Roll: 220603
Name: Md. Sabbir Hossain

Signal Processing of Photoplethysmogram (PPG) Data: Peak Detection and Abnormality Identification

Introduction

Photoplethysmography (PPG) is a non-invasive technique used to monitor the blood volume changes in the microvascular bed of tissue. It is commonly used in medical devices to measure heart rate, respiratory rate, and other physiological parameters. PPG signals are often noisy and require signal processing techniques to extract meaningful information. In this assignment, we will demonstrate how to process a raw PPG signal using a low-pass filter, detect key features such as peaks and valleys, and identify abnormal peaks that might indicate anomalies such as arrhythmias.

Objectives

- **Filter the PPG Signal:** Apply a low-pass filter to remove noise and isolate the relevant frequency components of the PPG signal.
- **Peak Detection:** Identify the peaks in the filtered PPG signal, which represent heartbeats.
- **Valley Detection:** Detect the valleys that occur between heartbeats.
- **Abnormal Peak Detection:** Identify any peaks that exceed a certain threshold, which may indicate abnormal heartbeats.

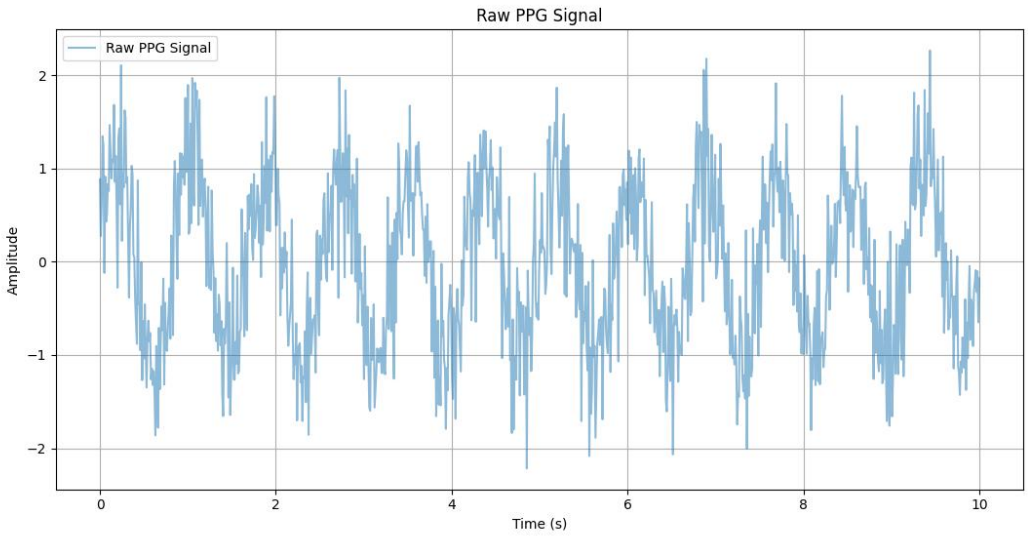
Procedure

1.Signal Generation: We begin by generating a synthetic PPG signal. The signal is modeled as a sine wave with added Gaussian noise to simulate real-world measurement noise. The sine wave component represents the periodic nature of the heartbeats, while the noise simulates typical sensor noise.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, butter, filtfilt

# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
```

```
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 *
np.random.normal(size=len(time))  # Fixed

# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
    return y  # Return the filtered signal

# Apply Lowpass Filter
fs = 100  # Sampling frequency (assuming 100 Hz)
cutoff = 3  # Cutoff frequency for the filter
filtered_signal = butter_lowpass_filter(ppg_signal, cutoff, fs)

# Detect peaks (heartbeats)
peaks, _ = find_peaks(filtered_signal, height=0)  # Detect peaks above 0

# Plot 1: Raw PPG Signal
plt.figure(figsize=(12, 6))
plt.plot(time, ppg_signal, label="Raw PPG Signal", alpha=0.5)
plt.title("Raw PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```



