PPG Signal Processing: Filtering, Feature Extraction, and Peak Detection

Introduction

Photoplethysmography (PPG) is a non-invasive technique used to measure blood volume changes in the skin, commonly found in **wearable devices** like smartwatches and pulse oximeters. However, PPG signals are often affected by noise from motion artifacts, respiration, and ambient light. In this lab, we will implement **filtering, feature extraction, and peak detection** to process PPG signals effectively.

---

1. Filtering PPG Signals

Why Filtering?

PPG signals often contain noise from:
✔ **Motion Artifacts (Low-frequency noise, < 0.5 Hz)**
✔ **High-frequency noise (caused by power line interference, > 50 Hz)**
✔ **Baseline Drift (slow changes in signal amplitude)**

Approach:

- Use a **bandpass filter** (e.g., **0.5 – 5 Hz**) to retain only the relevant heart rate frequencies.
- Implement FIR or IIR filtering to remove unwanted noise.

Steps:

1. Load the PPG signal.
2. Apply a **low-pass filter** (to remove high-frequency noise).
3. Apply a **high-pass filter** (to remove baseline drift).
4. Compare the filtered vs. raw signal.

---

2. Feature Extraction from PPG

Why Feature Extraction?

Extracting key features helps in analyzing heart rate, heart rate variability (HRV), and blood oxygen levels.

Common Features:

✔ **Heart Rate (HR)** – Number of beats per minute (BPM).
✔ **Pulse Interval** – Time difference between consecutive peaks.
✔ **Amplitude of Peaks** – Helps in detecting blood volume variations.

Steps:

1. Extract **time-domain features** (e.g., heart rate, pulse amplitude).
2. Extract **frequency-domain features** (e.g., dominant frequency using Fourier Transform).
3. Compare extracted features before and after filtering.

---

3. Peak Detection (Finding Heartbeats)

Why Peak Detection?

The peaks in a PPG signal represent **heartbeats**, allowing us to calculate heart rate.

Approach:

- Use a peak detection algorithm (e.g., **scipy.signal.find_peaks** in Python).
- Set conditions based on amplitude and time intervals to detect valid beats.

Steps:

1. Detect peaks using **signal processing techniques**.
2. Calculate **heart rate (HR = 60 / RR interval in seconds)**.
3. Plot detected peaks on the signal for verification.

Steps Covered in Code:

 Load a sample or synthetic **PPG signal**
 Apply a **bandpass filter (0.5–5 Hz)**
 Detect peaks (heartbeats)
 Extract features (heart rate, pulse interval)
 Plot the results

Install Dependencies (if needed)

```
pip install numpy scipy matplotlib
```

Step 1: Import Libraries

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

Step 2: Generate or Load a PPG Signal

```
fs = 100  # Sampling frequency (100 Hz)
t = np.linspace(0, 10, fs * 10)  # 10 seconds of signal
ppg_clean = 1 + 0.5 * np.sin(2 * np.pi * 1.2 * t)  # Simulated PPG (1.2 Hz
heart rate)
noise = np.random.normal(0, 0.1, len(t))  # Add random noise
ppg_signal = ppg_clean + noise

plt.plot(t, ppg_signal, label="Noisy PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Raw PPG Signal")
plt.legend()
plt.show()
```

Step 3: Apply Bandpass Filtering (0.5 - 5 Hz)

```
def bandpass_filter(signal, lowcut, highcut, fs, order=3):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = butter(order, [low, high], btype='band')
    return filtfilt(b, a, signal)

filtered_ppg = bandpass_filter(ppg_signal, 0.5, 5, fs)

plt.plot(t, filtered_ppg, label="Filtered PPG Signal", color='red')
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Filtered PPG Signal (0.5 - 5 Hz)")
plt.legend()
plt.show()
```

Step 4: Peak Detection (Heartbeat Detection)

