Journal of KIIT. Vol. 24, No. 7, pp. 35-43, Jul. 31, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.7.35

4족 보행 로봇의 다중 작업 동시 처리를 위한 루프 프리셋 시스템

김동연*, 정동원**[1], 정현준**[2]

Loop Preset System for Concurrent Multi-Task Execution in Quadruped Locomotion

Dongyeon Kim*, Dongwon Jeong**[1], and Hyunjun Jung**[2]

본 연구는 2024년도 과학기술정보통신부 및 정보통신기획평가원의 “SW중심대학사업” 지원을 받아 수행되었음 (2023-0-00065)

요 약

4족 보행 로봇이 요구하는 빈번한 관절 제어와 LLM의 수 초 응답 지연 사이의 시간 규모 차이로 인해, LLM을 플래닝에만 사용하거나 블로킹 방식으로 호출하는 기존 방법들은 보행 중 채팅과 같은 다른 작업을 동시에 처리할 수 없다. 이 연구에서는 LLM이 각 관절의 목표 각도와 반복 횟수로 구성된 루프 프리셋을 생성하고, PresetDispatcher가 이를 자율적으로 실행하는 구조를 제안한다. 생물학적 척수의 중앙 패턴 발생기(CPG)와 유사하게 디스패처가 보행을 담당하는 동안 LLM은 모터 제어로부터 독립된다. 실험 결과, 동일한 30초 현실 시간 조건에서 블로킹 방식(550 스텝, 채팅 11회, 평균 응답 2.653초) 대비 시뮬레이션 보행 스텝 수 기준 466배(256,799), 약 2배 많은 채팅 완료(23회), 약 절반의 응답 지연(1.293초)을 확인했다. 이는 모터 제어를 디스패처에 위임하는 방식이 보행 로봇에서 LLM의 동시 고수준 작업 처리를 가능하게 함을 보여준다.

Abstract

Quadruped locomotion requires high-frequency joint control while LLMs incur several-second inference latencies, preventing existing blocking or planning-only approaches from handling concurrent tasks such as chat. This paper proposes a system in which the LLM generates loop presets composed of per-joint target angles and a repeat count, and a PresetDispatcher executes them autonomously, decoupling the LLM from motor control analogous to biological spinal Central Pattern Generators (CPGs). Experimentally, under a 30-second wall-time condition, our method achieves 466x more simulation locomotion steps (256,799), ~2x more chat responses (23), and ~half the response latency (1.293s) compared to a blocking baseline (550 steps, 11 chats, 2.653s). These results demonstrate that delegating motor control to a dispatcher enables concurrent high-level task processing by LLMs on quadrupedal robots.

Keywords

distributed graph database, recommender systems, collaborative filtering, cloud computing

* 국립군산대학교 소프트웨어학전공 학사과정
- ORCID: https://orcid.org/0009-0006-8048-2696
** 국립군산대학교 소프트웨어학전공 교수(**[2] 교신저자)
- ORCID[1]: https://orcid.org/0000-0001-9881-5336
- ORCID[2]: https://orcid.org/0000-0002-6717-1395

· Received: Mar. 30, 2026 Revised: Apr. 15, 2026, Accepted: Apr. 18, 2026
· Corresponding Author: Hyunjun Jung
Dept. of Software at Kunsan National University, 558, Daehak-ro, Kunsan-si, Jeollabuk-do, Republic of Korea
Tel.: +82-63-469-8917, Email: junghj85@kunsan.ac.kr

Ⅰ. 서 론

보행 로봇이 사람과 함께 공간을 공유하는 상황이 늘어나면서, 로봇에게 요구되는 능력의 범위가 넓어지고 있다[1]. 물류 창고에서 짐을 나르거나 재난 현장을 탐사하는 로봇은 이동하면서 동시에 운영자와 대화하고, 상황을 보고하고, 주변의 변화에 반응해야 한다[2]. 이러한 요구는 로봇이 보행이라는 저수준 동작을 수행하는 동안 채팅, 감정 표현, 환경 인식과 같은 고수준 작업도 함께 처리할 수 있어야 함을 의미한다. 대형 언어 모델(LLM, Large Language Model)은 자연어 이해, 상황 판단, 다중 모달 추론 등에서 높은 성능을 보이며[3], 이러한 고수준 상호작용을 처리하는 데 적합한 후보로 주목받고 있다.

그러나 LLM을 보행 로봇의 실시간 제어에 직접 사용하는 데는 구조적인 어려움이 있다. 4족 보행 로봇은 12개의 관절에 빈번한 제어 명령을 필요로 하지만[4], LLM이 하나의 응답을 생성하는 데는 수 초가 걸린다. 두 과정의 시간 차이가 매우 크기 때문에, LLM 호출 1회가 완료되는 동안 제어 루프는 수백 번의 스텝을 실행해야 한다. LLM을 제어 루프 안에 직접 배치하면, 로봇이 LLM의 응답을 기다리는 동안 이전 명령을 반복하거나 정지하게 된다. 이 대기 시간이 누적되면 보행의 연속성이 끊기고, 환경 변화에 빠르게 대응하기 어려워진다.

