Dynamic Gait Cycle Calculation: A Novel Approach to Real-Time Cadence Analysis

Abstract

This paper presents a novel approach to dynamically calculate the gait cycle of individuals, specifically focusing on real-time cadence analysis. Unlike traditional methods that require a fixed number of steps or a predefined time interval, our method provides instant feedback on gait parameters, including cadence, total steps, step rates, walking speed, and stride length. This dynamic approach allows for more accurate and immediate gait analysis, which can be beneficial in various fields such as sports science, rehabilitation, and wearable technology.

Introduction

Gait analysis is a crucial aspect of various domains, including medical diagnostics, sports performance, and rehabilitation. Traditional methods of calculating gait parameters, especially cadence, often rely on a fixed number of steps or a specific duration, which can introduce delays and reduce the accuracy of real-time applications. Our approach addresses these limitations by dynamically calculating gait parameters, providing immediate and continuous feedback.

Methodology

Our system uses a combination of accelerometers, force-sensitive resistors (FSRs), and Bluetooth Low Energy (BLE) communication to collect and process data in real-time. The primary innovation lies in our method of calculating cadence and other gait parameters dynamically, without waiting for a set number of steps or time.

Gait Parameters and Formulas

1. **Total Steps**

- Calculated by counting each detected step from both feet.

2. **Step Rate**

- Left Step Rate: LeftCurrentStepRate = leftStepsCount / elapsedTime
- Right Step Rate: RightCurrentStepRate = rightStepsCount / elapsedTime

3. **Walking Speed (m/s)**

```
magnitude = sqrt(Xaxis^2 + Yaxis^2 + Zaxis^2)
deltaTime = 0.01 // 10ms time interval for updates
decayRate = 0.995 // Multiplier when no acceleration or motion detected
if (magnitude > 1.10) {
   walkingSpeed += magnitude * deltaTime;
} else {
   walkingSpeed *= decayRate;
}
```

4. **Step Length (m)**

- Left Step Length: leftStepLength = walkingSpeed / LeftCurrentStepRate
- Right Step Length: rightStepLength = walkingSpeed / RightCurrentStepRate

5. **Stride Length (m)**

- strideLength = leftStepLength + rightStepLength

6. Cadence (spm)

- cadence = (walkingSpeed / strideLength) * 60

Explanation of Formulas

- Total Steps: Simply the count of all detected steps, giving a measure of the total number of steps taken by the individual.

- Step Rates: These are calculated by dividing the number of steps detected for each foot by the elapsed time, providing a rate of steps per second.

- Walking Speed: This is dynamically updated based on the magnitude of the accelerometer data, which captures the acceleration in all three axes (X, Y, Z). The speed increases with detected motion and decays when no motion is detected.

- Step Length: This is derived by dividing the walking speed by the step rate for each foot, providing an average step length.

- Stride Length: The sum of the left and right step lengths, representing the total distance covered in one complete stride.

- Cadence: Calculated by dividing the walking speed by the stride length and then multiplying by 60 to convert the rate to steps per minute.

Implementation

The system is implemented using an Arduino with BLE capabilities. The setup involves multiple peripherals for data collection and communication.

Sample Code:

```
// Start gait calculation if the flag is set
if (isGaitCalculationStarted) {
  unsigned long currentTime = millis();
  // Detect steps based on force sensor readings and yaw threshold
  if ((left_fsr >= 200 || right_fsr >= 200) && yaw < STEP_THRESHOLD) {
    if (index == 0) {
      if (!stepDetectedLeft && currentTime - lastStepTime >= stepDelay) {
        stepDetectedLeft = true;
        stepCount++;
```

```
            String stepCountData = String(stepCount);
            stepCountCharacteristic.writeValue(stepCountData.c_str());
            leftStepsCount++;
            unsigned long elapsedTime = currentTime - startTime;
            if (elapsedTime > 0) {
              leftCurrentStepRate = leftStepsCount / ((float)elapsedTime / 1000);
            }
            lastStepTime = currentTime;
        }
    }
    else if (index == 1) {
        if (!stepDetectedRight && currentTime - lastStepTime >= stepDelay) {
            stepDetectedRight = true;
            stepCount++;
            String stepCountData = String(stepCount);
            stepCountCharacteristic.writeValue(stepCountData.c_str());
            rightStepsCount++;
            unsigned long elapsedTime = currentTime - startTime;
            if (elapsedTime > 0) {
              rightCurrentStepRate = rightStepsCount / ((float)elapsedTime / 1000);
            }
            lastStepTime = currentTime;
        }
    }
}
else {
    if (index == 0) {
        stepDetectedLeft = false;
    }
```

```
    else if (index == 1) {
        stepDetectedRight = false;
    }
}

// Calculate walking speed based on accelerometer data
magnitude = sqrt(hipAccX * hipAccX + hipAccY * hipAccY + hipAccZ * hipAccZ);
if (magnitude > 1.10) {
    walkingSpeed += magnitude * deltaTime;
}
else {
    walkingSpeed *= decayRate;
}
walkingSpeed = max(0.0, walkingSpeed);
walkingSpeedCharacteristic.writeValue(walkingSpeed);

// Calculate step lengths and stride length
if (leftCurrentStepRate != 0) {
    left_step_length = walkingSpeed / leftCurrentStepRate;
}
if (rightCurrentStepRate != 0) {
    right_step_length = walkingSpeed / rightCurrentStepRate;
}
stride_length = left_step_length + right_step_length;

// Calculate cadence
if (stride_length > 0) {
    cadence = (walkingSpeed * 60) / stride_length;
}
```

```
      else {
        cadence = 0;
      }

      // Write calculated values to BLE characteristics
      leftStepLengthCharacteristic.writeValue(left_step_length);
      rightStepLengthCharacteristic.writeValue(right_step_length);
      strideLengthCharacteristic.writeValue(stride_length);
      cadenceCharacteristic.writeValue(cadence);
    }
}
```