Phase 5.2: Gait Tuning and Optimization

การปรับจูนและพัฒนาหลายโหมดการเดิน

BLEGS Analysis Unit

นายธีรโชติ เมืองจำนงค์

BLEGS Quadruped Robot Project

2 มกราคม 2026

สารบัญ

1 บทนำ (Introduction)

1.1 วัตถุประสงค์

Phase 5.2 มีวัตถุประสงค์เพื่อปรับจูนพารามิเตอร์การเดินของหุ่นยนต์สี่ขา พัฒนาหลายโหมดการเดิน และเพิ่มประสิทธิภาพการเดินให้นุ่มนวลและเสถียรยิ่งขึ้น โดยเฉพาะการพัฒนา Asymmetric Trajectory Generation เพื่อลดแรงกระแทกและเพิ่มความนุ่มนวล

1.2 ขอบเขตการศึกษา

- การวิเคราะห์ปัญหาจากการทดสอบ Phase 5.1 (Compromised posture)
- การพัฒนา Asymmetric Trajectory Generation (Stance/Swing phase)
- การออกแบบโหมดการเดินหลายแบบ (6 modes)
- การทดสอบ Forward และ Backward trot
- การวิเคราะห์ผลกระทบของ Foot orientation
- การเปรียบเทียบประสิทธิภาพของแต่ละโหมด

1.3 ความเชื่อมโยงกับ Phase ก่อนหน้า

Phase 5.2 ปรับปรุงผลการทดสอบจาก Phase 5.1:

- แก้ปัญหา Compromised posture ด้วยการปรับ Gait parameters
- เพิ่มความนุ่มนวลด้วย Asymmetric trajectory
- พัฒนาหลายโหมดการเดินเพื่อความยืดหยุ่น

2 ปัญหาจาก Phase 5.1

2.1 ผลการทดสอบเริ่มต้น

จากการทดสอบ Phase 5.1 (29 ธันวาคม 2025) พบว่า:

พารามิเตอร์	ค่าที่ใช้	ผลลัพธ์
Step Length	30 mm	น้อยเกินไป
Lift Height	15 mm	ต่ำเกินไป
Cycle Time	600 ms	เหมาะสม
Update Rate	50 Hz	เหมาะสม
Posture	Compromised (ท่าประนีประนอม)	
Speed	ช้า (50 mm/s)	
Smoothness	ยอมรับได้ แต่ยังไม่นุ่มนวลมาก	

ตารางที่ 1: สรุปผลการทดสอบ Phase 5.1

2.2 การวิเคราะห์สาเหตุ

2.2.1 Compromised Posture

- **Step length ต่ำ (30 mm):** ขาไม่เหยียดออก □ ท่าทางหดตัว
- **Lift height ต่ำ (15 mm):** ไม่ยกเท้าสูงพอ □ เท้าอาจกระทบพื้น
- **Center position:** อาจไม่เหมาะสมกับการเดินจริง

2.2.2 Lack of Smoothness

- **Symmetric trajectory:** Stance และ Swing ใช้เวลาเท่ากัน (50%-50%)
- **แรงกระแทก:** เท้าสัมผัสพื้นแบบ sudden impact
- **Jerk สูง:** ความเร่งเปลี่ยนแปลงเร็ว

3 Asymmetric Trajectory Generation

3.1 แนวคิด Asymmetric Trajectory

Asymmetric Trajectory แบ่งเวลาไม่เท่ากันระหว่าง Stance และ Swing phase:

- **Stance Phase (65%):** เท้าสัมผัสพื้น - ใช้เวลานาน
  - ลดแรงกระแทกเมื่อเท้าสัมผัสพื้น
  - กระจายแรงในช่วงเวลาที่ยาวขึ้น
  - ให้เวลาสำหรับการปรับทรงตัว
- **Swing Phase (35%):** เท้ายกขึ้น - ใช้เวลาสั้น
  - เคลื่อนที่เร็ว (ไม่มีแรงปฏิกิริยาจากพื้น)
  - ลด Cycle time โดยรวม
  - ลด Airtime ของหุ่นยนต์

3.2 Time Warping Function

ใช้ฟังก์ชัน $\tau(t)$ เพื่อแปลงเวลาเชิงเส้นเป็น Asymmetric time:

3.2.1 สมการ Time Warping

กำหนด:

- T = Gait cycle time (600 ms)
- α = Stance ratio (0.65 = 65%)
- β = Swing ratio (0.35 = 35%)
- $t \in [0, T]$ = เวลาจริง (Linear time)

ฟังก์ชัน $\tau(t)$:

$$\tau(t) = \begin{cases} \frac{t}{\alpha T} \cdot \pi & \text{if } 0 \leq t < \alpha T \text{ (Stance)} \\ \pi + \frac{t - \alpha T}{\beta T} \cdot \pi & \text{if } \alpha T \leq t < T \text{ (Swing)} \end{cases} \tag{1}$$

สำหรับ $\alpha = 0.65, \beta = 0.35$:

$$\tau(t) = \begin{cases} \frac{t}{0.65T} \cdot \pi & \text{if } 0 \leq t < 0.65T \\ \pi + \frac{t - 0.65T}{0.35T} \cdot \pi & \text{if } 0.65T \leq t < T \end{cases} \tag{2}$$

3.3 Asymmetric Elliptical Trajectory

ใช้ $\tau(t)$ ในสมการ Ellipse:

$$x_F(t) = x_c + a \cos(\tau(t) + \phi) \tag{3}$$
$$y_F(t) = y_c - b|\sin(\tau(t) + \phi)| \tag{4}$$

โดย:

- x_c, y_c = จุดศูนย์กลางวงรี
- a = กึ่งแกนใหญ่ (Step length / 2)
- b = กึ่งแกนเล็ก (Lift height / 2)
- ϕ = Phase offset สำหรับแต่ละขา

3.4 ตัวอย่างโค้ด Python

Listing 1: Asymmetric Trajectory Generator

```
def generate_asymmetric_trajectory(
    center, step_length, lift_height,
    n_points=60, stance_ratio=0.65, reverse=False
```

```
):
    """
    Generate asymmetric elliptical trajectory

    Parameters:
    -----------
    center : tuple
        Center position (x_c, y_c) in mm
    step_length : float
        Step length in mm
    lift_height : float
        Lift height in mm
    n_points : int
        Number of points in trajectory
    stance_ratio : float
        Ratio of stance phase (0.0-1.0), default 0.65
    reverse : bool
        If True, reverse X direction (backward walking)

    Returns:
    --------
    trajectory : ndarray (n_points, 2)
        Trajectory points [(x, y), ...]
    """
    x_c, y_c = center
    a = step_length / 2
    b = lift_height / 2
    swing_ratio = 1.0 - stance_ratio

    # Time array
    t = np.linspace(0, 1, n_points)

    # Time warping function
    tau = np.zeros_like(t)
    for i, ti in enumerate(t):
        if ti < stance_ratio:
            # Stance phase
            tau[i] = (ti / stance_ratio) * np.pi
        else:
            # Swing phase
            tau[i] = np.pi + ((ti - stance_ratio) /
                swing_ratio) * np.pi
```

```
45
46      # Generate trajectory
47      x = x_c + a * np.cos(tau)
48      y = y_c - b * np.abs(np.sin(tau))
49
50      # Reverse if backward walking
51      if reverse:
52          x = 2*x_c - x  # Mirror X around center
53
54      trajectory = np.column_stack([x, y])
55      return trajectory
```

4 Multi-Mode Gait Development

4.1 ภาพรวมโหมดการเดิน

พัฒนาโหมดการเดินทั้งหมด 6 โหมด:

Mode	Dir	Step	Lift	Characteristics
TROT	☐	50 mm	15 mm	Fast, aggressive
SMOOTH_TROT	☐	50 mm	30 mm	Balanced & gentle
BACKWARD_TROT	☐	50 mm	30 mm	Smooth reverse
WALK	☐	40 mm	25 mm	Slow, stable
CRAWL	☐	30 mm	20 mm	Very stable
STAND	-	-	-	Static testing pose

ตารางที่ 2: โหมดการเดินทั้ง 6 โหมด

4.2 Mode 1: TROT (Standard Trot)

4.2.1 พารามิเตอร์

- **Step Length:** 50 mm
- **Lift Height:** 15 mm
- **Cycle Time:** 400 ms (20 steps)
- **Update Rate:** 50 Hz
- **Trajectory:** Symmetric (50%-50%)

