MD. Taz Warul Mulk
Roll: 210631

Feature Extraction of a PPG Signal

❖ **Title:** Feature Extraction and Analysis of Photoplethysmogram (PPG) Signals for Physiological Monitoring

❖ **Objectives:**

1. **To generate and filter a PPG signal** to simulate real-world data and remove noise for further analysis.
2. **To detect peaks and valleys** in the filtered PPG signal, representing systolic and diastolic phases of the heartbeat.
3. **To identify abnormal peaks** that could indicate irregularities in the heart's rhythm.
4. **To compute physiological features** such as heart rate, heart rate variability, and systolic/diastolic ratio, based on the PPG signal.
5. **To assess the signal quality** by calculating the Signal-to-Noise Ratio (SNR) and other statistical measures (skewness and kurtosis).
6. **To evaluate the PPG signal's variability** using metrics like the Peak-to-Peak Interval (PPI) and the Median Absolute Deviation (MAD).
7. **To visualize the extracted features** and understand the relationship between signal characteristics and heart function.

❖ **Theory:**

Photoplethysmogram (PPG) signals are optical measurements of blood volume changes within the microvascular bed of tissue. These signals are commonly used in health monitoring systems to assess various physiological parameters, such as heart rate, heart rate variability, and oxygen saturation. The PPG signal is typically recorded using a photodetector and a light source placed on the skin, usually on the fingertip or earlobe.

Key Concepts:

This project is a comprehensive analysis of a Photoplethysmogram (PPG) signal, which is commonly used in medical and health monitoring to measure heart rate and other cardiovascular metrics. Below is a detailed description of the features and functionalities implemented in the project:

1. Signal Generation and Initial Visualization

- Signal Generation: A synthetic PPG signal is generated using a sine wave with added noise to simulate real-world conditions.
- Raw Signal Plot: The raw PPG signal is plotted to visualize its initial form.

2. Signal Filtering

- Lowpass Filter: A Butterworth lowpass filter is applied to remove high-frequency noise from the PPG signal.
- Filtered Signal Plot: The filtered PPG signal is plotted to show the effect of the filtering process.

3. Peak and Valley Detection

- Peak Detection: Peaks in the filtered PPG signal are detected using the find_peaks function.
- Valley Detection: Valleys (troughs) in the filtered PPG signal are detected by finding peaks in the inverted signal.
- Peaks and Valleys Plot: The detected peaks and valleys are plotted on the filtered signal for visualization.

4. Abnormal Peaks Detection

- Abnormal Peaks: Peaks that exceed a certain height threshold are identified as abnormal peaks.
- Abnormal Peaks Plot: The abnormal peaks are marked on the filtered signal plot.

5. Counting Peaks, Valleys, and Abnormal Peaks

- Counts: The number of detected peaks, valleys, and abnormal peaks are counted and printed.

6. Heart Rate Calculation

- Heart Rate (HR): The heart rate in beats per minute (BPM) is calculated based on the intervals between detected peaks.
- Heart Rate Plot: The heart rate is displayed on a plot showing the filtered signal and detected peaks.

7. Peak-to-Peak Interval (PPI)

- PPI Calculation: The time intervals between consecutive peaks are calculated.
- PPI Plot: The peak-to-peak intervals are annotated on the filtered signal plot.

8. Signal-to-Noise Ratio (SNR)

- SNR Calculation: The SNR is calculated to measure the strength of the PPG signal relative to the noise.
- SNR Plot: The SNR value is displayed on a plot comparing the raw and filtered signals.

9. Heart Rate Variability (HRV)

- HRV Metrics: Standard Deviation of NN Intervals (SDNN) and Root Mean Square of Successive Differences (RMSSD) are calculated to measure HRV.
- HRV Plot: The HRV metrics are displayed on a plot showing the filtered signal and peak intervals.

10. Systolic to Diastolic Ratio (S/D Ratio)

- S/D Ratio Calculation: The ratio of systolic (peak) values to diastolic (valley) values is calculated.
- S/D Ratio Plot: The S/D ratio is displayed on a plot showing the filtered signal, peaks, and valleys.

11. Skewness and Kurtosis

- Skewness and Kurtosis Calculation: These statistical measures are calculated to describe the shape of the filtered PPG signal distribution.
- Histogram Plot: A histogram of the filtered PPG signal is plotted to visualize its distribution.

