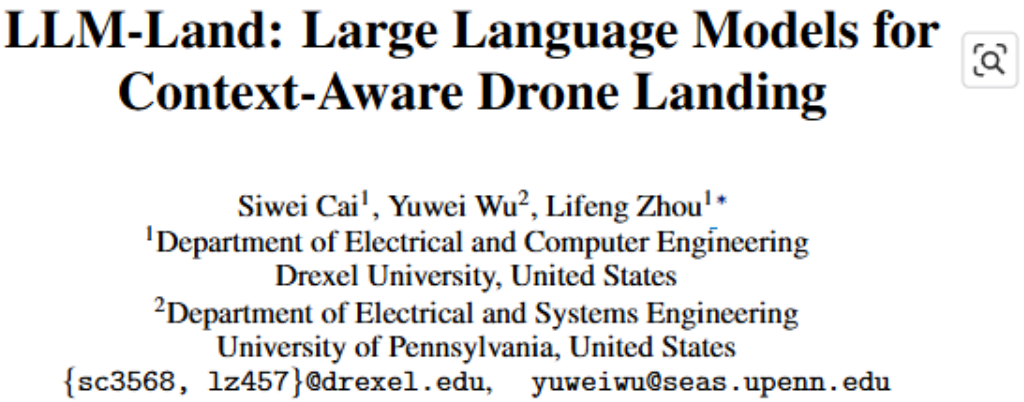
LLM-Land: Large Language Models for Context-Aware Drone Landing

Notebooks	DJI GCS project
보관함	☐
상태	인박스
생성 일시	@2025년 8월 8일 오후 1:48
★ 즐겨찾기	☐
최종 편집 일시	@2025년 8월 14일 오전 11:31



LLM-Land: Large Language Models for Context-Aware Drone Landing

Siwei Cai[1], Yuwei Wu[2], Lifeng Zhou[1]*
[1]Department of Electrical and Computer Engineering
Drexel University, United States
[2]Department of Electrical and Systems Engineering
University of Pennsylvania, United States
{sc3568, lz457}@drexel.edu, yuweiwu@seas.upenn.edu

이 연구는 기존 드론 착륙 방식의 한계를 극복하기 위해 대규모 언어 모델(LLM)과 모델 예측 제어(MPC)를 결합한 'LLM-Land' 프레임워크를 제안

Abstract

https://www.youtube.com/watch?v=9yGEpqmCtdA&feature=youtu.be

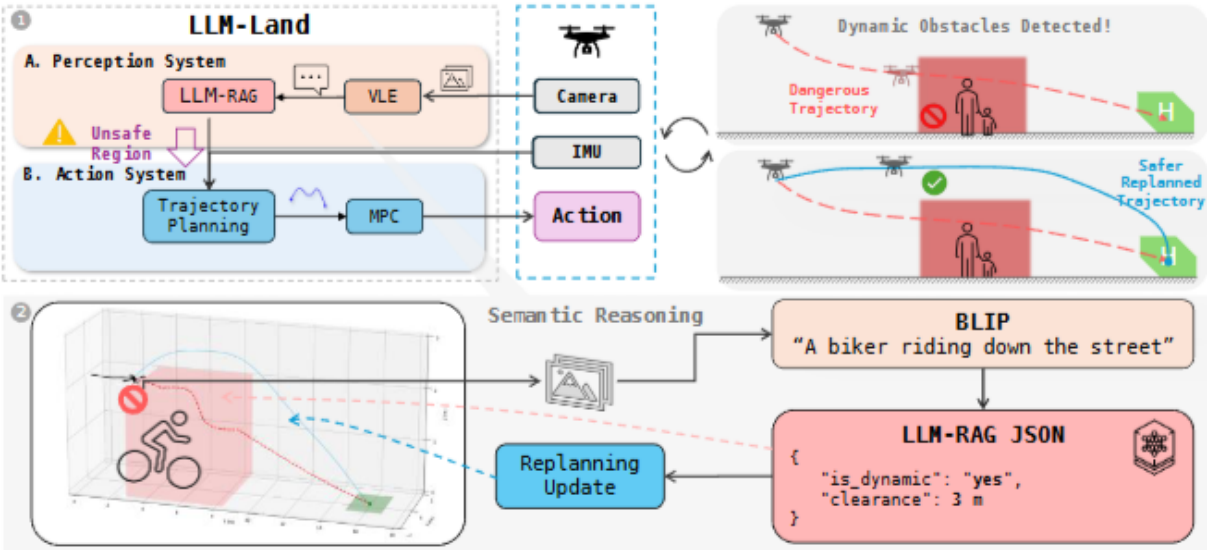
- **문제점**: 기존의 자율 드론 착륙 방식은 동적이고 구조화되지 않은 환경에서 다음과 같은 한계가 있었습니다.
  - **제한적인 시맨틱 인식**: 객체의 의미를 정확히 이해하지 못합니다. (예: 단순히 객체가 아닌 '사람' 또는 '차량'임을 인식)
  - **고정된 안전 여유**: 환경에 따라 유연하게 변경되지 않는 고정된 안전 거리에 의존합니다.
- **제안된 해결책 (LLM-Land 프레임워크)**: 이러한 한계를 극복하기 위해 대규모 언어 모델(LLM)과 모델 예측 제어(MPC)를 통합한 하이브리드 프레임워크를 제안합니다.
  - **Vision-Language Encoder (VLE)**: BLIP과 같은 VLE를 사용하여 드론의 실시간 이미지를 간결한 텍스트 장면 설명으로 변환합니다.
  - **경량 LLM (with RAG)**: Qwen 2.5 1.5B 또는 LLaMA 3.2 1B와 같은 경량 LLM이 검색 증강 생성(RAG) 기능과 함께 이 텍스트 설명을 처리합니다.
    - **장면 요소 분류**: 장면 내의 다양한 객체를 분류합니다.
    - **상황 인지 안전 버퍼 추론**: 보행자의 경우 3미터, 차량의 경우 5미터와 같이 상황에 따라 유연하게 안전 거리를 추론합니다. RAG는 LLM이 검증된 데이터에 기반하여 추론하도록 도와 '환각(hallucination)' 발생 위험을 줄입니다.
  - **모델 예측 제어 (MPC) 모듈**: LLM-RAG에서 생성된 시맨틱 플래그(semantic flags)와 안전하지 않은 영역(unsafe regions) 정보를 MPC 모듈에 전달합니다.
    - MPC는 이 정보를 바탕으로 실시간으로 궤적을 재계획하여 동적 장애물과의 충돌을 피하고 높은 착륙 정밀도를 유지합니다.
- **성과**: ROS-Gazebo 시뮬레이터에서 프레임워크를 검증한 결과, 다음과 같은 결과를 얻었습니다.
  - 기존의 비전 기반 MPC 기준선보다 일관되게 우수한 성능을 보였습니다.
  - 동적 장애물과의 근접 충돌(near-miss incidents)을 크게 줄였습니다.
  - 복잡한 환경에서도 정확한 착륙을 성공적으로 수행했습니다.