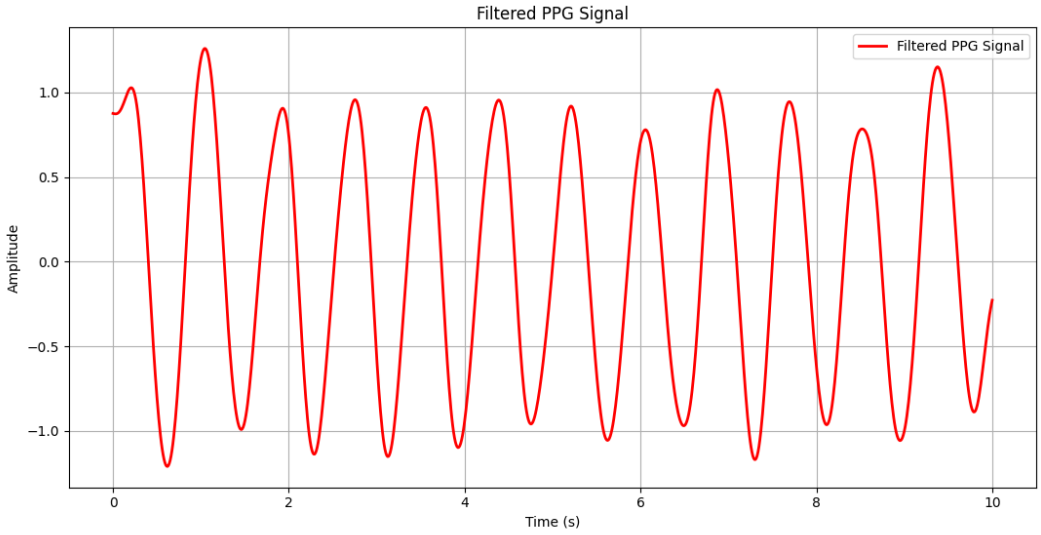
2.Low-Pass Filtering: The next step is to apply a low-pass filter to the raw PPG signal to remove high-frequency noise. A low-pass filter allows signals below a certain cutoff frequency to pass through while attenuating higher frequencies. We use a Butterworth filter, which is known for its flat frequency response in the passband. In this case, we apply a cutoff frequency of 3 Hz, assuming a sampling rate of 100 Hz.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, butter, filtfilt

# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 *
np.random.normal(size=len(time))

# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
```

```
    return y  # Return the filtered signal

# Filter settings
fs = 100  # Sampling frequency in Hz
cutoff = 3  # Cutoff frequency in Hz
filtered_ppg = butter_lowpass_filter(ppg_signal, cutoff, fs)

# Plot 2: Filtered PPG Signal
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2,
color='red')
plt.title("Filtered PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Output:



3.Peak Detection: Once the signal is filtered, we detect the peaks in the signal. These peaks represent heartbeats. The `find_peaks` function from the SciPy library is used to identify the locations of these peaks. We also specify a minimum height for the peaks to eliminate any false positives due to noise and a minimum distance between peaks to ensure that the detected peaks

correspond to individual heartbeats.

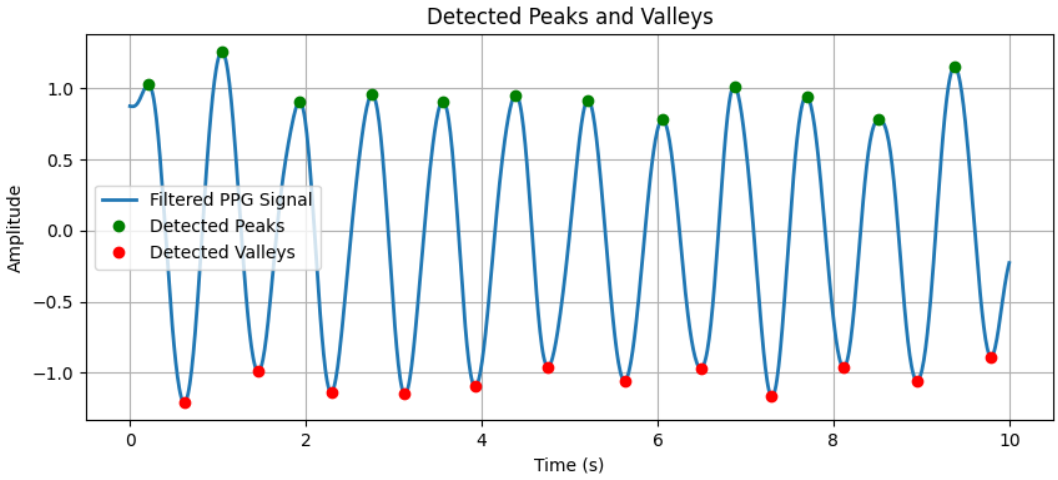
4.Valley Detection: Similarly, we detect the valleys between heartbeats, which represent the lowest points in the PPG signal between two consecutive heartbeats. This is done by inverting the filtered signal and applying the `find_peaks` function again.