12. Peak Area (AUC)

- AUC Calculation: The area under the curve (AUC) for the detected peaks is calculated using Simpson's rule.
- AUC Print: The AUC value is printed.

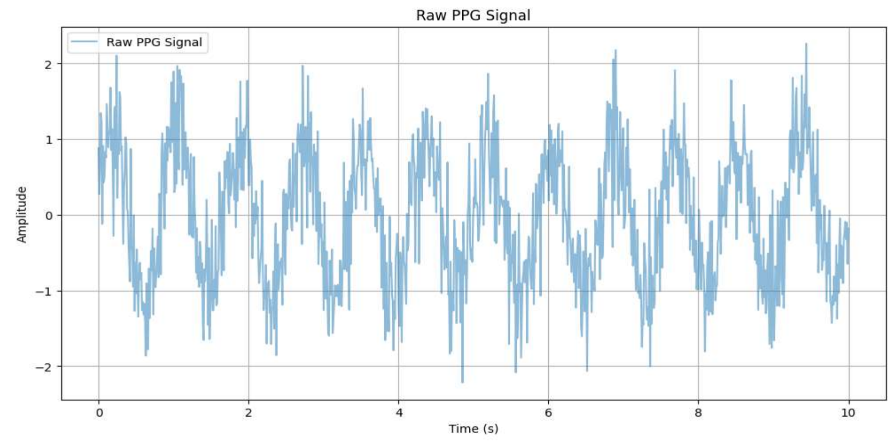
13. Median Absolute Deviation (MAD)

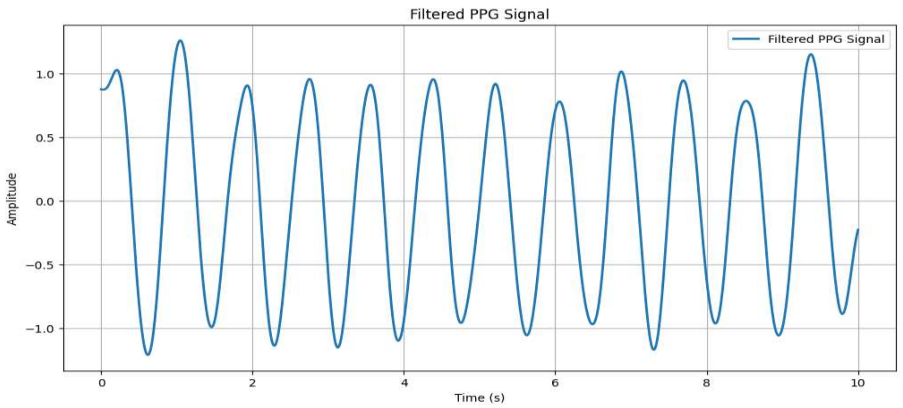
- MAD Calculation: The MAD is calculated to measure the variability of the filtered PPG signal.
- MAD Print: The MAD value is printed.

❖ Code With Visualization:

➢ Signal Generation and Initial Visualization and Signal Filtering

```
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))

# 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()

# 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

# 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)
plt.title("Filtered PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

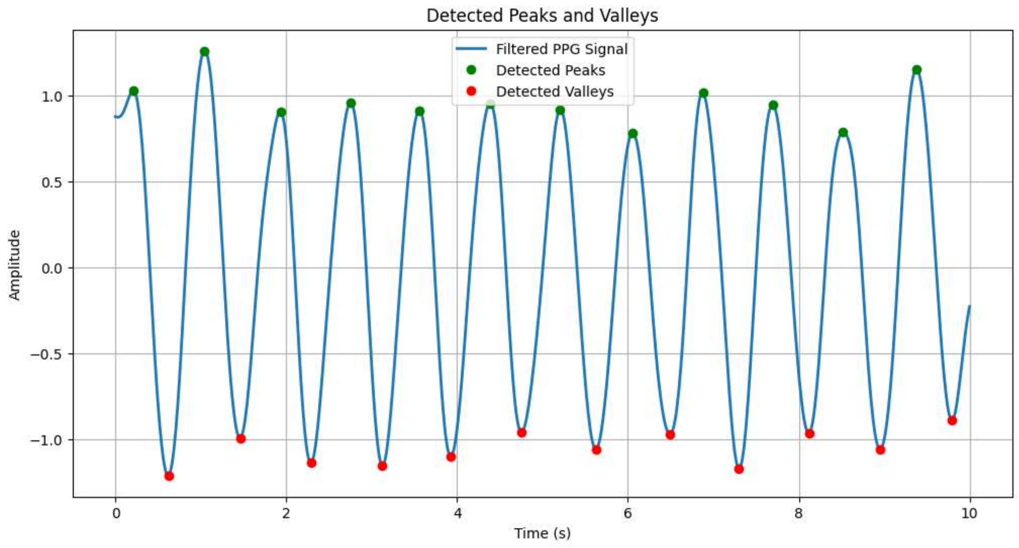


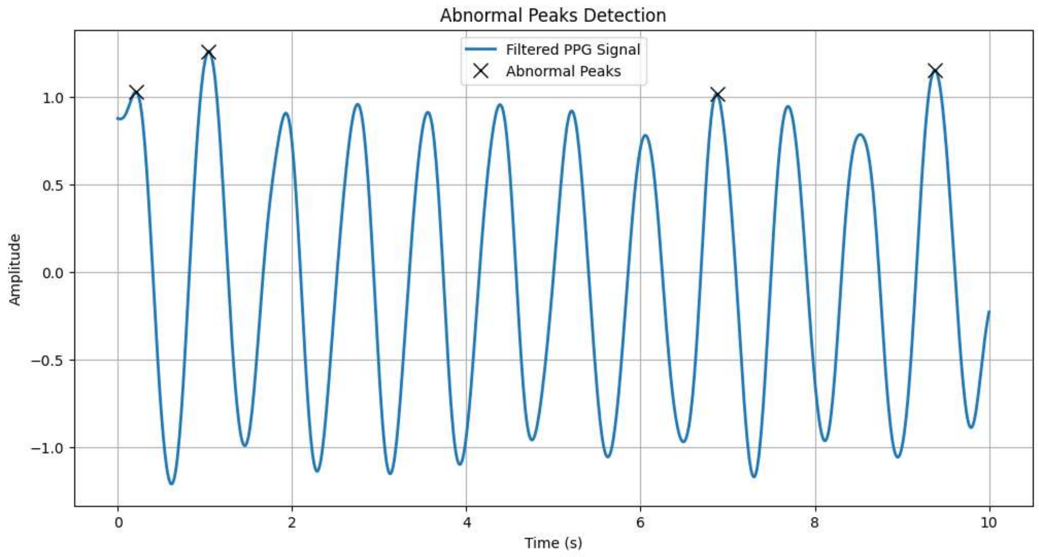


- **Peak, Valley and abnormality detection detection**

```
# 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=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")
plt.plot(time[valleys], filtered_ppg[valleys], "ro", label="Detected Valleys")
