PPG Signal Processing and Analysis

1. Introduction Photoplethysmography (PPG) is a non-invasive optical technique used to measure blood volume changes in the microvascular tissue. This experiment involves generating a synthetic PPG signal, applying signal processing techniques, and analyzing heart rate metrics to detect abnormalities.

2. Objectives

- To generate a synthetic PPG signal with realistic components.
- To apply a Butterworth low-pass filter to remove noise.
- To detect peaks and valleys in the signal corresponding to heartbeats.
- To compute heart rate (HR) and heart rate variability (HRV).
- To identify potential abnormalities in the PPG signal.
- To visualize and analyze the results.

3. Methodology The experiment was conducted using Python and signal processing libraries. The key steps are as follows:

3.1 Signal Generation

A synthetic PPG signal was generated using sinusoidal components and Gaussian noise:

- Base signal with a heart rate of approximately 60 BPM.
- Higher frequency components were added to simulate real-world variations.
- Random noise was introduced to enhance realism.

3.2 Signal Filtering

A Butterworth low-pass filter was applied:

- Sampling Rate: 100 Hz
- Cutoff Frequency: 5 Hz
- Filter Order: 4

3.3 Peak Detection and Heart Rate Calculation

- Peaks (representing heartbeats) were detected using scipy.signal.find_peaks().
- Peak-to-peak intervals (PPI) were used to calculate HR and HRV.
- Instantaneous heart rates were derived from the inverse of PPI.

3.4 Abnormality Detection

- Abnormal peaks were identified based on deviations in PPI.
- Early and delayed beats were detected using a statistical threshold (±2 standard deviations).

3.5 Visualization and Statistical Analysis

- The raw and filtered signals were plotted with detected peaks and valleys.
- Instantaneous heart rate over time was visualized.

- Statistical metrics were computed and displayed.

Code

```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from typing import Tuple, List, Dict
import os
from datetime import datetime

# [Previous functions remain the same until plot_ppg_analysis]

def plot_ppg_analysis(t, ppg_signal, filtered_ppg, peaks, valleys, abnormal_peaks=None, save_path=None):
    """
    Create visualization plots for PPG signal analysis and optionally save to PNG.

    Args:
        t (array): Time array
        ppg_signal (array): Raw PPG signal
        filtered_ppg (array): Filtered PPG signal
        peaks (array): Indices of detected peaks
        valleys (array): Indices of detected valleys
        abnormal_peaks (array, optional): Indices of abnormal peaks
        save_path (str, optional): Directory path to save the plots
    """
    plt.figure(figsize=(15, 12))

    # Plot 1: Raw PPG Signal
    plt.subplot(4, 1, 1)
    plt.plot(t, ppg_signal)
    plt.title('Raw PPG Signal')
    plt.xlabel('Time (s)')
```

```
plt.ylabel('Amplitude')
plt.grid(True)

# Plot 2: Filtered PPG Signal
plt.subplot(4, 1, 2)
plt.plot(t, filtered_ppg)
plt.title('Filtered PPG Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(True)

# Plot 3: Peaks and Valleys
plt.subplot(4, 1, 3)
plt.plot(t, filtered_ppg, label='Filtered Signal')
plt.plot(t[peaks], filtered_ppg[peaks], "rv", label='Normal Peaks')
plt.plot(t[valleys], filtered_ppg[valleys], "g^", label='Valleys')

if abnormal_peaks and len(abnormal_peaks) > 0:
    plt.plot(t[abnormal_peaks], filtered_ppg[abnormal_peaks], "yx",
          markersize=10, label='Abnormal Peaks')

plt.title('Filtered PPG Signal with Peaks and Valleys')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)

# Plot 4: Instantaneous Heart Rate
if len(peaks) > 1:
    inst_hr = 60 / np.diff(t[peaks])
    hr_times = t[peaks[1:]]
    plt.subplot(4, 1, 4)
```