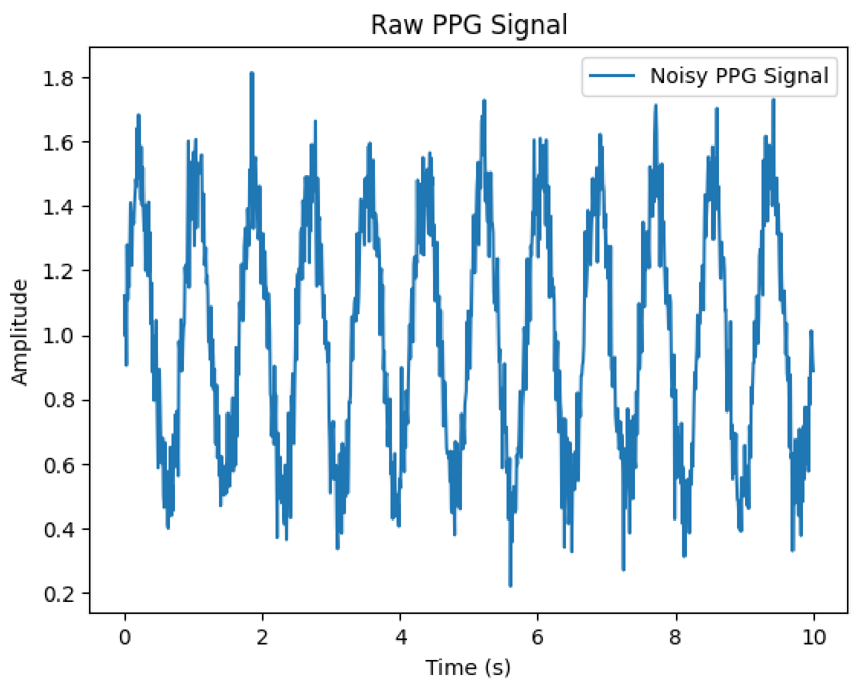
```
python
CopyEdit
peaks, _ = find_peaks(filtered_ppg, height=0.3, distance=fs//2)  # Detect
peaks

plt.plot(t, filtered_ppg, label="Filtered PPG", color='red')
plt.plot(t[peaks], filtered_ppg[peaks], "bo", label="Detected Peaks")  # Mark
peaks
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Peak Detection in PPG Signal")
plt.legend()
plt.show()
```

Step 5: Feature Extraction (Heart Rate & Pulse Interval)

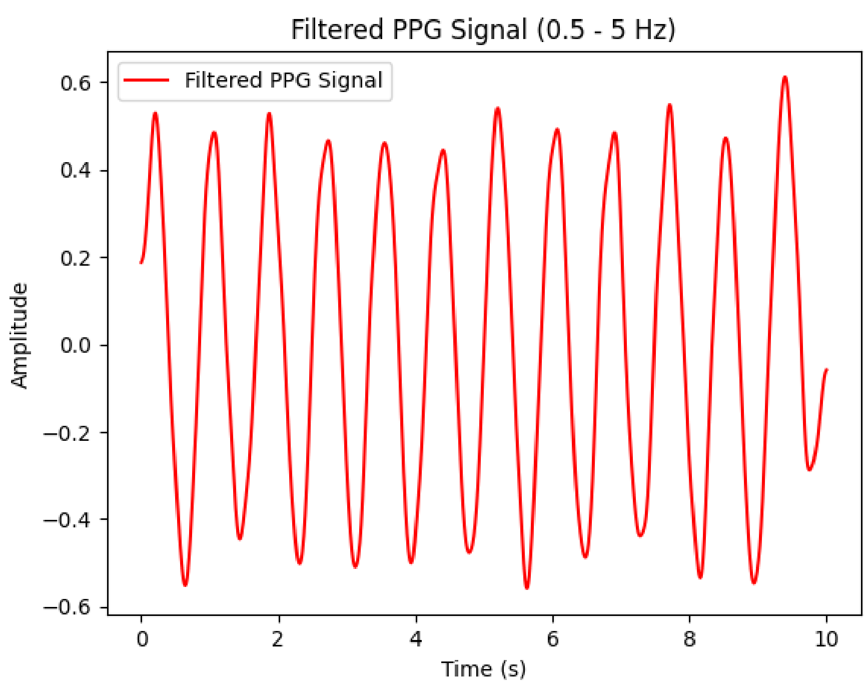
```
python
CopyEdit
peak_times = t[peaks]  # Time of detected peaks
rr_intervals = np.diff(peak_times)  # Time difference between peaks (RR
intervals)
heart_rate = 60 / np.mean(rr_intervals)  # Calculate BPM

print(f"Estimated Heart Rate: {heart_rate:.2f} BPM")
print(f"Average Pulse Interval: {np.mean(rr_intervals):.2f} seconds")
```