이 연구는 드론이 단순히 장애물을 인식하는 것을 넘어, 상황의 맥락을 이해하고 그에 맞춰 더욱 안전하고 지능적인 착륙 결정을 내릴 수 있도록 돕는 새로운 가능성을 제시합니다.

1 Introduction

- **드론의 중요성 및 자율 착륙의 필요성**: 무인 항공기(UAV)는 감시, 물류 배송, 인프라 검사, 환경 모니터링, 정밀 농업 등 광범위한 분야에서 필수적인 존재가 되었습니다. 이러한 애플리케이션에서 착륙은 가장 중요하고 기술적으로 까다로운 단계 중 하나입니다. 안전하고 자율적인 착륙 능력은 임무 완료를 보장할 뿐만 아니라 재충전 또는 페이로드(payload) 인계와 같은 고급 워크플로우를 가능하게 하여 드론의 지구력과 유연성을 크게 향상시킵니다.
- **기존 자율 착륙 방식의 한계**:
  - **동적이고 비정형적인 환경에서의 비효율성**: 기존의 착륙 방식은 일반적으로 고정된 환경 모델에 의존하며, 예측할 수 없는 변화나 급변하는 상황에 대한 적응성이 부족합니다.
  - **제한적인 시맨틱(Semantic) 인식**: 표준적인 인지 스택(perception stacks)은 물체의 존재와 위치는 감지할 수 있지만, '움직이지 않는 바위'와 '살아있는 존재'를 구별하는 것과 같은 의미론적 식별(semantic understanding) 능력이 부족합니다. 이는 안전한 착륙 결정에 매우 중요한 시맨틱 간극(semantic gap)을 야기합니다.
- **LLM-Land 프레임워크의 제안**: 이러한 한계를 극복하기 위해 본 논문은 Large Language Models(LLM)와 Model Predictive Control(MPC)을 통합한 하이브리드 프레임워크인 LLM-Land를 제안합니다.
  - **지각(Perception) 시스템**: 비전-언어 인코더(VLE, 예: BLIP)가 실시간 이미지를 간결한 텍스트 장면 설명으로 변환합니다. 이 설명은 Retrieval-Augmented Generation(RAG)이 적용된 경량 LLM(예: Qwen 2.5 1.5B 또는 LLaMA 3.2 1B)에 의해 처리되어 장면 요소를 분류하고 보행자 3미터, 차량 5미터와 같은 상황 인식 안전 버퍼(safety buffers)를 추론합니다.
  - **액션(Action) 시스템**: 결과로 생성된 시맨틱 플래그(semantic flags)와 안전하지 않은 영역(unsafe regions)은 MPC 모듈에 입력되어 실시간 궤적 재계획(trajectory replanning)을 가능하게 하며, 높은 착륙 정밀도를 유지하면서 충돌을 회피합니다.

- **프레임워크의 장점 및 검증**: LLM-Land 프레임워크는 ROS-Gazebo 시뮬레이터에서 기존 비전 기반 MPC 기준선(baselines)보다 지속적으로 우수한 성능을 보였습니다. 동적 장애물과의 근접 충돌(near-miss incidents)을 크게 줄이면서 혼잡한 환경에서도 정확한 착륙을 유지하는 능력을 입증했습니다.



에이전틱 AI

LLM -RAG 단에서 과거 정보 이전 스테이트 정보가 반영안됨

그걸 개선하거나 구현하면 성공률 높힐수 있음

체인적 RAG 과거 스테이트를 이해하고 진행하는 LLM-RAG 위에 만들어지는 에이전트 모듈 컨텍스트어웨어니스

실험환경

Figure 1은 LLM-Land 프레임워크의 전체적인 구조를 보여줍니다. 이 프레임워크는 드론의 안전한 자율 착륙을 위해 **인지 시스템(Perception System)**과 **액션 시스템(Action System)**의 두 가지 주요 구성 요소로 나뉩니다.

