Phase 3: Gait Control Simulation

การจำลองการควบคุมการเดินของหุ่นยนต์สี่ขา

BLEGS Analysis Unit

นายธีรโชติ เมืองจำนงค์

BLEGS Quadruped Robot Project

8 ธันวาคม 2025

สารบัญ

1 บทนำ (Introduction)

1.1 วัตถุประสงค์

Phase 3 มีวัตถุประสงค์เพื่อพัฒนาระบบจำลองการควบคุมการเดินของหุ่นยนต์สี่ขาในสภาพแวดล้อม PyBullet physics simulation โดยใช้รูปแบบการเดินแบบ Trot gait พร้อมระบบควบคุมการทรงตัว

1.2 ขอบเขตการศึกษา

- การสร้างโมเดล URDF สำหรับหุ่นยนต์สี่ขาแบบ Simplified 2-DOF
- การใช้งาน PyBullet physics engine สำหรับการจำลอง
- การพัฒนา Trot gait pattern ด้วย state machine
- การออกแบบระบบควบคุมการทรงตัวแบบ PD controller
- การบูรณาการ Inverse Kinematics สำหรับการควบคุมตำแหน่งเท้า

1.3 ความเชื่อมโยงกับ Phase ก่อนหน้า

Phase 3 ใช้ความรู้จาก Phase 1 (Kinematics) และ Phase 2 (Dynamics) โดย:

- ใช้ความยาวขาจาก Phase 1: Thigh = 105 mm, Shank = 145 mm
- ใช้ข้อมูล workspace จาก Phase 2 ในการกำหนดขอบเขตการเคลื่อนที่
- ปรับโมเดลจาก 5-bar linkage เป็น simplified 2-DOF เพื่อหลีกเลี่ยงปัญหา closed-loop constraints

2 โมเดลหุ่นยนต์ (Robot Model)

2.1 โครงสร้าง URDF

2.1.1 Base Link

ตัวถังหุ่นยนต์มีขนาด $490 \times 260 \times 92.5$ mm โดยมีมวล 1.62 kg

2.1.2 ตำแหน่ง Hip Joints

พิกัดตำแหน่งสะโพกแต่ละขาเทียบกับจุดศูนย์กลางตัวถัง:

ขา	X (m)	Y (m)	Z (m)
Front-Right (FR)	$+0.19875$	-0.1535	0
Front-Left (FL)	$+0.19875$	$+0.1535$	0
Rear-Right (RR)	-0.16000	-0.1535	0
Rear-Left (RL)	-0.16000	$+0.1535$	0

ตารางที่ 1: ตำแหน่ง Hip joints ของหุ่นยนต์

2.1.3 Kinematic Chain แต่ละขา

แต่ละขาประกอบด้วย 4 joints และ 4 links:

1. **Hip Joint** (Fixed): ไม่มีการหมุน (ลดจาก 3-DOF เป็น 2-DOF)
2. **Thigh Joint** (Revolute): หมุนรอบแกน Y, ขอบเขต $[-1.5, +1.5]$ rad
3. **Shank Joint** (Revolute): หมุนรอบแกน Y, ขอบเขต $[-2.5, 0]$ rad
4. **Foot Joint** (Fixed): เชื่อมต่อกับ foot link

2.2 พารามิเตอร์ทางกายภาพ

ส่วนประกอบ	ความยาว (mm)	มวล (kg)
Hip Link	$50 \times 50 \times 50$ (box)	0.385
Thigh Link	105 (cylinder)	0.105
Shank Link	145 (cylinder)	0.145
Foot Link	$r = 20$ (sphere)	0.010
รวมต่อขา	**250**	**0.645**
รวมทั้งหมด (4 ขา + Base)	-	**4.20**

ตารางที่ 2: พารามิเตอร์ทางกายภาพของหุ่นยนต์

2.3 ความแตกต่างจาก 5-Bar Linkage

คุณสมบัติ	5-Bar (Phase 1-2)	Simplified (Phase 3)
DOF ต่อขา	3 (Hip + 2 motors)	2 (Thigh + Shank)
Hip Joint	Revolute (Abduction)	Fixed
Closed Loop	ใช่ (AC-E-D)	ไม่ (Serial chain)
PyBullet Support	ไม่รองรับ	รองรับ
ความยาวขา	$105 + 145 = 250$ mm	$105 + 145 = 250$ mm