plt.title("Detected Peaks and Valleys")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()

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

# 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)
plt.title("Abnormal Peaks Detection")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()

# Count the number of peaks, valleys, and abnormal peaks
num_peaks = len(peaks)
num_valleys = len(valleys)
num_abnormal_peaks = len(abnormal_peaks)
```

```
# Print the counts
print(f"Number of Peaks: {num_peaks}")
print(f"Number of Valleys: {num_valleys}")
print(f"Number of Abnormal Peaks: {num_abnormal_peaks}")
```





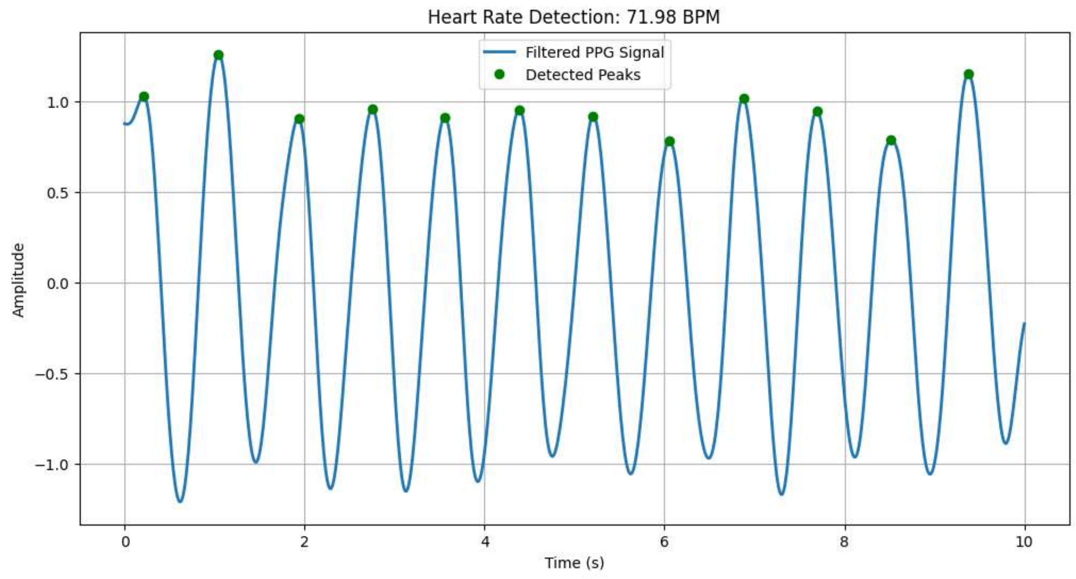
Number of Peaks: 12
Number of Valleys: 12
Number of Abnormal Peaks: 4

- **Heart Rate Calculation**

