

화성 개척자 생활 보조를 위한 Vision Language Model 기반 자연어 명령 수행 프레임워크

A Vision-Language Model-Based Framework for Executing Natural Language Commands to Assist Martian Settlers

이승규, 오승준, 박영주, 전상우, 장대성(지도교수)

한국항공대학교 항공우주 및 기계공학부

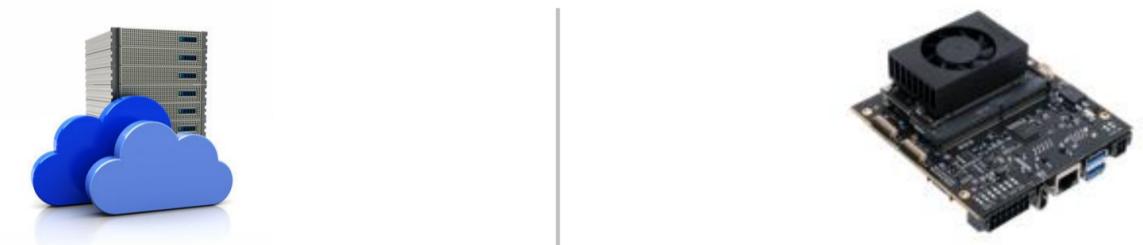


I. 연구 개요

연구 배경 및 목표

본 연구에서는 우주 거주지에서 생활 보조 작업을 자동화하기 위해, 자연어 기반 자율 로봇 프레임워크를 제안한다. 사용자 명령과 SLAM으로 구축한 환경 정보를 입력으로 활용하고, Vision Language Model(VLM)은 Actor-Critic 구조에서 로봇 동작 계획을 생성하고 타당성을 검증한다. 차량형 로봇에서 '탐지-계획-실행' 루프를 수행하며 자율적으로 임무를 처리한다. 제안된 프레임워크는 언어모델의 환경 인식 보조와 자연어 기반 고수준 계획 연구로 확장될 수 있다.