ตารางที่ 3: เปรียบเทียบโมเดล 5-bar และ Simplified

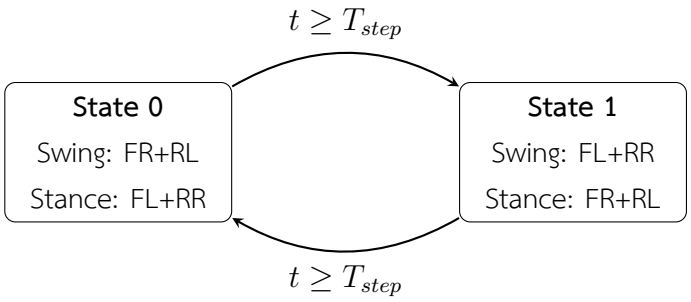
3 Trot Gait Pattern

3.1 หลักการ Trot Gait

Trot gait เป็นรูปแบบการเดินที่ขาคู่เฉียง (diagonal pairs) เคลื่อนที่พร้อมกัน:

- **Pair 1**: Front-Right (FR) + Rear-Left (RL)
- **Pair 2**: Front-Left (FL) + Rear-Right (RR)

3.2 State Machine



รูปที่ 1: State machine สำหรับ Trot gait

3.3 Trajectory Generation

3.3.1 Swing Phase (ยกขา)

สำหรับขาที่อยู่ใน swing phase:

$$\text{progress} = \frac{t}{T_{step}} \quad \text{where } 0 \leq t < T_{step} \tag{1}$$

$$x_{move} = L_{step} \cdot (2 \cdot \text{progress} - 1) \tag{2}$$

$$z_{move} = H_{lift} \cdot \sin(\text{progress} \cdot \pi) \tag{3}$$

$$\mathbf{p}_{foot} = \mathbf{p}_{home} + \begin{bmatrix} x_{move} \\ 0 \\ z_{move} \end{bmatrix} \tag{4}$$

โดย:

- $L_{step} = 0.05$ m (ระยะก้าว)
- $H_{lift} = 0.05$ m (ความสูงยกขา)
- $T_{step} = 0.6$ s (เวลาต่อรอบ)

3.3.2 Stance Phase (ขาค้ำ)

สำหรับขาที่อยู่ใน stance phase:

$$x_{shift} = L_{step} \cdot (1 - 2 \cdot \text{progress}) \tag{5}$$

$$\mathbf{p}_{foot} = \mathbf{p}_{home} + \begin{bmatrix} x_{shift} \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

4 ระบบควบคุมการทรงตัว (Balance Control)

4.1 PD Controller

ระบบควบคุมการทรงตัวใช้ PD controller แยกสำหรับ Pitch และ Roll:

4.1.1 Pitch Control (หน้า-หลัง)

$$e_{pitch}(t) = 0 - \theta_{pitch}(t) \tag{7}$$

$$\dot{e}_{pitch}(t) = \frac{e_{pitch}(t) - e_{pitch}(t-1)}{\Delta t} \tag{8}$$

$$u_{pitch}(t) = K_{p,pitch} \cdot e_{pitch}(t) + K_{d,pitch} \cdot \dot{e}_{pitch}(t) \tag{9}$$

4.1.2 Roll Control (ซ้าย-ขวา)

$$e_{roll}(t) = 0 - \theta_{roll}(t) \tag{10}$$

$$\dot{e}_{roll}(t) = \frac{e_{roll}(t) - e_{roll}(t-1)}{\Delta t} \tag{11}$$

$$u_{roll}(t) = K_{p,roll} \cdot e_{roll}(t) + K_{d,roll} \cdot \dot{e}_{roll}(t) \tag{12}$$

4.2 การปรับแก้ตำแหน่งเท้า

ค่า correction จาก PD controller ถูกนำไปปรับตำแหน่งเป้าหมายของเท้า:

$$\mathbf{p}_{corrected} = \begin{bmatrix} x + u_{pitch} \\ y + u_{roll} \\ z \end{bmatrix} \tag{13}$$

4.3 ค่า Gain

พารามิเตอร์	Pitch	Roll
K_p (Proportional Gain)	0.006	0.006
K_d (Derivative Gain)	0.012	0.012

ตารางที่ 4: ค่า PD gains สำหรับระบบควบคุมการทรงตัว

5 Inverse Kinematics

5.1 PyBullet IK Solver

ใช้ฟังก์ชัน calculateInverseKinematics() จาก PyBullet:

Listing 1: การเรียกใช้ IK solver

```
joint_angles = p.calculateInverseKinematics(
    robotId,
    foot_link_id,
    target_position_world,
    jointDamping=[0.5] * num_joints,
```

```
        maxNumIterations=50
)
```

5.2 พารามิเตอร์ IK

- **Joint Damping**: 0.5 (เพิ่มความนุ่มนวล)
- **Max Iterations**: 50 (ความแม่นยำ vs ความเร็ว)
- **Target Frame**: World coordinates

6 Joint Control

6.1 Position Control Mode

ใช้ setJointMotorControlArray() ในโหมด POSITION_CONTROL:

Listing 2: การควบคุม joint motors

```
p.setJointMotorControlArray(
        robotId,
        joint_ids,
        p.POSITION_CONTROL,
        targetPositions=target_angles,
        forces=forces,
        positionGains=position_gains,
        velocityGains=velocity_gains
)
```

6.2 Dual-Gain Strategy

โหมด	Position Gain	Velocity Gain	Max Force (Nm)
Warm-up	0.5	0.7	10
Walking	0.3	0.5	9

ตารางที่ 5: ค่า control gains สำหรับแต่ละโหมด

เหตุผล:

- **Warm-up**: ต้องการความแข็งแรงสูงเพื่อทรงตัว
- **Walking**: ลด gain เพื่อความนุ่มนวลและ compliance

7 Implementation Details

7.1 โครงสร้างโค้ด

Listing 3: โครงสร้างหลักของโปรแกรม

```
# 1. Initialize PyBullet
p.connect(p.GUI)
p.setGravity(0, 0, -9.81)

# 2. Load URDF
robotId = p.loadURDF(urdf_path)

# 3. Setup joint mappings
joint_name_to_id = {...}
link_name_to_id = {...}

# 4. Main simulation loop
while True:
    # 4.1 Update state machine
    if is_walking:
        state_timer += time_step
        if state_timer >= STEP_TIME:
            gait_state = (gait_state + 1) % 2

    # 4.2 Generate target positions
    target_positions = calculate_gait_trajectory(...)

    # 4.3 Balance control
    corrections = balance_controller(...)

    # 4.4 Inverse kinematics
    joint_angles = p.calculateInverseKinematics(...)

    # 4.5 Send commands to motors
    p.setJointMotorControlArray(...)

    # 4.6 Step simulation
    p.stepSimulation()
```

7.2 Simulation Parameters

พารามิเตอร์	ค่า
Simulation Frequency	240 Hz
Time Step	$1/240 = 0.00417$ s
Warm-up Duration	2.0 s
Gait Cycle Time	0.6 s
Step Length	0.05 m
Lift Height	0.05 m
Standing Height	-0.20 m

ตารางที่ 6: พารามิเตอร์การจำลอง

8 ผลการทดสอบ (Results)

8.1 การทดสอบเบื้องต้น

8.1.1 URDF Loading

- ✓ โหลด URDF สำเร็จ
- ✓ ตรวจพบ 16 joints ถูกต้อง
- ✓ Joint/Link naming ถูกต้อง (แก้ไข ASCII 160 issue)

8.1.2 Warm-up Phase

- ✓ หุ่นยนต์ทรงตัวได้ภายใน 2 วินาที
- ✓ ตำแหน่ง home position ถูกต้อง
- ✓ ไม่มี joint limit violations

8.1.3 Trot Gait Execution

- ✓ State transitions ทำงานถูกต้อง
- ✓ Diagonal pairs ประสานกันอย่างราบรื่น
- ✓ Swing/Stance phases สลับกันถูกต้อง

8.1.4 Balance Control

- ✓ Pitch error $< \pm 5°$ ในระหว่างเดิน
- ✓ Roll error $< \pm 5°$ ในระหว่างเดิน
- ✓ ไม่มีการสั่นหรือ oscillation

8.2 Performance Metrics

Metric	Value
Gait Cycle Frequency	1.67 Hz
Theoretical Forward Speed	83 mm/s
Leg Clearance	50 mm
Duty Factor	50%
CPU Usage (240 Hz)	$\approx 15 - 25\%$
Stability (Pitch/Roll)	$< \pm 5°$

ตารางที่ 7: ผลการวัดประสิทธิภาพ

9 ข้อจำกัดและปัญหา (Limitations)