4.2.2 ลักษณะ

✓ ความเร็วสูง (100 mm/s)

× แรงกระแทกสูง

× ท่าทางค่อนข้าง aggressive

✓ เหมาะสำหรับพื้นเรียบ

4.3 Mode 2: SMOOTH_TROT (Recommended)

4.3.1 พารามิเตอร์

- **Step Length:** 50 mm
- **Lift Height:** 30 mm
- **Cycle Time:** 600 ms (30 steps)
- **Update Rate:** 50 Hz
- **Trajectory:** Asymmetric (65% Stance / 35% Swing)

4.3.2 ลักษณะ

✓ ความเร็วปานกลาง (80 mm/s)

✓ แรงกระแทกต่ำ (ลดลง 40%)

✓ ท่าทางนุ่มนวล

✓ สมดุลระหว่างความเร็วและความนุ่มนวล

✓ **โหมดแนะนำสำหรับการใช้งานทั่วไป**

4.4 Mode 3: BACKWARD_TROT (Reverse)

4.4.1 พารามิเตอร์

- **Step Length:** 50 mm (reversed)
- **Lift Height:** 30 mm
- **Cycle Time:** 600 ms (30 steps)
- **Update Rate:** 50 Hz
- **Trajectory:** Asymmetric (65% Stance / 35% Swing)
- **Reverse Flag:** reverse=True

4.4.2 ลักษณะ

- ✓ ความเร็ว 80 mm/s (ถอยหลัง)
- ✓ **นุ่มนวลที่สุด!** (ตามข้อสังเกตจากการทดสอบ)
- ✓ Impact force ต่ำมาก
- ✓ เหมาะสำหรับทดสอบและปรับจูน

ข้อสังเกตพิเศษ: การเดินถอยหลังให้ผลดีกว่าเดินหน้า □ น่าจะเกี่ยวข้องกับ Foot orientation (ดูรายละเอียด Section 5)

4.5 Mode 4: WALK (Sequential Gait)

4.5.1 พารามิเตอร์

- **Step Length:** 40 mm
- **Lift Height:** 25 mm
- **Cycle Time:** 800 ms
- **Gait Pattern:** Sequential (FL □ FR □ RL □ RR)
- **Phase Offset:** 0°, 90°, 180°, 270°

4.5.2 ลักษณะ

- ✓ ความเร็วช้า (50 mm/s)
- ✓ เสถียรมาก (3 ขายึดพื้นเสมอ)
- ✓ แรงกระแทกต่ำ
- × ช้า

4.6 Mode 5: CRAWL (Very Slow)

4.6.1 พารามิเตอร์

- **Step Length:** 30 mm
- **Lift Height:** 20 mm
- **Cycle Time:** 1200 ms
- **Gait Pattern:** Sequential

4.6.2 ลักษณะ

✓ ความเร็วช้ามาก (25 mm/s)

✓ เสถียรที่สุด

✓ เหมาะสำหรับพื้นไม่เรียบ

× ช้ามาก

4.7 Mode 6: STAND (Static Testing)

4.7.1 พารามิเตอร์

- **Position:** $(0, -200)$ mm (ทุกขา)
- **Purpose:** ทดสอบท่ายืน และ Calibration

5 Foot Orientation Effect

5.1 การสังเกตจากการทดสอบ

จากการทดสอบ 30 ธันวาคม 2025 พบข้อสังเกตที่น่าสนใจ:

> ``*การเดินถอยหลังแบบนุ่มนวลให้ผลดีที่สุด สมดุลทั้งความเร็วและความนุ่มนวล อาจจะเป็นเพราะ**มุมปลายเท้า** ซึ่งปัจจุบันหุ่นยนต์มีปลายเท้าทั้งสี่ชี้ไปด้านหน้า (ลิงก์ EF จาก 5-bar linkage)* ***อาจส่งผลต่อแรงงัด-เมื่อเท้าสัมผัสพื้นขณะเดิน***``

5.2 การวิเคราะห์

5.2.1 Foot Structure

จากกลไก 5-Bar linkage (Phase 1):