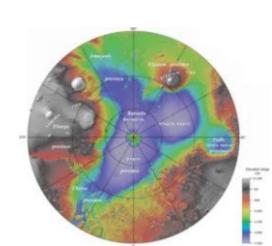
클라우드 기반 VS On-Device



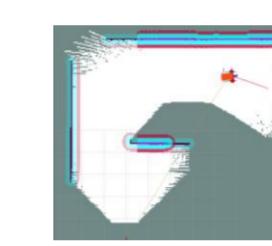
대규모 연산자원 요구
네트워크 의존성 높음

On-Device

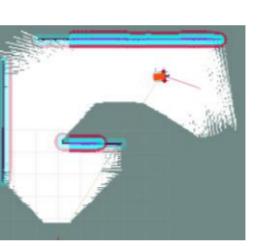
사전 정보 주어짐 VS 사전 정보 없음



네트워크 의존성 낮음
보드 내에서 실시간 처리



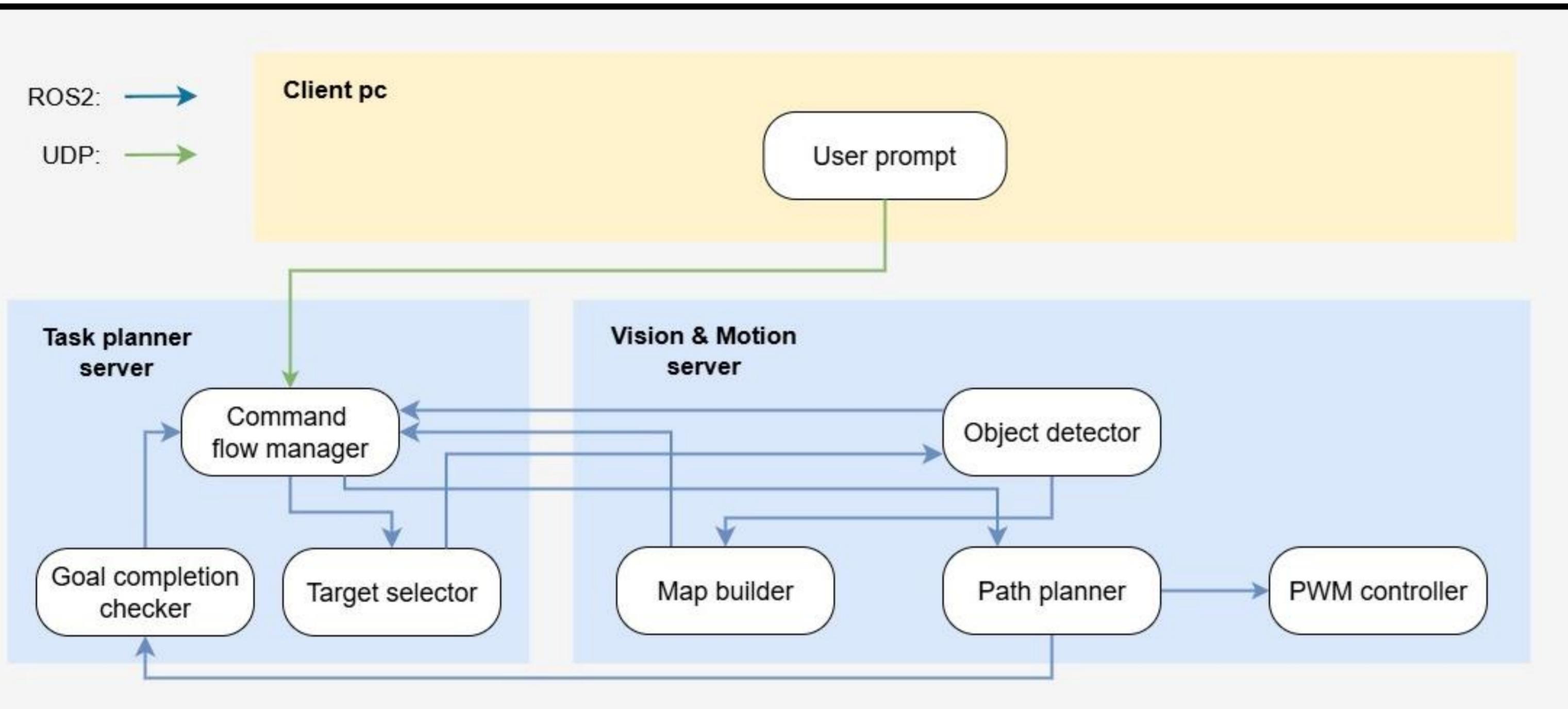
지형 데이터 제공