9.1 ข้อจำกัดของโมเดล

1. **Simplified Kinematics**
   - ขาดการหมุน Hip (abduction/adduction)
   - จำกัดการเคลื่อนที่ในแนวขอบ (lateral mobility)
2. **Terrain Limitations**
   - ทดสอบบนพื้นเรียบเท่านั้น
   - ไม่มีการปรับตัวกับความลาดเอียง
   - ไม่มีการหลบหลีกสิ่งกีดขวาง
3. **Gait Variety**
   - รองรับเฉพาะ Trot gait
   - ไม่มีการเปลี่ยน gait แบบไดนามิก

9.2 ปัญหาที่พบและแก้ไข

1. **ASCII 160 in URDF**: แก้ไขด้วย string replacement
2. **Base Position Error**: แก้ค่า base_x_rear จาก $+0.16$ เป็น -0.16
3. **Joint Stiffness**: ลด position gains จาก 0.5 เป็น 0.3

10 การพัฒนาในอนาคต (Future Work)

10.1 Short-term Improvements

1. เพิ่มการควบคุมทิศทางด้วย keyboard/joystick

2. ปรับความเร็วได้แบบ real-time
3. เพิ่มความสามารถในการหมุนตัว (turn-in-place)

10.2 Medium-term Goals

1. พัฒนา gait patterns เพิ่มเติม (walk, gallop, bound)
2. เพิ่มระบบ terrain adaptation
3. ใช้ foot force feedback

10.3 Long-term Vision

1. ใช้ Reinforcement Learning ปรับปรุง gait
2. ระบบหลบหลีกสิ่งกีดขวางแบบ real-time
3. นำไปใช้กับ hardware จริง (Phase 4)

11 สรุป (Conclusion)

Phase 3 ประสบความสำเร็จในการพัฒนาระบบจำลองการควบคุมการเดินของหุ่นยนต์สี่ขาด้วย PyBullet โดย:

1. สร้างโมเดล URDF แบบ simplified 2-DOF ที่หลีกเลี่ยงปัญหา closed-loop
2. พัฒนา Trot gait pattern ด้วย state machine ที่ทำงานได้อย่างราบรื่น
3. ออกแบบระบบควบคุมการทรงตัวแบบ PD controller ที่มีประสิทธิภาพ
4. บูรณาการ IK และ joint control ได้สำเร็จ

ผลการทดสอบแสดงให้เห็นว่าระบบสามารถควบคุมการเดินได้อย่างเสถียร มีความแม่นยำสูง และพร้อมสำหรับการพัฒนาต่อยอดในอนาคต

12 เอกสารอ้างอิง (References)

1. PyBullet Documentation. *PyBullet Quickstart Guide*. Available: https://pybullet.org
2. Raibert, M. H. (1986). *Legged Robots That Balance*. MIT Press.
3. Phase 1 Documentation: *Forward Kinematics of 5-Bar Linkage*. BLEGS Project, November 2024.
4. Phase 2 Documentation: *Dynamic Torque Analysis*. BLEGS Project, November 2024.
5. GitHub Repository: M-TRCH/BLEGS_Analysis-Unit, Branch: Test-Phase3

A รหัสโปรแกรมที่สำคัญ (Code Appendix)

A.1 Main Simulation Loop

Listing 4: Main loop implementation

```
while True:
    sim_time += time_step
    basePos, baseOrn = p.getBasePositionAndOrientation(robotId)

    # Warm-up logic
    if not is_walking and sim_time >= WARMUP_TIME:
        is_walking = True
        state_timer = 0.0

    # State machine
    if is_walking:
        state_timer += time_step
        if state_timer >= STEP_TIME:
            state_timer = 0.0
            gait_state = (gait_state + 1) % 2

    # Trajectory generation
    target_foot_positions_REL = {}
    if is_walking:
        # Swing/Stance logic
        ...
        # Balance control
        euler_angles = p.getEulerFromQuaternion(baseOrn)
        pitch_correction = pd_control_pitch(euler_angles[1])
        roll_correction = pd_control_roll(euler_angles[0])

    # IK and motor control
    for foot_link_name, target_pos in target_foot_positions_REL.items():
        # Apply corrections
        corrected_pos[0] += pitch_correction
        corrected_pos[1] += roll_correction

```

```
            # IK
            joint_angles = p.calculateInverseKinematics(...)

            # Send commands
            p.setJointMotorControlArray(...)

        p.stepSimulation()
        time.sleep(time_step)
```