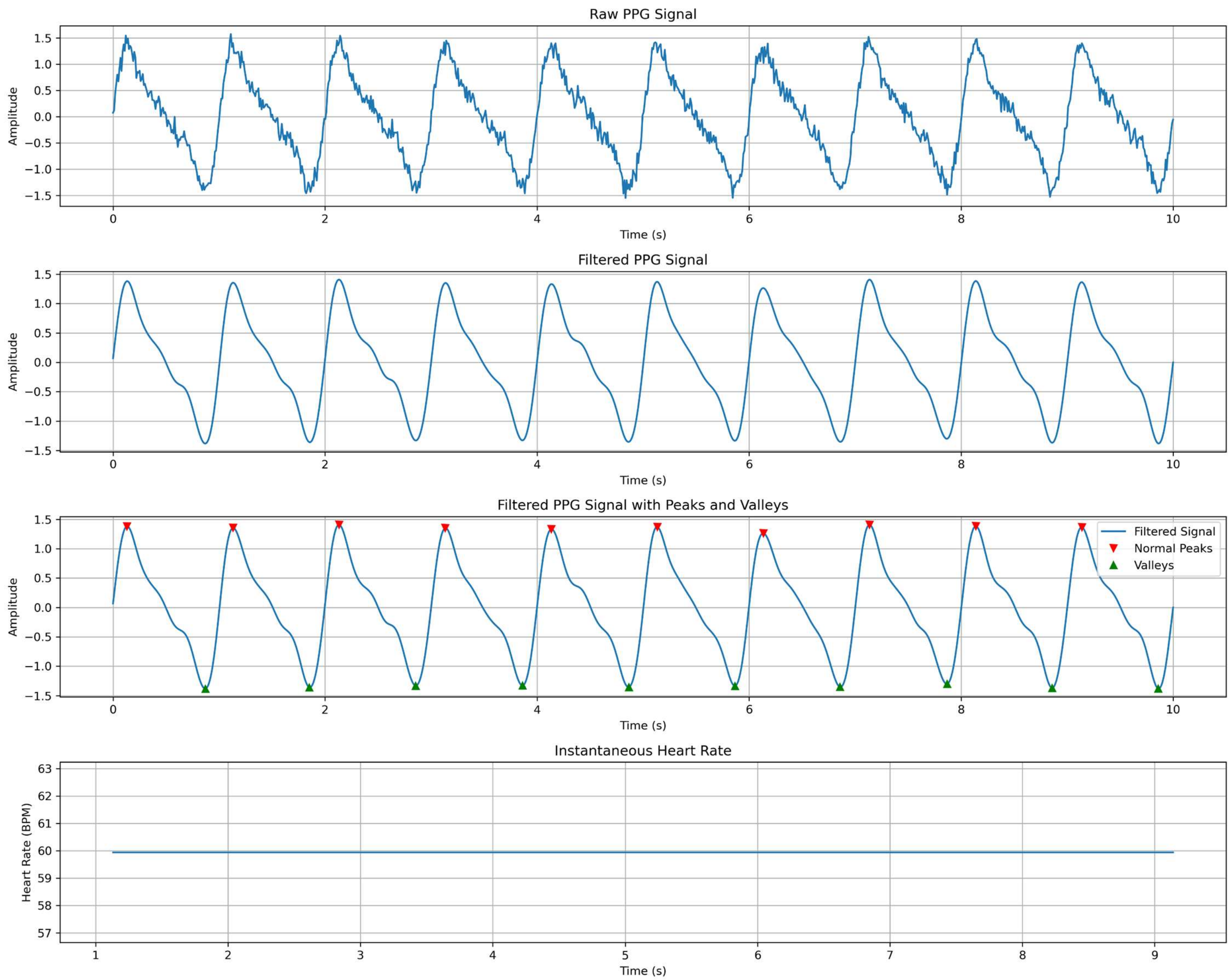
```python
        plt.plot(hr_times, inst_hr)
        plt.title('Instantaneous Heart Rate')
        plt.xlabel('Time (s)')
        plt.ylabel('Heart Rate (BPM)')
        plt.grid(True)

    plt.tight_layout()

    # Save plot if path is provided
    if save_path:
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        if not os.path.exists(save_path):
            os.makedirs(save_path)
        filename = os.path.join(save_path, f'ppg_analysis_{timestamp}.png')
        plt.savefig(filename, dpi=300, bbox_inches='tight')
        print(f"Plot saved as: {filename}")

    plt.show()

def save_statistics_to_file(hr_metrics: Dict, abnormal_peaks: List, abnormality_types: List, save_path: str):
    """
    Save statistics to a text file.

    Args:
        hr_metrics (Dict): Dictionary of heart rate metrics
        abnormal_peaks (List): List of abnormal peak indices
        abnormality_types (List): List of abnormality types
        save_path (str): Directory path to save the statistics
    """
    if not os.path.exists(save_path):
        os.makedirs(save_path)
```

```
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = os.path.join(save_path, f'ppg_statistics_{timestamp}.txt')

    with open(filename, 'w') as f:
        f.write("=== Heart Rate Statistics ===\n")
        f.write(f"Average Heart Rate: {hr_metrics['mean_hr']:.1f} BPM\n")
        f.write(f"Heart Rate Variability: {hr_metrics['hrv']:.1f} ms\n")
        f.write(f"Minimum Heart Rate: {hr_metrics['min_hr']:.1f} BPM\n")
        f.write(f"Maximum Heart Rate: {hr_metrics['max_hr']:.1f} BPM\n")
        f.write(f"Heart Rate Standard Deviation: {hr_metrics['std_hr']:.1f} BPM\n\n")

        f.write("=== Abnormality Detection ===\n")
        if len(abnormal_peaks) > 0:
            f.write(f"Number of abnormalities detected: {len(abnormal_peaks)}\n")
            for i, (peak, atype) in enumerate(zip(abnormal_peaks, abnormality_types)):
                f.write(f"Abnormality {i+1}: {atype} at time {peak:.2f}s\n")
        else:
            f.write("No abnormalities detected\n")

    print(f"Statistics saved as: {filename}")

def main():
    # Parameters
    sampling_rate = 100
    duration = 10
    cutoff_freq = 5  # Hz