기존 연구들은 이 문제를 두 가지 방향으로 해결하려 했다. 첫 번째 방향은 LLM을 모터 제어로부터 분리하여 고수준 플래닝에만 사용하는 방식이다[5][6]. 이 경우 LLM은 진행 방향이나 다음 동작을 선택하고, 실제 관절 제어는 별도의 저수준 컨트롤러가 담당한다. 이 구조는 보행의 연속성을 확보할 수 있지만, LLM이 플래닝 단계에서만 개입하기 때문에 보행 중 채팅이나 상태 보고를 동시에 처리하는 것이 설계상 어렵다. 두 번째 방향은 LLM을 제어 루프 안에 배치하는 방식으로, LLM 호출이 완료될 때까지 루프가 멈춘다[7]. 이 경우 LLM이 관절 명령을 직접 생성하므로 더 세밀한 제어가 가능하지만, 호출 중 로봇이 멈추거나 이전 명령을 반복하는 문제가 발생한다. 두 방향 모두 공통적인 한계를 가진다. 첫째, LLM이 모터 제어 흐름에 연결되어 있어, 보행이 진행되는 동안 LLM이 채팅 응답이나 상태 보고와 같은 다른 작업을 처리할 수 없다. 둘째, 보행 연구에서 LLM은 플래닝 또는 스킬 선택에만 사용되고, 인간-로봇 상호작용 연구에서는 LLM이 대화를 처리하지만 보행 중인 로봇에 적용된 사례가 없다. 보행 중인 로봇이 LLM으로 채팅과 같은 고수준 작업을 동시에 처리하는 구조는 구현되거나 측정된 바가 없다.

이 연구에서는 LLM이 JSON 형식의 루프 프리셋을 생성하고, PresetDispatcher가 이를 반복 실행하는 구조를 제안한다. 루프 프리셋은 각 관절의 목표 각도 배열과 반복 횟수로 구성되며, 프리셋 하나가 수십에서 수백 번의 관절 명령을 대신 처리한다. LLM은 현재 상황을 입력받아 적절한 프리셋을 생성하고, 이후 디스패처가 해당 프리셋을 자율적으로 반복 실행하기 때문에 LLM이 다음 프리셋이나 답변을 생성하는 동안에도 로봇은 계속 동작한다. 이 구조는 척수의 중앙 패턴 발생기(CPG, Central Pattern Generator)가 뇌의 개입 없이 보행 리듬을 유지하는 방식과 유사하다[8]. CPG는 외부 입력 없이 주기적인 운동 패턴을 생성하는 신경 회로로, 보행의 기본 리듬을 독립적으로 유지한다. 이 연구에서 디스패처는 CPG와 같은 역할을 하며, LLM은 보행 리듬에 개입하지 않고도 다른 작업을 처리할 수 있다.

이 연구는 위 구조가 실제로 보행과 채팅의 동시 처리를 가능하게 함을 시뮬레이션 환경에서 측정하고, LLM을 제어 루프에 직접 사용하는 방법과 비교한다. 동일한 30초 실험 시간 안에서 보행 스텝 수, 채팅 완료 수, 평균 응답 지연을 측정하여 두 방법을 정량적으로 비교한다. 이하 논문의 구성은 다음과 같다. 제2장에서는 LLM을 보행 로봇 제어 및 인간-로봇 상호작용에 적용한 관련 연구를 검토한다. 제3장에서는 Loop Preset System의 구조를 설명하며, 프리셋 생성 방식, PresetDispatcher의 동작, 그리고 보행과 채팅의 동시 처리 구조를 서술한다. 마지막으로 제4장에서는 블로킹 방식과의 비교 실험을 기술하고 결과를 분석한다.

Ⅱ. 관련 연구

2.1 LLM과 실시간 제어의 비동기 분리

LLM의 추론 주기는 실시간 제어 주기보다 느리기 때문에, 두 과정을 동기화할 경우 제어 성능이 저하되거나 계산 비용이 과도하게 증가한다는 문제가 공통적으로 제기된다.

AsyncDriver[9]는 자율주행에서 LLM의 추론 주기가 실시간 제어 주기보다 느려 두 과정이 동기화될 경우 계산 비용이 증가하고 성능이 저하된다는 점을 지적한다. 이를 해결하기 위해 LLM 추론과 실시간 플래너를 비동기로 분리하고, LLM이 생성한 장면 연관 특징을 실시간 플래너에 전달하는 구조를 제안하였다. 기존 동기 방식 대비 계산 비용을 줄이면서 유사한 궤적 예측 성능을 달성했다. 다만 자율주행 차량을 대상으로 하며, 비동기 분리를 통해 확보된 LLM의 처리 여유로 채팅과 같은 고수준 작업을 처리하는 방식은 다루지 않는다.