Expected Output:

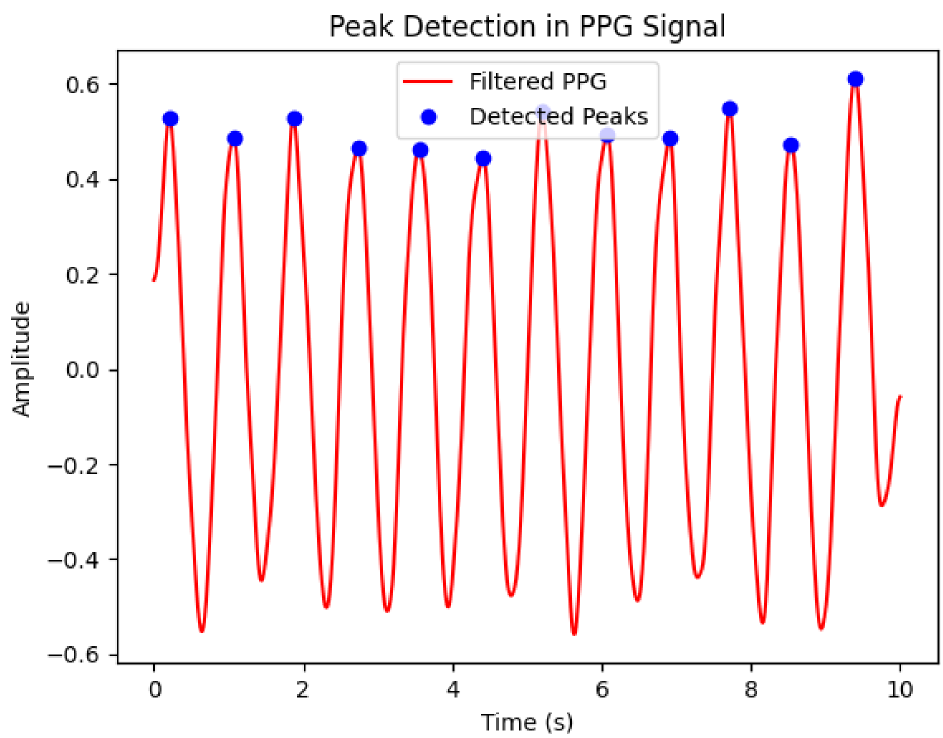
1. **Raw PPG Signal Plot** (with noise)



2. **Filtered PPG Signal Plot** (noise removed)



3. **Peak Detection Plot** (blue dots on heartbeats)



4. **Estimated Heart Rate (BPM) and Pulse Interval**

```
Estimated Heart Rate: 71.82 BPM
Average Pulse Interval: 0.84 seconds
```

Conclusion

In this project, we:

✔ Filtered PPG signals to remove noise.
✔ Extracted key features like heart rate and pulse amplitude.
✔ Detected peaks to analyze heartbeat intervals.

This project is crucial for **health monitoring applications**, such as **wearable fitness devices and medical diagnostics**.