A Framework Leveraging Large Language Models for Autonomous UAV Control in Flying Networks

Notebooks	DJI GCS project
보관함	☐
상태	작성 중
생성 일시	@2025년 8월 8일 오전 6:27
★ 즐겨찾기	☐
최종 편집 일시	@2025년 8월 8일 오후 3:00

***Abstract*—This paper proposes FLUC, a modular framework that integrates open-source Large Language Models (LLMs) with Unmanned Aerial Vehicle (UAV) autopilot systems to enable autonomous control in Flying Networks (FNs). FLUC translates high-level natural language commands into executable UAV mission code, bridging the gap between operator intent and UAV behaviour.**

FLUC is evaluated using three open-source LLMs – Qwen 2.5, Gemma 2, and LLaMA 3.2 – across scenarios involving code generation and mission planning. Results show that Qwen 2.5 excels in multi-step reasoning, Gemma 2 balances accuracy and latency, and LLaMA 3.2 offers faster responses with lower logical coherence. A case study on energy-aware UAV positioning confirms FLUC's ability to interpret structured prompts and autonomously execute domain-specific logic, showing its effectiveness in real-time, mission-driven control.

***Index Terms*—Flying Networks, Large Language Models**

FLUC 프레임워크 소개

- **FLUC란 무엇인가요?**
  - 오픈 소스 대규모 언어 모델(LLM)과 무인 항공기(UAV) 자동 조종 시스템을 통합하는 프레임워크입니다.
  - 플라잉 네트워크(FNs)에서 UAV의 자율 제어를 가능하게 하는 것을 목표로 합니다.
- **FLUC의 주요 기능은 무엇인가요?**
  - 높은 수준의 자연어 명령(예: "특정 위치로 이동")을 UAV가 실행할 수 있는 미션 코드로 번역합니다.
  - 이를 통해 오퍼레이터의 의도와 UAV의 실제 동작 사이의 간극을 연결해 줍니다.
- **어떤 LLM이 평가되었나요?**
  - Qwen 2.5, Gemma 2, LLaMA 3.2라는 세 가지 오픈 소스 LLM이 코드 생성 및 미션 계획 시나리오에서 FLUC와 함께 평가되었습니다.
- **각 LLM의 평가 결과는 어떠했나요?**
  - **Qwen 2.5:** 다단계 추론(multi-step reasoning)에서 뛰어난 성능을 보였습니다.
  - **Gemma 2:** 정확도(accuracy)와 지연 시간(latency) 사이에서 균형 잡힌 성능을 보여주었습니다.
  - **LLaMA 3.2:** 더 빠른 응답 속도를 제공했지만, 논리적 일관성(logical coherence)은 떨어졌습니다.
- **사례 연구를 통해 무엇을 확인했나요?**
  - 에너지 효율적인 UAV 위치 결정에 대한 사례 연구를 통해 FLUC가 구조화된 프롬프트를 해석하고 도메인별 논리를 자율적으로 실행할 수 있음을 확인했습니다.
  - 이는 실시간 미션 중심 제어에서 FLUC의 효과를 입증합니다.

제안된 프레임워크

- **기존 연구의 한계점 해결:**
  - 이전 연구들은 QoS(Quality of Service) 및 에너지 효율적인 UAV 경로에 초점을 맞추었지만, 주로 수동 제어 로직을 가정하고 계획된 미션의 자율 실행을 탐구하지 않았습니다.

    - 또한, 이 논문 저자들은 로컬에 배포된 LLM에 의해 구동되는 UAV 제어의 실제 구현이 부족하다고 지적했습니다.
    - FLUC는 이러한 한계점을 해결하기 위해 **클라우드 서비스에 의존하지 않고** 에지 디바이스가 자연어 명령을 자율적으로 해석하고 실행할 수 있도록 로컬에 배포된 LLM을 활용합니다.
- **주요 기능 및 특징:**
    - **모듈형 아키텍처:** 다양한 UAV 플랫폼 및 시뮬레이션 환경으로 확장할 수 있도록 설계되었습니다.
    - **로컬 LLM 실행:** Ollama 런타임을 사용하여 LLM을 오프라인에서 실행함으로써 낮은 지연 시간과 완전한 데이터 개인 정보 보호를 제공합니다.
    - **ArduPilot 통합:** 오픈 소스 자동 조종 시스템인 ArduPilot과 통합되어 제어 백엔드 역할을 합니다.
    - **MAVLink 호환성:** 드론 제어 프로토콜인 MAVLink와 호환되어 SITL (Software-In-The-Loop) 시뮬레이션 및 실제 UAV 배포를 모두 지원합니다. (GCS 소프트웨어)
    - **사용자 친화적 인터페이스:** 비전문가 사용자도 자연어 프롬프트를 통해 미션 목표를 정의할 수 있습니다.
    - **사전 정의된 함수 라이브러리:** `go_to_real_world_coords()`, `move_relative()`, `set_return()`와 같은 원시 함수를 포함하여 UAV 미션 로직의 의미론적 빌딩 블록 역할을 합니다.

