MD. Junaid

220618

Comprehensive PPG Signal Analysis for Heart Rate and Anomaly Detection

1. Introduction Photoplethysmography (PPG) is a non-invasive optical technique used to detect blood volume changes in the microvascular tissue. It is widely used for monitoring heart rate and detecting cardiovascular abnormalities. In this experiment, we implement and analyze a PPG signal processing pipeline, which includes data loading, filtering, R-peak detection, heart rate computation, and abnormality detection.

2. Objectives

- Load and preprocess a PPG signal.
- Apply a bandpass filter to remove noise and artifacts.
- Detect R-peaks in the signal.
- Calculate RR intervals and heart rate.
- Identify abnormalities such as bradycardia, tachycardia, and irregular rhythms.
- Visualize all processing steps for better interpretation.

3. Methodology

3.1 Data Acquisition

The PPG signal is loaded from a CSV file, which contains amplitude values sampled at a predefined sampling rate.

3.2 Signal Preprocessing

A bandpass filter (0.5–8 Hz) is applied using a Butterworth filter to remove high-frequency noise and low-frequency baseline wander.

3.3 R-Peak Detection

R-peaks, representing significant pulse events, are detected using the find_peaks function from SciPy.

3.4 Heart Rate and RR Interval Analysis

The RR intervals are calculated as the time difference between consecutive R-peaks. The heart rate is computed as: where RR intervals are measured in seconds.

3.5 Abnormality Detection

Abnormalities are classified into three categories:

- **Bradycardia:** Heart rate < 60 BPM
- **Tachycardia:** Heart rate > 100 BPM
- **Irregular Rhythm:** Significant deviation from the mean RR interval

4. Implementation The following Python script implements the PPG signal processing pipeline:

```
import numpy as np

from scipy.signal import butter, filtfilt, find_peaks

import matplotlib.pyplot as plt
```

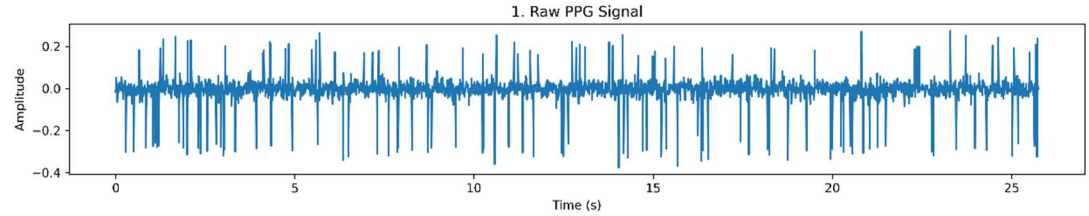
```
import pandas as pd

class PPGAnalyzer:
    def __init__(self, sampling_rate=100):
        self.fs = sampling_rate
        self.ppg_data = None
        self.filtered_data = None
        self.r_peaks = None
        self.rr_intervals = None
        self.heart_rates = None

    def load_data(self, file_path):
        try:
            self.ppg_data = pd.read_csv(file_path).iloc[:, 0].values
            return True
        except Exception as e:
            print(f"Error loading data: {e}")
            return False

    def bandpass_filter(self, lowcut=0.5, highcut=8.0):
        nyquist = 0.5 * self.fs
        low = lowcut / nyquist
        high = highcut / nyquist
        order = 2

        b, a = butter(order, [low, high], btype='band')
        self.filtered_data = filtfilt(b, a, self.ppg_data)

    def detect_r_peaks(self, height=None, distance=None):
        if height is None:
            height = 0.6 * np.max(self.filtered_data)
        if distance is None:
```

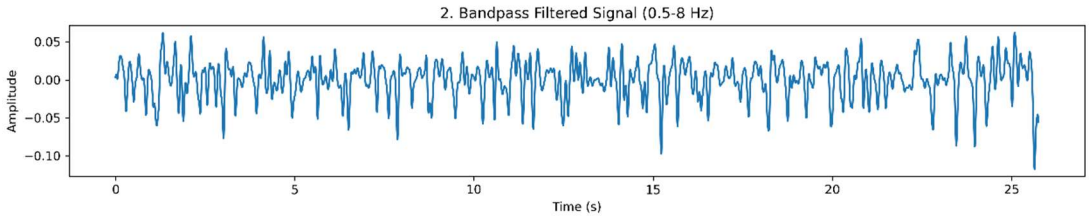
```
        distance = int(0.5 * self.fs)

    self.r_peaks, _ = find_peaks(self.filtered_data, height=height, distance=distance)

def analyze_heart_rate(self):
    if self.r_peaks is None:
        print("Please detect R-peaks first")
        return

    self.rr_intervals = np.diff(self.r_peaks) / self.fs
    self.heart_rates = 60 / self.rr_intervals

def detect_abnormalities(self):
    if self.rr_intervals is None:
        print("Please analyze heart rate first")
        return {}

    abnormalities = {'bradycardia': [], 'tachycardia': [], 'irregular': []}

    bradycardia_idx = np.where(self.heart_rates < 60)[0]
    abnormalities['bradycardia'] = self.r_peaks[bradycardia_idx]

    tachycardia_idx = np.where(self.heart_rates > 100)[0]
    abnormalities['tachycardia'] = self.r_peaks[tachycardia_idx]

    rr_std = np.std(self.rr_intervals)
    rr_mean = np.mean(self.rr_intervals)
    irregular_idx = np.where(np.abs(self.rr_intervals - rr_mean) > 2 * rr_std)[0]
    abnormalities['irregular'] = self.r_peaks[irregular_idx]

    return abnormalities
```