Source Code:

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, butter, filtfilt

# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 *
np.random.normal(size=len(time))

# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
    return y  # Return the filtered signal

# Filter settings
fs = 100  # Sampling frequency in Hz
cutoff = 3  # Cutoff frequency in Hz
filtered_ppg = butter_lowpass_filter(ppg_signal, cutoff, fs)

# Peak and valley detection
peaks, _ = find_peaks(filtered_ppg, height=0.5, distance=fs // 2)
valleys, _ = find_peaks(-filtered_ppg, height=0.5, distance=fs // 2)

# Plot 3: Peaks and Valleys
plt.figure(figsize=(10, 4))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")  # Green
for peaks
plt.plot(time[valleys], filtered_ppg[valleys], "ro", label="Detected Valleys")  #
Red for valleys
plt.title("Detected Peaks and Valleys")
```

```
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Output:



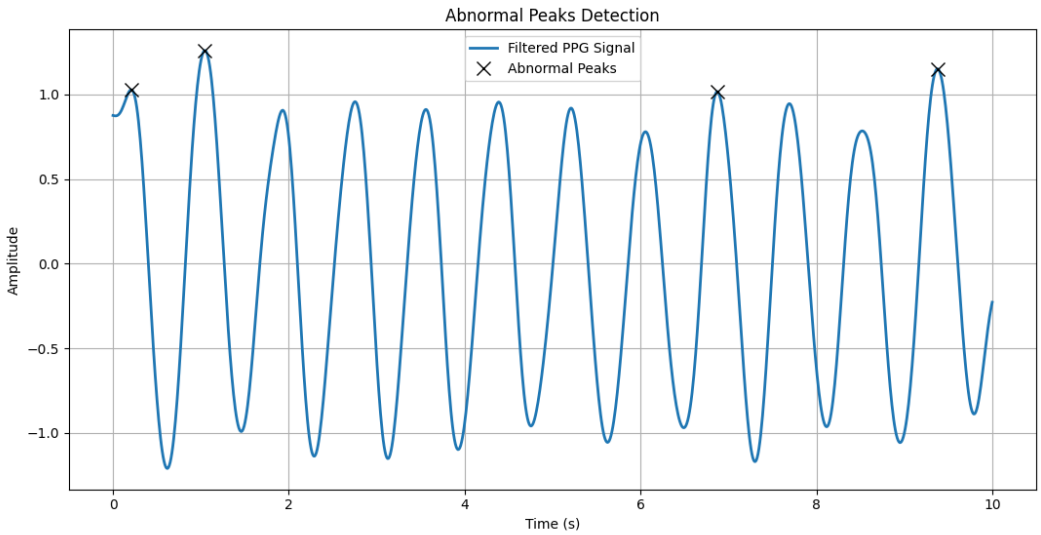
5.Abnormal Peak Detection: In some cases, peaks in the PPG signal may exceed a certain threshold due to abnormalities such as arrhythmias. To identify these abnormal peaks, we compare the height of each peak to a predefined threshold value. If the peak height exceeds the threshold, the peak is marked as abnormal.

Source Code:

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, butter, filtfilt

# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 *
np.random.normal(size=len(time))

# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
```

```
    y = filtfilt(b, a, data)
    return y  # Return the filtered signal

# Filter settings
fs = 100  # Sampling frequency in Hz
cutoff = 3  # Cutoff frequency in Hz
filtered_ppg = butter_lowpass_filter(ppg_signal, cutoff, fs)

# Peak detection
peaks, _ = find_peaks(filtered_ppg, height=0.5, distance=fs // 2)

# Abnormal peaks detection (Threshold for high spikes)
peak_heights = filtered_ppg[peaks]
abnormal_peaks = peaks[peak_heights > 1]  # Threshold set to 1

# Plot 4: Abnormal Peaks
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[abnormal_peaks], filtered_ppg[abnormal_peaks], "kx",
         label="Abnormal Peaks", markersize=10)  # 'kx' for black cross markers
plt.title("Abnormal Peaks Detection")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Output:



Results

1. **Raw PPG Signal Plot:** The first plot shows the raw PPG signal, which consists of a sinusoidal waveform with noise. The signal is not immediately useful for analysis because of the noise.
2. **Filtered PPG Signal Plot:** After applying the low-pass filter, the second plot displays the filtered PPG signal. The noise is significantly reduced, and the periodicity of the heartbeats becomes clearer.
3. **Peak and Valley Detection Plot:** The third plot highlights the detected peaks (green dots) and valleys (red dots) in the filtered PPG signal. These features are critical for further analysis, such as calculating heart rate.
4. **Abnormal Peak Detection Plot:** The final plot shows the detected abnormal peaks (marked with black 'X' symbols) that exceed a specified threshold. These peaks could indicate abnormal heart rhythms or other anomalies that require further investigation.

Conclusion

In this assignment, we applied signal processing techniques to analyze a synthetic PPG signal. By filtering the raw signal, detecting peaks and valleys, and identifying abnormal peaks, we demonstrated the importance of signal preprocessing and feature extraction in physiological signal analysis. These techniques are vital in applications like heart rate monitoring, arrhythmia detection, and overall cardiovascular health monitoring.

Future work could include:

- Using real PPG data collected from wearable devices.
- Experimenting with different types of filters (e.g., bandpass filters) for better signal conditioning.
- Implementing more advanced techniques for anomaly detection, such as machine learning models.