**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)

A screenshot of a graph

AI-generated content may be incorrect.

***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.