```
# Feature: Heart Rate (HR in BPM)
import numpy as np
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

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

# Compute Peak-to-Peak Intervals (PPI)
peak_intervals = np.diff(time[peaks])

# Compute Heart Rate (HR)
heart_rate = 60 / np.mean(peak_intervals)

# Print the Heart Rate
print(f"Heart Rate: {heart_rate:.2f} BPM")

plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")
plt.title(f"Heart Rate Detection: {heart_rate:.2f} BPM")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

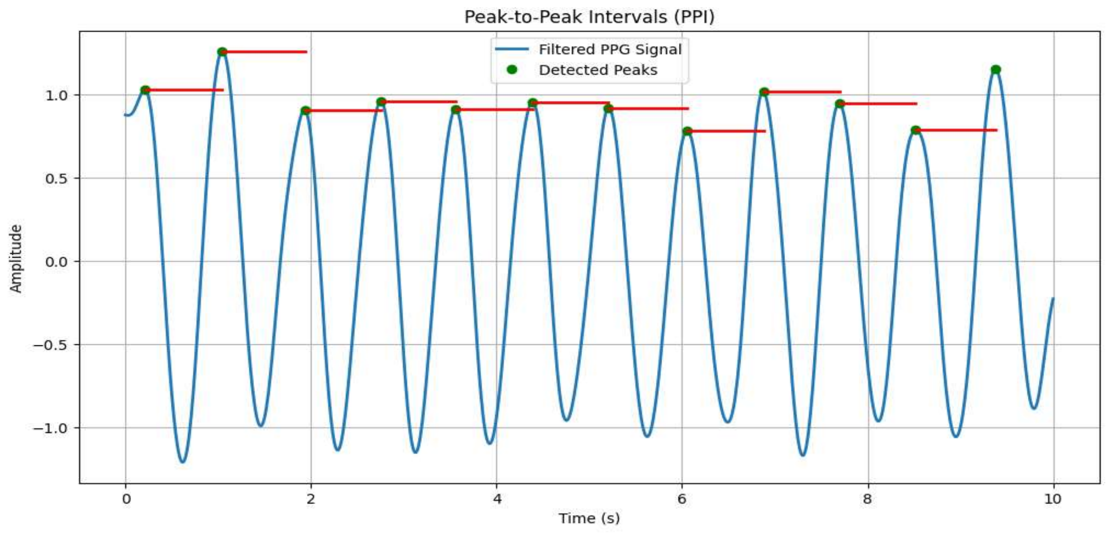
Heart Rate: 71.98 BPM



- **Peak-to-Peak Interval (PPI)**

```
# Feature: Peak-to-Peak Interval (PPI)
# It represents the time difference between consecutive peaks

# Compute Peak-to-Peak Intervals (PPI)
peak_intervals = np.diff(time[peaks])

# Print the Mean Peak-to-Peak Interval
print(f"Mean Peak-to-Peak Interval (PPI): {np.mean(peak_intervals):.3f} sec")
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")

# Annotate peak-to-peak intervals
for i in range(len(peaks) - 1):
    plt.plot([time[peaks[i]], time[peaks[i+1]]],
             [filtered_ppg[peaks[i]], filtered_ppg[peaks[i]]],
             "r-", linewidth=2)

plt.title("Peak-to-Peak Intervals (PPI)")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

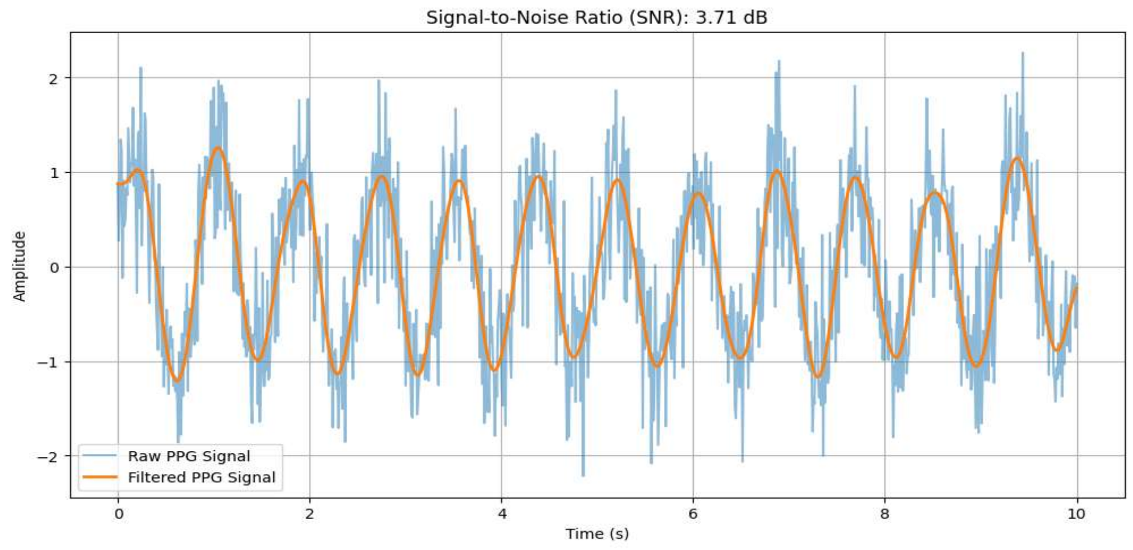
Mean Peak-to-Peak Interval (PPI): 0.834 sec



- **Signal-to-Noise Ratio (SNR)**

```
# Feature: Signal-to-Noise Ratio (SNR)
# SNR measures how strong the PPG signal is compared to noise

# Compute signal power (variance of filtered signal)
signal_power = np.var(filtered_ppg)

# Compute noise power (variance of residual noise)
noise_power = np.var(ppg_signal - filtered_ppg)

# Compute SNR in decibels (dB)
snr = 10 * np.log10(signal_power / noise_power)

# Print SNR value
print(f"Signal-to-Noise Ratio (SNR): {snr:.2f} dB")

plt.figure(figsize=(12, 6))
plt.plot(time, ppg_signal, label="Raw PPG Signal", alpha=0.5)
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.title(f"Signal-to-Noise Ratio (SNR): {snr:.2f} dB")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