HSAC-LLM[10]은 사회적 네비게이션에서 이동 제어와 자연어 대화를 별도로 처리할 경우 로봇의 반응이 부자연스러워진다는 점을 지적한다. 이를 해결하기 위해 강화학습 기반 이동 제어와 LLM 기반 음성 응답 생성을 동시에 수행하는 구조를 제안하였다. 보행자와의 상호작용 시나리오에서 이동과 음성 응답을 동시에 처리하는 실험으로 유효성을 보였다. 다만 2D 바퀴 기반 네비게이션을 대상으로 하며, 3차원 다관절 4족 보행 로봇에서 동시 작업 처리를 다루지 않는다.

2.2 4족 보행 로봇에서의 LLM 기반 제어

SayTap[5]은 LLM이 자연어 명령을 발 접촉 리듬 패턴으로 변환하여 저수준 컨트롤러에 전달할 때, 언어와 모터 제어 사이의 표현 간극이 발생한다는 점을 지적한다. 이를 해결하기 위해 발 접촉 패턴을 중간 인터페이스로 도입하고, LLM이 이 패턴을 생성하도록 설계하였다. 4족 보행 로봇에서 "천천히 트롯으로 전진해" 같은 명령부터 "오늘 소풍 가자" 같은 표현까지 보행 패턴으로 변환하는 것을 실험으로 보였다. 다만 명령 처리와 보행 실행이 순차적으로 이루어지며, 보행 중 LLM이 다른 작업을 처리하는 방식은 다루지 않는다.

다중 LLM 에이전트를 계층적으로 구성한 연구[6]는 4족 보행 로봇의 장기 태스크 수행에서 고수준 의미 이해와 다양한 이동 및 조작 스킬이 동시에 요구된다는 문제를 지적한다. 이를 해결하기 위해 의미 플래너, 파라미터 계산기, 코드 생성기, 재플래너로 구성된 다중 LLM 에이전트 계층을 제안하였다. 실제 4족 보행 로봇에서 이동과 조작을 포함한 장기 태스크를 수행하는 실험으로 유효성을 보였다. 다만 각 에이전트 단계는 순차적으로 실행되며, 보행 중 LLM이 별도 작업을 동시에 처리하는 구조는 포함하지 않는다.

2.3 LLM을 활용한 복합 태스크 처리

LaMI[11]는 대화형 로봇에서 조작, 제스처, 시선 조절 등 여러 모달리티를 일관되게 조율하는 것이 어렵다는 점을 지적한다. 이를 해결하기 위해 LLM을 중심 조율자로 두고, 각 모달리티 모듈을 LLM이 생성하는 응답 안에 자유롭게 삽입하는 구조를 제안하였다. NICOL 로봇에서 대화 중 조작과 감정 표현을 동시에 수행하는 실험으로 유효성을 보였다. 다만 대상이 정적인 대화형 로봇이며, 보행 로봇에서 이동 중 LLM이 다른 작업을 처리하는 상황은 다루지 않는다.

이중 레이어 메모리 아키텍처[12]는 단일 LLM만으로는 여러 태스크에 걸친 복잡한 추론을 처리하기에 내부 메모리가 부족하다는 점을 지적한다. 이를 해결하기 위해 인간 인지 모델에서 착안한 이중 레이어 메모리 아키텍처와 LLM을 결합하여 크로스 태스크 행동을 생성하는 방법을 제안하였다. 다수의 태스크 전환 시나리오에서 기존 방법 대비 향상된 성능을 보였다. 다만 태스크 간 전환 방식으로 동작하며, 하나의 태스크 실행 중 다른 작업을 동시에 처리하지는 않는다.

III. LLM 기반 Loop Preset을 통한 4족 보행 제어 시스템

3.1 시스템 개요

이 연구에서는 LLM이 생성한 루프 프리셋을 기

반으로 4족 보행 로봇을 제어하는 시스템을 제안한다. 시스템은 크게 프리셋 생성, 자율 실행, 병렬 처리의 세 요소로 구성된다. 각 요소는 독립적인 역할을 담당하며, 이 분리 구조가 LLM의 모터 제어 루프로부터의 독립을 가능하게 한다. 그림 1의 보행 스레드는 상황 기술(Situation description)을 LLM에 전달하여 JSON 프리셋을 생성하고, 생성된 프리셋을 PresetDispatcher가 자율적으로 반복 실행하며 관절 명령(Joint command)을 환경에 전달한다. 고수준 작업 스레드(HL task thread)는 이와 독립적으로 동작하며, 채팅 질문을 LLM에 전달하고 응답을 사용자에게 반환한다. 두 스레드는 LLM을 비동기로 공유하며, 어느 한 쪽의 LLM 호출이 다른 스레드의 실행을 블로킹하지 않는다.