- **A. 인지 시스템 (Perception System)**
  - **역할:** 드론의 온보드 카메라에서 실시간으로 이미지를 받아, 이를 **대규모 언어 모델(LLM)**이 이해할 수 있는 텍스트 설명으로 변환하고, 이 설명을 바탕으로 환경을 분석하여 안전 관련 정보를 생성합니다.
  - **과정:**
    - **이미지 캡션화:** 드론의 카메라(IMU Camera)가 원본 이미지를 획득합니다. 이 이미지는 **Vision-Language Encoder (VLE)** (예: BLIP 모델)를 통해 간결한 텍스트 형태의 장면 설명(캡션)으로 변환됩니다.
    - **시맨틱 추론:** 이렇게 생성된 텍스트 캡션은 **검색 증강 생성(RAG)**이 적용된 경량 LLM (예: Qwen 2.5 1.5B 또는 LLaMA 3.2 1B)에 입력됩니다. LLM-RAG 모듈은 이 캡션을 분석하여 동적 장애물을 감지하고, 보행자에게 3미터, 차량에 5미터와 같이 상황을 고려한 안전 버퍼(즉, "unsafe region")를 추론합니다.
    - **출력:** 최종적으로, `{"is_dynamic": "yes", "clearance": 3 m}`와 같은 구조화된 JSON 형식으로 시맨틱 플래그(예: `is_dynamic`)와 추론된 안전 버퍼(예: `clearance`)가 액션 시스템으로 전달됩니다.
- **B. 액션 시스템 (Action System)**
  - **역할:** 인지 시스템에서 받은 안전 정보를 활용하여 드론이 충돌을 피하고 높은 착륙 정밀도를 유지하며 안전하게 착륙할 수 있도록 실시간 궤적을 재계획하고 실행합니다.
  - **과정:**
    - 인지 시스템에서 생성된 JSON 형식의 입력(시맨틱 플래그 및 unsafe region)을 받습니다.
    - 이 정보를 바탕으로 **모델 예측 제어(MPC)** 기반 플래너가 드론의 궤적을 계획합니다. MPC는 시스템의 동역학(f)과 충돌 방지 제약 조건$(x_k \in \mathcal{X}),\ (p_k \in \mathcal{P}_c)$

      을 고려하여 최적의 제어 입력$((u))$을 결정합니다.
    - 이 과정에서 LLM-RAG가 식별한 안전하지 않은 영역 B를 궤적 계획 시 명시적으로 피하도록 합니다. 즉, 가능한 공간 $\tilde{\mathcal{P}}_c$이 원래의 복도 제약 $\mathcal{P}_c$을 만족하면서도 안전하지 않은 영역 B와 겹치지 않도록 $(\tilde{\mathcal{P}}_c \cap B = \emptyset)$로 수정됩니다.

        - 결과적으로, 드론은 안전하고 정밀하게 착륙할 수 있는 궤적을 따라 비행하게 됩니다.

2 Related Works

- **기존 UAV 자율 착륙 연구의 한계:**
    - 초기 연구들([13], [14], [15])은 주로 잘 모델링된 시스템 내에서 궤적 최적화에 효과적이었으나, 예측 불가능하거나 빠르게 변화하는 착륙 환경(예: 숲, 도시)에 대한 적응성이 부족했습니다.
    - 많은 방법([28], [29])이 정밀한 착륙 정확도를 달성했지만, 주로 '빈 테스트베드(empty testbed)' 즉, 장애물이 없는 환경을 가정하여 실제 복잡한 착륙 구역의 사람, 차량, 잔해물 등을 간과했습니다.
    - 일반적인 센서 데이터 처리 방식([17])은 물체의 존재와 위치를 감지할 수 있지만, '의미론적 이해(semantic understanding)'가 부족하여 움직이지 않는 바위와 살아있는 존재를 구별하지 못했습니다. 이는 안전한 착륙 결정에 중요한 '의미론적 간극(semantic gap)'을 만듭니다.
- **LLM 기반 제어 연구의 시작과 그 한계:**
    - 최근 로봇공학 분야에서 LLM([18], [19])을 환경 이해 파이프라인에 통합하려는 시도가 있었습니다.
    - 일부 연구([30], [31])는 LLM을 사용하여 이상 탐지 및 반응적 계획을 시도했지만, LLM의 느린 응답 속도로 인해 UAV가 대기해야 하는 등 운영 효율성이 저하되는 한계가 있었습니다. 특히 Real-time anomaly detection and reactive planning with large language models 논문은 빠른 이진 분류기와 느린 LLM을 이용한 2단계 접근 방식을 제안했지만, LLM 응답 대기 시간 동안 드론이 정지해야 하는 문제가 있었습니다.
- **본 논문(LLM-Land)의 차별점:**
    - LLM-Land는 '높은 빈도의 시각 기반 장애물 분류(high-frequency, vision-based obstacle classification)'와 '궤적 재계획(triggered trajectory replanning)'을 긴밀하게 결합하여 기존 연구의 한계를 극복합니다.
    - 환경을 제어 속도(control rates)로 직접 관찰하여 인지(perception)와 행동(action)을 통합함으로써, 착륙 과정에서 정적 및 동적 장애물을 모두 처리합니다.