변화 대응 제한적



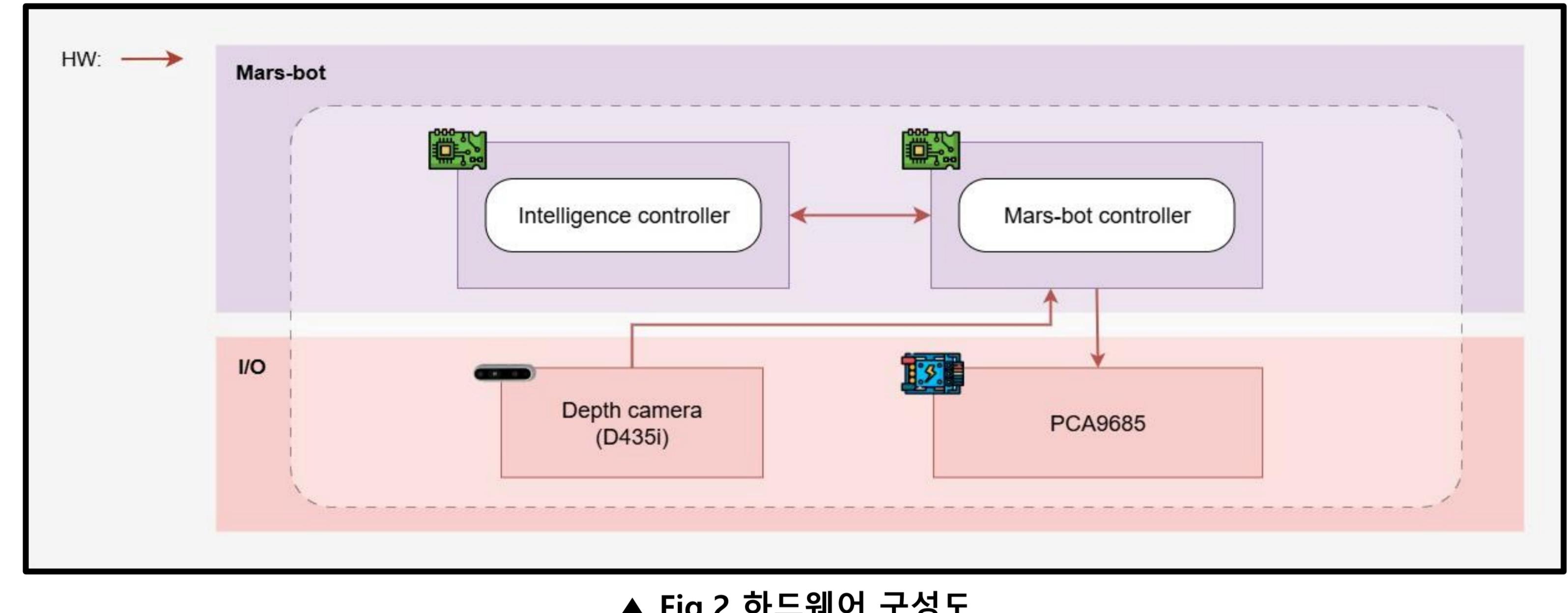
탐험을 통한 환경 정보 획득
행성, 우주 탐사에 필수적

II. 시스템 구성

1 시스템 구조

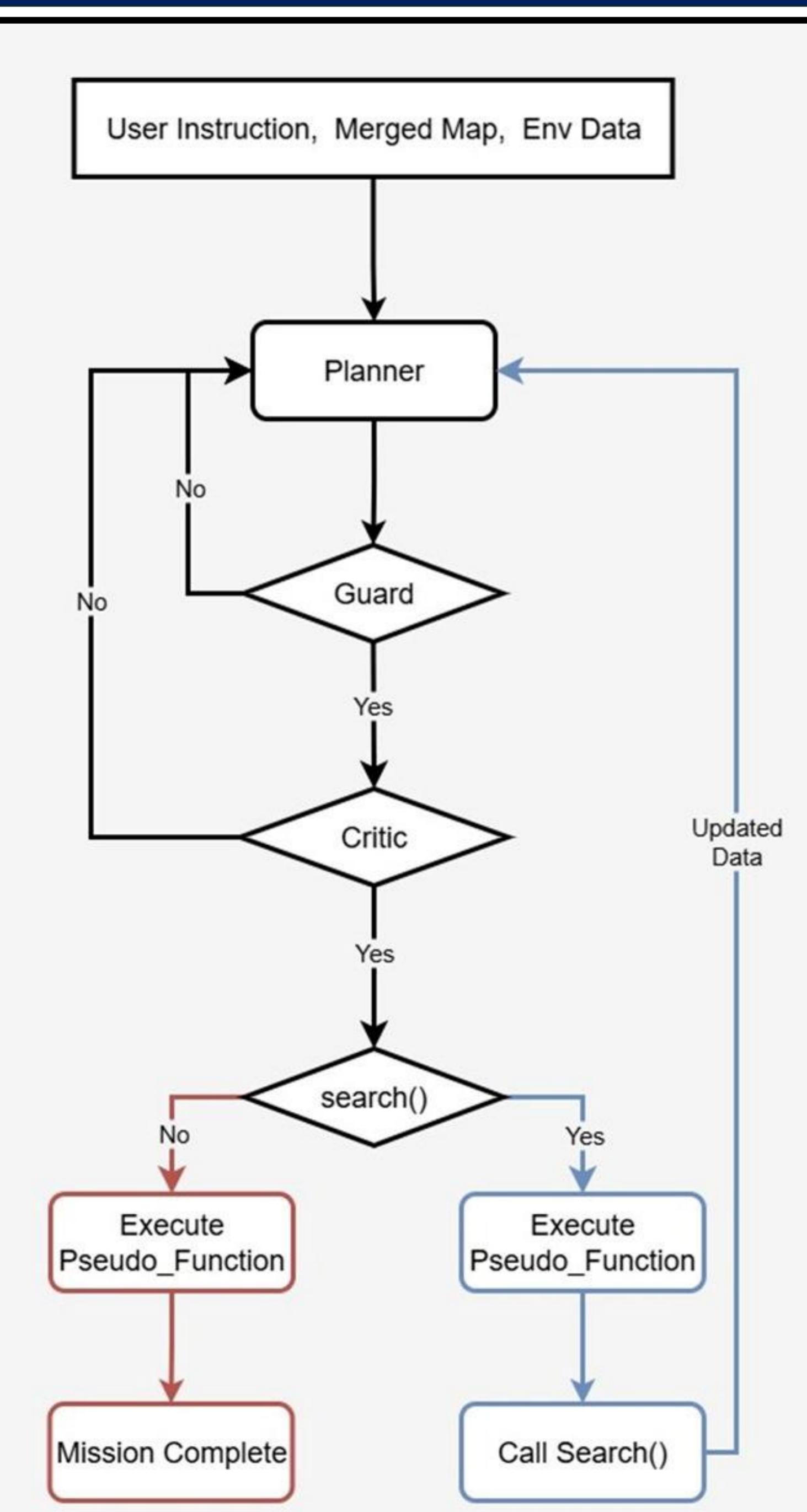


▲ Fig.1 소프트웨어 구성도



▲ Fig.2 하드웨어 구성도

2 행동 계획 알고리즘



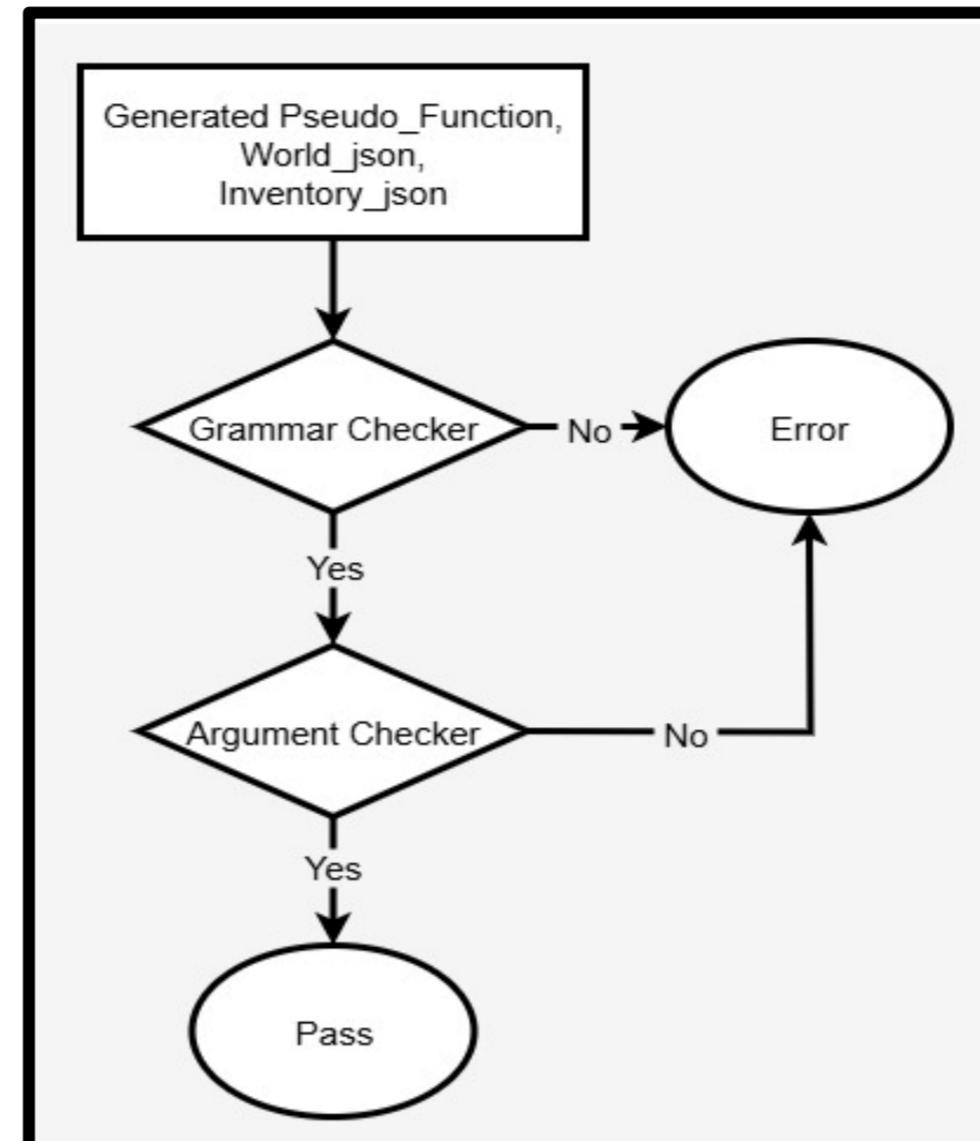
제어 함수	설명
move_to(object)	탐지된 대상(object)의 위치로 이동
pick(object)	대상을 짊어 올림 (대상 가까이에서만 실행 가능)
place(object)	대상을 자신의 자리에 내려 놓음
search(object)	임무 수행에 필요한 객체를 찾기 위해 이동, 환경 정보 업데이트
speak(message)	특정 메시지를 출력
return_to_base()	시작 지점으로 돌아옴

▲ Table.1 사전 정의된 함수 목록

요소	설명
User Instruction	사용자 자연어 명령
Merged Map	객체 검출-SLAM 결과를 통합한 2D 지도 이미지
Env Data	현재 환경의 객체 ID/클래스를 담은 JSON data
Planner	VLM 모델이 입력으로 들어온 User Instruction, Merged Map, Env Data를 기반으로 임무 수행을 위한 명령 생성
Guard	Critic 평가 전에 계획을 규칙 기반 타당성으로 점검하고, 위반 시 즉시 재계획을 요청하는 사전 필터
Critic	주위 환경 정보와 함수 실행 내역을 반영해 계획의 타당성을 점검하고, 피드백과 함께 재계획을 결정하는 평가 단계
Verdict	Critic이 생성된 계획을 검토해서 내린 최종 판결(승인 혹은 반려)

