options(options)
```

Kode 8: Setup model BlazeFace

Penjelasan:

- Model `.tflite` yang digunakan adalah `blaze_face_short_range`, cocok untuk deteksi wajah jarak dekat.
- `BaseOptions` menyimpan path model.
- `FaceDetectorOptions` digunakan untuk mengatur parameter model, seperti mode eksekusi (`IMAGE` untuk pemrosesan satu per satu).
- Objek `face_detector` kemudian dibuat dan siap digunakan untuk mendeteksi wajah dari gambar.

3.2.3 Fungsi `extract_forehead_roi`

```
def extract_forehead_roi(frame, detection):
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    bboxC = detection.bounding_box
    x, y, w, h = bboxC.origin_x, bboxC.origin_y, bboxC.width, bboxC.height

    scale_w = 0.85
    scale_h = 1.1
    offset_y = 3
    new_w = int(w * scale_w)
    new_h = int(h * scale_h)
    center_x = int(x + w / 2)
    center_y = int(y + h * 0.35 + offset_y)
    new_x = max(0, center_x - new_w // 2)
    new_y = max(0, center_y - new_h // 2)
    new_x = min(new_x, frame.shape[1] - new_w)
    new_y = min(new_y, frame.shape[0] - new_h)
    roi = rgb_frame[new_y:new_y + new_h, new_x:new_x + new_w]
    return roi, (new_x, new_y, new_w, new_h)
```

Kode 9: Fungsi ekstraksi ROI dahi

Penjelasan fungsi:

- Fungsi ini menerima dua parameter: `frame` (gambar dari kamera) dan `detection` (hasil deteksi wajah).
- Mengonversi `frame` dari format BGR ke RGB.
- Mengambil informasi bounding box dari hasil deteksi wajah.
- ROI dahi dihitung dengan mengatur ulang posisi berdasarkan proporsi tertentu (`scale_w`, `scale_h`) dan `offset_y` ke atas.

- Dilakukan validasi agar ROI tidak keluar dari batas gambar.
- ROI dikembalikan dalam bentuk potongan gambar RGB dan tuple koordinat lokasi ROI.

3.3 Penjelasan `main.py`

3.3.1 Import Library

```
import cv2
import threading
import time
import numpy as np
from mediapipe.tasks import python
from mediapipe.tasks.python import vision
import mediapipe as mp
from collections import deque
from rPPG import extract_forehead_roi, face_detector
from respirasi import get_respiration_roi
from signal_processing import estimate_bpm
```

Kode 10: Import library

Penjelasan:

- `OpenCV (cv2)`: menangkap video dan pemrosesan gambar.
- `threading` dan `time`: untuk proses paralel dan delay.
- `MediaPipe`: untuk deteksi wajah menggunakan BlazeFace.
- `deque`: digunakan sebagai buffer sinyal dengan panjang terbatas.
- `extract_forehead_roi`: fungsi dari `rPPG.py` untuk ROI dahi.
- `get_respiration_roi`: menentukan area pernapasan (dada).
- `estimate_bpm`: menghitung detak jantung atau RR dari sinyal.

3.3.2 Inisialisasi Variabel Global

```
fps = 30
buffer_len = 300
rgb_buffer = deque(maxlen=buffer_len)
r_signal = deque(maxlen=buffer_len)
g_signal = deque(maxlen=buffer_len)
b_signal = deque(maxlen=buffer_len)
resp_signal = deque(maxlen=buffer_len)
features = None
old_gray = None
hr = 0.0
rr = 0.0
monitoring_active = True
resp_roi_coords = None
frame_display = None
rgb_full = []
resp_full = []
```

Buffer digunakan untuk menyimpan sinyal rPPG dan respirasi secara real-time. Nilai `fps` menentukan laju frame per detik.

3.3.3 Parameter Lucas-Kanade Optical Flow

```
lk_params = dict(winSize=(15, 15), maxLevel=2,
                 criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
```

Digunakan untuk konfigurasi pelacakan titik fitur di area dada guna estimasi sinyal respirasi.

3.3.4 Fungsi `apply_pos`

