LLM for Robotics:

Two papers:

- A Survey on Integration of Large Language Models with Intelligent Robots

- Large Language Models for Robotics: Opportunities, Challenges, and Perspectives

A Survey on Integration of Large Language Models with Intelligent Robots

Kim, Y., Kim, D., Choi, J. et al. A survey on integration of large language models with intelligent robots. Intel Serv Robotics 17, 1091–1107 (2024)

Presented By
Rishov Paul (vst2hb)
Radowan Mahmud Redoy (snf4za)

Rishov Paul (vst2hb)

Importance of Surveys in Robotics Research

1 DATA COLLECTION

Surveys effectively gather quantitative and qualitative data, crucial for informed decision-making.

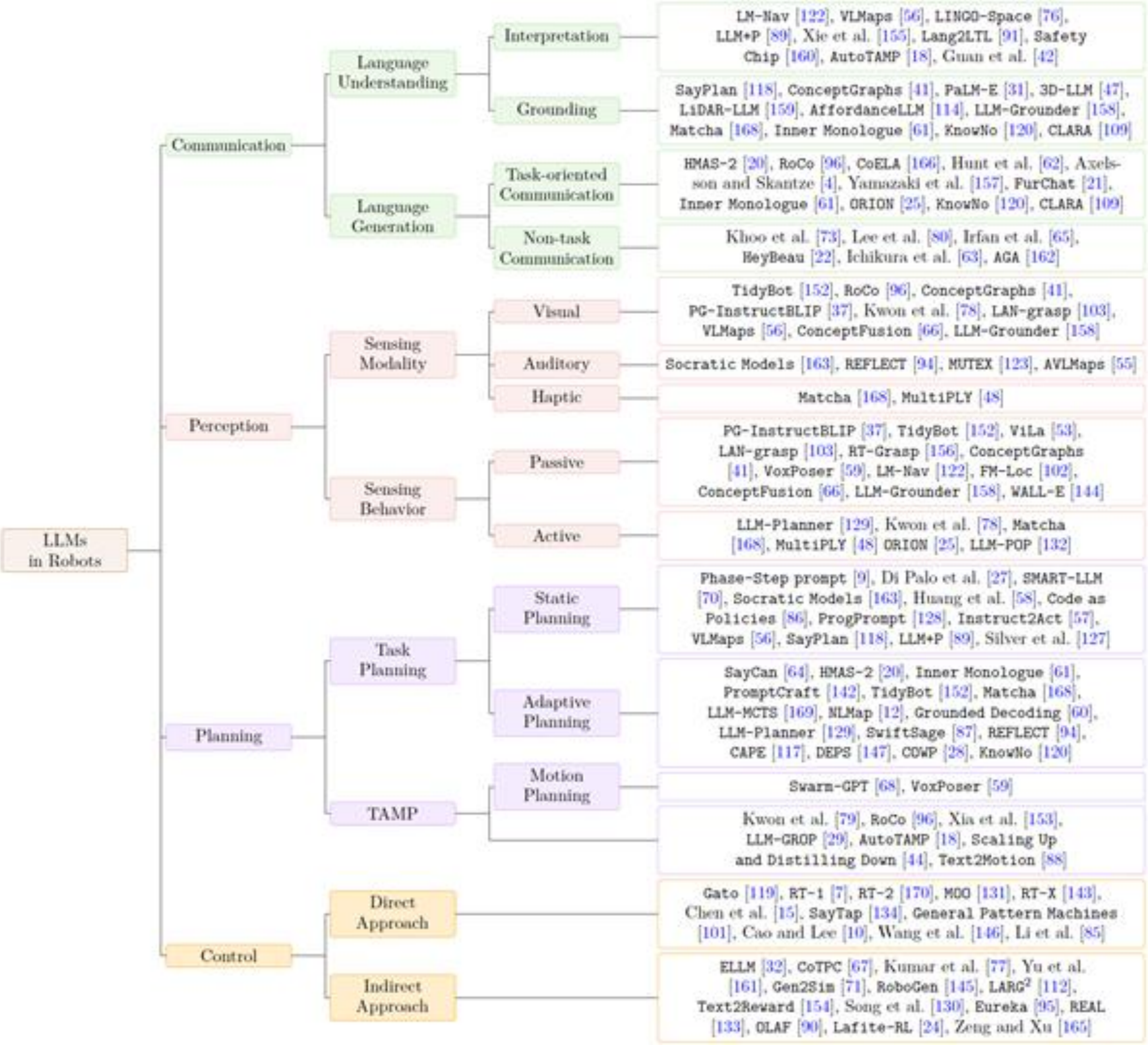
2 TREND ANALYSIS

Surveys help identify trends over time, especially in the adoption of advanced technologies.

3 FEEDBACK MECHANISM

Surveys provide a channel for stakeholders to express their views for system improvements.

Overview structure of intelligent robotics research integrated with LLMs in this survey



LLMs and Their Impact on Robotics

Enhancing Robotics with LLM Capabilities

Challenges

- LLMs often generate **inaccurate** or unexpected responses
- Raises **safety** of robot execution
- Must include **filtering and correction mechanisms** to ensure safe deployment
- Lack of **systematic prompt design guidelines** hindering seamless integration of key components

Research Focus of the Survey

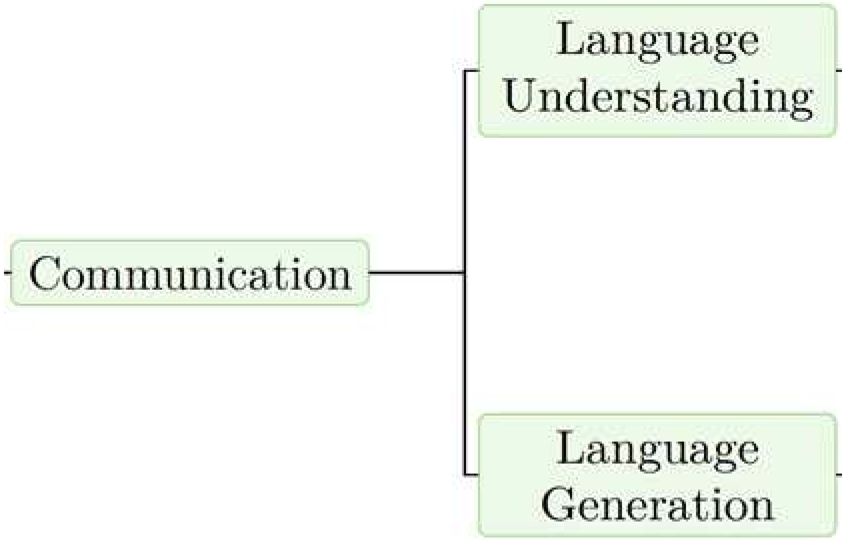
Question 1	Question 2	Question 3
How are LLMs utilized in robotics domains?	What are the integration limitations, and how can they be addressed?	What prompt structures are required for basic functionality?

Communication in Robotics

Enables robots to **interact effectively** with humans and other agents

Two Key Areas:

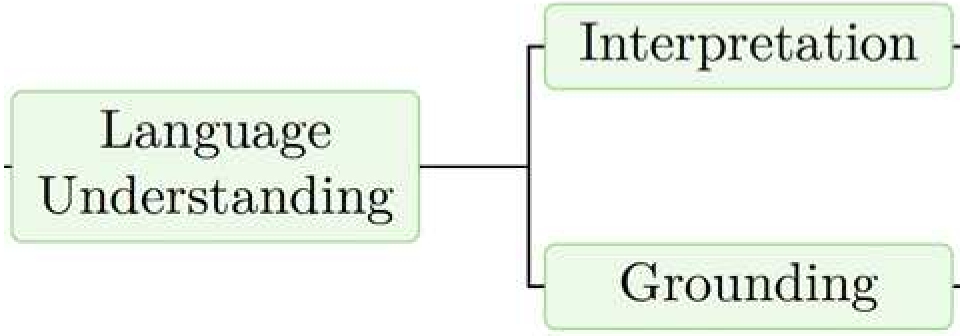
- Language understanding
- Language generation



Language Understanding

Challenges:

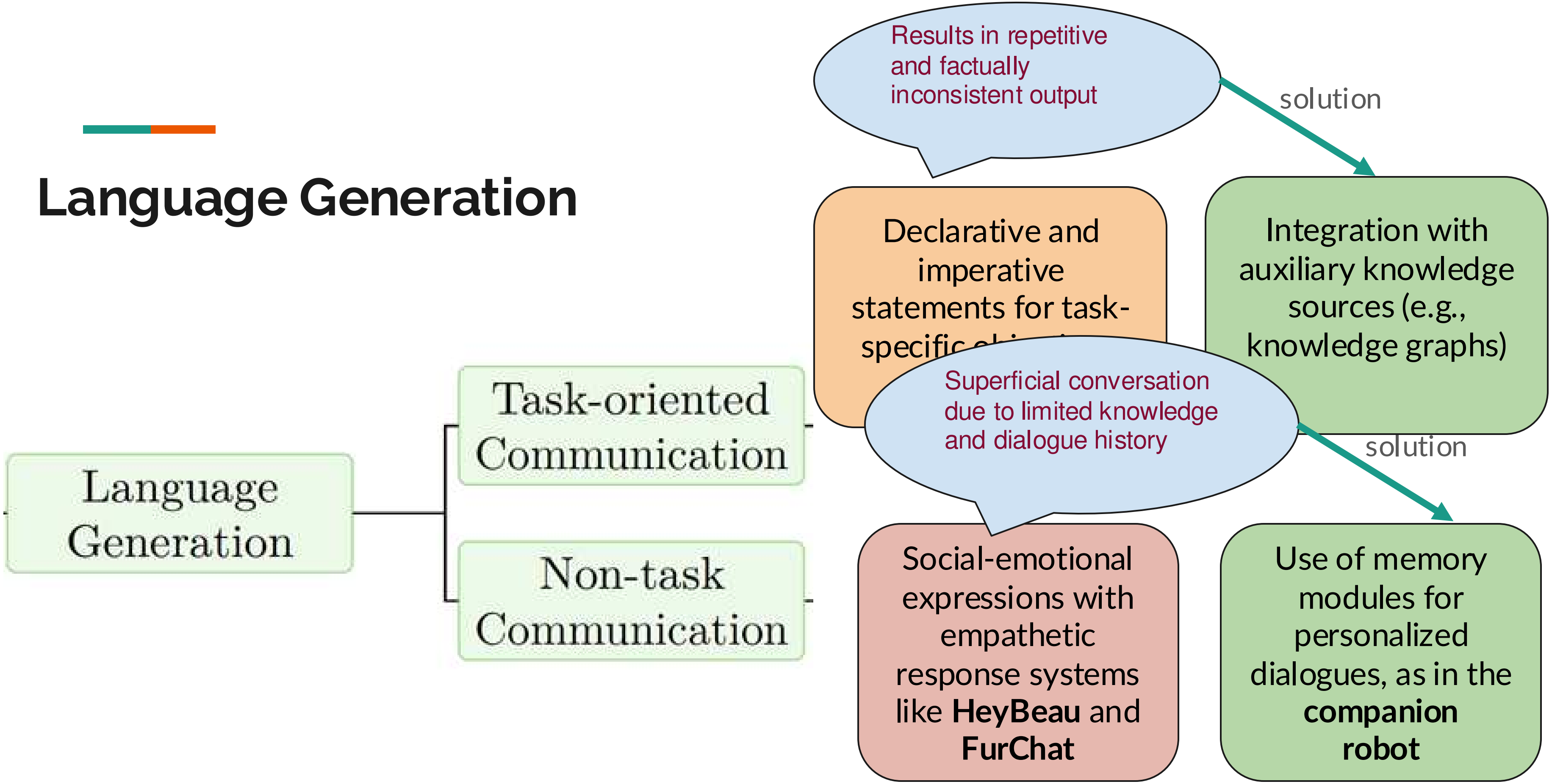
- Syntax and semantics inconsistencies
- Dependency on detailed world models



Transforming natural language into semantic representations like **PDDL** and **LTL**

Mapping linguistic expressions to targets such as behaviours or objects using tools like **ConceptGraphs** and **LLM-Grounder**

Language Generation

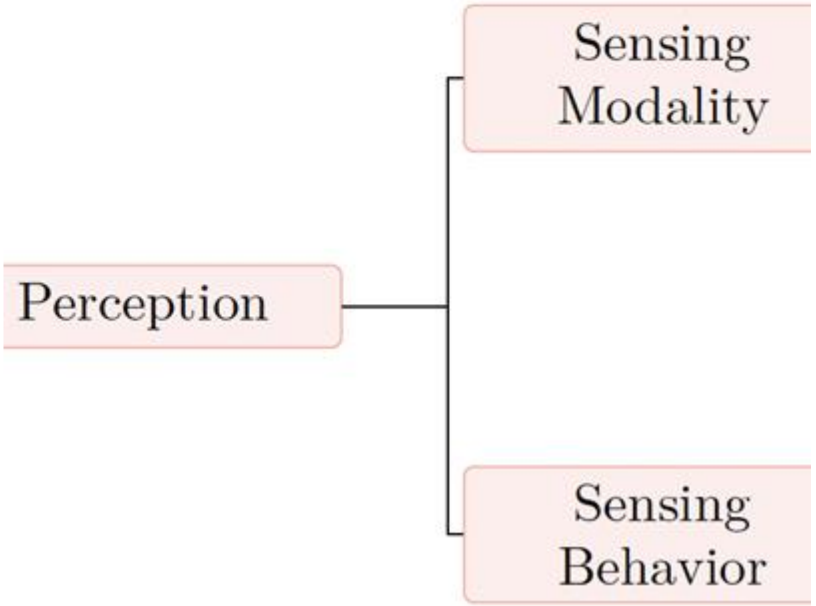


Perception in Robotics

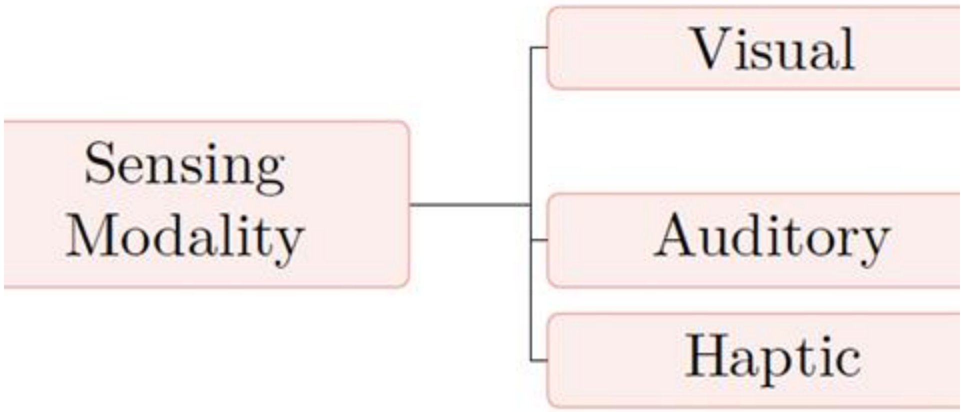
Plays a crucial role in enabling robots to **make decisions**, **plan their actions**, and **navigate** the real world

Two Key Areas:

- Sensing Modality
- Sensing Behavior



Sensing Modality



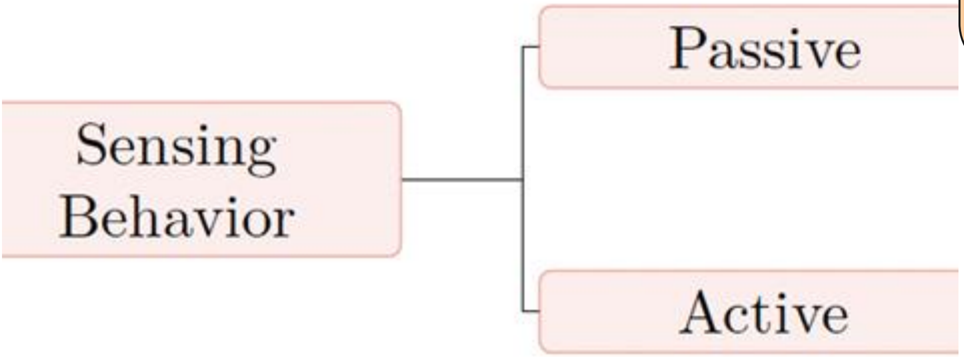
Models like **CLIP** and **InstructBLIP** enable object identification and scene understanding

Involves the interpretation of sound. **AudioCLIP** and **AVLMaps** integrate sound-based context

Involves the interpretation of contact information. **MultiPLY** combines tactile data with language for interactive perception

Sensing Behavior

limits the ability to perform tasks when information is unobserved or unavailable (e.g., unseen area, weight)



Gathering sensory data without taking active actions

TidyBot identifies the **nearest object** from an **overhead view** and **classifies** its category using a **closer camera view**