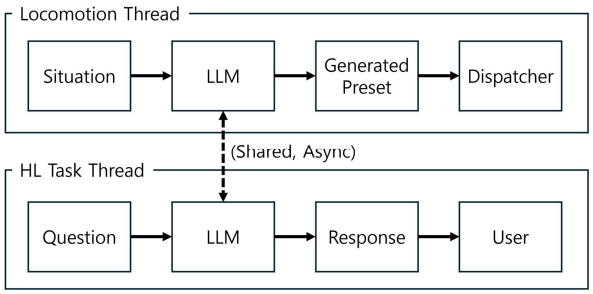

그림 1. Loop preset system 전체 구조
Fig. 1. Overview of the loop preset system

3.2 프리셋 구조 및 생성

루프 프리셋은 JSON 형식으로 표현되며, 로봇의 각 관절에 대한 목표 각도 배열과 반복 횟수로 구성된다. LLM이 실제로 생성한 루프 프리셋의 예시는 표 1과 같다. 관절의 수와 의미는 로봇 구조에 따라 달라지며, 프리셋 형식은 특정 로봇에 종속되지 않는다. 반복 횟수는 각도 배열을 몇 번 반복할지를 지정한다. 프리셋 하나가 수십에서 수백 번의 관절 명령을 대신 처리하기 때문에, LLM이 한 번의 생성 호출로 제어 루프의 긴 구간을 커버할 수 있다. 그림 2는 루프 프리셋 생성의 전체적인 흐름을 보여준다. 먼저 LLM은 현재 로봇의 상황을 자연어로 입력받아 해당 상황에 적합한 프리셋을 생성한다. 입력에는 속도, 자세, 지형 정보 등이 포함될 수 있으며, LLM은 이 정보를 바탕으로 현재 상황에서 어떤 관절 동작 패턴이 적합한지를 판단한다. 생성된 프리셋은 큐를 통해 디스패처에 전달된다. 이 생성 과정이 완료되기까지 수 초가 소요될 수 있지만, 디스패처가 이전 프리셋을 계속 실행하기 때문에 생성 지연이 보행 중단으로 이어지지 않는다.

표 1. LLM이 생성한 루프 프리셋 예시
Table 1. Example of LLM-Generated Loop Preset

```
{
  "repeat": 30,
  "actions": [
    {
      "joint_targets": [0.0, 0.4, -0.8, 0.0, 0.4, -0.8,
                        0.0, 0.4, -0.8, 0.0, 0.4, -0.8],
      "duration_steps": 5
    },
    {
      "joint_targets": [0.2, 0.2, -0.6, -0.2, 0.6, -1.0,
                        -0.2, 0.2, -0.6,  0.2, 0.6,
-1.0],
      "duration_steps": 5
    }
  ]
}
```

그림 2. 상황 기술 기반 루프 프리셋 생성 흐름
Fig. 2. Loop preset generation from situation description

3.3 PresetDispatcher

PresetDispatcher는 표 2와 같이 LLM이 생성한 프리셋을 입력받아 지정된 횟수만큼 반복 실행하고, 매 스텝마다 환경에 관절 명령을 전달하는 역할을 담당한다. 디스패처는 그림 3과 같이 실행(Executing), 요청(Requesting), 대기(Fallback)의 세 상태로 동작한다. 프리셋이 정상적으로 공급되는 동안에는 실행 상태를 유지하고, 프리셋 교체 조건이 충족되면 LLM에 다음 프리셋을 비동기로 요청한다. LLM 응답이 지연될 경우에는 직전 프리셋을 계속 반복하는 대기 상태로 전환하여 로봇이 정지하지 않도록 한다. 이 구조에서 LLM은 모터 제어 루프에 종속되지 않는다. LLM은 프리셋을 생성하는 역할만 담당하며, 실제 관절 명령의 반복 실행은 디스패처가 자율적으로 처리한다. 이는 척수의 중앙 패턴 발생기(CPG)가 뇌의 개입 없이 보행 리듬을 독

립적으로 유지하는 방식과 유사하다[8]. CPG는 외부 입력 없이도 주기적인 운동 패턴을 생성할 수 있으며, 보행 리듬의 유지와 상위 명령 처리를 분리한다. 이 연구에서 디스패처는 이와 유사한 역할을 하며, LLM은 보행 리듬에 개입하지 않고 다른 작업을 처리할 수 있다. 디스패처가 LLM과 관절 제어 사이의 결합을 끊는다는 점은 시스템 설계의 핵심이다. LLM 호출이 느리거나 실패하더라도 디스패처는 보행을 중단하지 않으며, LLM이 생성하는 프리셋의 품질과 무관하게 마지막으로 유효한 프리셋을 계속 실행한다.