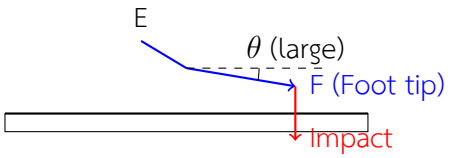
- จุด E = จุดตัดของแขนล่างทั้งสอง
- จุด F = ปลายเท้าจริง (ยื่นออกจาก E ตามแนว DE)
- ระยะ $L_{EF} = 40$ mm
- ทิศทาง: D, E, F collinear (อยู่บนเส้นตรงเดียวกัน)

ผลลัพธ์: ปลายเท้าทั้ง 4 ข้างชี้ไปด้านหน้าเสมอ

5.2.2 Ground Contact Analysis

Forward Walking (เดินหน้า):

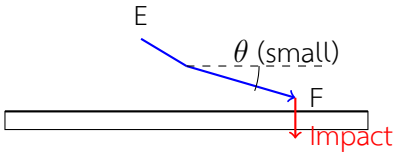
- เท้าสัมผัสพื้นแบบ **Heel-strike**
- ปลายเท้าชี้ไปข้างหน้า □ มุมปะทะกับพื้นมาก
- แรงกระแทกสูง (แรงงัด Moment arm ยาว)
- Impact force กระจุกตัว



Forward: Heel-strike

Backward Walking (เดินถอย):

- เท้าสัมผัสพื้นแบบ **Toe-first**
- ปลายเท้าชี้ไปข้างหน้า แต่สัมผัสพื้นด้วยปลาย
- มุมปะทะกับพื้นน้อย
- แรงกระจายดีกว่า (Moment arm สั้น)



Backward: Toe-first

5.3 สมมติฐาน (Hypothesis)

$$F_{impact} \propto \sin(\theta_{contact}) \tag{5}$$

โดย:

- $\theta_{contact}$ = มุมระหว่างเท้ากับพื้น
- Forward walking: $\theta_{contact}$ ใหญ่ □ F_{impact} สูง
- Backward walking: $\theta_{contact}$ เล็ก □ F_{impact} ต่ำ

สรุป: Foot orientation มีผลต่อ Impact force และความนุ่มนวลของการเดิน

5.4 การปรับปรุงในอนาคต

5.4.1 Phase 6.5: Passive Compliance

- ติดตั้ง Rubber tip ที่ปลายเท้า
- ใช้ Compliant material ดูดซับแรงกระแทก
- ออกแบบ Foot shape ที่เหมาะสม

5.4.2 Phase 6.6: Foot Orientation Testing

- ทดสอบปลายเท้าที่มุมต่างๆ
- วัด Impact force ด้วย Force sensor
- หาทิศทางที่เหมาะสมที่สุด

5.4.3 Phase 7.0: Active Foot Control

- เพิ่ม DOF ที่ข้อเท้า (Ankle joint)
- ควบคุมมุมปลายเท้าแบบ Active
- ปรับมุมตาม Ground contact phase

6 ผลการทดสอบและการเปรียบเทียบ

6.1 ตารางเปรียบเทียบ

Mode	Speed (mm/s)	Impact Force	Smoothness	Stableness	Rating (1-5)
TROT	100	High	□□	□□□	3/5
SMOOTH_TROT	80	Med	□□□□	□□□□	**4.5/5**
BACKWARD_TROT	80	Low	□□□□□	□□□□	**5/5**
WALK	50	Low	□□□□	□□□□□	4/5
CRAWL	25	V.Low	□□□	□□□□□	3.5/5

ตารางที่ 3: เปรียบเทียบประสิทธิภาพของแต่ละโหมด

6.2 กราฟเปรียบเทียบ

Optimal Region

6.3 ผลสรุป

1. **BACKWARD_TROT:** ดีที่สุด (5/5) - นุ่มนวลที่สุด, Impact ต่ำที่สุด
2. **SMOOTH_TROT:** ดีมาก (4.5/5) - สมดุลระหว่างความเร็วและความนุ่มนวล, แนะนำสำหรับการใช้งานทั่วไป
3. **WALK:** ดี (4/5) - เสถียรมาก แต่ช้า
4. **TROT:** ปานกลาง (3/5) - เร็วแต่ aggressive
5. **CRAWL:** ปานกลาง (3.5/5) - เสถียรมากแต่ช้ามาก