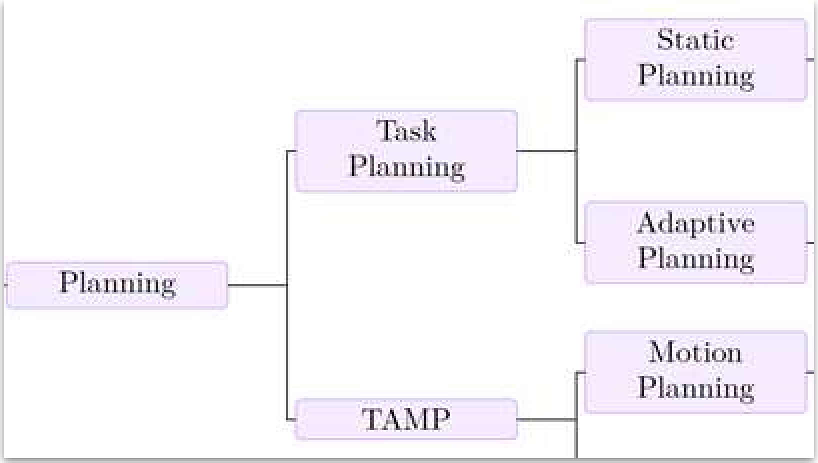
Actively seeking out sensory information to improve understanding

LLM-Planner generates seeking actions such as '**open the refrigerator**' to locate invisible object

Planning in Robotics

The process of organizing and sequencing actions to achieve specific objectives

Encompasses high-level task planning and low-level motion planning

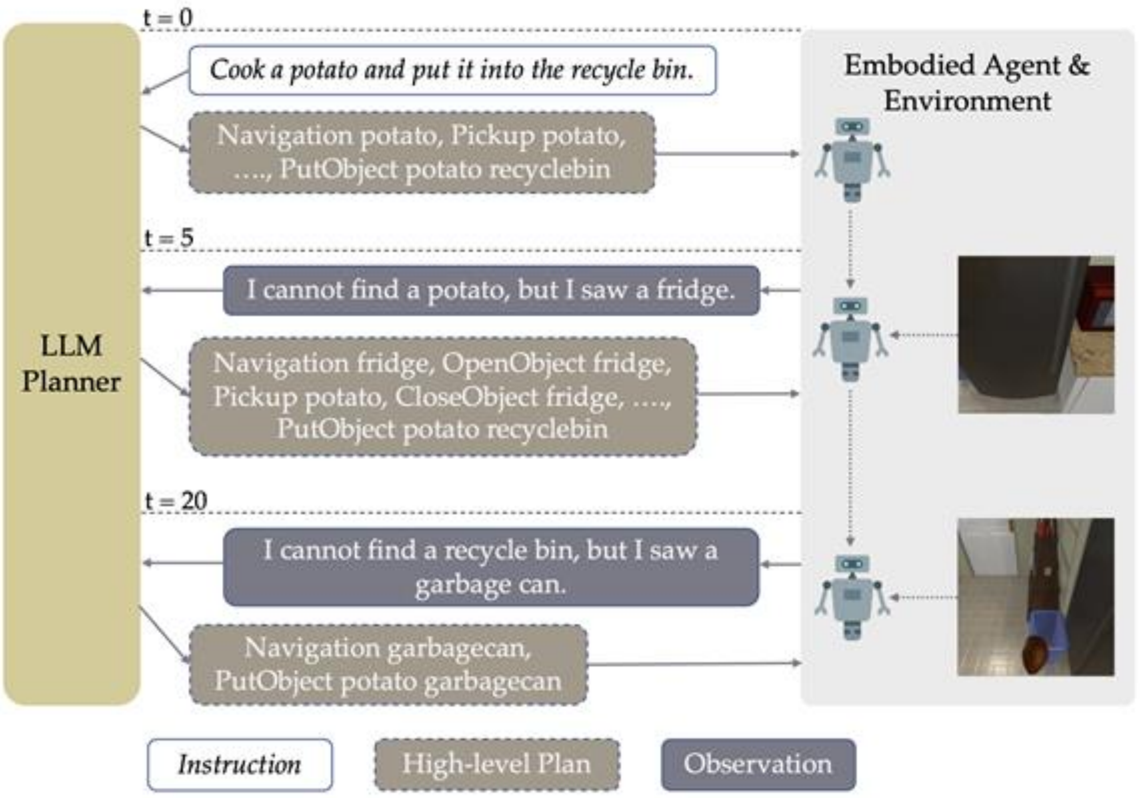


Task Planning

- **Static Planning**
  - Fixed plans generated without adapting to environmental changes.
  - Example: **SayPlan** uses PDDL planners to create predefined action sequences.
- **Adaptive Planning**
  - Plans dynamically adjusted based on real-time feedback and observations.
  - Example: **LLM-Planner** adapts to new scenarios by integrating environmental feedback

LLM-Planner

Figure 1: An illustration of LLM-Planner for high-level planning. After receiving the natural language instruction ($t = 0$), LLM-Planner first generates a high-level plan by prompting a large language model (*e.g.*, GPT-3). When the embodied agent gets stuck during the execution of the current plan ($t = 5$ and 20), LLM-Planner re-plans based on observations from the environment to generate a more grounded plan, which may help the agent get unstuck. The commonsense knowledge in the LLM (*e.g.*, food is often stored in a fridge) allows it to produce plausible high-level plans and re-plan based on new environmental perception.

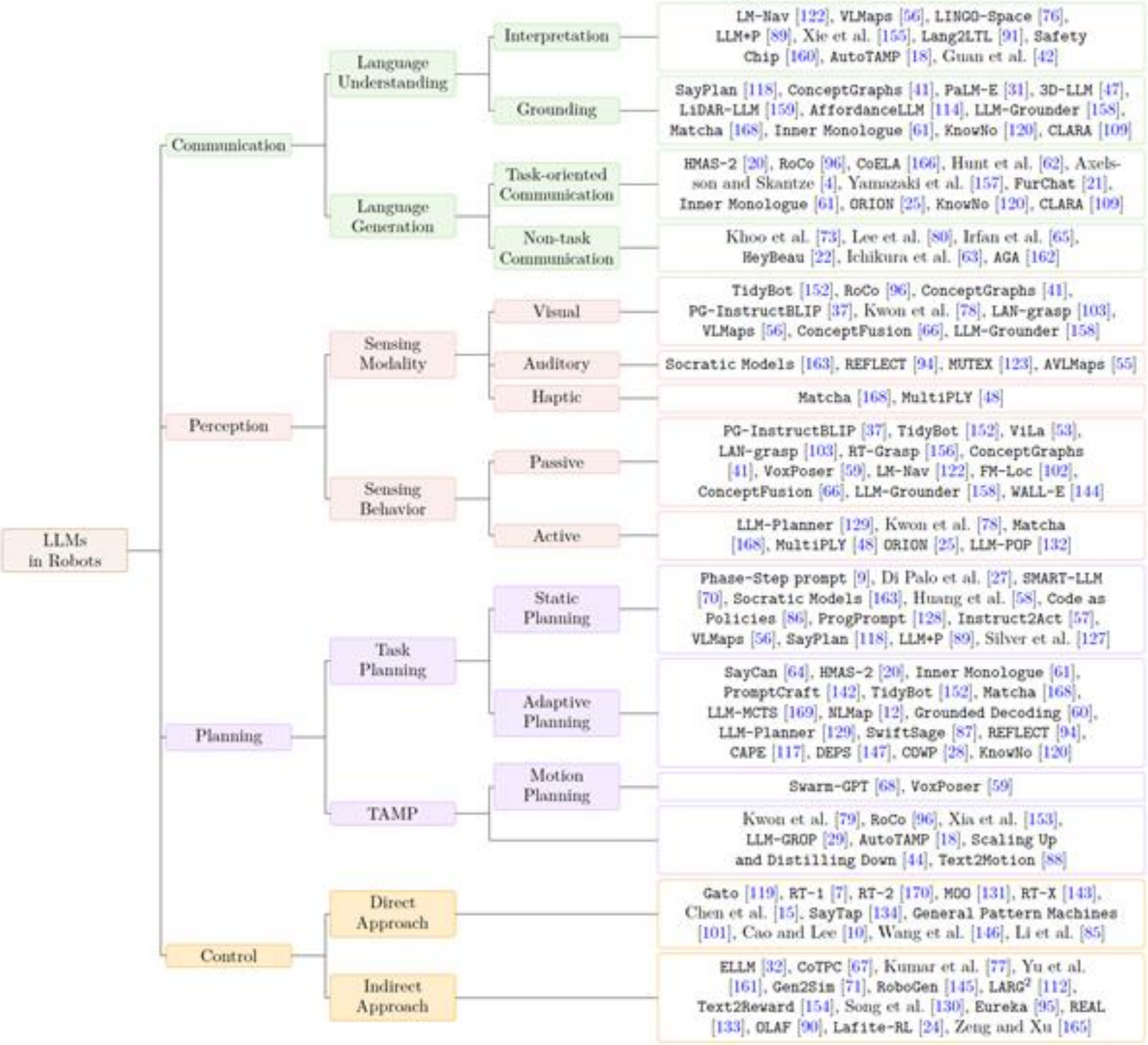


Task and Motion Planning (TAMP)

- Motion planning refers to the process of generating a path by computing sequential waypoints within configuration or task spaces
- Combines high-level task planning with low-level motion execution.
- Example Models:
  - **Text2Motion**: Generates feasible high-level actions combined with learned motion skills.
  - **AutoTAMP**: Integrates logical and physical reasoning to plan complex tasks, like articulated object manipulations

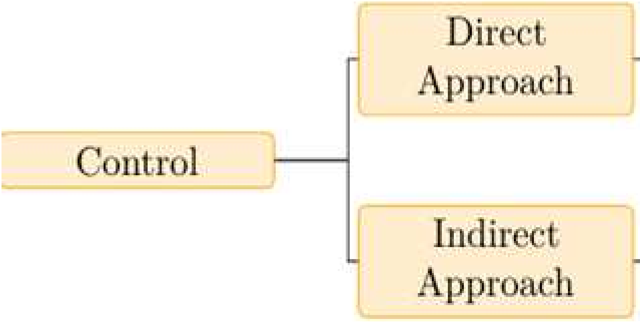
Radowan Mahmud Redoy (snf4za)

Revisiting the structure of intelligent robotics research integrated with LLMs in this survey



Control

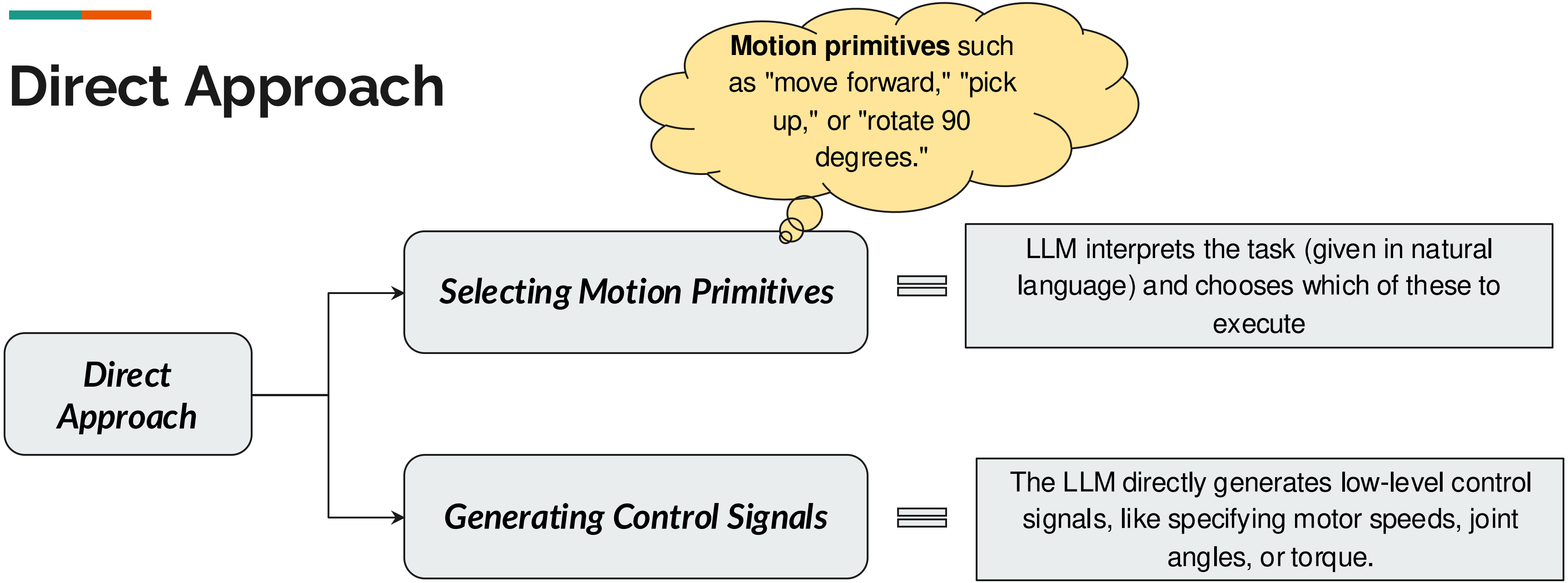
Explores the application of LLMs in robotic control systems.



*LLM itself is responsible for **understanding a task and translating** it into a command.*

*LLMs can generate **indirect representations** of control commands*

Direct Approach



Direct Approach

.Action Tokens for Control Policies:

- Early work focuses on generating **action tokens** to train Transformer architectures for control policies.
- These models, such as **Gato**, **RT-1**, and **MOO**, are trained using task-specific expert demonstrations.

<u>Generalization over Unseen Tasks</u>

However, this requires ***extensive data collection and long training times***, limiting scalability

Direct Approach

Address the data collection challenge

Researchers leverage **vision and language datasets**

This approach preserves general visual-language knowledge while adapting to control tasks

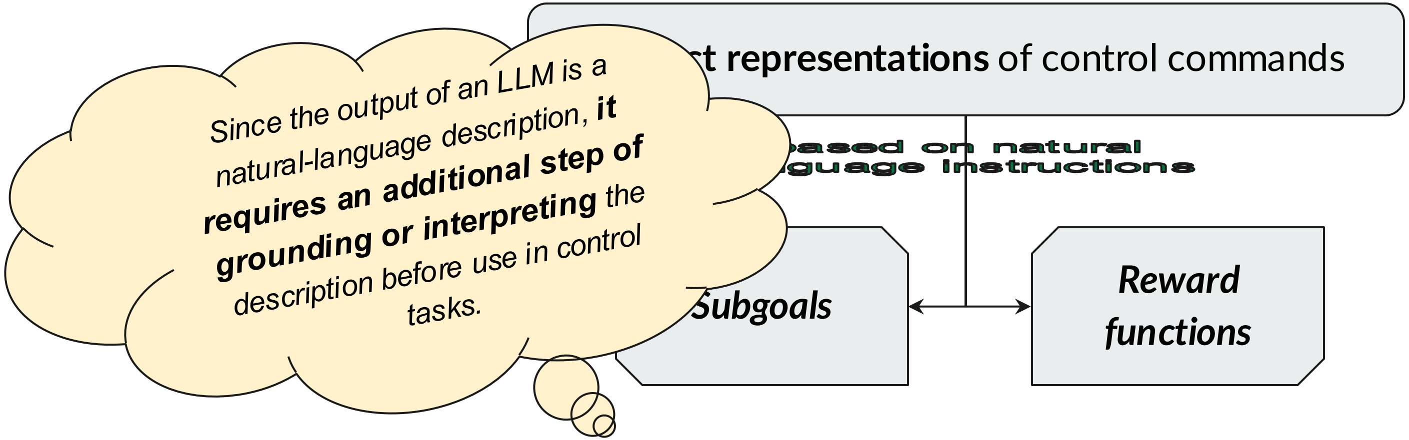
To minimize the training burden

low-rank adaptation (LoRA) is employed for fine-tuning

Direct Approach

Challenges	*Focus*
LLMs struggle to generate continuous action-level commands (e.g., **joint positions** or **torque values**) due to their discrete token-based generation process.	To address this, researchers use LLMs to generate **task-level outputs** instead of low-level control signals.

Indirect Approach



Guiding Learning with Goal Descriptions:

- Researchers use **goal descriptions** in natural language to explain desired behaviors.

Indirect Approach

This method requires **pre-defined reward formats**

Generated reward functions may not always be **accurate** or **optimal** for direct use in training

...on

Generate corresponding reward function

Recent Work Prompts LLMs to infer new reward functions based on **human-designed examples**

Indirect Approach

Refinement Loop to Improve Accuracy:

- ❖ *To improve the accuracy of reward functions, a* ***refinement loop*** *is added to validate both:*
  - ➢ **Syntax**
  - ➢ **Semantics**

Other Refinement Methods:

- ❖ *Some approaches refine motion by*:
  - ➢ **Adjusting control parameters** based on the **error state**.
  - ➢ **Selecting a suitable motion target** from human feedback.