    # Set output directory for saving files
    save_path = 'ppg_output'

    # Generate sample data
    t, ppg_signal = generate_ppg_signal(duration, sampling_rate)
```

```python
    # Apply Butterworth filter
    filtered_ppg = apply_butterworth_filter(ppg_signal, sampling_rate, cutoff_freq)

    # Find peaks and valleys
    peaks, _ = find_peaks(filtered_ppg, distance=50)
    valleys, _ = find_peaks(-filtered_ppg, distance=50)

    # Calculate heart rate metrics
    hr_metrics = calculate_heart_rate_metrics(t, peaks)

    # Detect abnormalities
    abnormal_peaks, abnormality_types = detect_abnormalities(t, peaks, hr_metrics)

    # Visualize results and save plot
    plot_ppg_analysis(t, ppg_signal, filtered_ppg, peaks, valleys, abnormal_peaks, save_path)

    # Display and save statistics
    display_statistics(hr_metrics, abnormal_peaks, abnormality_types)
    save_statistics_to_file(hr_metrics, abnormal_peaks, abnormality_types, save_path)

if __name__ == "__main__":
    main()
```

4. Results and Discussion The analysis provided the following results:

- **Average Heart Rate (HR):** Computed from detected peaks.
- **Heart Rate Variability (HRV):** Derived from standard deviation of PPI.
- **Minimum & Maximum HR:** Identified from instantaneous HR variations.
- **Abnormal Beats:** Early or delayed beats detected in the signal.
- **Visualization:** The processed PPG signal showed clear periodic peaks corresponding to heartbeats, and abnormal peaks were marked in the plots.



=== Heart Rate Statistics ===

Average Heart Rate: 59.9 BPM

Heart Rate Variability: 0.0 ms

Minimum Heart Rate: 59.9 BPM

Maximum Heart Rate: 59.9 BPM

Heart Rate Standard Deviation: 0.0 BPM

=== Abnormality Detection ===

No abnormalities detected

5. Conclusion The experiment successfully demonstrated the steps involved in PPG signal processing, filtering, and heart rate analysis. The methodology provided a structured approach to detect abnormalities in heart rate. The results showed the effectiveness of signal processing techniques in analyzing physiological signals.

6. Future Work

- Implementing real-time PPG signal analysis.

- Enhancing abnormality detection using machine learning techniques.
- Comparing different filtering techniques for improved signal quality.