- 반복적인 대규모 모델 추론(large-model inference)으로 인한 지연 없이 실시간으로 안전 버퍼와 비행 경로를 업데이트할 수 있어, 복잡하고 예측 불가능한 실제 시나리오의 요구 사항에 더 가깝습니다.

3 Methods

- **전반적인 시스템 아키텍처**
  - LLM-Land 프레임워크는 UAV의 상황 인식 자율 착륙 문제를 해결하기 위해 **인지 (Perception)** 및 **행동(Action)**이라는 두 가지 긴밀하게 연결된 서브시스템으로 구성됩니다.
  - **인지 서브시스템**은 원시 이미지(raw images)를 Vision-Language Encoder(VLE)와 RAG(Retrieval-Augmented Generation)가 결합된 경량 LLM을 통해 구조화된 안전 매개변수로 변환합니다.
  - **행동 서브시스템**은 이 매개변수들을 3D MPC(Model Predictive Control) 기반 계획 프레임워크에서 활용하여 실시간으로 충돌 없는 착륙 궤적을 생성하고 실행합니다.
- **3.1 인지(Perception) 시스템**
  - **역할**: 카메라에서 들어오는 픽셀 데이터를 JSON 형식의 안전 제약 조건으로 변환합니다.
  - **VLE를 통한 장면 이해 (Scene Understanding via VLE)**
    - 드론에 탑재된 전방 카메라로부터 실시간 이미지를 수신합니다.

    ▼ 각 프레임은 BLIP(Bootstrapping Language-Image Pre-training) 모델의 요구 해상도(384x384px)로 조정되어 BLIP 모델로 전달됩니다.

    BLIP(Bootstrapping Language-Image Pre-training)은 이미지와 텍스트 데이터를 함께 학습하여 두 가지 모달리티(시각 정보와 언어 정보)를 모두 이해하고 생성할 수 있도록 설계된 비전-언어 사전 학습 모델입니다.

    - BLIP은 Vision Transformer 인코더와 자동회귀(auto-regressive) 디코더를 사용하여 간결한 "zero-shot caption"(장면 설명 텍스트)을 생성합니다.
    - BLIP-1을 사용하는 이유는 자원 제약이 있는 UAV의 온보드 메모리 및 계산 오버헤드를 최소화하기 위함입니다.

- **LLM-RAG를 통한 시맨틱 추론 (Semantic Reasoning via LLM–RAG)**
  - BLIP이 생성한 캡션은 4단계 파이프라인을 거쳐 `is_dynamic`(동적 여부)과 `z_min`(최소 고도)이라는 두 가지 JSON 필드를 출력합니다.
  - **지식 기반 및 인덱싱 (Knowledge Base & Indexing)**:
    - 드론에 내장된 JSON 구조의 지식 기반(KB)을 유지합니다. 이 KB에는 장애물 클래스(정적/동적, 사람/무생물), 최소 안전 고도, 공칭 안전 버퍼 등의 정보가 포함됩니다.
    - 문서들은 512-토큰 세그먼트로 분할되어 LlamaIndex의 VectorStoreIndex와 BAAI/bge-small-en-v1.5 임베딩을 사용하여 인덱싱됩니다.
  - **검색 (Retrieval)**:
    - BLIP 캡션은 지식 기반 인덱싱에 사용된 것과 동일한 임베딩으로 변환된 후 하이브리드 모드(시맨틱 + 키워드)로 쿼리됩니다.
    - 개념적 관련성과 정확한 키워드 일치를 모두 고려하여 결과를 병합합니다.
    - 실시간 요구사항을 충족하기 위해 상위 3개의 관련 세그먼트를 검색하여 LLM에 충분한 컨텍스트를 제공합니다.
  - **LLM 추론 및 최적화 (LLM Inference & Optimization)**:
    - Unsloth를 통해 4비트 GPTQ 양자화 및 명령-튜닝된 경량 LLM(예: Llama3.2-1B-Instruct 또는 Qwen2-1.5B-Instruct)이 실행됩니다.
    - 정적 KV 캐싱(caching)과 양자화(quantization)를 통해 VRAM 사용량을 줄이고 처리량을 두 배로 높입니다.
    - 결정론적 출력을 위해 온도를$\tau \leq 0.5$ 로 설정하고, 출력 토큰 수를 20개로 제한하며, 시퀀스 종료 시 조기 중단을 활성화하여 일정한 응답 지연 시간을 보장합니다.
  - **프롬프팅 및 게시 (Prompting & Publishing)**:
    - LLM이 `is_dynamic`과 `z_min`의 두 가지 JSON 필드만 출력하도록 최소한의 시스템 프롬프트를 사용하여 지시합니다 (자세한 예시는 Appendix A.1 참조).
    - 생성된 JSON 출력을 파싱하여 이 값을 전용 ROS(Robot Operating System) 토픽에 게시합니다.
    - 이 정보는 다운스트림 MPC 플래너에서 착륙 궤적을 조정하고 적절한 안전 버퍼 및 고도 제한을 적용하는 데 사용됩니다.