채팅 스레드가 LLM을 수 초간 점유하여 새로운 프리셋 생성이 지연되는 경우에도, 디스패처는 마지막으로 유효한 프리셋을 fallback 상태에서 계속 반복 실행하므로 보행이 중단되지 않는다. 이는 CPG가 상위 신경계의 명령 없이도 보행 리듬을 독립적으로 유지하는 원리와 동일하다. 프리셋 하나가 수십에서 수백 번의 관절 명령을 커버하도록 설계되어 있어, 생성 지연이 발생하더라도 로봇은 이전 보행 패턴을 안정적으로 유지한다.

표 2. PresetDispatcher 스텝 실행 수도코드
Table 2. PresetDispatcher step execution pseudocode

```
Input:  env, preset_queue, current_preset, action_idx,
step_count, cycle_count
Output: time_step, action

if preset_queue is not empty then
    current_preset ← preset_queue.get()
    step_count  ← 0
    action_idx  ← 0
    cycle_count ← 0
if current_preset is None then
    action ← 0 // zero action
else
    phase  ← current_preset.actions[action_idx
mod K]
    action ← phase.joint_targets
time_step  ← env.step(action)
step_count ← step_count + 1
if step_count ≥ phase.duration_steps then
    step_count ← 0
    action_idx ← action_idx + 1
    if action_idx ≥ K then
        action_idx  ← 0
        cycle_count ← cycle_count + 1
        if preset is one-shot or cycle_count ≥
C_max then
            current_preset ← None
return time_step, action
```

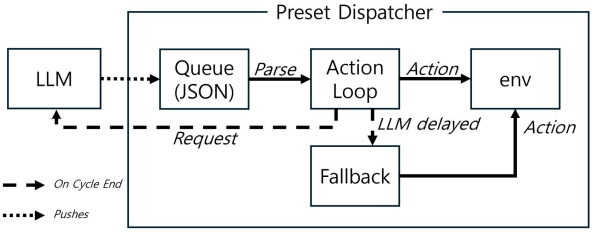

그림 3. PresetDispatcher 내부 구조 및 동작 흐름
Fig. 3. Internal architecture of the PresetDispatcher

3.4 동시 처리 구조

시스템은 표 3과 같이 두 개의 병렬 스레드로 동작한다. 보행 스레드는 PresetDispatcher를 통해 보행을 처리하고, 별도의 고수준 작업 스레드는 채팅 질문을 순차적으로 LLM에 전달하고 응답을 수집한다. 두 스레드는 독립적으로 실행되며, 고수준 작업 스레드의 LLM 호출이 보행 루프를 블로킹하지 않는다. 이 구조가 동작하기 위해서는 두 스레드가 LLM 처리 자원을 경합 없이 공유할 수 있어야 한다. 보행 스레드는 프리셋 교체 시 LLM을 호출하고, 고수준 작업 스레드는 채팅 응답을 위해 LLM을 호출한다. 두 호출이 동시에 집중되면 단일 GPU에서 동작하는 LLM이 요청을 큐잉하게 되어 채팅 응답 지연이 증가한다.

표 3. 보행과 고수준 작업의 병렬 실행
Table 3. Concurrent execution of locomotion and high-level task

```
Input:  dispatcher, chat_questions, duration T
Output: locomotion steps, chat responses

// Thread 1: Locomotion loop
while elapsed time < T do
    time_step, action ← dispatcher.step()
    if preset replacement condition met then
        dispatcher.request_preset(situation)     //
async, non-blocking

// Thread 2: High-level task (chat)
for each question q in chat_questions do
    t0       ← current time
    response ← llm.chat(q)
// blocking within Thread 2 only
    latency  ← current time − t0
    record(response, latency)

// Thread 1 and Thread 2 run independently
```

이를 방지하기 위해 프리셋 요청에는 시간 기반 쿨다운을 적용하여 교체 빈도를 제한한다. 시뮬레이션에서는 물리 연산이 실제 시간보다 빠르게 진행되기 때문에, 스텝 수 기반으로 쿨다운을 설정하면 프리셋 교체 요청이 실제 시간 기준으로 과도하게 집중된다. 시간 기반 쿨다운은 이 문제를 피하여, 채팅 스레드가 LLM 처리 자원을 안정적으로 확보할 수 있도록 한다.