Prompt Guideline

“Our survey offers ***guidelines for prompt engineering*** in each component area, supplemented with key examples of prompt components, to provide practical insights for researcher entering this field”

Prompt Guideline

Well-designed prompts	• *Include clear, concise, and specific statements without using technical jargon*
	• *Incorporate examples that allow anticipating the model's process*
	• *Specify the format that we want the output to be presented*
	• *Contain instructions to constrain actions*

Prompt Guideline

Guidelines over ***four robotics fields*** have been provided

Conversation Prompt: Interactive grounding

Directive Prompt: Scene-graph generation

Planning Prompt: Few-shot planning

Code Generation Prompt: Reward function generation.

Conversation Prompt: Interactive Grounding

Task Description
You are a conversational agent that interactively grounds referenced objects within tasks to a specific world model. You should engage with humans for clarifications and logically infer the target objects. Keep your response concise without explanation.
Task Procedure
You should: 1. Identify the target objects and their details from the task. 2. Iteratively ask for additional detail with a new criterion if multiple options within the world model meet the criteria. 3. Select the most appropriate object within the world model when only a single option meets the criteria.
Task Context
Task: "Bring me something to eat" World Model: ['water bottle', 'plate', 'napkin', 'coke can', 'potato chips', 'candle', 'sandwiches', 'pepper shaker', 'salt shaker', 'fork', 'banana', 'cookie', 'apple', 'cereal box', 'juice box', 'cup']
Interaction
LLM: Do you prefer something savory or sweet to eat? **User:** sweet **LLM:** Do you prefer something crunchy or soft as a sweet snack? **User:** crunchy **LLM:** Based on your preference for something sweet and crunchy, I suggest bringing the "cookie."

Directive prompt: Scene-graph Generation

Node Creation Prompt

Task Description

You are an agent responsible for generating nodes in a scene graph based on images. Each object's image and a unique ID will be provided to you. Your task is to generate the name of the central object. The output should be formatted as 'ObjectName(ID), ObjectName(ID), ...', without any space within each object name. No additional explanation is required.

Task Context

IDs: 0, 1, 2, 3, 4.

Entire Scene Visualization

Side View Robot View

VLM Output

VLM: YellowCube(0), BlueCube(1), Keyboard(2), Mouse(3), Phone(4)

Directive prompt: Scene-graph Generation

Edge Creation Prompt

Task Description

You are an agent responsible for inferring edge relations in a scene graph. Based on the provided names and 3D coordinate information of each object, infer one major {spatial_relation} from {source} to {target} (i.e., source->target: spatial_relation), which means the source is located to spatial_relation of the target. Spatial relations are limited to 'left,' 'right,' 'forward,' 'back,' 'up,' and 'down'. The bbox_extent represents the object's dimensions along the X, Y, and Z axes in meters, and the bbox_center specifies the object's central position in 3D space. For example, if the y-value of the source's center is bigger than that of the target, then the source is located to the forward of the target (i.e., source->target: forward). Similarly, if the x-value of the source is bigger than the x-value of the target, then the source is located to the right of the target (i.e., target->source: right). You can determine 'left' and 'right' based on the x-value, 'forward' and 'back' based on the y-value, and 'up' and 'down' based on the z-value.
Provide the output in a simple format, as shown in the example below. Take a deep breath and work on this problem step-by-step.

Example

Example_Input:
- object_list:
(cup(23), bbox_extent: [0.1, 0.1, 0.1], bbox_center: [10, 0.5, 0.53])
(box(20), bbox_extent: [0.6, 0.5, 0.32], bbox_center: [10, 0.5, 0.2])
- [source, target]: [box(20), cup(23)]?

Example_Output:
- box(20)->cup(23): down

Directive prompt: Scene-graph Generation

Task Context

Input:
- object_list:

(YellowCube(0), bbox_extent: [0.05, 0.05, 0.05], bbox_center: [0.52, 0.52, 1.07])
(BlueCube(1), bbox_extent: [0.05, 0.05, 0.05], bbox_center: [0.52, 0.52, 1])
(Keyboard(2), bbox_extent: [0.45, 0.15, 0.02], bbox_center: [0.53, 0.45, 1])
(Mouse(3), bbox_extent: [0.12, 0.06, 0.04], bbox_center: [0.75, 0.46, 1])
(Phone(4), bbox_extent: [0.07, 0.14, 0.007], bbox_center: [0.51, 0.2, 1])

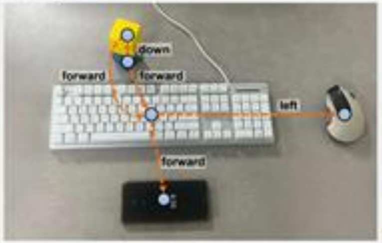
- [source, target]: [BlueCube(1), Keyboard(2)]?
- [source, target]: [BlueCube(1), YellowCube(0)]?
- [source, target]: [YellowCube(0), Keyboard(2)]?
- [source, target]: [Keyboard(2), Mouse(3)]?
- [source, target]: [Keyboard(2), Phone(4)]?

Output:

LLM Output

LLM:
(Skip explanations)
- BlueCube(1)->Keyboard(2): forward
- BlueCube(1)->YellowCube(0): down
- YellowCube(0)->Keyboard(2): forward
- Keyboard(2)->Mouse(3): left
- Keyboard(2)->Phone(4): forward

Scene-Graph Visualization



Planning prompt: Few-shot Planning

Task Description
Create a next plan for a robot to complete a household task using the allowed actions and visible objects. The robot is able to pick up and hold only one object. The robot CANNOT open or close some objects when the robot holds an object (after PickUp and before PutInto). Initially, the robot does not hold anything, and every openable object is closed. All plan sequences start with 'Initial,' which means initial condition. When the task is completed, use the action 'Done' to finish it. Do not do anything other than what the task requires. Answer only one step of the next plan without other things.
Examples
For example, Task: Put paper in the drawer Allowed actions: Initial, Open (object), Close (object), PickUp (object), PutInto (object), Done Visible objects: paper, drawer, carrot Executed plans: Initial, Open (drawer), PickUp (paper), PutInto (drawer) Next plan: Close (drawer) Task: Put only stationery in the drawer Allowed actions: Initial, Open (object), Close (object), PickUp (object), PutInto (object), Done Visible objects: paper, drawer, carrot Executed plans: Initial, Open (drawer), PickUp (paper), PutInto (drawer), Close (drawer) Next plan: Done

Planning prompt: Few-shot Planning

Task Context
Task: Put only food in the refrigerator Allowed actions: Initial, Open (object), Close (object), PickUp (object), PutInto (object), Done Visible objects: refrigerator, carrot, pencil Executed plans: Initial Next plan:
LLM Output
LLM: Open (refrigerator)
Interaction
User: Executed plans: Initial, Open (refrigerator) Next plan: **LLM:** PickUp (carrot) **User:** Executed plans: Initial, Open (refrigerator), PickUp (carrot) Next plan: **LLM:** PutInto (refrigerator) **User:** Executed plans: Initial, Open (refrigerator), PickUp (carrot), PutInto (refrigerator) Next plan: **LLM:** Close (refrigerator) **User:** Executed plans: Initial, Open (refrigerator), PickUp (carrot), PutInto (refrigerator), Close (refrigerator) Next plan: **LLM:** Done

Code Generation Prompt: Reward Design

Task Description
You need to control a two-jointed robot arm to touch the target with its fingertip in 2d xy plane. The action space is Box(-1.0, 1.0, (2,), float32) as a torque applied to joints. Please design a reward function for training RL agents in Python based on the following information.
Available APIs
You can use the following functions to design a reward function. - def get_joint_angles(env) -> tuple[float, float] # get the current angles of the joints in radian - def get_joint_angular_velocities(env) -> tuple[float, float] # get the current angular velocities of the joints in rad/s - def get_current_action(env) -> tuple[float, float] # get the current action torque in Nm - def get_current_position(env) -> tuple[float, float] # get the current x, y position of the fingertip - def get_target_position(env) -> tuple[float, float] # get the target x, y position

Code Generation Prompt: Reward Design

Goals and Constraints
The trained RL agent should accomplish the following goals and constraints. - The starting joint angles and angular velocities are all initialized to zero - The target position is randomly initialized - Finish the task with less joint torque consumption as possible - Do not consider other goals or constraints except given ones
Generation Rules
You should follow the rules below. - Write the Python function in the form of 'def compute_reward(env) -> float' and return the 'reward: float' only - Do not use redundant constant variables, class instances, or functions that are not given, but you can use popular Python libraries like numpy if you import them - Consider the scale of each reward component - Return the generated Python function only

Code Generation Prompt: Reward Design

LLM Output

```
import numpy as np

def compute_reward(env) -> float:
    # Get current position and target position
    current_pos = np.array(get_current_position(env))
    target_pos = np.array(get_target_position(env))
    # Calculate Euclidean distance between current and target position
    distance_to_target = np.linalg.norm(current_pos - target_pos)
    # Get current joint angles and angular velocities
    joint_angles = np.array(get_joint_angles(env))
    joint_velocities = np.array(get_joint_angular_velocities(env))
    # Get current action
    current_action = np.array(get_current_action(env))
    # Calculate penalty based on joint torque
    torque_penalty = -0.01 * np.linalg.norm(current_action)
    # Calculate reward based on distance to target
    distance_reward = -0.1 * distance_to_target
    # Combine rewards
    reward = distance_reward + torque_penalty
    return reward
```

Conclusion

LLMs in Robotics	*Methodological Advancements*	*Future Impact*
This survey investigates the integration of LLMs in robotics, focusing on key components like ***communication, perception, planning, and control****.*	*It examines methodological advancements that maximize LLM capabilities, enhance response integrity, and provides* ***practical prompt engineering guidelines with key example****s.*	*The survey showcases LLMs'* ***transformative role in robotics*** *and guides future research directions.*

Large Language Models for Robotics: Opportunities, Challenges, and Perspectives

20250129

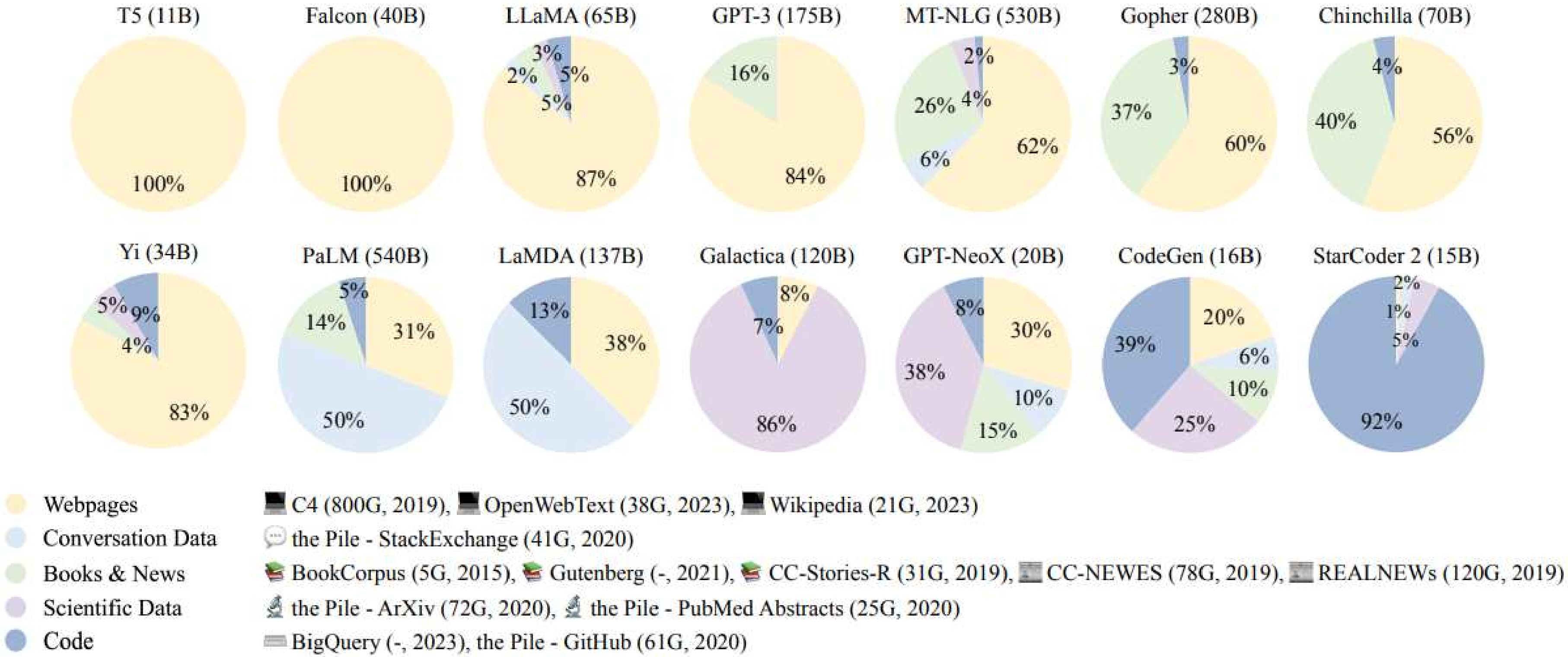
Presented by UVA – GenAI – Survey course

First Half

Presented by: Swakshar Deb(swd9tc)

Introduction

Ration of various data sources in the existing pre-trained Large Language Models (LLMs)

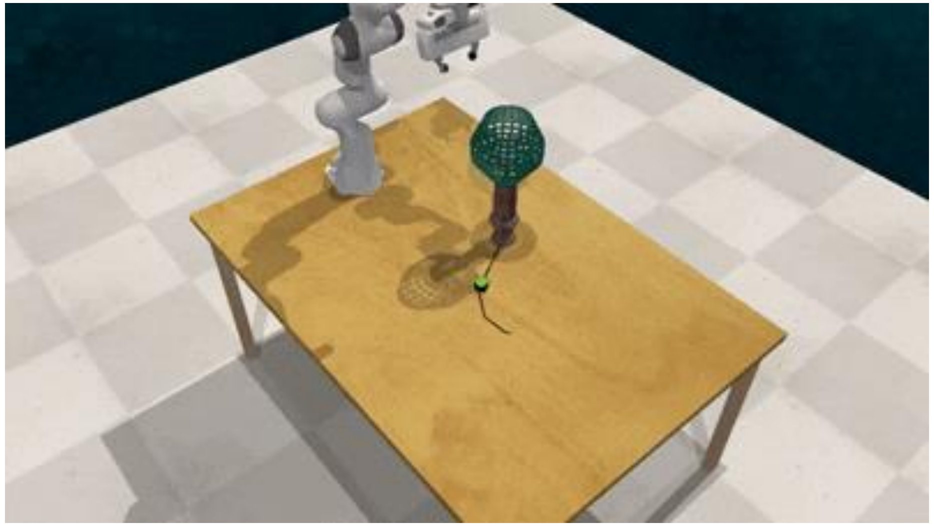


Zhao et. al., A Survey of Large Language Models, 2024

Challenge: Limited Real-World Exposure of LLMs

- Robot actions are grounded in real world through constant physical interactions
- Text only LLMs unable to process signal from various sensory input thus is not suitable for Long Horizon Tasks