- **3.2 행동(Action) 시스템**
  - **역할**: 인지 시스템으로부터 피드백을 받아 안전하고 실행 가능한 제어 동작을 생성합니다.
  - **3D 비선형 MPC (3-D Nonlinear MPC)**

- 시스템의 상태 x는 위치 p, 속도 v, 롤 ϕ, 피치 θ, 요 ψ 각도를 포함하는 **15차원 벡터**

  $x = [p, v, \phi, \theta, \psi]\top \in Xx = [p, v, \phi, \theta, \psi]^\top \in \mathcal{X}$

  로 정의됩니다.
- 제어 입력 u는 명령된 각속도 $(\dot{\phi}_c, \dot{\theta}_c, \dot{\psi}_c)$와 추력 T_c를 포함하는 **4차원 벡터**

  $u = [\phi \dot{} c, \theta \dot{} c, \psi \dot{} c, Tc]\top \in Uu = [\dot{\phi}_c, \dot{\theta}_c, \dot{\psi}_c, T_c]^\top \in \mathcal{U}$

  입니다.
- MPC 문제는 주어진 참조 궤적 $x_k^{ref}{}_{k=0}^{N}$에 대해 유한한 예측 구간 N 동안 다음 목적 함수를 최소화하도록 공식화됩니다.

$$\min_{\boldsymbol{x},\boldsymbol{u}} J^N(\boldsymbol{x}^N, \boldsymbol{x}_{\text{ref}}^N) + \sum_{k=0}^{N-1} \left(J_{\boldsymbol{x}}^k(\boldsymbol{x}^k, \boldsymbol{x}_{\text{ref}}^k) + J_{\boldsymbol{u}}^k(\boldsymbol{u}^k) + J_{\Delta \boldsymbol{u}}^k(\boldsymbol{u}^{k+1}, \boldsymbol{u}^k) \right), \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x}^{k+1} = f(\boldsymbol{x}^k, \boldsymbol{u}^k),\ \boldsymbol{x}^0 = \boldsymbol{x}_0, \tag{1b}$$

$$\boldsymbol{u}^k \in \mathcal{U}, \quad \boldsymbol{x}^k \in \mathcal{X}, \quad \boldsymbol{p}^k \in \mathcal{P}_c, \tag{1c}$$

- $J_N(x_N, x_N^{ref})$: 예측 구간 끝(시간 단계 N)에서의 최종 상태 x_N과 최종 참조 상태 x_N^{ref} 사이 오차에 대한 페널티.
- $J_x(x_k, x_k^{ref})$: 각 시간 단계 k에서의 현재 상태 x_k와 참조 상태 x_k^{ref} 사이 추적 오차에 대한 페널티.
- $J_u(u_k)$: 제어 입력 u_k의 크기에 대한 페널티(제어 노력).
- $J_{\Delta u}(u_{k+1}, u_k)$: 연속된 제어 입력 변화량에 대한 페널티로 제어 동작의 부드러움 유도.

- 이 최적화 문제는 다음 제약 조건을 만족해야 합니다.

  $xk + 1 = f(xk, uk), x0 = x0(1b)x_{k+1} = f(x_k, u_k), \quad x_0 = x_0 \quad (1b)$

- $x_{k+1} = f(x_k, u_k)$: 시스템의 비선형 동역학 모델.
- $x_0 = x_0$: 드론의 초기 상태.

$uk \in U, xk \in X, pk \in Pc(1c) u_k \in \mathcal{U}, \quad x_k \in \mathcal{X}, \quad p_k \in P_c \quad (1c)$

- $u_k \in \mathcal{U}$: 제어 입력이 허용된 제어 입력 공간 내에 존재.
- $x_k \in \mathcal{X}$: 시스템 상태가 허용된 상태 공간 내에 존재.
- $p_k \in P_c$: 드론 위치가 충돌 없는 안전 영역 P_c 내에 존재.

  P_c는 볼록 다면체(convex polytopes)로 표현되며 실시간 동적 업데이트.

- 참조 상태 x_k^{ref}에는 원하는 위치 및 방향 구성 요소만 포함되며, 속도나 고차 미분값은 MPC 최적화를 통해 **동적 실현 가능성**(dynamic feasibility)이 보장됩니다.

원하시면 여기에 **LLM-RAG 기반 안전 영역 회피 제약**도 같은 형식으로 바로 이어 붙여 드릴 수 있습니다.

그렇게 하면 전체 MPC 공식이 완성됩니다.

- **상황 인식 실시간 궤적 업데이트 (Context-Aware Real-Time Trajectory Update)**

- 잠재적인 동적 장애물이 장면에 진입하면, 시스템은 **프런트엔드 궤적 검색**(trajectory search)과 **회랑 제약**(corridor constraints)을 업데이트하여 **실시간 재계획** 프로세스를 시작합니다.
- **LLM-RAG** 모듈은 동적 장애물의 의도를 추론하여 *안전하지 않은 영역* (unsafe regions) B를 식별하고, 이 정보를 행동 시스템에 전달합니다.
- 안전하지 않은 영역 B는 또 다른 **볼록 집합**으로 모델링됩니다.
- 장애물 회피를 보장하기 위해, 실행 가능한 공간 $\tilde{P}_c$는 기존 회랑 제약 조건 P_c를 만족하면서 B의 내부를 명시적으로 제외해야 합니다. 즉,