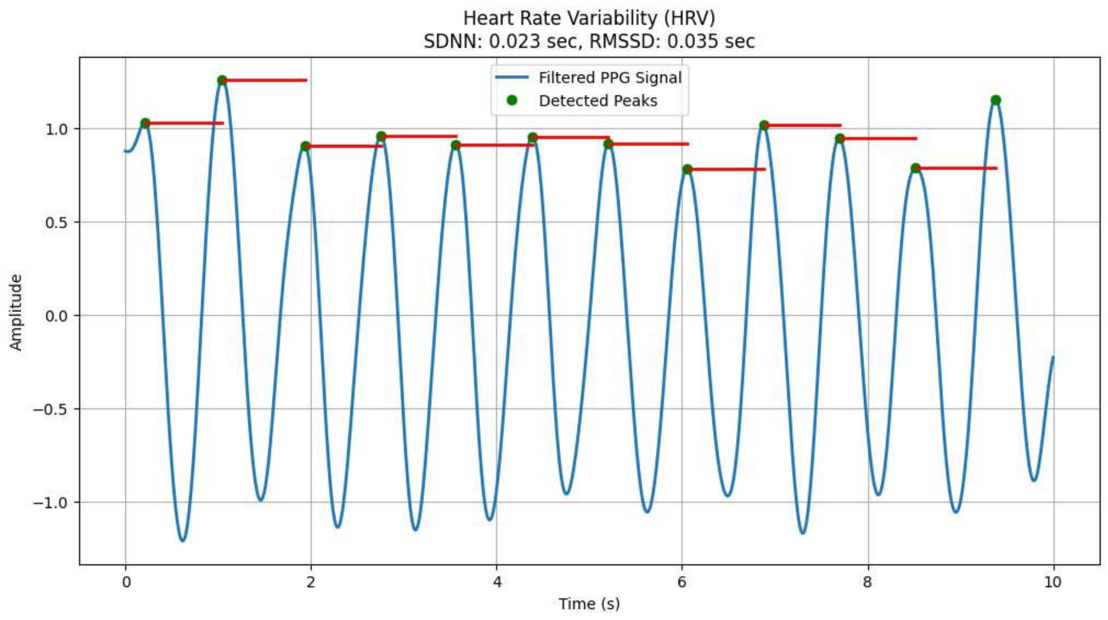
Signal-to-Noise Ratio (SNR): 3.71 dB



- **Heart Rate Variability (HRV)**

```
# Feature: Heart Rate Variability (HRV)
# HRV measures the variation in time intervals between heartbeats

# Compute Standard Deviation of Peak Intervals (SDNN - Standard Deviation of NN Intervals)
sdnn = np.std(peak_intervals)

# Compute Root Mean Square of Successive Differences (RMSSD)
rmssd = np.sqrt(np.mean(np.diff(peak_intervals) ** 2))

# Print HRV values
print(f"Heart Rate Variability (HRV):")
print(f"  SDNN: {sdnn:.3f} sec")
print(f"  RMSSD: {rmssd:.3f} sec")
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")

# Annotate peak intervals
for i in range(len(peaks) - 1):
   plt.plot([time[peaks[i]], time[peaks[i+1]]],
        [filtered_ppg[peaks[i]], filtered_ppg[peaks[i]]],
        "r-", linewidth=2)

plt.title(f"Heart Rate Variability (HRV)\nSDNN: {sdnn:.3f} sec, RMSSD: {rmssd:.3f} sec")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Heart Rate Variability (HRV):
SDNN: 0.023 sec
RMSSD: 0.035 sec

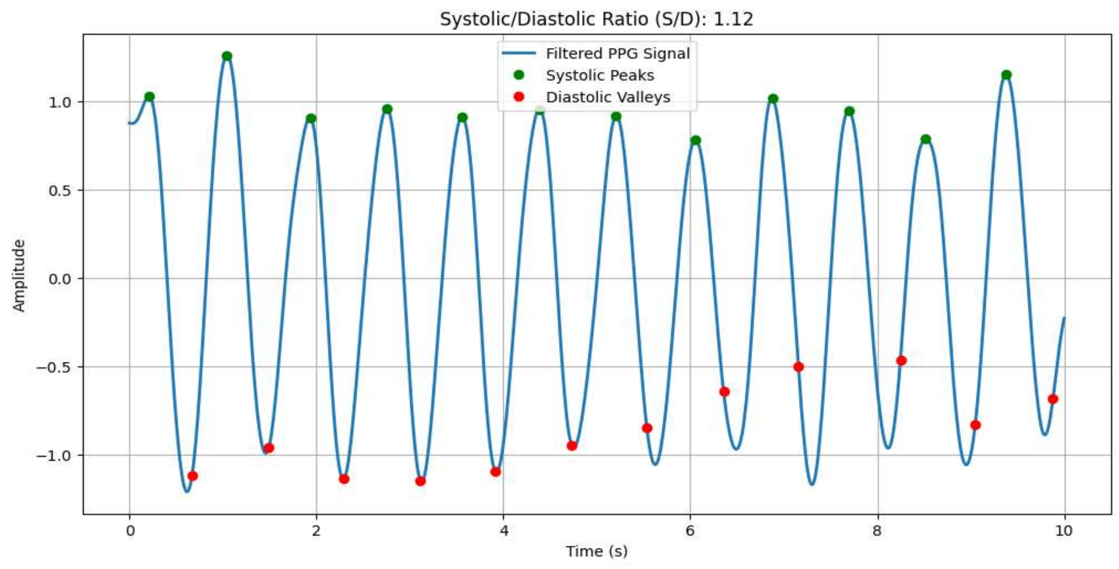


- **Systolic to Diastolic Ratio (S/D Ratio)**

```
# Feature: Systolic to Diastolic Ratio (S/D Ratio)
# This ratio represents the relationship between peak (systolic) and valley (diastolic) values.