IV. 실험 및 평가

4.1 실험 설계

표 4. 실험 환경 구성
Table 4. Experimental environment configuration

Operating system	Windows 11 Pro
Framework	Python 3.12.0
Library	dm_control 1.0.37, MuJoCo 3.5.0
CPU	Intel Core i9-13900K
GPU	NVIDIA RTX A6000
Memory	56 GB
LLM	Qwen3.5 9B
Wall-time budget	30s

그림 4. dm_control 시뮬레이션 환경 MuJoCo 기반 4족 보행 로봇
Fig. 4. dm_control simulation environment with the quadruped robot model

실험은 DeepMind의 물리 시뮬레이션 프레임워크인 dm_control의 quadruped/walk 태스크에서 수행했다. 로봇은 cheetah 계열의 4족 보행 로봇으로 12개의 액추에이터를 가지며, 제어 주기는 0.01초(초당 100회)이다. 12개 관절은 4개 다리 각각의 좌우 회전(yaw), 들기(lift), 뻗기(extend) 3개로 구성된다. 매 스텝마다 각 관절에 목표 각도값이 전달되며, MuJoCo의 내부 PD 컨트롤러가 현재 각도에서 목표 각도로의 힘을 계산하여 관절을 구동한다. 그 외 실험 환경은 표 4과 같으며, 실험에 사용된 4족 보행 로봇 모델은 그림 4과 같다.

4.2 비교 실험 설계

Baseline은 Code-as-Policies 계열의 방법이다. 매 50스텝마다 LLM을 호출하여 관절 목표값을 계산하고, 호출이 완료된 후 채팅을 순차적으로 1회 처리한다. 제안 모델인 Loop Preset System은 PresetDispatcher로 보행을 처리하고, 별도 스레드에서 채팅을 처리한다. 채팅 응답이 완료되면 즉시 다음 질문을 전달하는 방식으로 30초 동안 연속으로 채팅을 수행한다. 동일한 30초 실험 시간 안에서 두 방법을 비교했다.

4.3 실험 결과

표 5와 그림 5는 실험 결과를 보여준다. Locomotion Steps는 실험 시간 동안 로봇에 전달된 관절 명령의 총 횟수이며, Chats Completed는 질문 전송부터 응답 수신까지 완전히 처리된 채팅 횟수다. Avg. Chat Latency는 질문이 전송된 시점부터 응답이 도착한 시점까지의 평균 소요 시간이며, Walking Stability는 전체 스텝 중 torso_upright 관측값이 안정성 기준을 초과한 비율이다. LLM Wait Ratio는 전체 실험 시간 중 메인 루프가 LLM 응답을 기다리며 블로킹된 시간의 비율이다.

표 5. 실험 결과
Table 5. Experimental results

Metric	Baseline	Loop preset
Locomotion steps	550	256,799
Chats completed	11	23
Avg. chat latency	2.653s	1.293s
Walking stability	100%	100%
LLM wait ratio	99.6%	-

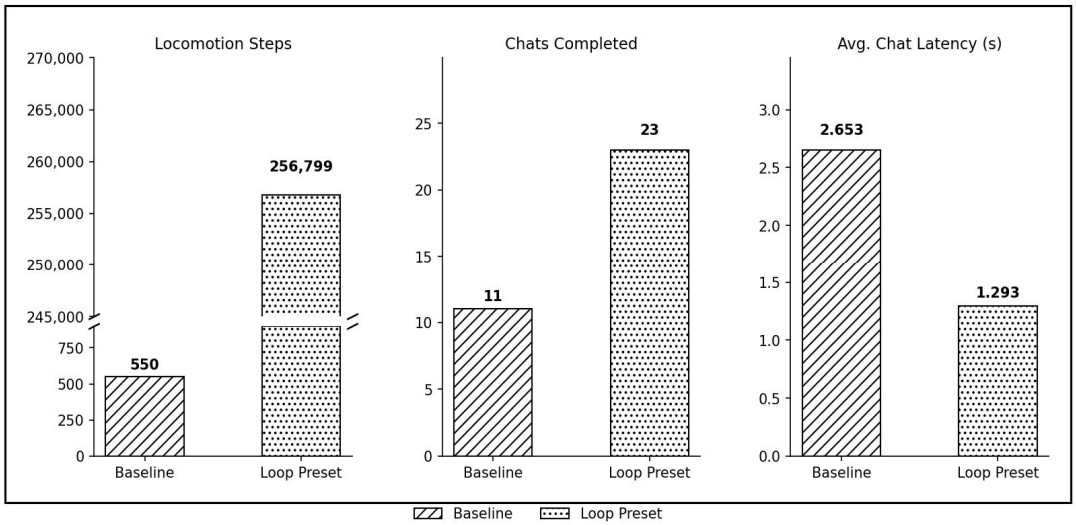

그림 5. Baseline과 Loop Preset 성능 비교
Fig. 5. Performance comparison between baseline and loop preset

Baseline은 30초 중 99.6%를 LLM 대기에 소모했으며, 총 550번의 관절 명령을 수행했다. 제안한 Loop Preset은 같은 시간에 256,799번의 관절 명령을 수행하면서 23건의 채팅을 완료했다. 평균 채팅 응답 지연은 Baseline이 2.653초, 제안 방법이 1.293초였다. 두 방법 모두 보행 안정성은 100%를 유지했다.