User Request: Put the apple on the plate

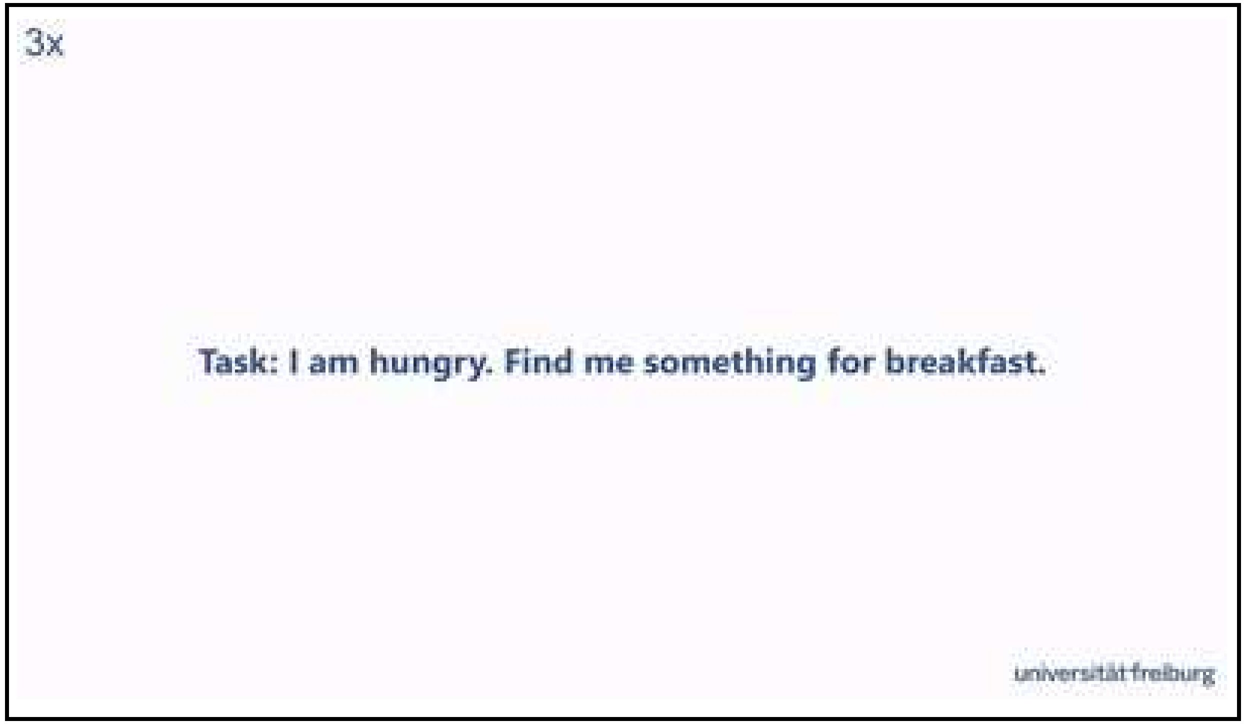


User Request: Put the ball on the basket

Images taken from Honkerkamp et. al., Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation, 2024

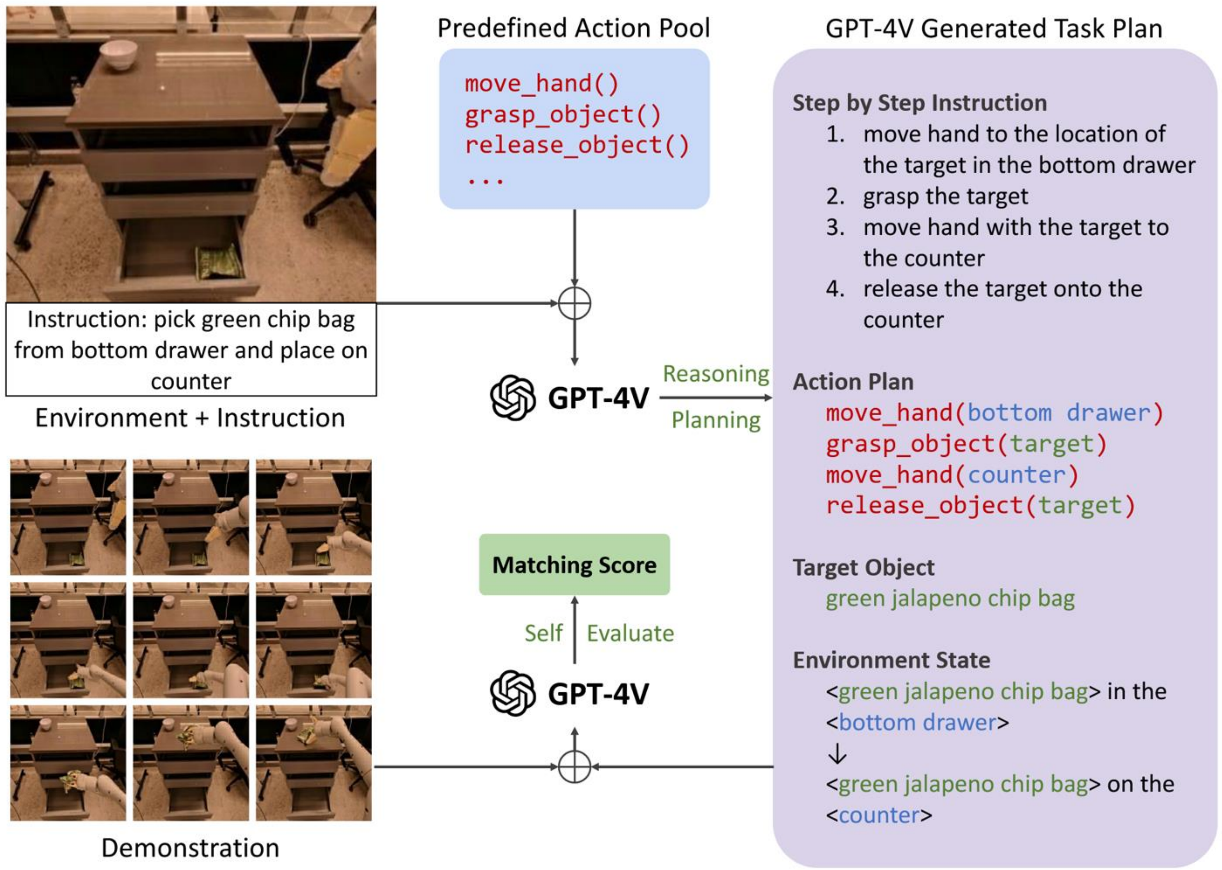
Opportunity: Multimodal LLMs Bridge Reality

- Multimodal LLMs can process various sensory information thus suitable for real-world tasks



Images taken from [Honkerkamp et. al., Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation, 2024]

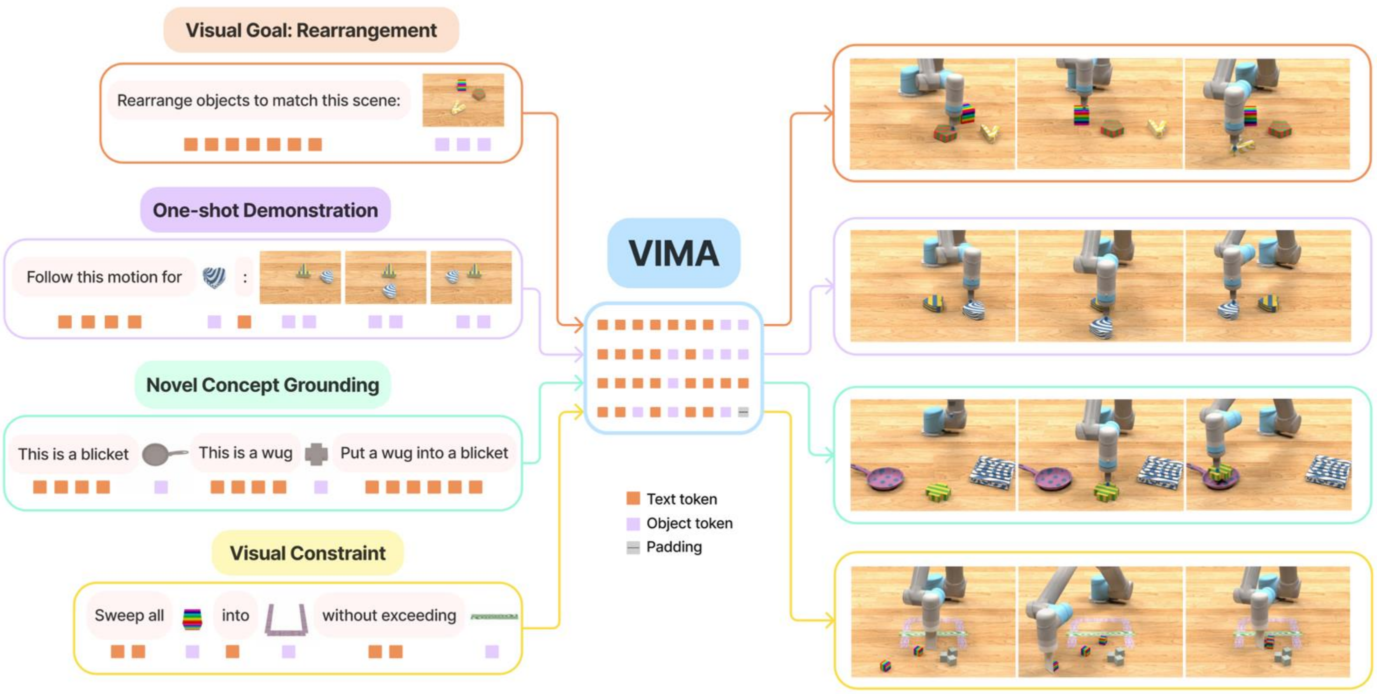
Solution: Proposed Framework



The Scope of Multimodal Approaches with LLMs

VIMA: Robot Manipulation with Multimodal Prompt

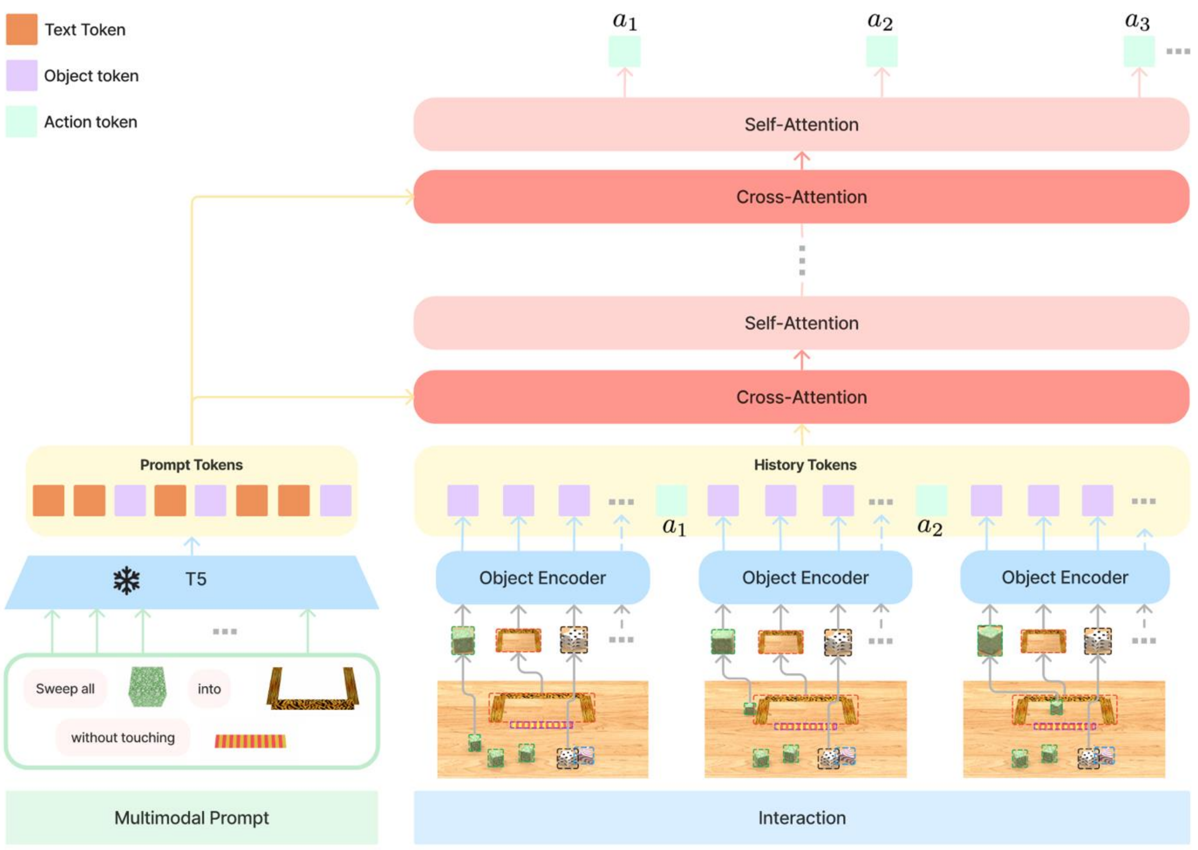
- Multimodal prompts can be a combination of both natural text instruction and the visual scene
- Multimodal prompts capture rich information about the task



Jiang et. al., VIMA: General Robot Manipulation with Multimodal Prompts, 2022

Method: VIMA

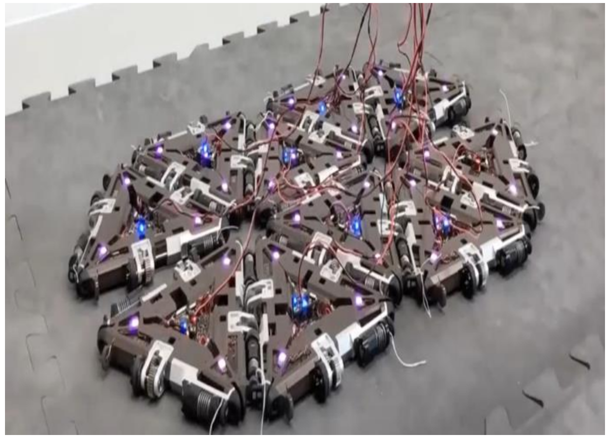
- Historical token is a set of past observation and action (a_i)
- Learn robot policy condition on prompt and historical token
- Utilized causal transformer (i.e., previous actions are the cause of present action)



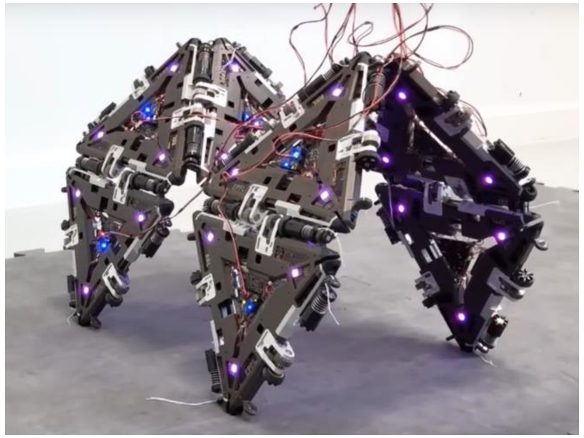
Modular Robot

- A modular robot can change its shape according to the task
- To operate a modular robot, different controller is required for each individual shapes

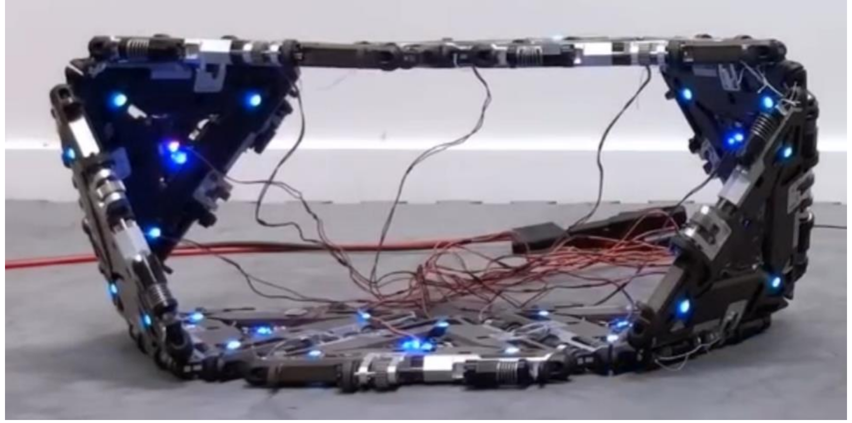
"Mori3" is a polygon based modular robot that can dynamically change it shape