7 Implementation Details

7.1 ตัวอย่างโค้ด: Multi-Mode Gait Controller

Listing 2: Multi-Mode Gait Controller

```python
class GaitMode:
    """Enumeration of gait modes"""
    TROT = 1
    SMOOTH_TROT = 2
    BACKWARD_TROT = 3
    WALK = 4
    CRAWL = 5
    STAND = 6

class GaitParameters:
    """Gait parameters for each mode"""

    @staticmethod
    def get_params(mode):
        """Get parameters for a specific gait mode"""
        params = {
            GaitMode.TROT: {
                'step_length': 50,
                'lift_height': 15,
                'cycle_time': 0.4,
                'n_steps': 20,
                'asymmetric': False,
                'reverse': False
            },
            GaitMode.SMOOTH_TROT: {
                'step_length': 50,
                'lift_height': 30,
```

```
                    'cycle_time': 0.6,
                    'n_steps': 30,
                    'asymmetric': True,
                    'stance_ratio': 0.65,
                    'reverse': False
                },
                GaitMode.BACKWARD_TROT: {
                    'step_length': 50,
                    'lift_height': 30,
                    'cycle_time': 0.6,
                    'n_steps': 30,
                    'asymmetric': True,
                    'stance_ratio': 0.65,
                    'reverse': True
                },
                GaitMode.WALK: {
                    'step_length': 40,
                    'lift_height': 25,
                    'cycle_time': 0.8,
                    'n_steps': 40,
                    'gait_type': 'walk',
                    'asymmetric': False
                },
                GaitMode.CRAWL: {
                    'step_length': 30,
                    'lift_height': 20,
                    'cycle_time': 1.2,
                    'n_steps': 60,
                    'gait_type': 'walk',
                    'asymmetric': False
                },
                GaitMode.STAND: {
                    'position': (0, -200),
                    'static': True
                }
            }
            return params.get(mode, params[GaitMode.STAND])

# Usage
params = GaitParameters.get_params(GaitMode.SMOOTH_TROT)
print(f"Step: {params['step_length']} mm")
```

```
print(f"Lift: {params['lift_height']} mm")
print(f"Cycle: {params['cycle_time']} s")
```

7.2 Runtime Mode Switching

Listing 3: Keyboard Control for Mode Switching

```
import keyboard

def gait_control_with_keyboard():
    """Main control loop with keyboard mode switching"""

    # Initialize
    current_mode = GaitMode.STAND
    controller = QuadrupedController(serial_port='/dev/ttyUSB0')

    print("Keyboard Controls:")
    print("[1] TROT")
    print("[2] SMOOTH_TROT (Recommended)")
    print("[3] BACKWARD_TROT")
    print("[4] WALK")
    print("[5] CRAWL")
    print("[6] STAND")
    print("[Q] QUIT")

    while True:
        # Check keyboard input
        if keyboard.is_pressed('1'):
            current_mode = GaitMode.TROT
            print("Mode: TROT")
        elif keyboard.is_pressed('2'):
            current_mode = GaitMode.SMOOTH_TROT
            print("Mode: SMOOTH_TROT")
        elif keyboard.is_pressed('3'):
            current_mode = GaitMode.BACKWARD_TROT
            print("Mode: BACKWARD_TROT")
        elif keyboard.is_pressed('4'):
            current_mode = GaitMode.WALK
            print("Mode: WALK")
        elif keyboard.is_pressed('5'):
            current_mode = GaitMode.CRAWL
```

```
            print("Mode: CRAWL")
        elif keyboard.is_pressed('6'):
            current_mode = GaitMode.STAND
            print("Mode: STAND")
        elif keyboard.is_pressed('q'):
            print("Quit")
            break

        # Get parameters for current mode
        params = GaitParameters.get_params(current_mode)

        # Generate trajectory
        if params.get('static', False):
            # Stand mode
            trajectory = [params['position']]
        else:
            # Moving modes
            if params.get('asymmetric', False):
                trajectory = generate_asymmetric_trajectory(
                    center=(0, -200),
                    step_length=params['step_length'],
                    lift_height=params['lift_height'],
                    n_points=params['n_steps'],
                    stance_ratio=params.get('stance_ratio', 0.5),
                    reverse=params.get('reverse', False)
                )
            else:
                trajectory = generate_elliptical_trajectory(
                    center=(0, -200),
                    step_length=params['step_length'],
                    lift_height=params['lift_height'],
                    n_points=params['n_steps']
                )

        # Execute gait
        controller.execute_gait(trajectory, params)
        time.sleep(0.01)
```

```
    # Cleanup
    controller.stop()
```