Ⅴ. 결론 및 향후 과제

이 연구에서는 LLM이 생성한 루프 프리셋을 PresetDispatcher가 자율적으로 반복 실행하는 구조를 제안했다. 이 구조는 LLM을 모터 제어 루프로부터 분리함으로써, 보행이 진행되는 동안 LLM이 채팅과 같은 고수준 작업을 병렬로 처리할 수 있게 한다. 기존 연구들이 LLM을 플래닝이나 스킬 선택에만 사용하거나, 제어 루프 안에 블로킹 방식으로 배치하는 방식을 취했던 것과 달리, 이 연구는 디스패처가 CPG와 유사한 방식으로 보행 리듬을 독립적으로 유지하는 동안 LLM이 다른 작업에 할당될 수 있는 구조를 설계하고 측정했다. dm_control 시뮬레이션에서 수행한 실험 결과, 제안 방법은 동일한 30초 시간 안에서 Baseline 대비 466배 많은 보행 스텝을 처리하면서, 동시에 2배 많은 채팅을 절반의 응답 지연으로 완료했다. Baseline이 실험 시간의 99.6%를 LLM 대기에 소모한 것과 달리, 제안 방법은 보행 루프와 채팅 스레드가 독립적으로 동작했으며, 보행 안정성은 두 방법 모두 100%로 동일하게 유지되었다. 이 결과는 루프 프리셋을 통한 모터 제어 위임이 보행 품질을 유지하면서도 LLM의 고수준 작업 처리 능력을 실질적으로 확보함을 보여준다. 이 연구에는 다음과 같은 한계가 있다. 실험이 dm_control 시뮬레이션 환경에서만 수행되었으며, 실제 4족 보행 로봇 하드웨어에서의 검증은 이루어지지 않았다. 시뮬레이션에서는 물리 연산이 실시간보다 빠르게 진행되기 때문에, 실제 환경에서 나타나는 센서 노이즈, 관절 마찰, 지형 불규칙성과 같은 요인이 반영되지 않았다. 또한 프리셋의 품질은 LLM이 대상 로봇의 관절 구조를 얼마나 정확하게 이해하는지에 의존하며, 보다 복잡한 보행 패턴이나 지형 적응이 요구되는 상황에서 LLM이 적절한 프리셋을 생성하는지는 추가 검증이 필요하다. 실험이 단일 로컬 GPU에서 Ollama를 통해 진행되었으므로, 서버 기반 추론 환경이나 더 빠른 추론 환경에서는 응답 지연 수치가 달라질 수 있다.

향후 연구는 다음과 같다. 첫째, 프리셋 교체 전략의 고도화이다. 이 연구에서는 시간 기반 쿨다운으로 프리셋 교체 주기를 제어했지만, 실제 환경에서는 지형 변화나 로봇 상태에 따라 교체 시점을 동적으로 결정하는 방식이 더 적합할 수 있다. 또한 프리셋 교체가 장시간 이루어지지 않을 경우 보행 패턴이 고정되어 지형 변화에 대한 적응성이 저하

될 수 있으며, 이에 대한 정량적 평가와 대응 전략도 함께 다룰 예정이다. 둘째, 보행 품질의 정량적 평가이다. 본 실험에서는 torso_upright 기반의 안정성 지표를 사용하였으나, dm_control quadruped/walk 태스크의 reward 값 및 전진 거리 등 보행 품질을 직접 반영하는 지표를 활용한 평가는 향후 연구에서 수행할 예정이다. 셋째, 실험의 반복성 및 시간 조건 다양화이다. 본 실험은 단일 조건(30초, 1회 수행)에서 진행되었으며, 반복 실험을 통한 통계적 신뢰성 확보와 다양한 시간 조건에서의 초당 처리율 비교는 후속 연구에서 보완할 예정이다. 넷째, 비차단형 LLM 제어 방식과의 비교이다. 툴 호출, 코드 실행, 안전 검증 기반의 비차단 멀티태스크 구현 방식 등 최신 접근법과의 정량적 비교는 향후 연구 과제로 남긴다.

References

[1] E. Woo, T. Kim, and D. Lee, "Design of Indoor Security Robot System Based on Autonomous Driving and Object Detection", Journal of KIIT, Vol. 21, No. 8, pp. 19-25, Aug. 2023. https://doi.org/10.14801/jkiit.2023.21.8.19.

[2] J. Gong and S. Lee, "Hierarchical Coverage Path Planning for Active SLAM", Journal of KIIT, Vol. 22, No. 3, pp. 59-70, Mar. 2024. https://doi.org/10.14801/jkiit.2024.22.3.59.