Nautral Position



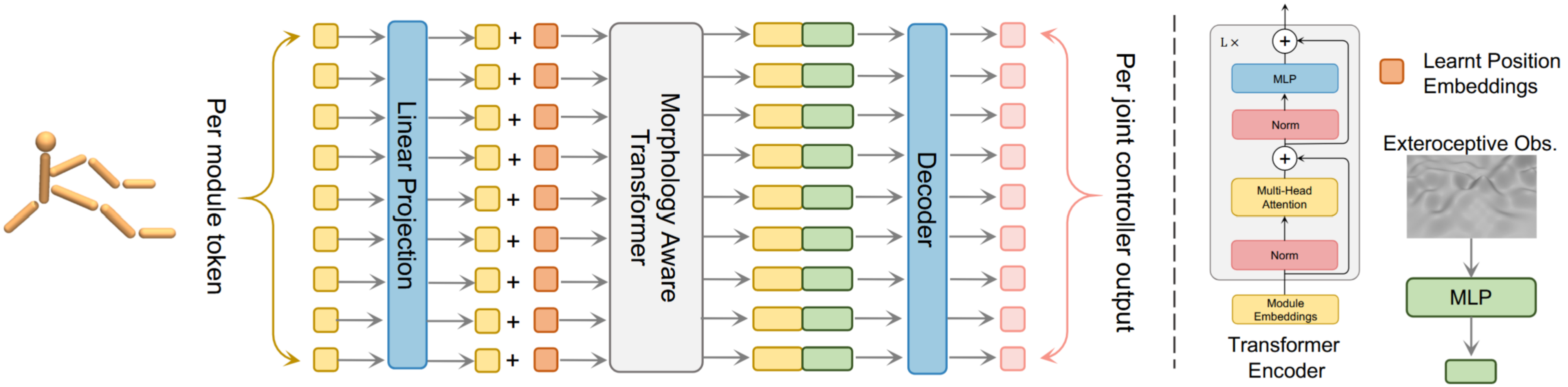
Quadruped



Rolling Track

An Universal Controller: MetaMorph

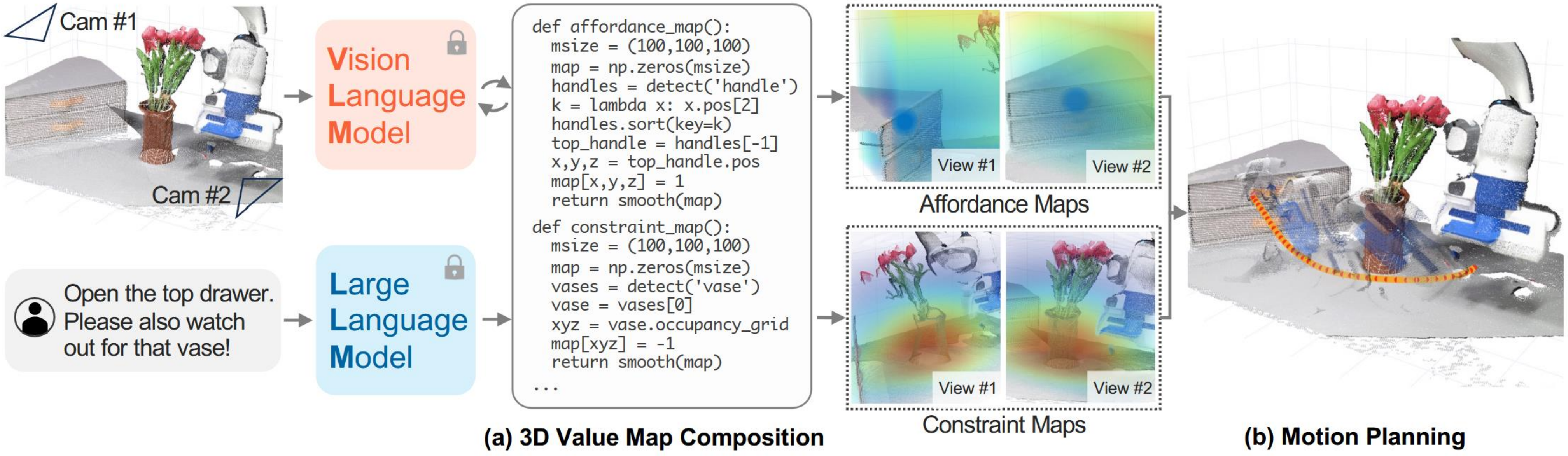
- A Transformer based architecture to learn universal controller for a modular robot shape space
- Exteroceptive Observation: Observations of various sensory devices



[Gupta et. al., METAMORPH: LEARNING UNIVERSAL CONTROLLERS WITH TRANSFORMERS, 2022](#)

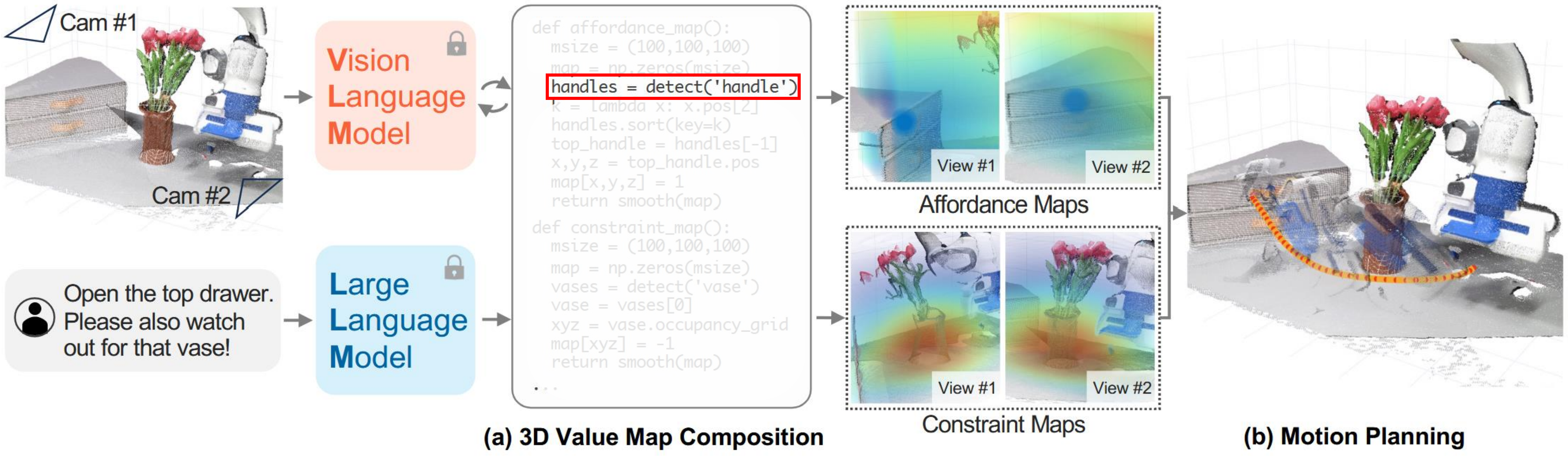
VoxPoser: Robot Manipulation with Value Map

- Enhance robot motion planning with affordance maps and constraint maps

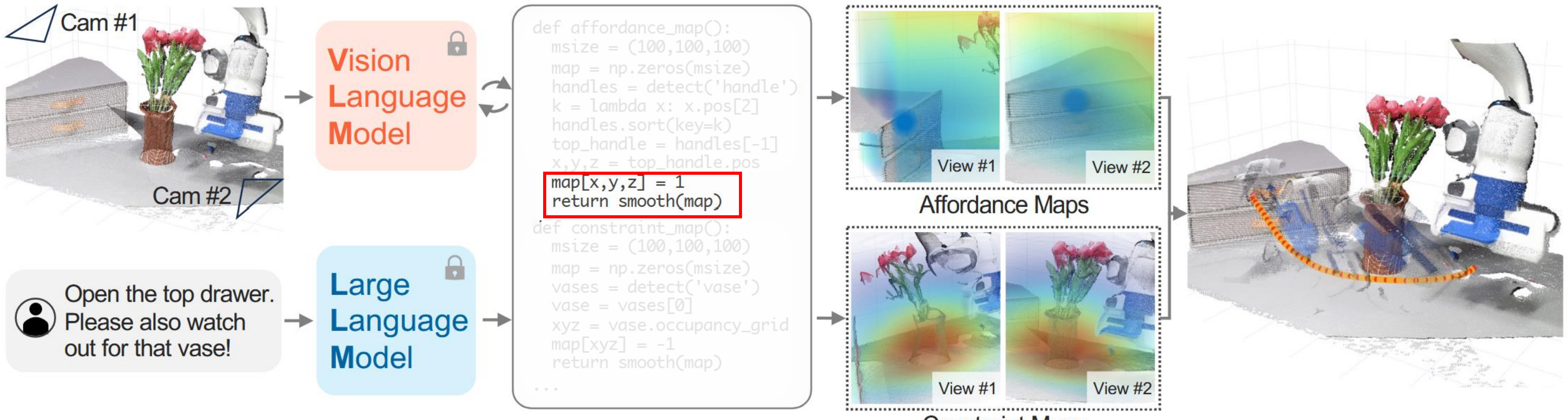


Huang et. al., VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models, 2023

Method: VoxPoser

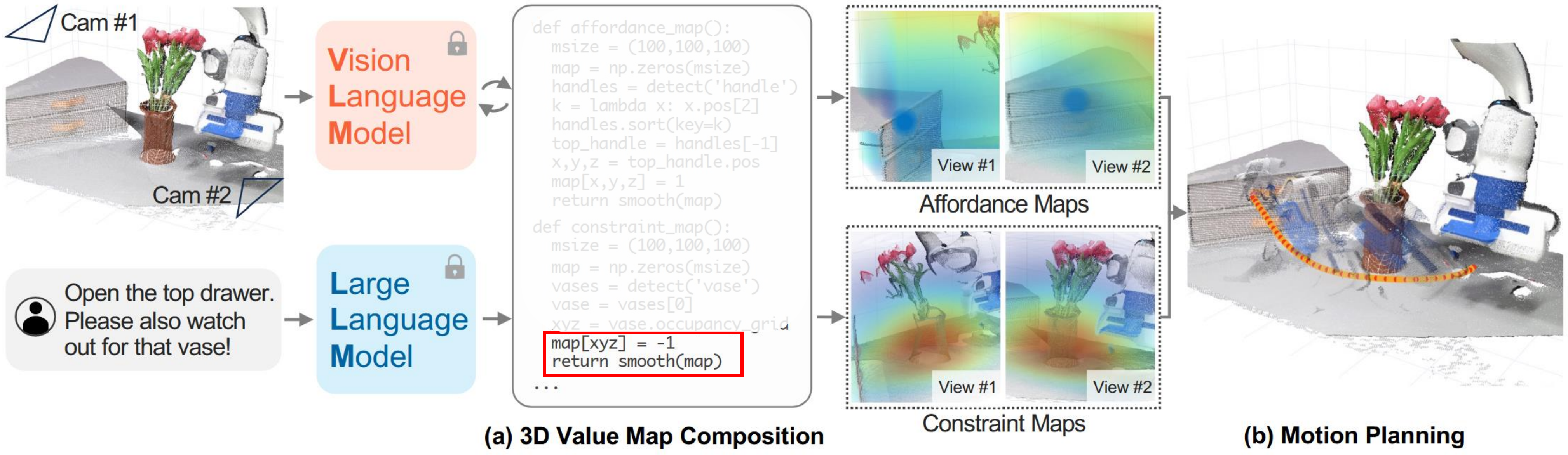


Continue...



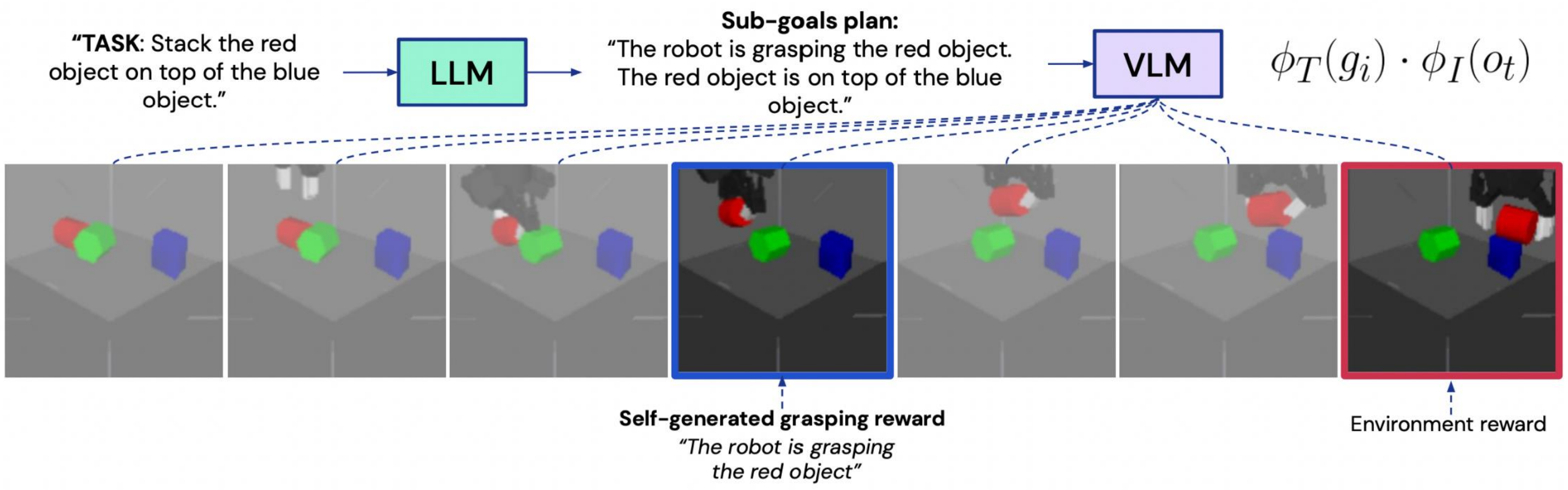
(a) 3D Value Map Composition **(b) Motion Planning**

Continue...


Cam #1
Cam #2
Open the top drawer. Please also watch out for that vase!
Vision Language Model
Large Language Model
def affordance_map():
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
k = lambda x: x.pos[2]
handles.sort(key=k)
top_handle = handles[-1]
x,y,z = top_handle.pos
map[x,y,z] = 1
return smooth(map)
def constraint_map():
msize = (100,100,100)
map = np.zeros(msize)
vases = detect('vase')
vase = vases[0]
xyz = vase.occupancy_grid
map[xyz] = -1
return smooth(map)
...
View #1
View #2
Affordance Maps
View #1
View #2
Constraint Maps
(a) 3D Value Map Composition
(b) Motion Planning

VLM as Internal Reward Model

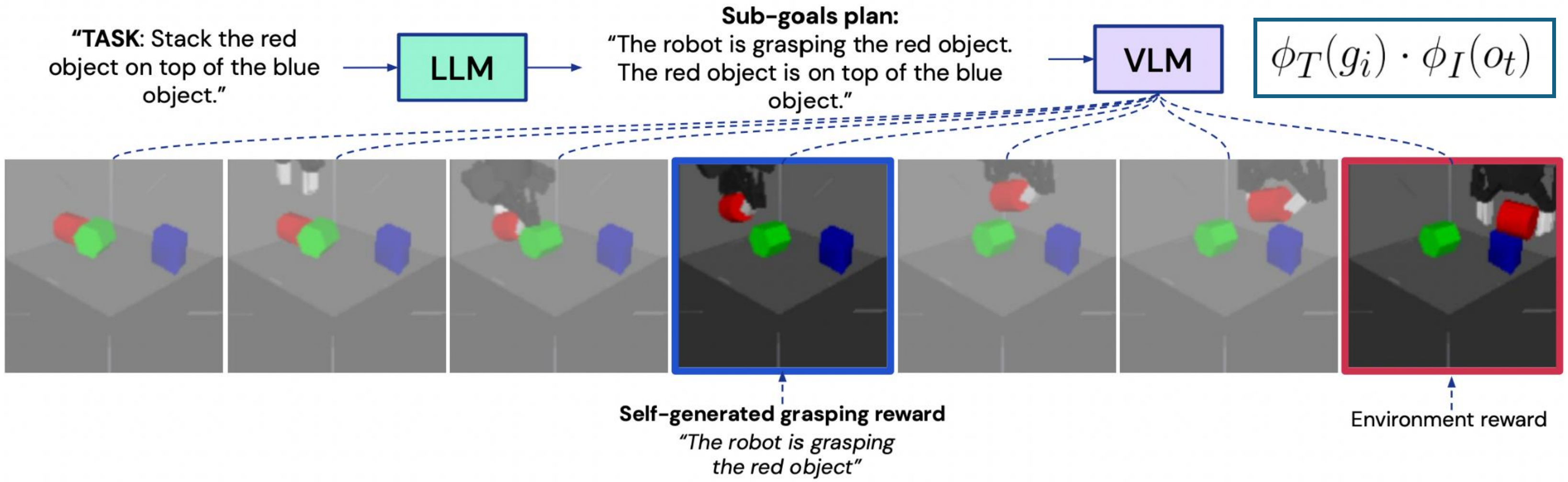
- Traditional Reinforcement Learning has limited feedback (i.e., environmental reward) during learning process
- Self generated dense reward with Vision Language Model (VLM) from the sparse reward