  $\tilde{P}_c \subseteq P_c, \quad \tilde{P}_c \cap B = \emptyset$

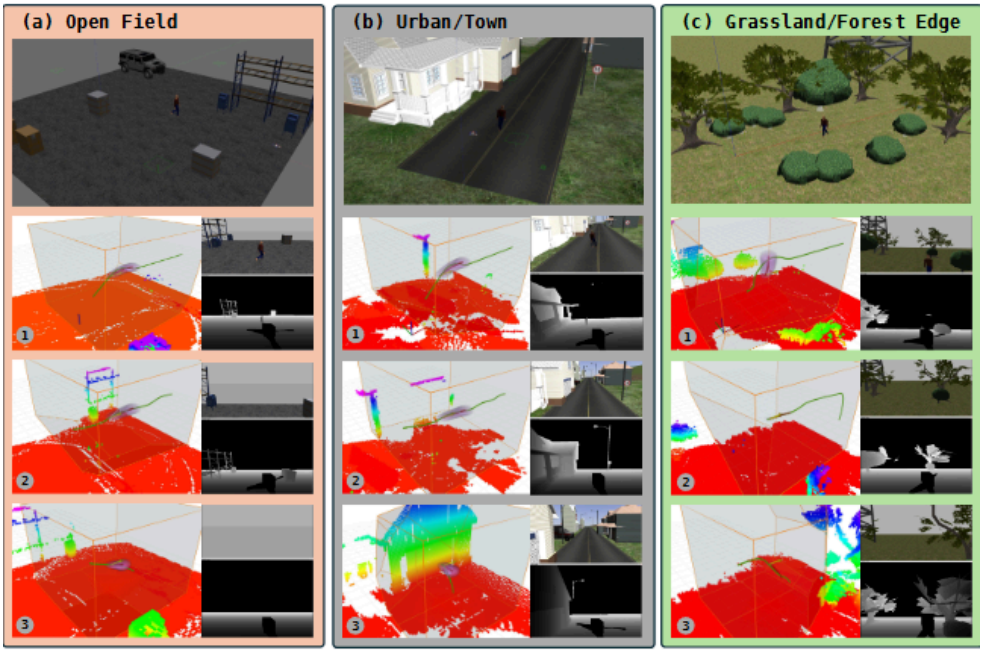
  가 성립해야 합니다.
- 궤적 검색 단계에서는 후보 경로가 식별된 안전하지 않은 영역과 충돌하는지 명시적으로 검사하여, 해당 경로를 폐기합니다.

- MPC 모듈에서는 제약 조건 공식에 사용되는 **볼록 다면체**를 수정해 안전하지 않은 영역을 명시적으로 제외함으로써, 최종 궤적이 원래 회랑의 안전한 하위 집합으로 제한되도록 보장합니다.

4 Results

4 Results

- **실험 설정 (Experimental Setup)**
  - **시뮬레이션 환경**: 모든 실험은 ROS Noetic 및 Gazebo 시뮬레이터와 rotors sim 패키지를 통합하여 구현되었습니다. 이 환경은 깊이 및 RGB-D 카메라를 포함한 사실적인 물리 및 센서 모델링을 제공합니다.
  - **하드웨어 구성**: 시뮬레이션은 Intel Core i7-12800H CPU, 16GB RAM 및 NVIDIA GeForce RTX 3070 Ti Mobile GPU (8GB VRAM)를 탑재한 랩톱에서 실행되어 실제 온보드 하드웨어 제약을 반영했습니다.
  - **평가 시나리오**: LLM-Land는 그림 2에 표시된 세 가지 대표적인 시나리오에서 평가되었습니다.



- **(a) Open Field**: 최소한의 정적 장애물이 있는 넓고 평평한 지역.
- **(b) Urban / Town**: 건물, 거리, 주차된 차량이 있는 도심 환경.
- **(c) Grassland / Forest Edge**: 야외 구조물, 나무, 덤불이 있는 반자연적 환경.

- **비교 방법 (Compared Methods)**
  이 연구에서는 세 가지 착륙 방법을 벤치마킹했습니다.
  - **Baseline MPC**: 깊이 센서의 기하학적 장애물 정보에만 의존하며 시맨틱 추론이 없는 표준 Model Predictive Control.
  - **LLM–MPC**: Baseline MPC에 BLIP으로 생성된 장면 캡션을 처리하여 상위 수준의 지침을 제공하지만 RAG가 없는 경량 LLM이 추가된 방법.
  - **LLM-Land**: RAG 메커니즘을 LLM 파이프라인에 통합하여 안정적이고 상황을 인지하는 추론을 통해 MPC를 안내하는 본 논문의 제안 프레임워크.
- **실험 1: 동적 장애물 회피 착륙 (Dynamic Obstacle Avoidance During Landing)**
  - **목표**: 동적 장애물(시뮬레이션된 사람 형태)이 착륙 경로 근처에서 예측 불가능하게 움직일 때 UAV의 착륙 성능을 테스트합니다.
  - **측정 지표**: UAV가 동적 장애물과 충돌하거나 1m 안전 버퍼에 진입하지 않고 목표 착륙 지점 0.5m 반경 내에 착륙하는 경우를 성공으로 분류했습니다.
  - **결과**:
    - Baseline MPC는 동적 장애물 존재 시 34%의 성공률을 보였습니다.
    - LLM-MPC는 LLM의 일관성 없고 의미론적으로 비일관적인 출력으로 인해 성능이 6%로 더욱 저하되었습니다.
    - LLM-Land는 가장 높은 96%의 성공률을 달성하여, 착륙 작업에서 신뢰할 수 있는 장애물 회피를 위한 상황 인지 추론의 가치를 입증했습니다. 이는 Compressed Context Memory와 같이 제한된 환경 인식에 의존하는 기존 방식의 한계를 LLM-Land가 성공적으로 극복했음을 보여줍니다.
- **실험 2: 다양한 환경에서의 성능 (Performance Across Diverse Environments)**