▲ Table.2 알고리즘 내부 요소

3 Actor-Critic 구조 개선



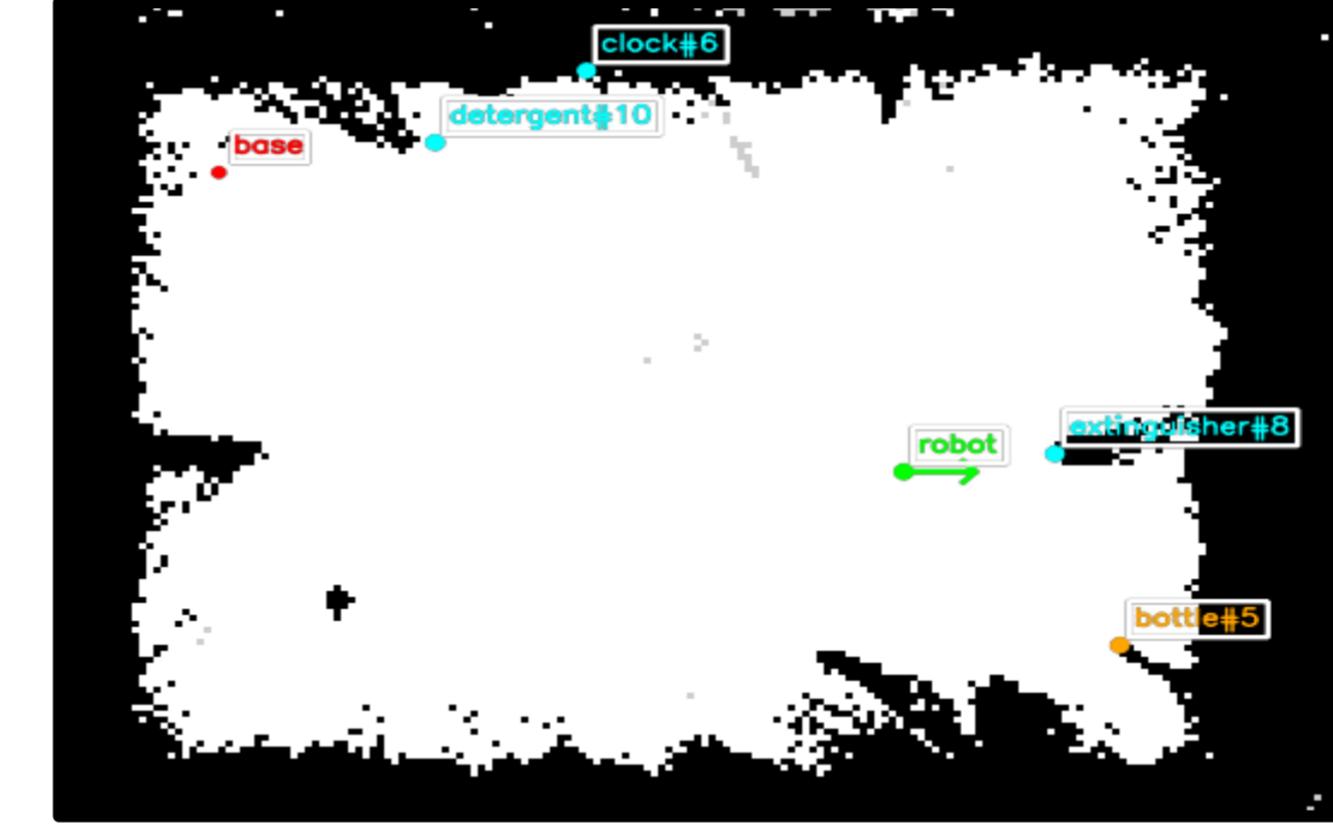
▲ Fig.4 Guard 단계 작동 구조

- On-Device 환경에서는 사용할 수 있는 언어모델의 크기가 제한되어, 상용 클라우드 기반 언어모델에 비해 성능이 낮음
- Planner가 환각이 발생하거나 지시사항을 준수하지 않아 유효하지 않은 계획을 생성했을 때 Critic이 이를 반려하지 못하는 상황 발생
⇒ 규칙 기반 안전장치인 Guard로 오류 정보 메시지를 생성하고, 이 메시지를 Planner에게 피드백하여 올바른 재계획을 유도

4 환경 인식 보조 기능

```
<World JSON>
  "objects": [
    {"id": "laundry#1", "class": "laundry"},
    {"id": "washing_machine#1", "class": "washing_machine"}
  ]
<Inventory JSON>
  "holding": [
    {"id": "laundry#1", "class": "laundry"},
    {"id": "sponge#1", "class": "sponge"}
  ]
```

▲ Fig.5 world/inventory JSON 예



▲ Fig.6 2D 지도 이미지 예

- 물체의 위치가 파악되면 Fig.5 형식으로 World JSON에 저장하고, 로봇이 pick() 함수로 집은 물체는 Inventory JSON에 따로 기록하여, 이 두 정보를 VLM에 입력한다.