```
def apply_pos(rgb_matrix):
    mean_centered = rgb_matrix - np.mean(rgb_matrix, axis=0)
    S = np.array([[0, 1, -1], [-2, 1, 1]])
    X = np.dot(S, mean_centered.T)
    h = X[0] + X[1]
    return h
```

Fungsi ini menerapkan metode POS (Plane-Orthogonal-to-Skin) untuk menggabungkan saluran RGB menjadi satu sinyal peka terhadap denyut nadi.

3.3.5 Fungsi `update_metrics`

```
def update_metrics():
    global hr, rr
    while monitoring_active:
        if len(rgb_buffer) >= 100:
            rgb_np = np.array(rgb_buffer)
            h = apply_pos(rgb_np)
            hr, _ = estimate_bpm(h, fs=fps, lowcut=0.7, highcut=3.0)
        if len(resp_signal) >= 100:
            rr, _ = estimate_bpm(list(resp_signal), fs=fps, lowcut=0.1, highcut=0.7)
        time.sleep(2)
```

Fungsi ini dijalankan pada thread paralel dan memperbarui nilai HR dan RR setiap dua detik.

Penjelasan Bandpass Filter pada Estimasi HR dan RR Pada fungsi `update_metrics`, filtering dilakukan menggunakan bandpass filter untuk memilih hanya sinyal dalam rentang frekuensi tertentu yang relevan secara fisiologis. Hal ini dilakukan dengan parameter `lowcut` dan `highcut` yang diteruskan ke fungsi `estimate_bpm`.

- Untuk sinyal rPPG (detak jantung), digunakan:
  - `lowcut = 0.7` Hz, yaitu 42 bpm
  - `highcut = 3.0` Hz, yaitu 180 bpm

  Sehingga hanya komponen sinyal dengan frekuensi antara **42–180 bpm** yang akan dianalisis.
- Untuk sinyal respirasi (pernapasan), digunakan:
  - `lowcut = 0.1` Hz, yaitu 6 bpm
  - `highcut = 0.7` Hz, yaitu 42 bpm

  Artinya hanya frekuensi antara **6–42 napas per menit** yang dipertahankan.

Penggunaan filter ini sangat penting untuk membuang noise dan komponen sinyal di luar rentang normal manusia. Filter dilakukan dengan menggunakan filter *Butterworth* orde ke-3 tanpa fase delay (`filtfilt`), yang sudah diimplementasikan dalam fungsi `bandpass_filter` pada modul `signal_processing.py`.

$$BPM = 60 \times \frac{jumlahpuncak}{durasi(detik)} \tag{1}$$

Rumus di atas digunakan untuk menghitung BPM berdasarkan jumlah puncak dari sinyal hasil filter.

3.3.6 Fungsi Utama `run_main`

```
def run_main():
    ...
```

Penjelasan isi fungsi:

1. Membuka webcam dan memulai thread `update_metrics()`.
2. Mengambil frame dari kamera secara loop.
3. Mendeteksi wajah, mengekstraksi ROI dahi, menghitung mean RGB dan menyimpannya.
4. Menentukan ROI dada untuk respirasi dan melacak fitur optik.
5. Menyimpan sinyal RGB dan sinyal pergerakan vertikal untuk estimasi.

3.3.7 Pelacakan Fitur dan Optical Flow

Fitur di ROI pernapasan dideteksi dengan `cv2.goodFeaturesToTrack`, dan dilacak antar frame dengan `cv2.calcOpticalFlowPyrLK`. Rata-rata posisi Y dari fitur digunakan sebagai sinyal respirasi mentah.

3.4 Modul `signal_processing.py`

3.4.1 Import Library

```
import numpy as np
from scipy.signal import butter, filtfilt, find_peaks
from scipy.ndimage import uniform_filter1d
```

Kode 11: Import library

Penjelasan:

- `numpy`: manipulasi array dan operasi numerik dasar.
- `scipy.signal.butter` dan `filtfilt`: untuk merancang dan menerapkan filter bandpass.
- `find_peaks`: mendeteksi puncak dalam sinyal untuk menghitung BPM/RR.
- `uniform_filter1d`: meratakan sinyal agar lebih halus untuk estimasi puncak.

3.4.2 Fungsi `bandpass_filter`