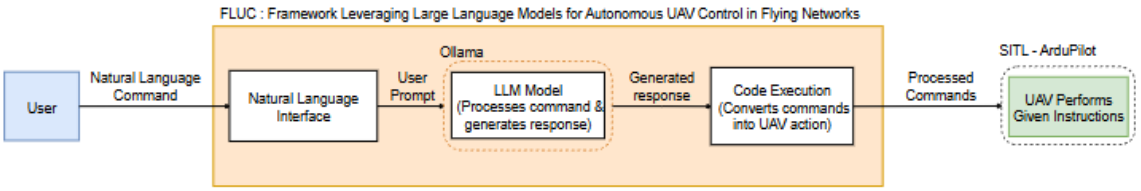

Fig. 2. FLUC system architecture: from user input to mission execution via local LLM processing and UAV control logic.

- **User (사용자)**:
    - 사용자는 FLUC 시스템과 상호작용하는 주체입니다.
    - UAV에 수행할 임무 목표를 **자연어 명령(Natural Language Command)** 형태로 제시합니다. 예를 들어, "200미터를 대각선으로 비행해"와 같은 명령이 될 수 있습니다.

- **Natural Language Interface (자연어 인터페이스)**:
  - 사용자가 입력한 자연어 명령을 시스템으로 전달하는 역할을 합니다.
  - 이 인터페이스를 통해 사용자의 명령은 다음 단계인 LLM 모델로 전달될 **사용자 프롬프트(User Prompt)**로 변환됩니다.
- **Ollama (올라마)**:
  - FLUC 프레임워크의 핵심 부분으로, 로컬에서 대규모 언어 모델(LLM)을 실행하는 경량 추론 엔진입니다.
  - **LLM Model (Processes command & generates response) (LLM 모델: 명령 처리 및 응답 생성)**:
    - 사용자 프롬프트를 입력받아 이를 처리하고, UAV가 실행할 수 있는 형태의 **생성된 응답(Generated response)**을 만들어냅니다.
    - 이 응답은 UAV 미션 코드를 포함하는 파이썬(Python) 코드와 같은 구조화된 제어 논리입니다.
    - LLM 모델은 `go_to_real_world_coords()`, `move_relative()`, `set_return()`과 같은 미리 정의된 함수 라이브러리를 활용하여 UAV 임무 논리를 구성합니다.
- **Code Execution (Converts commands into UAV action) (코드 실행: 명령을 UAV 행동으로 변환)**:
  - LLM 모델에서 생성된 응답, 즉 UAV 미션 코드를 입력받아 이를 실행 가능한 형태로 변환합니다.
  - 이 모듈은 MAVProxy를 통해 ArduPilot과 인터페이스하여, 생성된 코드를 UAV가 이해하고 수행할 수 있는 **처리된 명령(Processed Commands)**으로 바꿉니다.
- **SITL - ArduPilot (UAV Performs Given Instructions) (SITL - 아두파일럿: UAV가 주어진 지침 수행)**:
  - FLUC 프레임워크의 제어 백엔드입니다.
  - **ArduPilot**은 드론, 로버 등 다양한 무인 시스템을 제어하는 오픈 소스 자동 조종 시스템입니다.
  - *SITL (Software-In-The-Loop)**은 실제 하드웨어 없이 소프트웨어 시뮬레이션을 통해 UAV 시스템을 테스트하는 환경입니다.
  - 처리된 명령을 받아 UAV 시뮬레이션 환경(SITL)이나 실제 UAV 하드웨어에서 UAV가 주어진 임무를 자율적으로 수행하도록 합니다. 이는 UAV가 에너지 효율적인 비행 경로를 따르거나, 특정 위치로 이동하는 등 다양한 임무를 포함합니다.

Software-In-The-Loop (SITL) 에서 생성된 비행 경로

LLM 평가 시나리오

- **목표:** 각 LLM이 구조화된 미션 지향 UAV 작업을 얼마나 잘 수행할 수 있는지 평가합니다. 특히 실시간 제약 조건이 있는 시나리오에 중점을 둡니다.
- **시나리오 복잡도 증가:** 이 시나리오들은 기본적인 내비게이션 작업부터 공간 인식 및 동적 경로 계획과 관련된 고급 추론 문제에 이르기까지 복잡도가 점진적으로 증가하도록 설계되었습니다.