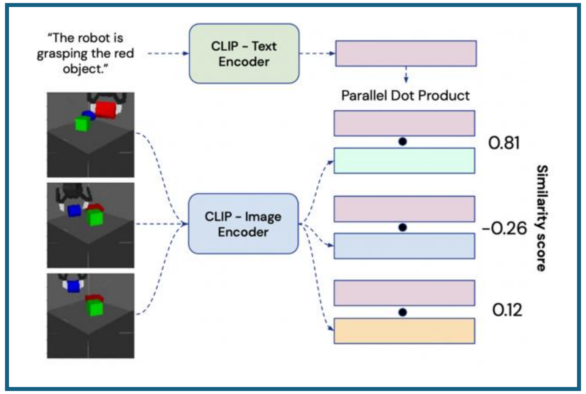


Palo et. al., Towards A Unified Agent with Foundation Models, 2023

VLM as Internal Reward Model

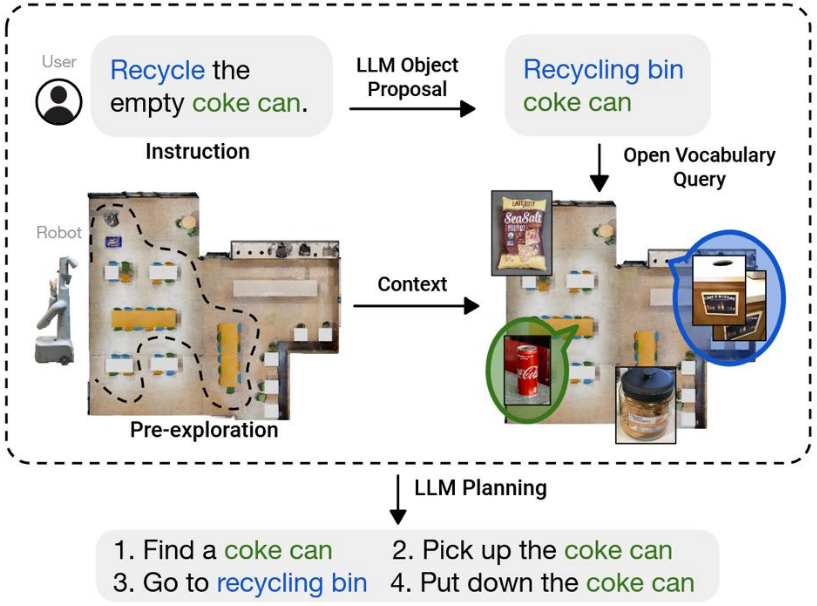
- Traditional Reinforcement Learning has limited feedback (i.e., environmental reward) during learning process
- Self generated dense reward with Vision Language Model (VLM) from the sparse reward





NLMap: Open Vocabulary Queryable Map

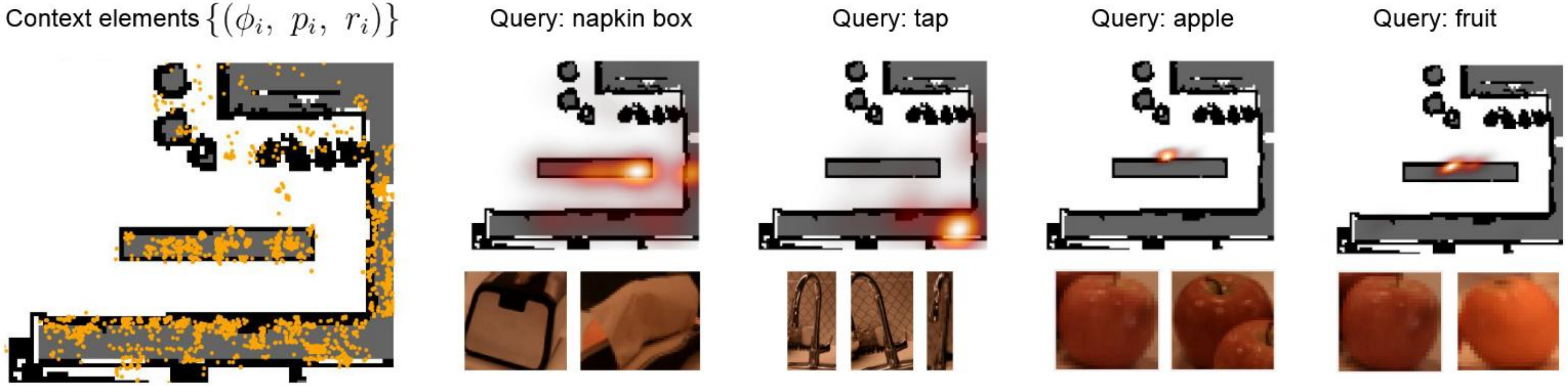
- In the pre-exploration stage, robot constructs the semantic map (i.e., voxel)
- After the pre-exploration stage, user can query any object within the scene



Cheng et al., Open-vocabulary Queryable Scene Representations for Real World Planning, 2022

Method: NLMap

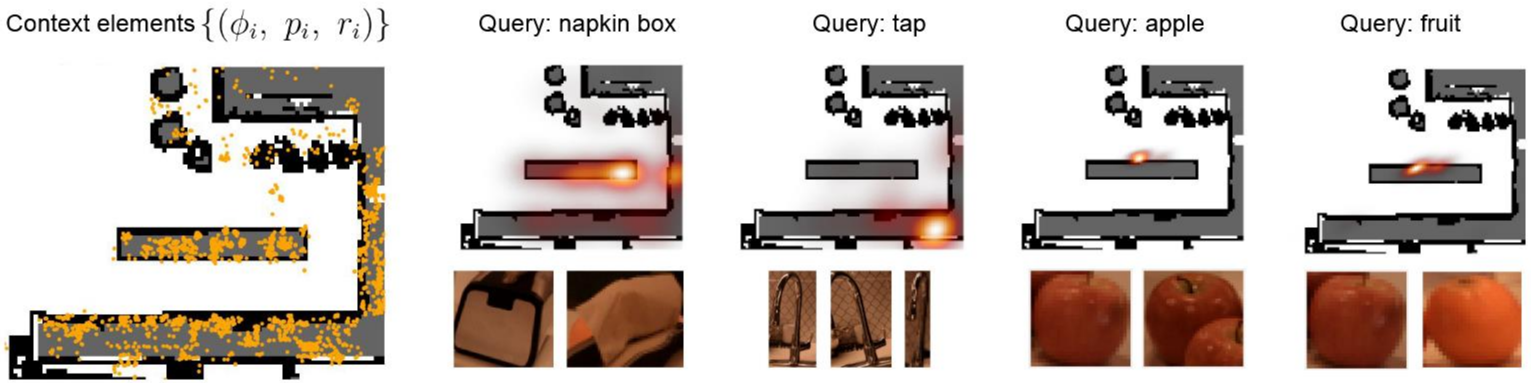
- Uses both the CLIP and VILD image encoder to extract object features, φ_i
- p_i is the position and r_i is the size of the i-th object



Continue...

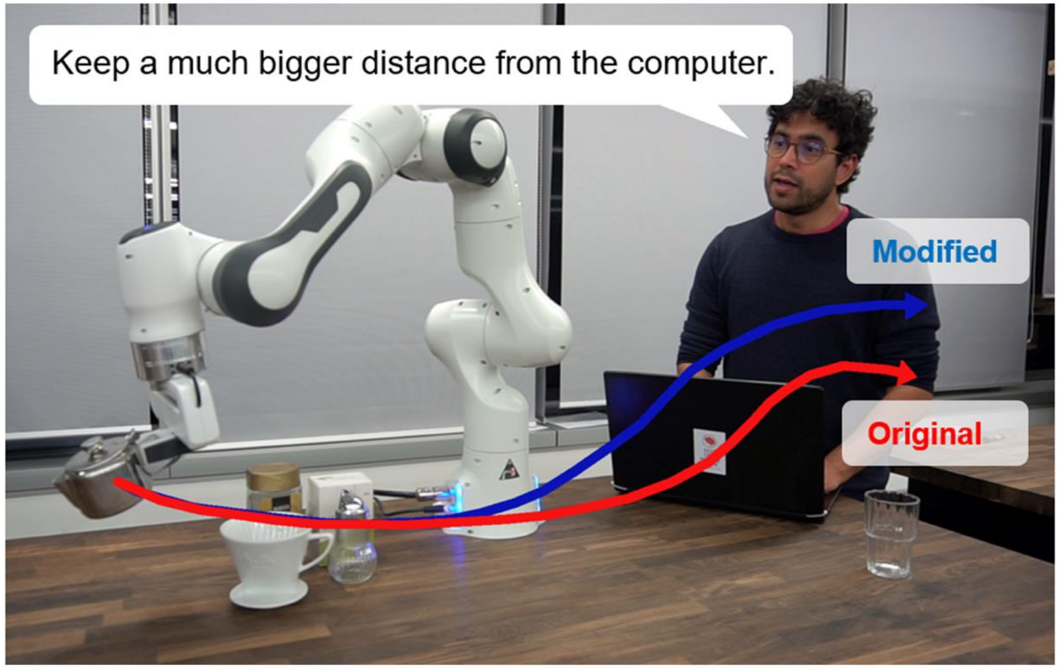
- Object query: find similarity between image - text encoding and highlight the position with highest similarity

$$D:(\phi_i, p_i, r_i), y_i \mapsto \max(D_{\text{clip}}, D_{\text{vild}}), \text{ where}$$
$$D_{\text{clip}} = \langle \Phi_{clip\_img}(I_i), \Phi_{clip\_text}(I_i) \rangle$$
$$D_{\text{vild}} = \langle \Phi_{vild\_img}(I_i), \Phi_{clip\_text}(I_i) \rangle$$



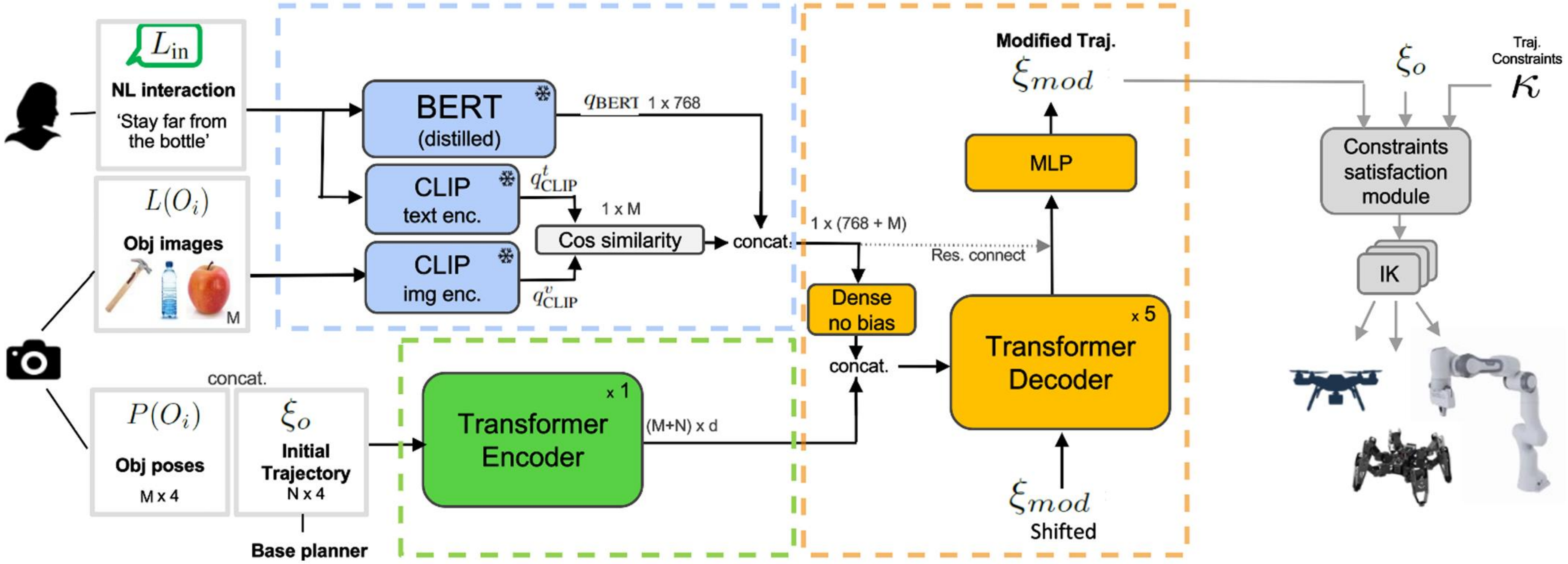
Language Guided Trajectory Modification

- **LATTE** proposes to modify the robot end-effector trajectory based on the natural language instruction



Bucker et. al., LATTE: LAnguage Trajectory TransformEr, 2022

Method: LATTE



Second Half of the paper Large Language Models for Robotics: Opportunities, Challenges, and Perspectives

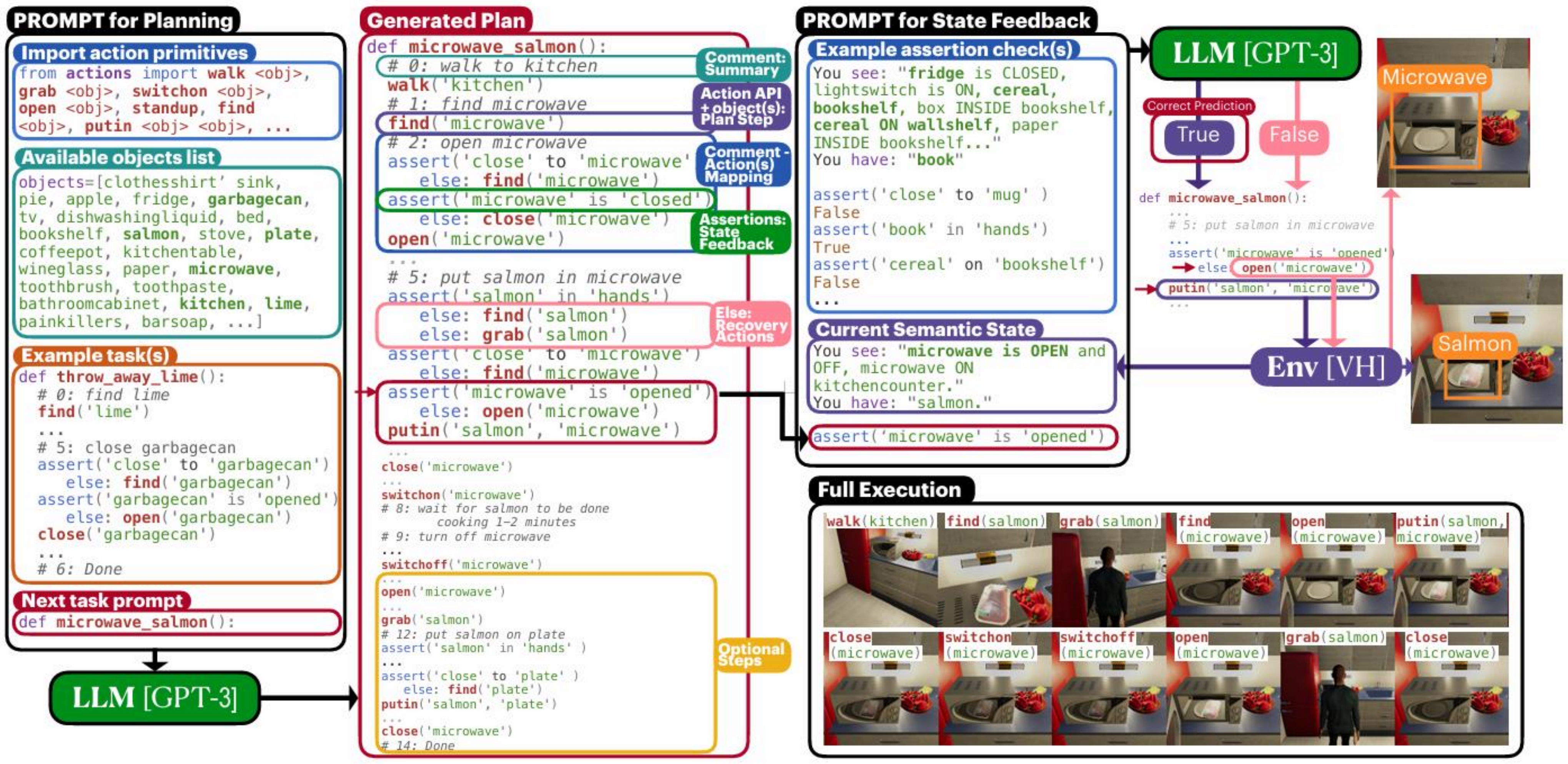
Presented By : Md. Mahir Ashhab (ftm2nu)