[3] W. X. Zhao, et al., "A Survey of Large Language Models", arXiv preprint arXiv:2303.18223, Mar. 2023. https://doi.org/10.48550/arXiv.2303.18223.

[4] Y. Huang, Y. Hao, B. Yu, F. Yan, Y. Yang, F. Min, Y. Han, L. Ma, S. Liu, Q. Liu, and Y. Gan, "DaDu-Corki: Algorithm-Architecture Co-Design for Embodied AI-Powered Robotic Manipulation", Proc. Int. Symp. Comput. Archit. (ISCA), Tokyo, Japan, pp. 327-343, Jun. 2025. https://doi.org/10.1145/3695053.3731099.

[5] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, "SayTap: Language to Quadrupedal Locomotion", Proc. Conf. Robot Learning (CoRL), PMLR 229, Atlanta, USApp. 3556-3570, Nov. 2023.

[6] Y. Ouyang, J. Li, Y. Li, Z. Li, C. Yu, K. Sreenath, and Y. Wu, "Long-Horizon Locomotion and Manipulation on a Quadrupedal Robot with Large Language Models", arXiv preprint arXiv:2404.05291, Apr. 2024. https://doi.org/10.48550/arXiv.2404.05291.

[7] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as Policies: Language Model Programs for Embodied Control", Proc. IEEE Int. Conf. Robot. Autom. (ICRA), London, United Kingdom, pp. 9493-9500, May 2023. https://doi.org/10.1109/ICRA48891.2023.10160591.

[8] A. J. Ijspeert, "Central Pattern Generators for Locomotion Control in Animals and Robots: A Review", Neural Networks, Vol. 21, No. 4, pp. 642-653, May 2008. https://doi.org/10.1016/j.neunet.2008.03.014.

[9] Y. Chen, Z.-H. Ding, Z. Wang, Y. Wang, L. Zhang, and S. Liu, "Asynchronous Large Language Model Enhanced Planner for Autonomous Driving", Proc. Eur. Conf. Comput. Vis. (ECCV), Milan, Italy, pp. 22-38, Oct. 2024. https://doi.org/10.1007/978-3-031-72764-1_2.

[10] C. Wen, Y. Liu, G. C. R. Bethala, S. Yuan, H. Huang, Y. Hao, M. Wang, Y.-S. Liu, A. Tzes, and Y. Fang, "Socially-Aware Robot Navigation Enhanced by Bidirectional Natural Language Conversations Using Large Language Models", arXiv preprint arXiv:2409.04965, Sep. 2024. https://doi.org/10.48550/arXiv.2409.04965.

[11] C. Wang, S. Hasler, D. Tanneberg, F. Ocker, F. Joublin, A. Ceravola, J. Deigmoeller, and M. Gienger, "LaMI: Large Language Models for Multi-Modal Human-Robot Interaction", arXiv preprint arXiv:2401.15174, Jan. 2024. https://doi.org/10.48550/arXiv.2401.15174.

[12] H. Ali, P. Allgeuer, C. Mazzola, G. Belgiovine,

B. C. Kaplan, L. Gajdosech, and S. Wermter, "Robots Can Multitask Too: Integrating a Memory Architecture and LLMs for Enhanced Cross-Task Robot Action Generation", arXiv preprint arXiv:2407.13505, Jul. 2024. https://doi.org/10.48550/arXiv.2407.13505.

저자소개

김 동 연 (Dongyeon Kim)

2021년 3월 ~ 현재 : 국립군산대학교 소프트웨어학과 학사과정
관심분야 : LLM, 머신러닝

정 동 원 (Dongwon Jeong)

1997년 2월 : 군산대학교 컴퓨터과학과(학사)
1999년 2월 : 충북대학교 전자계산학과(석사)
2004년 2월 : 고려대학교 컴퓨터학과(박사)
2005년 4월 ~ 현재 : 국립군산대학교 소프트웨어학과 교수
관심분야 : 데이터베이스, 시맨틱 서비스, 빅데이터, 사물인터넷, 엣지컴퓨팅, 지능형 융합 서비스

정 현 준 (Hyunjun Jung)

2008년 3월 : 삼육대학교 컴퓨터과학과(공학사)
2010년 3월 : 숭실대학교 컴퓨터학과(공학석사)
2017년 9월 : 고려대학교 컴퓨터·전파통신공학과(공학박사)
2017년 8월 ~ 2020년 8월 : 광주과학기술원 블록체인인터넷경제연구센터 연구원
2021년 3월 ~ 현재 : 국립군산대학교 소프트웨어학과 교수
관심분야 : 블록체인, 데이터 사이언스, 센서 네트워크, 사물인터넷, 머신러닝