```
def bandpass_filter(data, fs, lowcut, highcut, order=3):
    b, a = butter(order, [lowcut, highcut], btype='band', fs=fs)
    return filtfilt(b, a, data)
```

Kode 12: Fungsi bandpass_filter

Fungsi ini menerapkan filter bandpass ke sinyal input:

- `butter` membuat filter Butterworth orde 3.
- `filtfilt` melakukan filtering maju-mundur agar tidak menimbulkan delay fase.
- Cocok digunakan untuk menyaring noise dari sinyal rPPG dan respirasi.

3.4.3 Fungsi `estimate_bpm`

```
def estimate_bpm(signal, fs, lowcut, highcut):
    if len(signal) < fs * 3:
        return 0.0, np.zeros_like(signal)
    filtered = bandpass_filter(signal, fs, lowcut, highcut)
    filtered = uniform_filter1d(filtered, size=5)
    peaks, _ = find_peaks(filtered, distance=fs//2)
    bpm = 60 * len(peaks) / (len(filtered) / fs)
    return bpm, filtered
```

Kode 13: Fungsi estimate_bpm

Penjelasan fungsi:

- Mengecek apakah panjang sinyal cukup untuk dihitung BPM (minimal 3 detik).
- Sinyal difilter dengan bandpass sesuai rentang frekuensi target.
- Digunakan perataan (smoothing) dengan `uniform_filter1d`.
- `find_peaks` digunakan untuk mendeteksi jumlah denyut (puncak).
- BPM dihitung sebagai:

$$BPM = 60 \times \frac{jumlahpuncak}{durasi(detik)}$$

- Mengembalikan nilai BPM serta sinyal yang sudah difilter.

3.5 Penjelasan respirasi.py

3.5.1 Import library

Berikut adalah penjelasan library yang dipakai pada respirasi.py:

```
    import cv2
    import mediapipe as mp
    from mediapipe.tasks import python
    from mediapipe.tasks.python import vision

```

Kode 14: Import library pada respirasi.py

Pada respirasi.py menggunakan dua library yaitu cv (opencv) dan mediapipe. Library cv (opencv) digunakan untuk pemrosesan gambar dan video, serta MediaPipe (mp) dari Google, sebuah pustaka machine learning real-time untuk visi komputer. from mediapipe.tasks import python dan from mediapipe.tasks.python import vision diimpor untuk mengakses modul tugas visi komputer MediaPipe berbasis Python, seperti PoseLandmarker, yang krusial untuk aplikasi pemantauan detak jantung dan pernapasan berbasis video.

3.5.2 Inisialisasi Pose Landmarker

Berikut penjelasan dari menginisialisasi pose landmarer:

```
    # Inisialisasi PoseLandmarker
    model_path = "Model/pose_landmarker.task"
    BaseOptions = mp.tasks.BaseOptions
    PoseLandmarkerOptions = vision.PoseLandmarkerOptions
    VisionRunningMode = vision.RunningMode

    options = PoseLandmarkerOptions(
```

```
        base_options=BaseOptions(model_asset_path=model_path),
        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
    )
    pose_landmarker = vision.PoseLandmarker.create_from_options(options)

```

Kode 15: Inisialisasi pose landmarker

Bagian ini untuk menginisialisasi pose landmarker. Pertama menentukan path dari model nya. kemudian mengonfigurasi opsi-opsi seperti BaseOptions, PoseLandmarkerOptions, dan VisionRunningMode. Opsi-opsi ini mengatur cara kerja model, termasuk lokasi model, running_mode=IMAGE (pemrosesan per gambar), dan num_poses=1 (deteksi satu pose). Setelah semua opsi diatur, objek pose_landmarker dibuat menggunakan PoseLandmarker.create_from_options(options), menghasilkan instansi detektor pose yang siap pakai untuk mengekstraksi informasi posisi tubuh manusia dari gambar masukan, penting untuk aplikasi seperti pemantauan detak jantung atau pernapasan.

3.5.3 Fungsi get_respiration_roi

Berikut adalah penjelasan fungsi get_respiration_roi:

```
    def get_respiration_roi(frame, scale_x=1.5, roi_height=120, shift_y=30, draw_bahu=True):
    image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    height, width = frame.shape[:2]

    mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=image_rgb)
    result = pose_landmarker.detect(mp_image)

    if not result.pose_landmarks:
        raise ValueError("Pose tidak terdeteksi.")

    landmarks = result.pose_landmarks[0]
    center_0 = landmarks[0]
    left_shoulder = landmarks[11]
    right_shoulder = landmarks[12]

    left_x = int(left_shoulder.x * width)
    right_x = int(right_shoulder.x * width)
    shoulder_width = abs(right_x - left_x)
    if shoulder_width < 20:
        raise ValueError("Pose gagal: bahu terlalu dekat")

    center_dada_x = (left_shoulder.x + right_shoulder.x) / 2
    center_dada_y = (left_shoulder.y + right_shoulder.y) / 2

    final_center_x = int(((center_0.x + center_dada_x) / 2) * width)
    final_center_y = int(((center_0.y + center_dada_y) / 2) * height) + shift_y

    roi_width = int(shoulder_width * scale_x)
    left_roi = max(0, final_center_x - roi_width // 2)
    right_roi = min(width, final_center_x + roi_width // 2)
    top_roi = max(0, final_center_y - roi_height // 2)
    bottom_roi = min(height, final_center_y + roi_height // 2)

    if draw_bahu:
        for pt in [left_shoulder, right_shoulder]:
            px = int(pt.x * width)
```

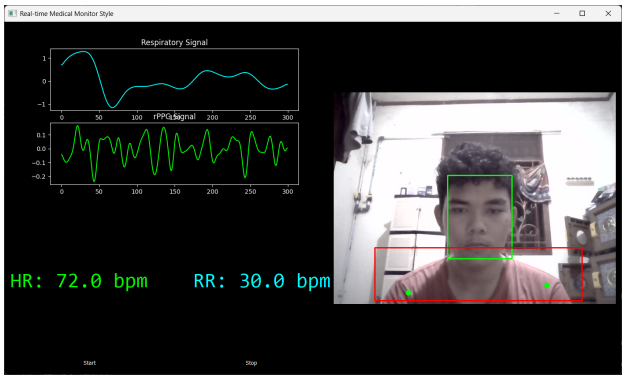
```
            py = int(pt.y * height)
            cv2.circle(frame, (px, py), 6, (0, 255, 0), -1)