Natural Language Understanding in Robotics

Core Insights

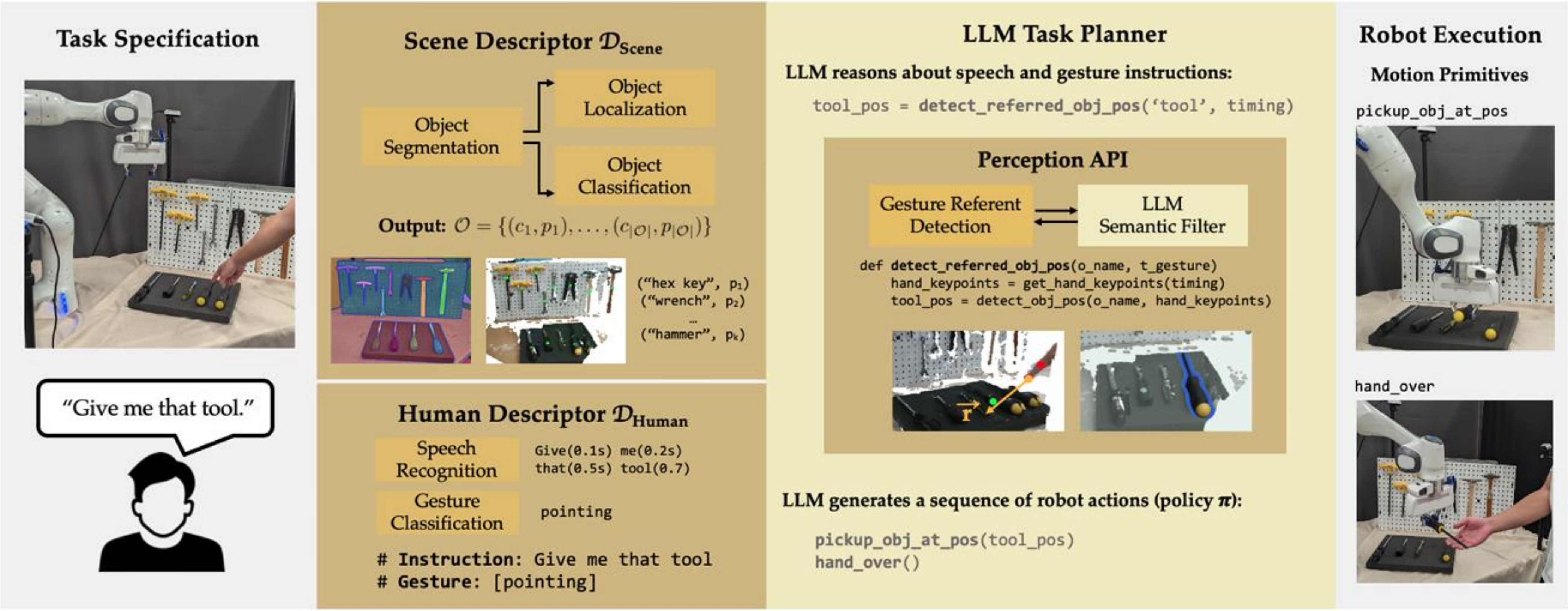
- LLMs serve as a knowledge base for robotic tasks, offering common sense reasoning and enhancing code comprehension.
- Examples include-
  - **ProgPrompt** : Programming language features in LLMs for better task performance.
  - **GIRAF** : Interprets gestures and language commands for human-machine collaboration.
  - **Cap (Code as Policies)** : Adapts language model programs for robots, enabling layered control and complex task execution.
- Key Advantages:
  - Reduced need for data collection and model training compared to traditional robotics.
  - Flexibility and adaptability to diverse scenarios and tasks.
- Applications: From home cleaning to technical robot programming.

ProgPrompt



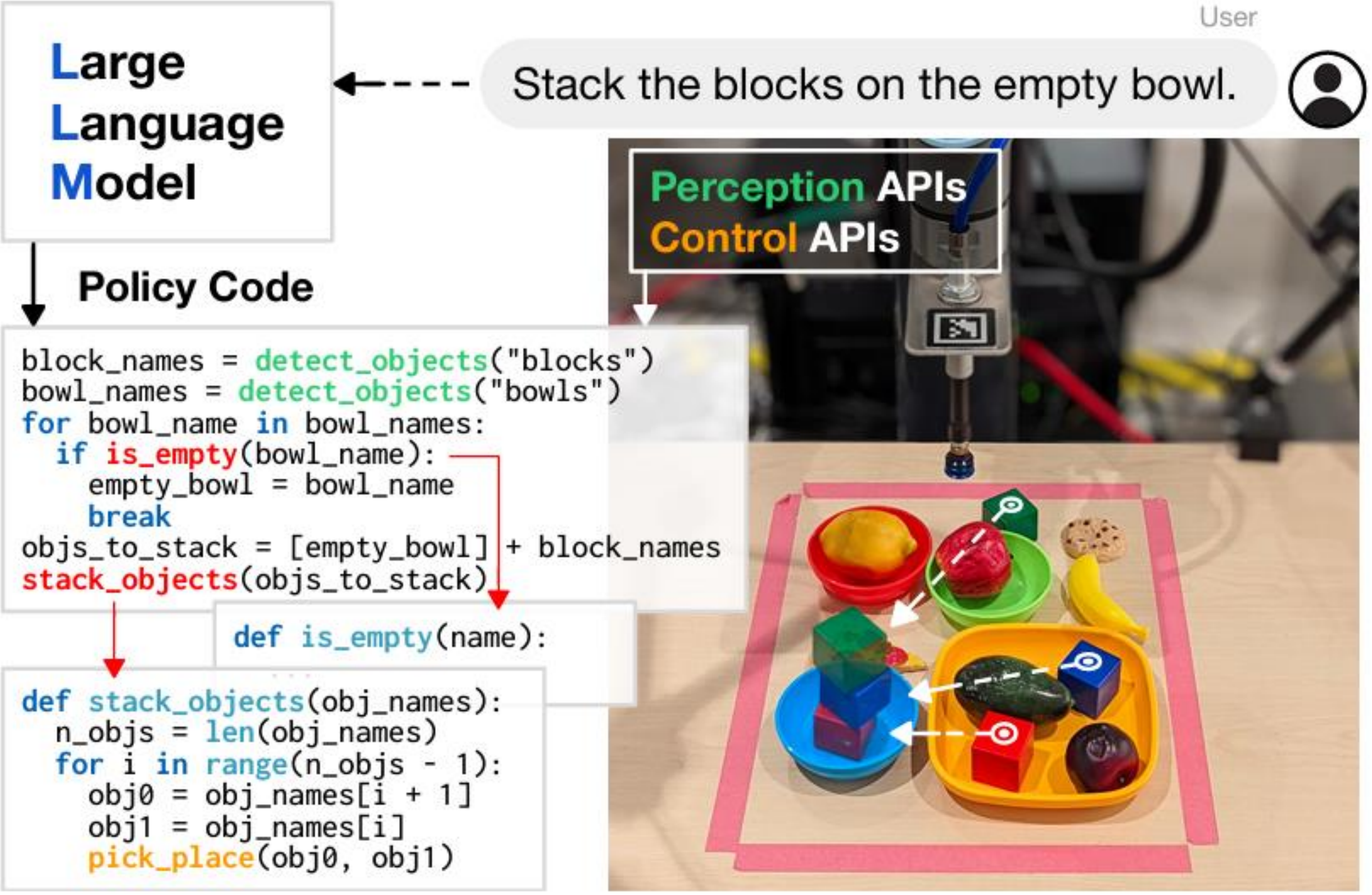
PROMPT for Planning
Import action primitives
from actions import walk <obj>, grab <obj>, switchon <obj>, open <obj>, standup, find <obj>, putin <obj> <obj>, ...
Available objects list
objects=[clothesshirt, sink, pie, apple, fridge, garbagecan, tv, dishwashingliquid, bed, bookshelf, salmon, stove, plate, coffeepot, kitchentable, wineglass, paper, microwave, toothbrush, toothpaste, bathroomcabinet, kitchen, lime, painkillers, barsoap, ...]
Example task(s)
def throw_away_lime():
# 0: find lime
find('lime')
# 5: close garbagecan
assert('close' to 'garbagecan')
else: find('garbagecan')
assert('garbagecan' is 'opened')
else: open('garbagecan')
close('garbagecan')
# 6: Done
Next task prompt
def microwave_salmon():
LLM [GPT-3]
Generated Plan
def microwave_salmon():
# 0: walk to kitchen
walk('kitchen')
# 1: find microwave
find('microwave')
# 2: open microwave
assert('close' to 'microwave')
else: find('microwave')
assert('microwave' is 'closed')
else: close('microwave')
open('microwave')
# 5: put salmon in microwave
assert('salmon' in 'hands')
else: find('salmon')
else: grab('salmon')
assert('close' to 'microwave')
else: find('microwave')
assert('microwave' is 'opened')
else: open('microwave')
putin('salmon', 'microwave')
close('microwave')
switchon('microwave')
# 8: wait for salmon to be done cooking 1-2 minutes
# 9: turn off microwave
switchoff('microwave')
open('microwave')
grab('salmon')
# 12: put salmon on plate
assert('salmon' in 'hands')
assert('close' to 'plate')
else: find('plate')
putin('salmon', 'plate')
close('microwave')
# 14: Done
Comment: Summary
Action API + object(s): Plan Step
Comment - Action(s) Mapping
Assertions: State Feedback
Else: Recovery Actions
Optional Steps
PROMPT for State Feedback
Example assertion check(s)
You see: "fridge is CLOSED, lightswitch is ON, cereal, bookshelf, box INSIDE bookshelf, cereal ON wallshelf, paper INSIDE bookshelf..."
You have: "book"
assert('close' to 'mug')
False
assert('book' in 'hands')
True
assert('cereal' on 'bookshelf')
False
...
Current Semantic State
You see: "microwave is OPEN and OFF, microwave ON kitchencounter."
You have: "salmon."
assert('microwave' is 'opened')
LLM [GPT-3]
Correct Prediction
True
False
def microwave_salmon():
# 5: put salmon in microwave
assert('microwave' is 'opened')
else open('microwave')
putin('salmon', 'microwave')
Env [VH]
Microwave
Salmon
Full Execution
walk(kitchen)
find(salmon)
grab(salmon)
find (microwave)
open (microwave)
putin(salmon, microwave)
close (microwave)
switchon (microwave)
switchoff (microwave)
open (microwave)
grab(salmon)
close (microwave)

GIRAF



Task Specification
"Give me that tool."
Scene Descriptor $\mathcal{D}_{\text{Scene}}$
Object Segmentation
Object Localization
Object Classification
Output: $\mathcal{O} = \{(c_1, p_1), \ldots, (c_{|\mathcal{O}|}, p_{|\mathcal{O}|})\}$
("hex key", p_1)
("wrench", p_2)
...
("hammer", p_k)
Human Descriptor $\mathcal{D}_{\text{Human}}$
Speech Recognition
Give(0.1s) me(0.2s) that(0.5s) tool(0.7)
Gesture Classification
pointing
Instruction: Give me that tool
Gesture: [pointing]
LLM Task Planner
LLM reasons about speech and gesture instructions:
tool_pos = detect_referred_obj_pos('tool', timing)
Perception API
Gesture Referent Detection
LLM Semantic Filter
def detect_referred_obj_pos(o_name, t_gesture)
hand_keypoints = get_hand_keypoints(timing)
tool_pos = detect_obj_pos(o_name, hand_keypoints)
LLM generates a sequence of robot actions (policy π):
pickup_obj_at_pos(tool_pos)
hand_over()
Robot Execution
Motion Primitives
pickup_obj_at_pos
hand_over

CAP



User
Stack the blocks on the empty bowl.
Large Language Model
Policy Code
Perception APIs
Control APIs
block_names = detect_objects("blocks")
bowl_names = detect_objects("bowls")
for bowl_name in bowl_names:
if is_empty(bowl_name):
empty_bowl = bowl_name
break
objs_to_stack = [empty_bowl] + block_names
stack_objects(objs_to_stack)
def is_empty(name):
def stack_objects(obj_names):
n_objs = len(obj_names)
for i in range(n_objs - 1):
obj0 = obj_names[i + 1]
obj1 = obj_names[i]
pick_place(obj0, obj1)

Complex Task Reasoning and Decision-making

Key Techniques

- SayCan: Combines LLMs with reinforcement learning for task refinement.
- Instruct2Act: Translates multimodal commands into action sequences via LLM-generated policy codes.
- PDDL Planning: Guides heuristic search planners using pretrained LLM outputs.
- REFLECT: Uses LLMs for failure explanation and task correction.
- Visual-Linguistic-Action Models: Integrates multimodal pretraining for improved robotic strategies.
- Socratic Models: Uses dialogue between pre-trained models for zero-shot task performance.

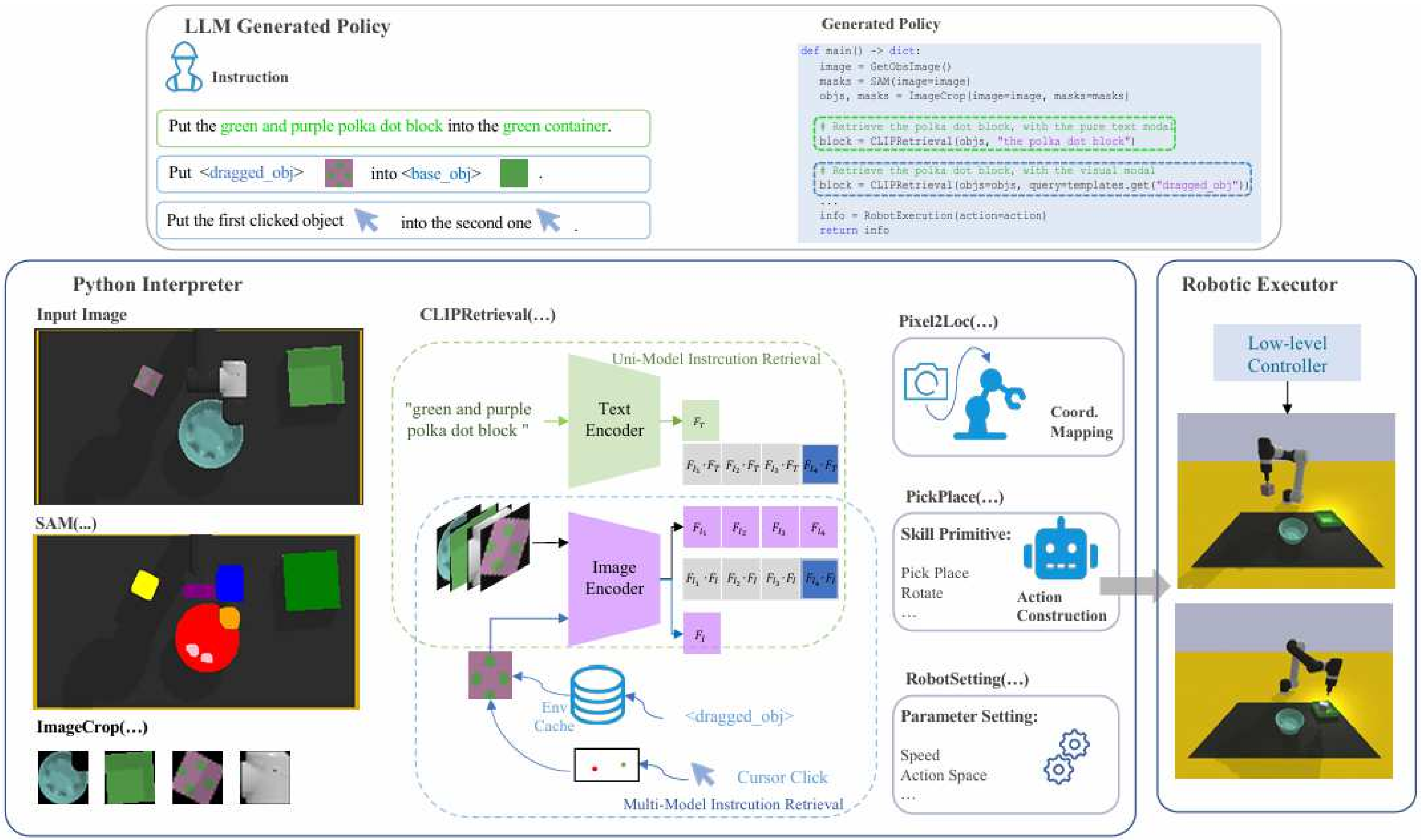
Advantages

- Enhanced semantic understanding.
- Integration of multimodal inputs (e.g., vision, language).
- Zero-shot and few-shot learning capabilities.