- Fig.6와 같이, 주위 물체의 위치를 주행 중 SLAM으로 얻는 2D 지도 이미지 위에 표시한다. 이때 VLM은 환경 정보에 업데이트 할 객체의 종류들을 직접 결정한다.

⇒ 위 두 정보를 통해 VLM은 자신 주위에 있는 물체들 중, 임무 수행과 관련된 물체들만의 위치 정보를 선별적으로 받을 수 있다.

III. 실험 및 결과분석

본 프레임워크의 개발 목적은 화성 개척자 생활 보조이므로, 일상에서 일어날 법한 상황을 가정하여 환경을 구성하고 실험을 진행하였다. 기존 Actor-Critic 구조에 Guard를 추가한 이후 불필요하게 Critic이 호출되는 빈도가 감소하였다.

```
User Instruction: "Go to the trash bin."
=====
Reasoning: (생략)
Pseudo_Function
"python
move_to("trash bin#4")
place(" trash bin#4")
return_to_base() "
=====
[RUN] Guard 실패 → 재계획 시도 1
[GUARD] invalid: place(" trash bin#4")
Place target extinguisher#4' is not present in inventory.json.holding.
Currently holding: []
=====
PLAN (guard replan 1) =====
Reasoning: (생략)
Pseudo_Function
"python
move_to(" trash bin#4")
return_to_base() "
=====
```

▲ Fig.7 Guard의 피드백으로 재계획하는 예

```
[초기 환경 정보]
"objects": [
  {"id": "person#1", "class": "person", "map_xy": [2.01, 3.46]}
]

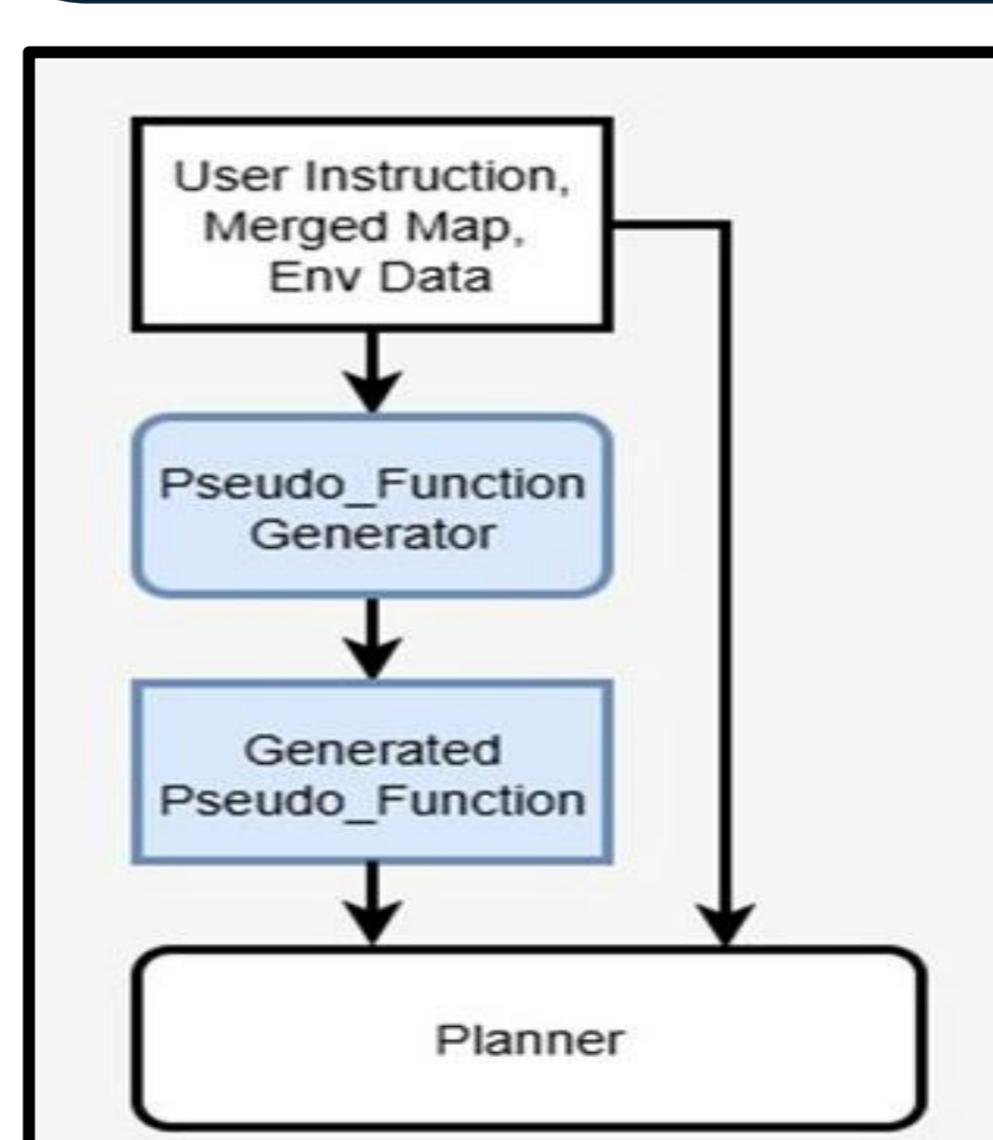
[Iteration 1]
  search("dog")
[Iteration 2]
  move_to("dog#1")
  pick("dog#1")
  move_to("person#1")
  place("dog#1")
  search("apple")
[Iteration 3]
  move_to("apple#1")
  pick("apple#1")
  move_to("person#1")
  place("apple#1")
  return_to_base()
```

