Tugas Implementasi rPPG (Real-time Heart Rate Detection)

Nama: Cindy Nadila Putri
Mata Kuliah: Sistem Teknologi Multimedia

Alur:

1. **Deteksi Wajah:** Menggunakan MediaPipe Face Detection.
2. **ROI Selection:** Memilih area dahi secara otomatis.
3. **Signal Extraction:** Mengambil rata-rata kanal Hijau (Green Channel).
4. **Filtering & BPM:** Menggunakan Bandpass Filter dan Peak Detection untuk estimasi detak jantung secara real-time.

In [1]:
```
import cv2
import numpy as np
import mediapipe as mp
from scipy.signal import butter, filtfilt, find_peaks
import time

print(f"OpenCV Version: {cv2.__version__}")
print(f"MediaPipe Version: {mp.__version__}")
```

```
OpenCV Version: 4.11.0
MediaPipe Version: 0.10.21
```

Pemrosesan Sinyal

Bagian ini berisi fungsi untuk:

1. **Normalisasi:** Agar sinyal memiliki mean 0.
2. **Bandpass Filter:** Mengambil frekuensi detak jantung manusia (0.9 Hz - 3.0 Hz / setara 54 - 180 BPM).
3. **Calculate Heart Rate:** Menghitung BPM berdasarkan puncak sinyal.

In [2]:
```
def normalize_signal(signal):
    """Normalize signal to zero mean and unit variance."""
    signal = np.array(signal)
    if np.std(signal) == 0:
        return signal - np.mean(signal)
    return (signal - np.mean(signal)) / (np.std(signal) + 1e-8)

def bandpass_filter_rppg(data, fs=30, low=0.9, high=3.0, order=3):
    """
    Bandpass filter for rPPG signal.
    Range diperluas sedikit ke 3.0 Hz (180 BPM) untuk cover range lebih luas.
    """
    if len(data) < 15: # Safety check jika data terlalu sedikit
        return data
```

```
    nyquist = 0.5 * fs
    low = low / nyquist
    high = high / nyquist

    # Pastikan batas filter valid
    if low <= 0 or high >= 1:
        return data

    b, a = butter(order, [low, high], btype='band')
    return filtfilt(b, a, data)

def calculate_heart_rate(signal, fs=30, prominence=0.5):
    """
    Calculate heart rate (BPM) from filtered rPPG signal using Peak Detection.
    """
    signal = normalize_signal(signal)
    # Cari puncak (peaks)
    peaks, _ = find_peaks(signal, prominence=prominence)

    if len(peaks) < 2:
        return 0, peaks

    # Hitung rata-rata jarak antar puncak
    diffs = np.diff(peaks)
    mean_diff = np.mean(diffs) # dalam satuan frames

    # Konversi ke detik lalu ke BPM
    bpm = 60 * (fs / mean_diff)
    return bpm, peaks
```

Fungsi normalize_signal digunakan untuk menstandarkan sinyal dengan membuat nilai rata-ratanya menjadi 0 dan standar deviasinya 1. Tujuannya agar perbedaan pencahayaan atau intensitas warna tidak memengaruhi proses analisis. Selanjutnya, bandpass_filter_rppg berperan sebagai penyaring utama yang menggunakan Butterworth filter untuk membuang noise dan hanya mempertahankan frekuensi antara 0.9 hingga 3.0 Hz—rentang yang sesuai dengan detak jantung manusia (sekitar 54–180 BPM). Terakhir, calculate_heart_rate memproses sinyal yang sudah bersih tersebut dengan mencari puncak-puncak gelombang menggunakan algoritma find_peaks, kemudian menghitung jarak antar puncak untuk mendapatkan nilai detak jantung dalam satuan BPM.

Class RPPGProcessor

Kelas utama yang menangani:

- Inisialisasi MediaPipe.
- Ekstraksi ROI (Region of Interest) pada dahi.
- Pengumpulan buffer sinyal RGB.