- **목표**: Open Field, Urban/Town, Grassland/Forest Edge 세 가지 환경에서 각 착륙 방법의 성능을 평가합니다.
- **측정 지표**: 실험 1과 동일한 성공률 지표를 사용했습니다.
- **결과**:
-
    - Baseline MPC는 가장 간단한 환경(Open Field에서 34%)에서만 보통의 성공률을 보였고, 복잡한 환경에서는 더욱 저하되었습니다.
    - LLM-MPC는 LLM의 '환각(hallucinations)' 문제로 인해 모든 시나리오에서 8%~12%의 저조한 성능을 보였습니다.
    - LLM-Land는 Open Field에서 94%, Urban에서 74%, Grassland/Forest에서 80%의 일관적으로 높은 착륙 성공률을 유지하여, 강력한 시맨틱 기반 추론과 다양한 환경 복잡성에 대한 적응성을 입증했습니다. 이는 Real-Time Anomaly Detection and Reactive Planning with Large Language Models와 같은 이전 연구에서 LLM의 실시간 적용에 대한 우려를 완화하고 있습니다.

Method	Success Rate (%)
Baseline MPC	34.0
LLM–MPC	6.0
LLM-Land	**96.0**

Method	Open Field	Urban / Town	Grassland / Forest
Baseline MPC	34.0	28.0	32.0
LLM–MPC	8.0	12.0	6.0
LLM-Land	**94.0**	**74.0**	**80.0**

- **실험 3: 제약된 하드웨어에서의 LLM 성능 벤치마크 (LLM Performance Benchmark on Constrained Hardware)**
    - **목표**: 8GB VRAM 제약 조건에서 LLM+RAG-MPC 프레임워크에 통합된 다양한 경량 LLM의 성능 트레이드오프를 평가합니다.
    - **측정 지표**:

    - **평균 피드백 지연 시간 (Avg. Latency)**: 센서 데이터 도착부터 ROS 토픽에 시맨틱 추론 결과 게시까지의 평균 경과 시간 (밀리초).
    - **최대 VRAM 사용량 (Peak VRAM Usage)**: 지각 및 추론 파이프라인에서 소비된 최대 GPU 메모리 (기가바이트).
    - **작업 성공률 (Task Success Rate)**: 충돌이나 근접 통화 없이 목표 지점 0.5m 이내에 성공적으로 착륙한 비율.
- **결과**:
    - 가장 작은 모델인 Llama-3.2-1B-Instruct는 가장 낮은 지연 시간(1453.6 ms)과 최소 메모리 사용량(1.5 GB)을 보이면서 94.0%의 성공률을 달성했습니다.
    - 중간 크기 모델(Qwen2.5-1.5B-R1-Distilled 및 Qwen2.5-3B-Instruct)은 지연 시간과 VRAM이 다소 증가하고 성공률은 낮아졌습니다.
    - 가장 큰 모델인 Llama-3.2-3B-Instruct는 가장 높은 지연 시간(2252.7 ms)과 최대 VRAM 사용량(2.9 GB)을 보여 실시간 응답성을 저해하고 MPC 입력 지연으로 인해 60.0%의 성공률을 기록했습니다.
    - 결과적으로 Llama-3.2-1B-Instruct가 제한된 하드웨어에서 가장 빠른 추론 속도, 가장 낮은 리소스 소비, 그리고 가장 높은 착륙 성공률을 제공하는 최적의 성능을 보였습니다.

LLM Model (Quantized)	Avg. Latency (ms)	Peak VRAM (GB)	Success Rate (%)
Llama-3.2-1B-Instruct	1453.6	1.5	94.0
Qwen2.5-1.5B-R1-Distilled	1510.2	2.3	70.0
Qwen2.5–3B	1889.5	2.9	82.0
Llama-3.2–3B-Instruct	2252.7	2.9	60.0

5 Conclusion and Discussion

- **문제점 인식**: 역동적인 환경에서 드론의 안전하고 신뢰할 수 있는 자율 착륙은 복잡한 과제입니다.

- **LLM-Land 프레임워크 제안**: 본 논문은 경량화된 RAG(Retrieval-Augmented Generation) 강화 LLM(Large Language Model)을 MPC(Model Predictive Control) 기반 로컬 플래너에 직접 통합하는 새로운 프레임워크를 소개합니다.
    - **목표**: 이 방법은 LLM의 시맨틱 추론 능력을 활용하여 장애물 식별, 분류 및 후속 착륙 결정을 개선하는 것을 목표로 합니다.
- **주요 실험 결과**:
    - **성능 우위**: LLM-Land는 동적 시뮬레이션 환경에서 기존의 Baseline MPC 및 RAG가 적용되지 않은 LLM-MPC 접근 방식보다 훨씬 뛰어난 성능을 보였습니다.
    - **안전성 및 성공률 향상**: 시맨틱 컨텍스트(semantic context) 통합을 통해 착륙 안전성과 성공률이 크게 향상되었습니다.
    - **경량 LLM의 효율성**: 최적화 및 양자화된 경량 LLM이 NVIDIA Jetson Orin NX와 같은 자원 제약이 있는 하드웨어에서도 실시간으로 효과적으로 실행될 수 있음을 입증하여, 고급 AI 추론과 실제 로봇 애플리케이션 간의 격차를 해소했습니다.
- **결론**: 이 연구는 UAV 제어 루프 내에서 Grounded LLM(현실에 기반한 LLM) 추론의 잠재력을 검증하며, 실제 복잡한 시나리오에 적합한 더 안전하고 상황 인지적인 항공 로봇 공학의 길을 열었습니다.
- **향후 연구 방향**:
    - RAG 지식 기반의 자율 업데이트를 통해 운영 경험으로부터 지속적인 학습을 가능하게 하는 것.
    - 컴퓨팅 자원이 개선됨에 따라 더 크고 유능한 모델을 탐색하는 것.
    - 고급 비디오 이해 모델 또는 훨씬 더 큰 파운데이션 모델(Foundation Models)을 통합하여 시스템의 상황 이해 및 실시간 의사 결정 능력을 더욱 향상시키는 것.