▲ Fig.8 임무 수행 내역 예

- Fig.7에서는 최초에 Planner가 잘못 생성한 계획을 Guard에서 탐지하여 오류 메시지를 Planner에게 feedback하는 사례를 보여준다. 이후 Planner는 올바른 계획을 성공적으로 생성한다.
이처럼 규칙 기반의 필터링이 On-Device 환경 제약 하에서 적은 컴퓨팅 비용으로 계획 실행에 있어 최소한의 안정성을 확보할 수 있게 해준다.

- Fig.8에서는 Planner가 비교적 복잡한 임무를 수행한 사례를 보여준다.
- Planner는 시야 정보가 부족하다고 판단되면 search()를 호출해 추가 환경 탐색을 수행하고, 그 결과를 반영해 계획을 재구성한다.
- search()함수를 기점으로 계획하게 하여 한 번에 많은 명령을 생성해야 하는 부담을 줄인다.

IV. 결론



▲ Fig.9 제어 함수 생성 알고리즘

본 연구는 VLM이 자연어 명령을 받아 계획을 생성하고 이를 순차적으로 실행해 차량형 로봇을 제어하는 프레임워크를 제안한다. VLM은 객체 검출과 SLAM을 통합한 2D 지도 이미지로 환경을 인식하며, Actor-Critic 구조로 계획을 수립·검증·개선한다.

향후 연구에서는 Fig.9와 같이 고정된 제어 함수 목록 뿐만 아니라, 언어 모델이 상황에 맞는 새로운 제어 함수를 스스로 설계·정의하는 방향으로 확장할 예정이다. 이를 통해 수행 가능한 임무의 범위를 넓히고, 예상치 못한 환경에 대해서도 최소한의 사전작업으로 신속한 임무 수행이 가능하게 하는 것을 목표로 한다.