In [3]:
```
class RPPGProcessor:
    def __init__(self, fps=30, buffer_size=300):
        self.fps = fps
        self.buffer_size = buffer_size
```

```
        # Buffer Sinyal
        self.g = [] # Kita fokus ke Green channel untuk tugas ini

        # Variabel Hasil
        self.filtered_signal = []
        self.bpm = 0

        # Inisialisasi MediaPipe
        self.mp_face_detection = mp.solutions.face_detection
        self.face_detection = self.mp_face_detection.FaceDetection(
            model_selection=0, # 0 untuk jarak dekat (webcam laptop)
            min_detection_confidence=0.5
        )
        self.last_forehead_rect = None

    def extract_signal(self, frame):
        """Proses frame: Deteksi Wajah -> Ambil ROI -> Ambil rata-rata Green"""
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.face_detection.process(frame_rgb)

        h, w, _ = frame.shape
        g_mean = 0

        if results.detections:
            detection = results.detections[0]
            bbox = detection.location_data.relative_bounding_box

            x = int(bbox.xmin * w)
            y = int(bbox.ymin * h)
            width = int(bbox.width * w)
            height = int(bbox.height * h)

            # --- ROI Jidat ---
            fx1 = x + int(0.25 * width)
            fx2 = x + int(0.75 * width)
            fy1 = y + int(0.05 * height)
            fy2 = y + int(0.20 * height)

            # Clamp agar tidak error out of bound
            fx1, fx2 = max(0, fx1), min(w, fx2)
            fy1, fy2 = max(0, fy1), min(h, fy2)

            self.last_forehead_rect = (fx1, fy1, fx2, fy2)

            # Ambil rata-rata kanal Hijau (Index 1)
            roi = frame[fy1:fy2, fx1:fx2]
            if roi.size > 0:
                g_mean = np.mean(roi[:, :, 1])
        else:
            self.last_forehead_rect = None
            # Jika wajah hilang, masukkan nilai terakhir atau 0
            if self.g: g_mean = self.g[-1]

        # Simpan ke buffer
        self.g.append(g_mean)

        # Jaga ukuran buffer (Sliding Window)
        if len(self.g) > self.buffer_size:
            self.g.pop(0)
```

```
    def update_bpm(self):
        """Hitung BPM berdasarkan data di buffer saat ini"""
        if len(self.g) < self.fps * 3: # Butuh minimal 3 detik data
            return 0

        # Ambil data buffer
        raw_signal = np.array(self.g)

        # 1. Detrending simple (mengurangi rata-rata)
        detrended = signal.detrend(raw_signal) if 'signal' in globals() else (ra

        # 2. Bandpass Filter
        self.filtered_signal = bandpass_filter_rppg(detrended, fs=self.fps)

        # 3. Hitung BPM
        bpm, _ = calculate_heart_rate(self.filtered_signal, fs=self.fps)

        # Smoothing BPM agar tidak lompat-lompat
        if 40 < bpm < 200: # Validasi range manusia
            if self.bpm == 0:
                self.bpm = bpm
            else:
                self.bpm = 0.9 * self.bpm + 0.1 * bpm # Alpha smoothing

        return self.bpm

    def draw_overlay(self, frame):
        """
        Visualisasi Pengganti: Menggambar kotak ROI dan Grafik Sinyal
        langsung di frame video menggunakan OpenCV.
        """
        # 1. Gambar Kotak ROI
        if self.last_forehead_rect:
            x1, y1, x2, y2 = self.last_forehead_rect
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, "ROI", (x1, y1-5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,

        # 2. Gambar Grafik Sinyal di Pojok Bawah
        if len(self.filtered_signal) > 50:
            h, w, _ = frame.shape
            graph_w = 200
            graph_h = 100
            x_start = 20
            y_start = h - 20

            # Background Hitam Transparan
            overlay = frame.copy()
            cv2.rectangle(overlay, (x_start, y_start - graph_h), (x_start + grap
            cv2.addWeighted(overlay, 0.6, frame, 0.4, 0, frame)

            # Normalisasi sinyal agar muat di kotak grafik
            sig = normalize_signal(self.filtered_signal[-100:]) # Ambil 100 data

            # Gambar garis
            points = []
            for i, val in enumerate(sig):
                px = int(x_start + (i / 100) * graph_w)
                # Map nilai -2..2 ke tinggi grafik
                normalized_y = (val + 2) / 4
                py = int(y_start - (normalized_y * graph_h))
```

```
                py = max(y_start - graph_h, min(y_start, py)) # Clamp
                points.append((px, py))

            if len(points) > 1:
                cv2.polylines(frame, [np.array(points)], False, (0, 255, 255), 2
```

Pada metode init, dilakukan beberapa inisialisasi penting. Komponen face_detection digunakan untuk memulai model deteksi wajah dari MediaPipe, dengan parameter model_selection=0 yang memang dirancang khusus untuk mendeteksi wajah dari jarak dekat, seperti pada penggunaan webcam laptop. Selain itu, dibuat buffer data (self.g) berupa list kosong yang nantinya akan menyimpan nilai rata-rata warna hijau dari setiap frame video. Buffer ini berfungsi sebagai memori sementara atau sliding window untuk membentuk sinyal rPPG dari perubahan warna kulit.