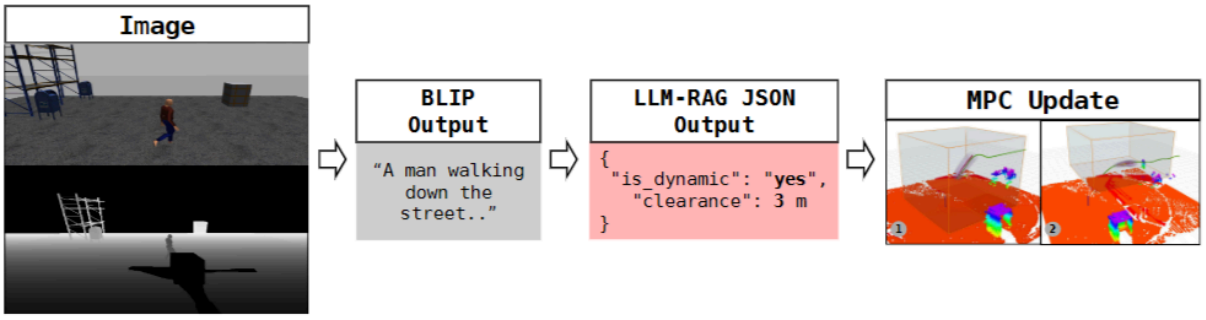
6 Limitations

- **정확한 장애물 형상 및 공간 관계 부족**:
    - 현재 인지 파이프라인은 이미지 캡션을 통해 시맨틱(의미론적) 컨텍스트만 제공합니다.
    - 이는 LLM이 장애물의 정확한 기하학적 형상이나 공간적 관계를 추론하는 데 필요한 정보를 포함하지 않습니다.

- 예를 들어, "사람"이라고 인식할 수는 있지만, 그 사람이 드론으로부터 정확히 몇 미터 떨어져 있는지, 어떤 자세로 있는지 등 정밀한 공간 정보를 파악하기 어렵습니다.
- 이러한 한계는 Describe Anything: Detailed Localized Image and Video Captioning과 같은 더욱 강력한 비전 인코더(VLE)를 사용하여 극복할 수 있습니다.

- **경량 LLM의 본질적 능력 한계**:
  - 드론에 탑재하기 위해 경량 LLM(Large Language Model)을 사용하는데, 이는 자원 제약으로 인해 불가피한 선택입니다.
  - 그러나 이 경량 모델들은 매개변수 규모나 아키텍처 복잡성 면에서 대규모 최신 모델에 비해 떨어집니다.
  - 결과적으로, 깊이 있는 논리적 추론 능력이나 복잡한 추론 능력이 제한됩니다.
  - 이는 깊은 시맨틱 이해나 복잡한 경로 계획을 필요로 하는 작업에서 시스템의 성능을 제약할 수 있습니다.
- **임베디드 하드웨어의 운영 속도 병목 현상**:
  - 실시간 자율 비행은 낮은 지연 시간(low-latency)의 의사 결정을 필요로 합니다.
  - 하지만 현재 드론에 탑재되는 임베디드 프로세서의 계산 제약으로 인해 LLM 추론 속도가 느려집니다.
  - 이러한 높은 지연 시간은 동적인 착륙 상황과 같이 제때 대응해야 하는 상황에서 복잡한 추론을 실제적으로 사용하기 어렵게 만듭니다.
  - 따라서, 중요한 시간 제약 내에서 더 정교한 추론 작업을 실행하기 위해서는 하드웨어 처리 능력과 효율성 개선이 필수적입니다.

부록



A.1 Prompt Design and LLM Behavior

The performance of lightweight language models in our system is sensitive to prompt structure and clarity. To ensure consistent output, we employ a typed system prompt that explicitly instructs the LLM to extract structured metadata only, avoiding any free-text generation. An example prompt is shown below:

```
You are a drone safety metadata extractor. Do NOT read any free text.
From the context, pull exactly two fields:
  • Classification: ‘yes’ or ‘no’
  • Minimum Altitude: a float (strip ‘meters’)

Context:
{context_str}

Examples:
Context: [Classification: no | Minimum Altitude: 0.0 meters | Text: brick building]
Q: a room with tree on the floor?
A: ‘‘‘json
{"is_dynamic": "no", "z_min": 0.0}

Return only JSON:
‘‘‘json
{
  "is_dynamic": "{Classification}",
  "z_min": {Minimum_Altitude}
}
‘‘‘
```