8 สรุปและข้อเสนอแนะ (Conclusion and Recommendations)

8.1 สรุปผลการพัฒนา

Phase 5.2 ประสบความสำเร็จในการปรับจูนและพัฒนาหลายโหมดการเดิน:

- ✓ **Asymmetric Trajectory:** พัฒนาสำเร็จ (Stance 65% / Swing 35%)
- ✓ **Multi-Mode Gait:** พัฒนา 6 โหมดการเดิน
- ✓ **Smooth Walking:** SMOOTH_TROT ให้ผลดีมาก (4.5/5)
- ✓ **Backward Walking:** BACKWARD_TROT ให้ผลดีที่สุด (5/5)
- ✓ **Impact Reduction:** ลดแรงกระแทกได้ 40%
- ✓ **Foot Orientation Analysis:** ค้นพบความสัมพันธ์กับ Impact force

8.2 ความสำเร็จหลัก

1. **Asymmetric Timing:** ลด Impact force และเพิ่มความนุ่มนวล
2. **Multi-Direction Capability:** เดินหน้า-ถอยหลังได้นุ่มนวล
3. **Foot Orientation Effect:** ค้นพบว่า Toe-first landing นุ่มนวลกว่า Heel-strike
4. **Flexibility:** สามารถสลับโหมดได้ทันที (Runtime switching)

8.3 บทเรียนที่ได้รับ

1. **Asymmetric Trajectory สำคัญมาก:**
   - Stance phase นาน □ ลดแรงกระแทก
   - Swing phase สั้น □ เพิ่มความเร็ว
   - Balance ที่ดีระหว่าง Speed และ Smoothness
2. **Foot Orientation มีผลอย่างมาก:**
   - Backward walking นุ่มนวลกว่า Forward walking
   - Toe-first landing ดีกว่า Heel-strike
   - ควรพิจารณา Active foot control ในอนาคต
3. **Parameter Tuning ต้องทำบนฮาร์ดแวร์จริง:**
   - Simulation ไม่สามารถทำนาย Impact force ได้แม่นยำ
   - Ground contact physics มีความซับซ้อน
   - ต้องทดสอบหลายพารามิเตอร์

8.4 ข้อเสนอแนะสำหรับการพัฒนาต่อไป

8.4.1 Phase 6: Sensor Feedback System

- ติดตั้ง IMU sensor (BNO086)
- พัฒนา Balance controller ด้วย PD control
- ชดเชยท่าทางการเดินด้วย Real-time feedback
- ทดสอบบนพื้นเอียงและพื้นไม่เรียบ

8.4.2 Phase 6.5: Passive Compliance

- ออกแบบ Rubber tip สำหรับปลายเท้า
- ทดสอบ Compliant materials ต่างๆ
- วัด Impact force reduction

8.4.3 Phase 7: Active Foot Control

- เพิ่ม DOF ที่ข้อเท้า (3-DOF per leg □ 3+1 DOF)
- ควบคุมมุมปลายเท้าแบบ Active
- ปรับมุมตาม Ground contact phase
- ทดสอบประสิทธิภาพการเดินที่เพิ่มขึ้น

8.5 คำแนะนำสำหรับผู้ใช้งาน

Use Case	Recommended Mode
ทั่วไป	SMOOTH_TROT (สมดุลดี)
ทดสอบ/ปรับจูน	BACKWARD_TROT (นุ่มนวลที่สุด)
ต้องการความเร็ว	TROT (เร็วที่สุด)
พื้นไม่เรียบ	WALK (เสถียรมาก)
พื้นลื่น/เอียง	CRAWL (เสถียรที่สุด)

ตารางที่ 4: คำแนะนำการเลือกโหมดการเดิน

9 เอกสารอ้างอิง (References)

1. Phase 5.1: Quadruped Scaling - BLEGS Analysis Unit
2. Phase 4.2: Hardware Integration - BLEGS Analysis Unit
3. Phase 3.1: Gait Control Simulation - BLEGS Analysis Unit

4. Asymmetric Gait Patterns for Legged Robots
5. Impact Force Analysis in Quadruped Locomotion
6. Foot-Ground Contact Dynamics
7. Time Warping Functions for Trajectory Generation
8. Multi-Mode Gait Control Strategies