    return (left_roi, top_roi, right_roi, bottom_roi)

```

Kode 16: Fungsi get_respiration_roi

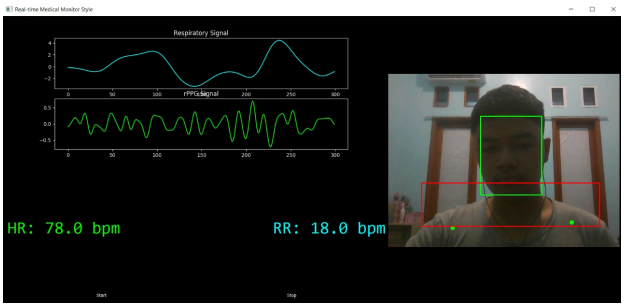
Fungsi get_respiration_roi mengidentifikasi region of interest (ROI) pada area dada dalam frame video untuk analisis sinyal pernapasan. Frame dikonversi ke RGB untuk diproses oleh PoseLandmarker MediaPipe guna mendeteksi landmark tubuh yang dimana bahu kiri nilai landmark nya 11 dan bahu kanan nilai landmark nya 12. Jika tidak ada pose terdeteksi, atau lebar bahu terlalu kecil, fungsi akan melemparkan error. ROI ditentukan berdasarkan posisi tengah antara kepala dan dada, disesuaikan secara vertikal, dengan lebar yang dihitung dari lebar bahu, dan tinggi tetap. ROI ini dibatasi agar tidak melebihi batas frame. Jika draw_bahu True, lingkaran kecil digambar pada bahu sebagai umpan balik visual. Akhirnya, fungsi mengembalikan koordinat ROI (left, top, right, bottom) untuk analisis area pernapasan.

4 Hasil

Eksperimen dilakukan pada dua subjek dengan dua kondisi pencahayaan berbeda: terang dan redup. Setiap subjek direkam menggunakan webcam untuk menangkap sinyal rPPG dan sinyal respirasi secara real-time. Sinyal-sinyal tersebut diproses menggunakan metode POS dan optical flow untuk mengekstraksi detak jantung (HR) dan laju pernapasan (RR).

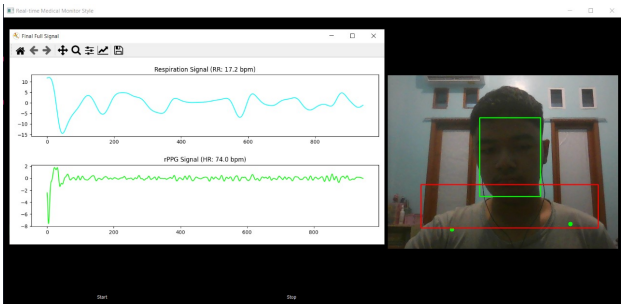


(a) rPPG - Subjek A (Terang)



(b) Respirasi - Subjek B (Redup)

(c) rPPG - Subjek A (Terang)



(d) Respirasi - Subjek B (Redup)

Gambar 1: Perbandingan sinyal rPPG dan respirasi dari dua subjek dalam dua kondisi pencahayaan berbeda.

Berdasarkan Gambar 1, terlihat bahwa kondisi pencahayaan berpengaruh terhadap kualitas sinyal yang diperoleh. Pada kondisi terang (Subjek A), sinyal rPPG tampak lebih halus dan memiliki amplitudo yang konsisten. Hal ini memungkinkan deteksi puncak yang lebih akurat dalam estimasi detak jantung.

Sebaliknya, sinyal rPPG pada Subjek B dalam kondisi redup mengalami fluktuasi yang lebih kasar dan tidak stabil. Ini menunjukkan bahwa intensitas cahaya mempengaruhi kestabilan pantulan cahaya dari kulit, yang merupakan dasar dari sinyal rPPG.

Sementara itu, sinyal respirasi dari kedua subjek terlihat cukup stabil meskipun terdapat sedikit noise pada kondisi redup. Hal ini karena metode optical flow yang digunakan untuk pelacakan gerakan dada masih mampu mendeteksi pergerakan vertikal secara konsisten.

Gambar a dan b merupkn keadan window ketika msih mengmbi gambar, sementara itu gmbr c dan d adalah keadan ketika sudah selesai mengambil gambar