# Extract Systolic (Peak) and Diastolic (Valley) Values
systolic_values = filtered_ppg[peaks]
diastolic_values = filtered_ppg[valleys]

# Compute the S/D Ratio
sd_ratio = np.mean(systolic_values) / np.mean(np.abs(diastolic_values))
```

```
# Print S/D Ratio
print(f"Systolic/Diastolic Ratio (S/D): {sd_ratio:.2f}")

plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Systolic Peaks")
plt.plot(time[valleys], filtered_ppg[valleys], "ro", label="Diastolic Valleys")

plt.title(f"Systolic/Diastolic Ratio (S/D): {sd_ratio:.2f}")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```

Systolic/Diastolic Ratio (S/D): 1.12

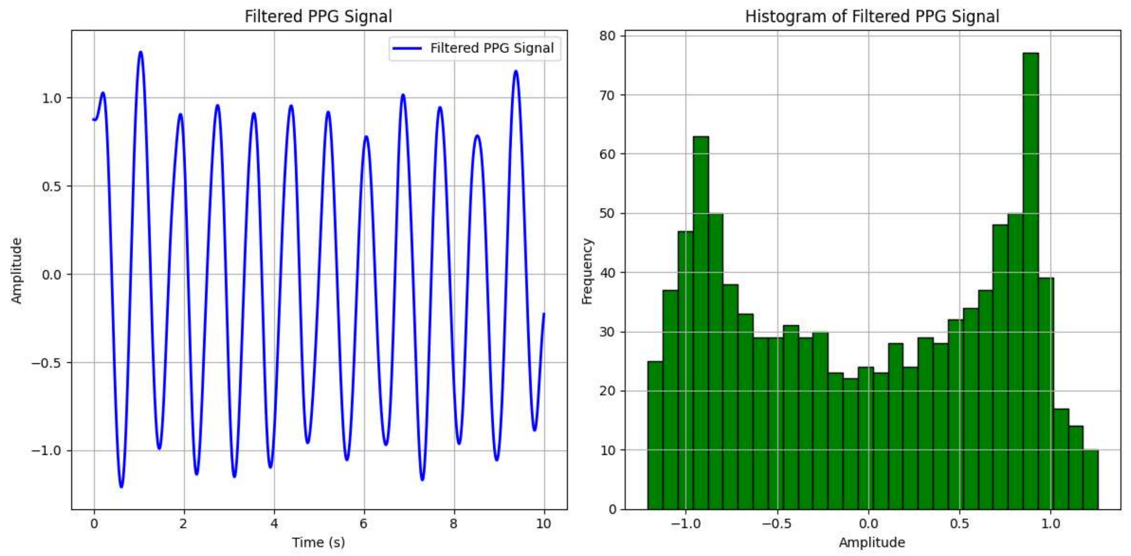


➢ <u>**Skewness and Kurtosis**</u>

```
# Feature: Skewness and Kurtosis
from scipy.stats import skew, kurtosis

# Calculate Skewness and Kurtosis
skewness = skew(filtered_ppg)
kurt = kurtosis(filtered_ppg)

# Print the Skewness and Kurtosis values
```

```
print(f"Skewness: {skewness:.3f}")
print(f"Kurtosis: {kurt:.3f}")

# Plotting the filtered PPG signal and its histogram
plt.figure(figsize=(12, 6))

# Plot the filtered PPG signal
plt.subplot(1, 2, 1)
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", color="b", linewidth=2)
plt.title("Filtered PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()

# Plot the histogram to visualize skewness and kurtosis
plt.subplot(1, 2, 2)
plt.hist(filtered_ppg, bins=30, color="g", edgecolor="black")
plt.title("Histogram of Filtered PPG Signal")
plt.xlabel("Amplitude")
plt.ylabel("Frequency")
plt.grid()

plt.tight_layout()
plt.show()
```

Skewness: -0.017
Kurtosis: -1.442



- **Peak Area (AUC) and Median Absolute Deviation (MAD)**

```
# Feature: Peak Area (AUC)
from scipy.integrate import simpson

# Calculate Area Under the Curve (AUC) for the peaks
auc = simpson(filtered_ppg[peaks])
print(f"Area Under Curve (AUC) for Peaks: {auc:.3f}")

# Feature: Median Absolute Deviation (MAD)
mad = np.median(np.abs(filtered_ppg - np.median(filtered_ppg)))
print(f"Median Absolute Deviation (MAD): {mad:.3f}")
```

```
Area Under Curve (AUC) for Peaks: 10.552
Median Absolute Deviation (MAD): 0.712
```