- **세부 시나리오:**
  - **좌표 기반 내비게이션 (Coordinates-based navigation):**
    - LLM이 "Go to 41.1783107 -8.591609 17"과 같이 원시 GPS 입력을 실행 가능한 UAV 명령으로 변환하는 능력을 테스트했습니다.
  - **의미론적 지리 위치 (Semantic geolocation):**
    - "Go to FEUP"와 같은 프롬프트를 사용하여 LLM이 OpenStreetMap [19]을 통해 명명된 위치를 해석하고 적절한 제어 로직을 생성하는 능력을 평가했습니다.
  - **상대적 움직임 (Relative motion):**
    - "Fly in a straight diagonal line for 200 meters"와 같은 지시를 통해 비행 좌표 생성 능력을 탐구했습니다.
  - **경로 최적화 (Path optimization):**

        - LLM에게 3D 좌표를 가진 5개의 웨이포인트(waypoints)를 통해 가장 효율적인 경로를 합성하도록 요청하여 공간 추론 능력을 테스트했습니다.
    - **장애물 회피 (Obstacle avoidance):**
        - LLM이 환경적 제약 조건에 따라 UAV 경로를 조정하기 위해 조건부 로직을 적용해야 하는 시나리오입니다.

Scenario	LLaMA 3.2	Gemma 2	Qwen 2.5
1: Coordinates	148.5 ± 20.7	45.8 ± 0.3	90.6 ± 39.7
2: Location	275.9 ± 57.2	33.7 ± 3.9	140.6 ± 39.8
3: Diagonal	108.0 ± 18.0	22.4 ± 0.5	55.8 ± 24.8
4: Waypoints	633.4 ± 89.4	339.0 ± 55.9	582.1 ± 105.8
5: Obstacle	411.6 ± 75.8	97.0 ± 22.7	225.6 ± 72.7

평균 토큰 사용량 비교

- **LLM별 토큰 사용량 분석:**
    - **LLaMA 3.2:** 이 모델은 모든 시나리오에서 일관되게 가장 많은 토큰을 생성했습니다. 이는 LLaMA 3.2가 다른 모델에 비해 더 장황하거나 상세한 출력을 생성하는 경향이 있음을 나타냅니다. 특히 'Waypoints' 시나리오에서는 633.4 ± 89.4 토큰으로 가장 높은 수치를 보였습니다.
    - **Gemma 2:** Gemma 2는 모든 시나리오에서 가장 적은 토큰을 사용했습니다. 이는 Gemma 2가 비교적 간결하고 효율적인 출력을 생성한다는 것을 의미하며, 특히 단순한 시나리오('Coordinates', 'Location', 'Diagonal')에서 매우 낮은 토큰 사용량을 보여주었습니다.
    - **Qwen 2.5:** Qwen 2.5는 LLaMA 3.2보다는 적지만 Gemma 2보다는 많은 토큰을 생성했습니다. 이 모델은 특히 복잡한 작업에서 더 상세하고 문맥이 풍부한 출력을 생성하는 경향이 있다고 언급되어 있습니다.

Scenario	LLaMA 3.2	Gemma 2	Qwen 2.5
1: Coordinates	**12.4 ± 1.8**	13.4 ± 0.4	32.9 ± 13.2
2: Location	25.1 ± 5.6	**10.5 ± 0.9**	49.7 ± 13.7
3: Diagonal	9.0 ± 1.4	**8.0 ± 0.9**	21.7 ± 8.7
4: Waypoints	**68.2 ± 11.4**	110.5 ± 21.4	236.4 ± 40.8
5: Obstacle	41.1 ± 9.0	**29.0 ± 5.7**	83.0 ± 25.8

LLM별 평균 실행 시간

- **모델별 성능 요약**:
  - **LLaMA 3.2**: 일반적으로 가장 빠른 실행 시간을 보였습니다. 특히 "Coordinates" 시나리오에서는 12.4 ± 1.8초로 가장 빨랐고, "Waypoints" 시나리오에서도 68.2 ± 11.4초로 다른 모델보다 훨씬 빨랐습니다. 이는 작고 가벼운 모델 아키텍처 덕분에 빠른 응답이 가능함을 시사합니다. 하지만, 논문에서는 LLaMA 3.2가 복잡한 작업에서 논리적 일관성이 낮고 불완전하거나 부정확한 출력을 자주 생성하여 재프롬프팅이 많이 필요했다고 언급합니다.
  - **Gemma 2**: 모든 시나리오에서 비교적 안정적인 실행 시간을 보였습니다. "Location" (10.5 ± 0.9초), "Diagonal" (8.0 ± 0.9초), "Obstacle" (29.0 ± 5.7초) 시나리오에서는 가장 빠른 실행 시간을 기록하며, 속도와 정확성 사이의 균형을 잘 맞추는 것으로 평가됩니다.
  - **Qwen 2.5 Coder**: 모든 시나리오에서 일관되게 가장 긴 실행 시간을 보였습니다. "Waypoints" 시나리오에서는 236.4 ± 40.8초로 다른 두 모델보다 훨씬 느렸습니다. 이는 Qwen 2.5 Coder가 더 상세하고 상황에 맞는 출력을 생성하는 경향이 있기 때문이며, 복잡한 코드 생성 작업에 최적화된 모델의 특성에서 기인합니다. 논문에서는 Qwen 2.5가 다단계 추론과 코드 생성에서 탁월한 성능을 보인다고 설명합니다.
- **가장 시간이 오래 걸리는 작업**: 모든 모델에게 "Waypoints"(경유지 최적화) 시나리오는 가장 시간이 많이 소요되는 작업이었습니다. 이는 3D 공간 추론 및 효율적인 경로 합성과 같은 복잡한 계산이 필요하기 때문입니다. 논문에서는 이처럼 높은 실행 시간이 요구되는 작업의 경우 사전 계획된, 실시간성이 중요하지 않은 미션에 더 적합할 수 있다고 언급합니다.