Secara keseluruhan, sistem menunjukkan performa yang baik dalam menangkap kedua jenis sinyal fisiologis, namun kualitas pencahayaan tetap menjadi faktor penting dalam akurasi estimasi.

5 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem deteksi sinyal fisiologis berbasis video, dapat disimpulkan bahwa:

1. Sistem mampu mendeteksi dan mengekstraksi sinyal rPPG dari area dahi serta sinyal respirasi dari area dada secara real-time menggunakan kamera webcam standar.

2. Metode *Plane-Orthogonal-to-Skin* (POS) efektif dalam menggabungkan saluran RGB untuk menghasilkan sinyal rPPG yang dapat digunakan dalam estimasi detak jantung (Heart Rate).

3. Penggunaan pelacakan fitur optik (*optical flow*) pada area dada memungkinkan estimasi laju pernapasan (Respiratory Rate) yang cukup akurat, meskipun tetap bergantung pada kualitas deteksi pose.

4. Hasil percobaan menunjukkan bahwa kualitas pencahayaan sangat memengaruhi akurasi sinyal rPPG. Pencahayaan yang cukup menghasilkan sinyal yang lebih stabil, sementara pencahayaan redup menyebabkan noise yang lebih tinggi.

5. Visualisasi sinyal melalui antarmuka grafis (GUI) memungkinkan pengguna untuk memantau detak jantung dan respirasi secara simultan dalam waktu nyata.

Secara keseluruhan, sistem ini menunjukkan potensi untuk digunakan sebagai alat bantu monitoring fisiologis non-invasif. Namun, untuk keperluan medis yang lebih presisi, pengembangan lebih lanjut dibutuhkan, khususnya dalam hal kalibrasi sensor, kompensasi gerakan, dan validasi terhadap data klinis.

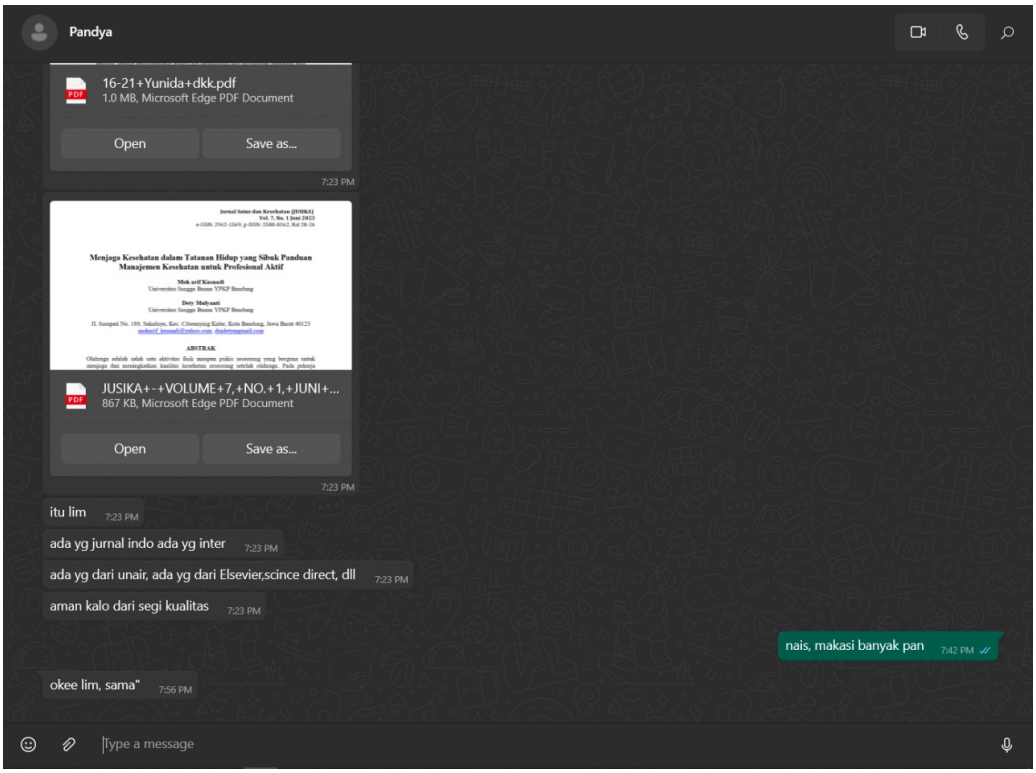
References

[1] S. Rollo, O. Antsygina, and M. S. Tremblay, "The whole day matters: understanding 24-hour movement guideline adherence and relationships with health indicators across the lifespan," *Journal of sport and health science*, vol. 9, no. 6, pp. 493–510, 2020.

[2] S. Sulistiarini, "Hubungan perilaku hidup sehat dengan status kesehatan pada masyarakat kelurahan ujung," *Jurnal Promkes*, vol. 6, no. 1, p. 12, 2018.

[3] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, "An iot-aware architecture for smart healthcare systems," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515–526, 2015.

Credit:
Jurnal yang digunakan sebagai referensi adalah jurnal yang diberikan dari teman saya yang bernama Pandya dari Universitas Lampung Jurusan Kedokteran.



Gambar 2: Sumber Jurnal

Link LLM/ChatGPT/Claude:
Penjelasan GUI.py
Penjelasan respirasi.py
GPT