Metode extract_signal merupakan tahap paling penting dalam proses rPPG. Di sini, sistem terlebih dahulu mendeteksi posisi wajah pada setiap frame video. Setelah itu, dipilih Region of Interest (ROI) yang fokus pada area dahi, karena bagian ini memiliki kulit yang tipis, kaya pembuluh darah, dan relatif bebas dari gangguan gerakan seperti mulut atau mata. Dari area dahi tersebut, sistem kemudian melakukan ekstraksi kanal hijau dengan menghitung rata-rata nilai piksel warna hijau. Kanal hijau dipilih karena cahaya hijau paling sensitif terhadap hemoglobin: ketika jantung memompa darah, penyerapan cahaya hijau meningkat, dan ketika aliran darah menurun, penyerapan ikut turun. Pola perubahan inilah yang membentuk sinyal rPPG yang digunakan untuk memperkirakan detak jantung.

Metode update_bpm dijalankan pada setiap frame untuk memperbarui nilai detak jantung. Pertama, dilakukan validasi data untuk memastikan buffer sinyal sudah berisi cukup informasi, setidaknya sekitar tiga detik, sebelum perhitungan dimulai. Setelah itu, sinyal menjalani proses detrending untuk mengurangi pengaruh perubahan pencahayaan yang lambat sehingga sinyal kembali berpusat di sekitar nol. Langkah berikutnya adalah filtering dan perhitungan menggunakan fungsi utilitas yang telah dibangun sebelumnya. Karena hasil BPM mentah biasanya masih berfluktuasi, diterapkan proses smoothing menggunakan weighted average (0.9 × BPM lama + 0.1 × BPM baru) agar nilai detak jantung yang tampil di layar berubah lebih stabil dan tidak melonjak-lonjak.

Sistem ini menggambarkan grafik sinyal detak jantung secara real-time di pojok tampilan video. Seratus data sinyal terakhir dinormalisasi agar pas di dalam kotak grafik dengan tinggi 100 piksel. Garis sinyal kemudian digambar menggunakan cv2.polylines. Selain itu, sebuah kotak hijau juga ditampilkan pada area dahi untuk menandai region yang sedang diproses.

Eksekusi Program (Main Loop)

Jalankan cell di bawah ini untuk memulai kamera.

- **Buffer:** 300 frame (sekitar 10 detik).
- **FPS:** 30 FPS.

- **Tekan 'q'** pada keyboard untuk berhenti.

In [4]:
```
import scipy.signal as signal

def main():
    # Inisialisasi Processor
    processor = RPPGProcessor(fps=30, buffer_size=300)

    cap = cv2.VideoCapture(0)

    # Cek kamera
    if not cap.isOpened():
        print("Error: Kamera tidak terdeteksi.")
        return

    print("Mulai program rPPG...")
    print("Tekan 'q' di window kamera untuk keluar.")

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Mirror frame agar enak dilihat
        frame = cv2.flip(frame, 1)

        # 1. Ekstraksi Sinyal
        processor.extract_signal(frame)

        # 2. Update Perhitungan BPM (setiap frame)
        current_bpm = processor.update_bpm()

        # 3. Visualisasi (Overlay Grafik & Teks)
        processor.draw_overlay(frame)

        # Tampilkan BPM
        cv2.putText(frame, f"Heart Rate: {int(current_bpm)} BPM", (20, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        cv2.imshow("Tugas rPPG - Real Time", frame)

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

    cap.release()
    cv2.destroyAllWindows()

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

```
Mulai program rPPG...
Tekan 'q' di window kamera untuk keluar.
```

Dalam pengerjaan tugas ini, saya menggunakan tools Gemini AI untuk membantu refactor code dari tugas sebelumnya. Berikut link percakapan terlampir: https://gemini.google.com/share/bd1b66603cfe