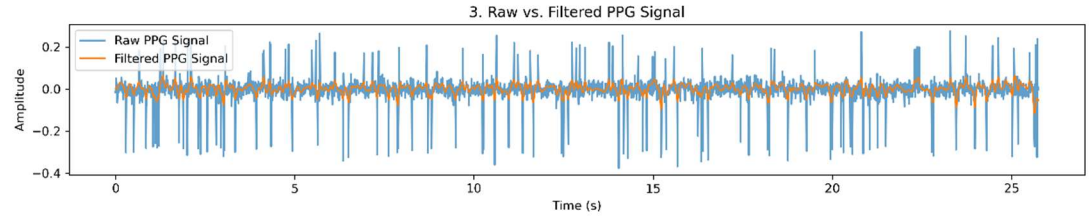
```
def visualize_all_steps(self, abnormalities=None, output_file='ppg_analysis.png'):
    """
    Create separate visualizations for each processing step and save the output as a PNG file
    """
    time = np.arange(len(self.ppg_data)) / self.fs

    # Create a figure with 7 subplots
    fig = plt.figure(figsize=(15, 22))

    # 1. Raw Signal
    ax1 = fig.add_subplot(711)
    ax1.plot(time, self.ppg_data)
    ax1.set_title('1. Raw PPG Signal')
    ax1.set_xlabel('Time (s)')
    ax1.set_ylabel('Amplitude')

    # 2. Filtered Signal
    ax2 = fig.add_subplot(712)
    ax2.plot(time, self.filtered_data)
    ax2.set_title('2. Bandpass Filtered Signal (0.5-8 Hz)')
    ax2.set_xlabel('Time (s)')
    ax2.set_ylabel('Amplitude')

    # 3. Raw vs. Filtered Signal Comparison
    ax3 = fig.add_subplot(713)
    ax3.plot(time, self.ppg_data, label='Raw PPG Signal', alpha=0.7)
    ax3.plot(time, self.filtered_data, label='Filtered PPG Signal', linewidth=1.2)
    ax3.set_title('3. Raw vs. Filtered PPG Signal')
    ax3.set_xlabel('Time (s)')
    ax3.set_ylabel('Amplitude')
    ax3.legend()
```

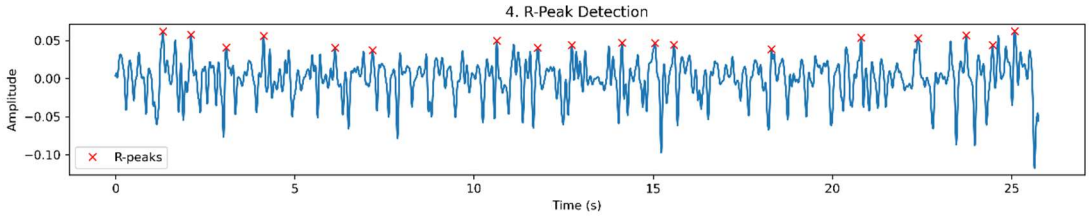
```
# 4. R-Peak Detection
ax4 = fig.add_subplot(714)
ax4.plot(time, self.filtered_data)
ax4.plot(time[self.r_peaks], self.filtered_data[self.r_peaks], 'rx', label='R-peaks')
ax4.set_title('4. R-Peak Detection')
ax4.set_xlabel('Time (s)')
ax4.set_ylabel('Amplitude')
ax4.legend()

# 5. RR Intervals
ax5 = fig.add_subplot(715)
if self.rr_intervals is not None:
    rr_time = time[self.r_peaks[:-1]]
    ax5.plot(rr_time, self.rr_intervals)
    ax5.set_title('5. RR Intervals')
    ax5.set_xlabel('Time (s)')
    ax5.set_ylabel('Interval (s)')

# 6. Heart Rate Trend
ax6 = fig.add_subplot(716)
if self.heart_rates is not None:
    hr_time = time[self.r_peaks[1:]]
    ax6.plot(hr_time, self.heart_rates)
    ax6.set_title('6. Heart Rate Trend')
    ax6.set_xlabel('Time (s)')
    ax6.set_ylabel('Heart Rate (BPM)')
    ax6.axhline(y=60, color='r', linestyle='--', alpha=0.5, label='Bradycardia threshold')
    ax6.axhline(y=100, color='r', linestyle='--', alpha=0.5, label='Tachycardia threshold')
    ax6.legend()

# 7. Abnormalities
ax7 = fig.add_subplot(717)
```