Scenario	LLaMA 3.2	Gemma 2	Qwen 2.5
1: Coordinates	4 / 3 / 3	**10 / 0 / 0**	5 / 4 / 1
2: Location	2 / 1 / 7	**9 / 1 / 0**	3 / 1 / 6
3: Diagonal	2 / 3 / 5	**10 / 0 / 0**	4 / 0 / 6
4: Waypoints	0 / 1 / 9	0 / 8 / 2	**5 / 5 / 0**
5: Obstacle	0 / 3 / 7	0 / 4 / 6	**4 / 4 / 2**

LLM 모델별 출력 품질 분류

- **모델별 출력 분류 (Model Output Classification)**: 각 셀의 숫자는 해당 시나리오에서 각 LLM이 10번의 시도 중 어떤 유형의 출력을 생성했는지 보여줍니다. 숫자는 '성공(Successful) / 부분적으로 올바름(Partially Correct) / 실패(Unsuccessful)' 순서로 표시됩니다.
  - **성공(Successful)**: 의도된 작업을 첫 시도에 정확하게 수행하고 추가적인 조정이 필요 없었던 출력입니다.
  - **부분적으로 올바름(Partially Correct)**: 구문은 유효했지만, 전체 작업 실행에 영향을 미치는 사소한 오류, 누락 또는 오해석이 있었던 출력입니다.
  - **실패(Unsuccessful)**: 유효하지 않거나, 프롬프트와 관련이 없거나, 실행 가능한 미션 코드를 전혀 생성하지 못한 출력입니다.

결과 분석:

- **Qwen 2.5**: 복잡한 작업(경유지 최적화, 장애물 회피)에서 특히 뛰어난 가장 일관된 성능을 보였습니다. 이는 Qwen 2.5 Coder 모델이 코드 합성에 특화되어 있기 때문이며, 출력 정확도가 중요한 미션에 적합함을 시사합니다. 예를 들어, Waypoints 시나리오에서는 5번 성공, 5번 부분적으로 올바름을 기록하며 실패가 없었습니다.
- **Gemma 2**: 단순한 작업에서는 신뢰할 수 있는 성능을 보여주었지만, 작업 복잡성이 증가함에 따라 성공률이 감소했습니다. Coordinates, Location, Diagonal과 같은 간단한 시나리오에서는 모두 10번의 성공을 기록했지만, Waypoints 및 Obstacle 시나리오에서는 성공이 없었습니다. 이는 일반적인 목적으로 설계된 모델의 한계를 보여줍니다.
- **LLaMA 3.2**: 구조화된 작업에서 실행 가능한 출력을 생성하는 데 자주 실패했으며, 종종 불완전하거나 부정확한 응답을 보였습니다. 특히 복잡한 시나리오에서는 실패율이

매우 높았습니다. 이는 모델 크기가 작아 응답 속도는 빠르지만, 복잡한 추론이나 정확한 코드 생성 능력은 떨어진다는 것을 의미합니다.

TABLE V
OUTPUT CLASSIFICATION IN THE SUPPLY CASE STUDY (10 ATTEMPTS PER MODEL).

Model	Successful	Partial	Unsuccessful
LLaMA 3.2	0	4	6
Gemma 2	**10**	0	0
Qwen 2.5	**10**	0	0

TABLE VI
AVERAGE TOKEN USAGE AND EXECUTION TIME IN THE SUPPLY CASE STUDY, WITH 95% CONFIDENCE INTERVALS

Model	Tokens	Time (s)
LLaMA 3.2	113.9 ± 10.7	9.7 ± 0.9
Gemma 2	67.9 ± 3.1	20.2 ± 1.0
Qwen 2.5	91.8 ± 29.0	35.5 ± 10.6

SUPPLY 알고리즘을 사용한 사례 연구에서 각 모델의 평균 토큰 사용량과 평균 실행 시간(초)을 95% 신뢰 구간

https://www.youtube.com/watch?v=sZbJOSk8Cc4