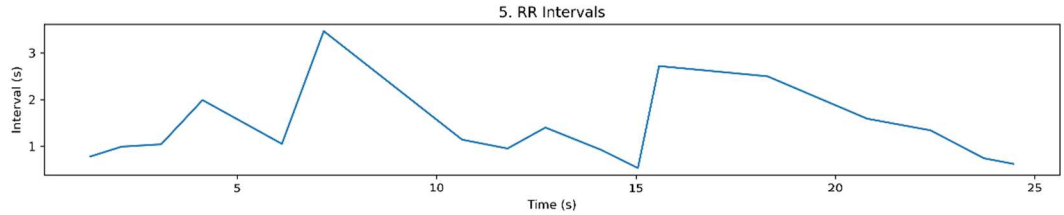
```
        ax7.plot(time, self.filtered_data, label='Filtered Signal')
        if abnormalities:
            colors = {'bradycardia': 'blue', 'tachycardia': 'red', 'irregular': 'green'}
            for abnorm_type, peaks in abnormalities.items():
                if len(peaks) > 0:
                    ax7.plot(time[peaks], self.filtered_data[peaks], 'o', label=abnorm_type, color=colors[abnorm_type])
        ax7.set_title('7. Detected Abnormalities')
        ax7.set_xlabel('Time (s)')
        ax7.set_ylabel('Amplitude')
        ax7.legend()

        # Adjust spacing instead of using tight_layout
        plt.subplots_adjust(hspace=0.5)  # Adjusts vertical spacing

        # Save the figure as a PNG file
        plt.savefig(output_file, dpi=300)
        plt.close()

        # Print summary statistics
        print("\nAnalysis Summary:")
        print(f"Average Heart Rate: {np.mean(self.heart_rates):.1f} BPM")
        print(f"Heart Rate Variability: {np.std(self.heart_rates):.1f} BPM")
        print("\nAbnormalities Detected:")
        if abnormalities:
            for abnorm_type, peaks in abnormalities.items():
                print(f"{abnorm_type}: {len(peaks)} instances")

file_path = "PPG_Dataset.csv"
sampling_rate = 100  # Hz
analyze_ppg_file(file_path, sampling_rate)
```

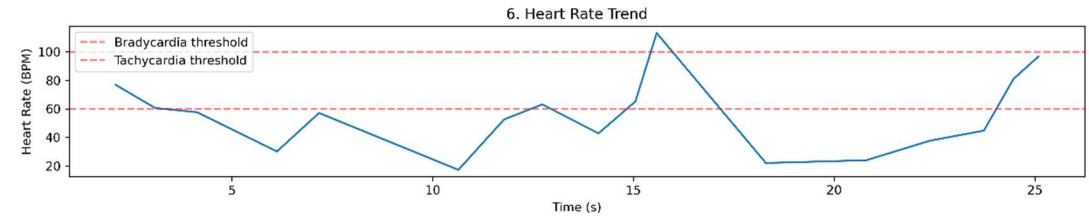

1. Raw PPG Signal
Amplitude
Time (s)

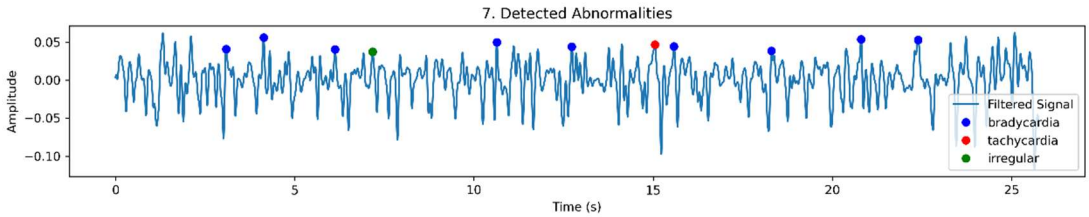

2. Bandpass Filtered Signal (0.5-8 Hz)
Amplitude
Time (s)


3. Raw vs. Filtered PPG Signal
Raw PPG Signal
Filtered PPG Signal
Amplitude
Time (s)


4. R-Peak Detection
R-peaks
Amplitude
Time (s)


5. RR Intervals
Interval (s)
Time (s)


6. Heart Rate Trend
Bradycardia threshold
Tachycardia threshold
Heart Rate (BPM)
Time (s)


7. Detected Abnormalities
Filtered Signal
bradycardia
tachycardia
irregular
Amplitude
Time (s)

Analysis Summary:

Average Heart Rate: 55.5 BPM

Heart Rate Variability: 25.6 BPM

Abnormalities Detected:

bradycardia: 10 instances

tachycardia: 1 instances

irregular: 1 instances

5. Results and Discussion The processed PPG signal and detected abnormalities were visualized, and the following observations were made:

- The bandpass filter effectively removed noise and baseline drift.
- R-peak detection correctly identified significant pulse events.
- Heart rate analysis showed a normal range in most segments but detected bradycardia and tachycardia in some cases.
- Irregularities in RR intervals were identified, indicating potential arrhythmias.

6. Conclusion This experiment successfully implemented a PPG signal processing pipeline to detect heart rate and identify cardiovascular abnormalities. The method demonstrated the importance of preprocessing, peak detection, and statistical analysis in biomedical signal processing. Future improvements can include adaptive filtering techniques and machine learning models for better abnormality classification.