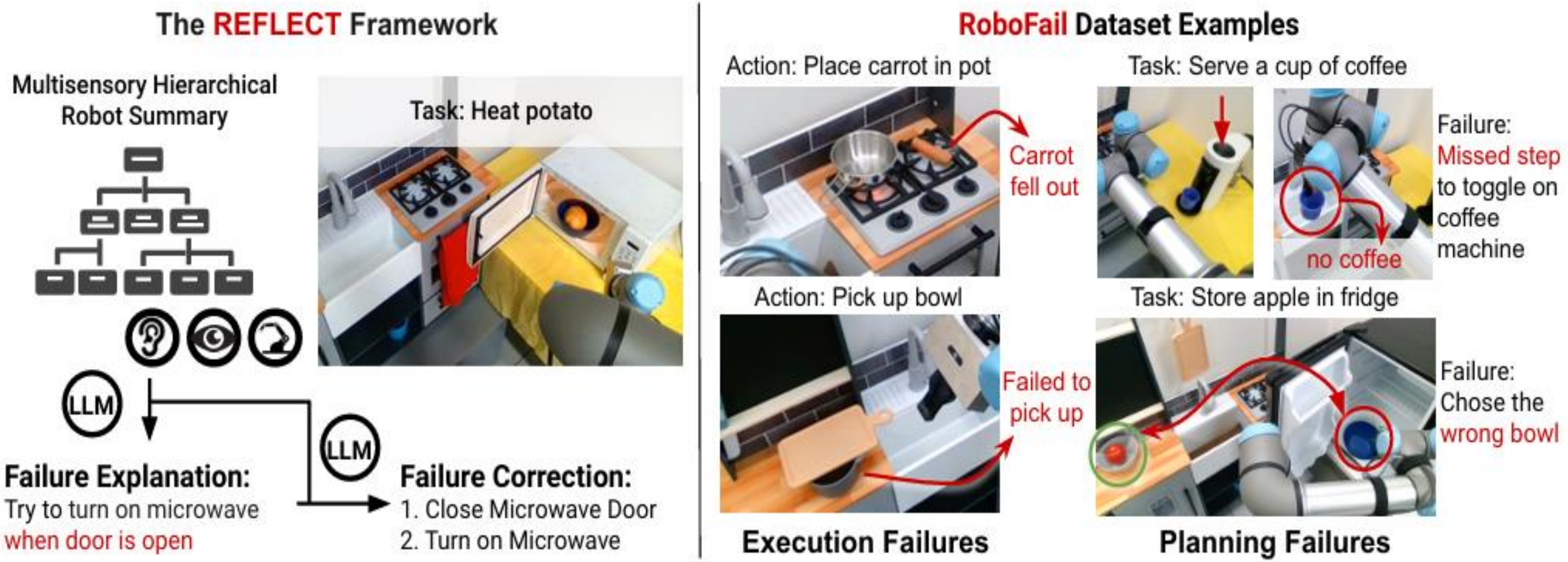
SayCan


I spilled my drink, can you help?
GPT3
You could try using a vacuum cleaner.
LaMDA
Do you want me to find a cleaner?
FLAN
I'm sorry, I didn't mean to spill it.
I spilled my drink, can you help?
LLM
"find a cleaner"
"find a sponge"
"go to the trash can"
"pick up the sponge"
"try using the vacuum"
Value Functions
"find a sponge"
"go to the trash can"
"pick up the sponge"
SayCan
"find a sponge"
"go to the trash can"
"pick up the sponge"
I would:
1. find a sponge
2. pick up the sponge
3. come to you
4. put down the sponge
5. done

Instruct2Act

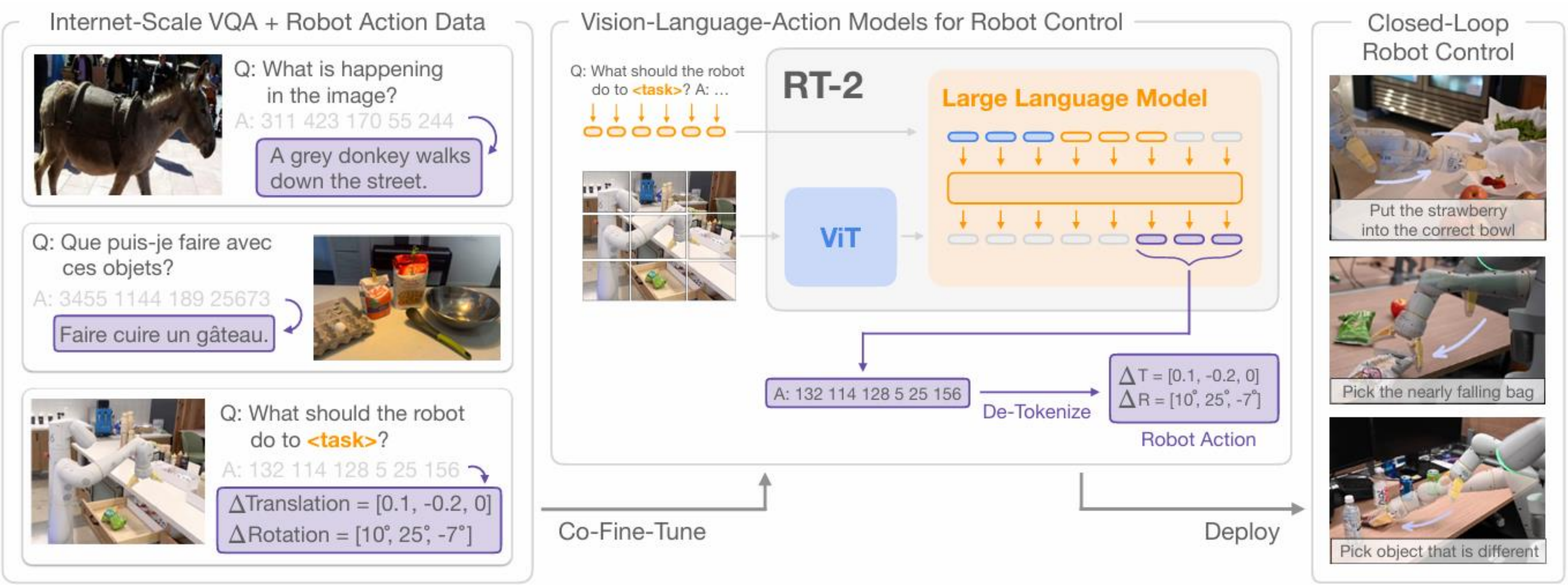


Reflect



The REFLECT Framework
Multisensory Hierarchical
Robot Summary
Task: Heat potato
LLM
LLM
Failure Explanation:
Try to turn on microwave
when door is open
Failure Correction:
1. Close Microwave Door
2. Turn on Microwave
RoboFail Dataset Examples
Action: Place carrot in pot
Carrot
fell out
Task: Serve a cup of coffee
no coffee
Failure:
Missed step
to toggle on
coffee
machine
Action: Pick up bowl
Failed to
pick up
Task: Store apple in fridge
Failure:
Chose the
wrong bowl
Execution Failures
Planning Failures

Vision-Language-Action Model



Interactive Strategies

Innovations

- Matcha: Combines LLMs with multimodal perception for better interaction.
- Generative Agents: Simulate human behavior using memory synthesis and LLMs for realistic interaction patterns.

Focus Areas

- Multimodal perception and human behavior simulation.
- Integration with perceptual systems like image recognition and speech processing.

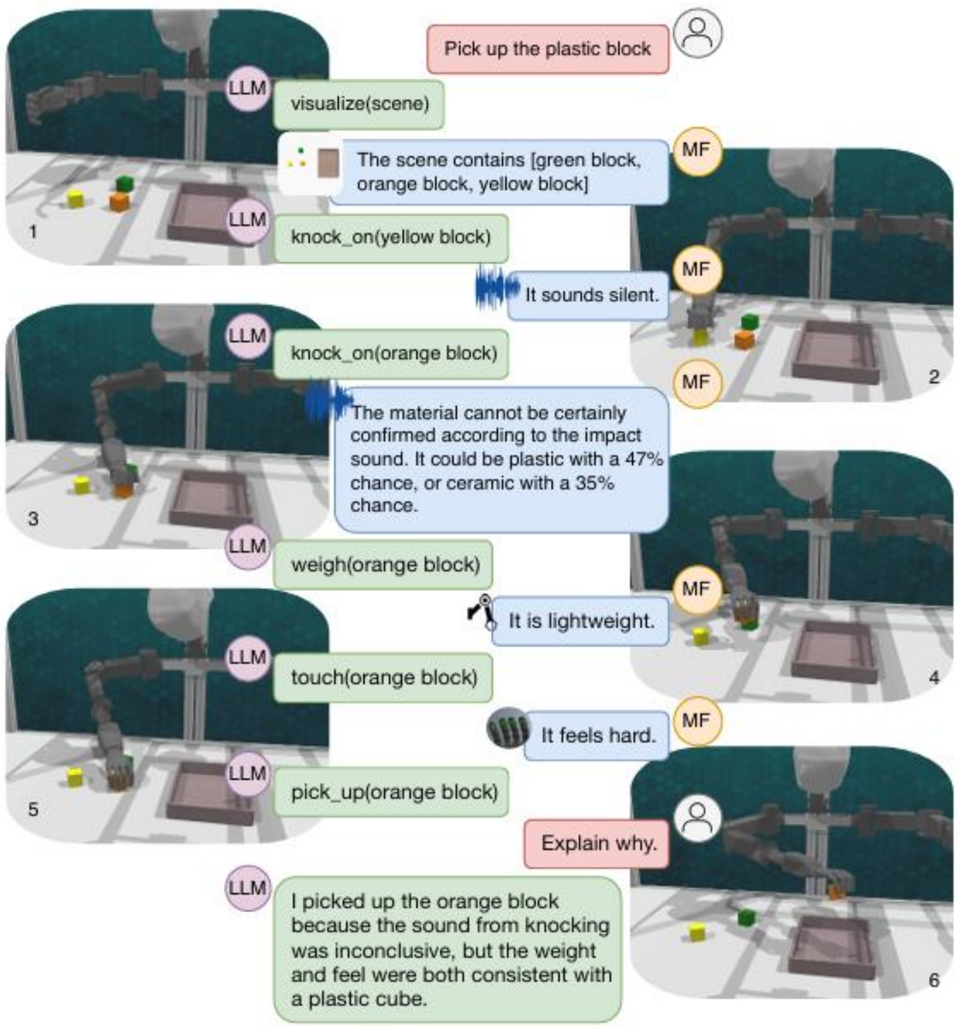
Potential Applications

Intelligent assistants, robotics, augmented reality systems.

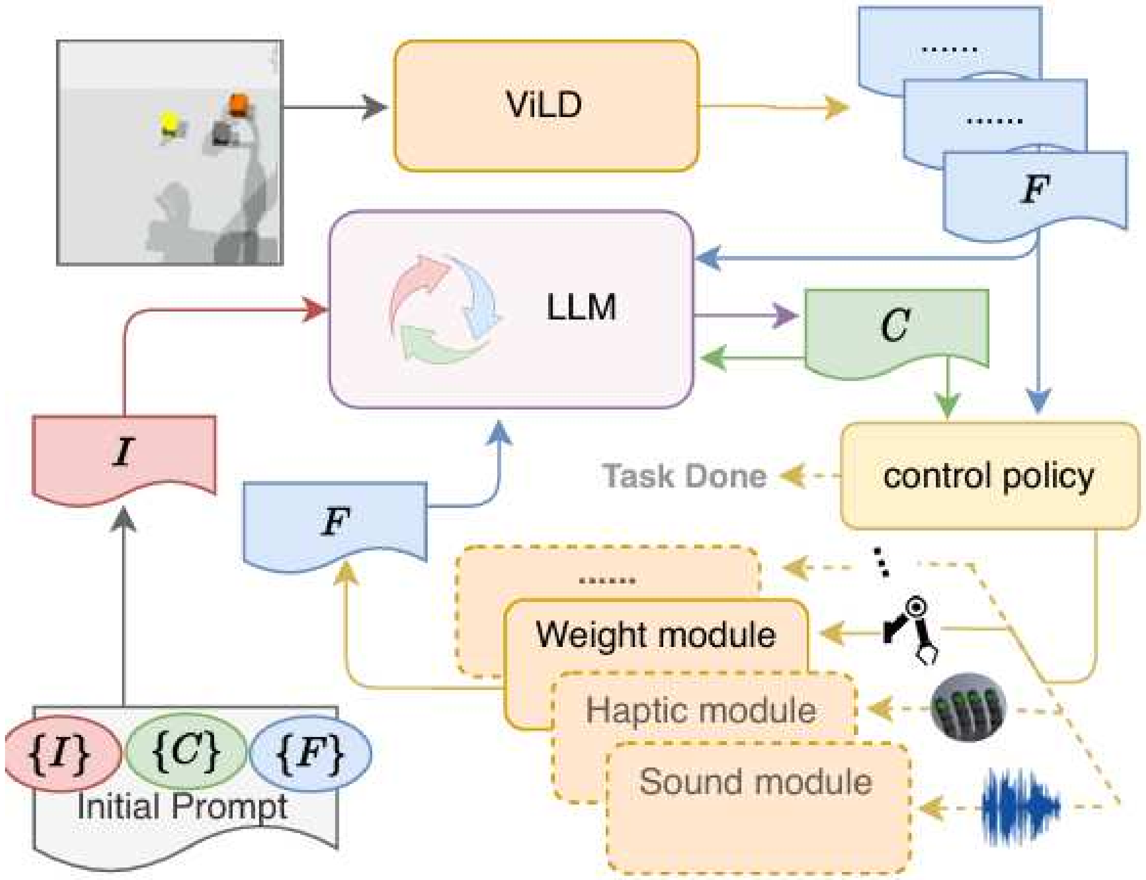
Challenges

Ethical and social implications of human-like AI behavior.

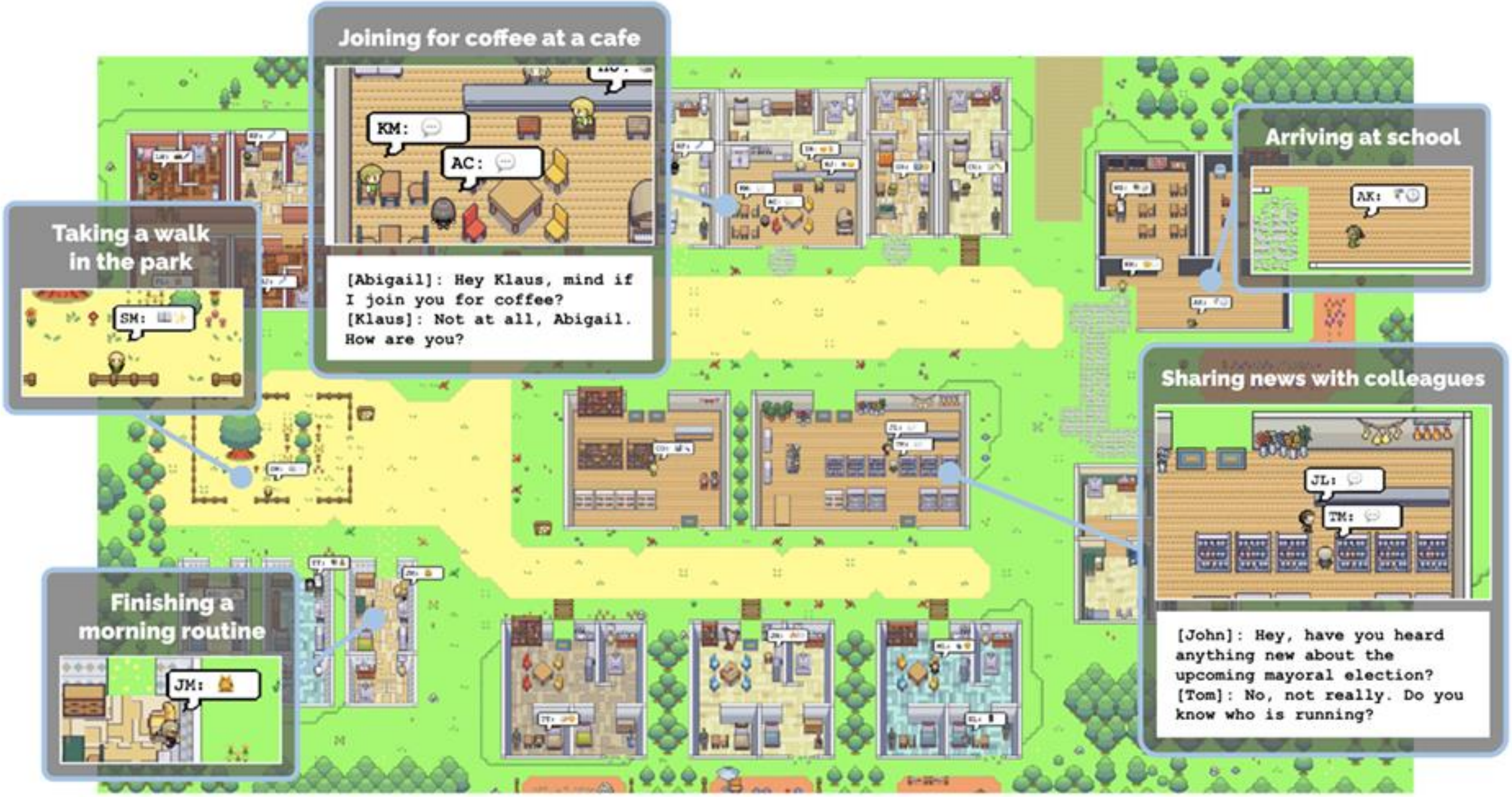
MATCHA



Overview of MATCHA



Generative Agents



Key Takeaways

LLMs are transforming robotics and AI by

- Enhancing generalization, decision-making, and interaction capabilities.
- Reducing reliance on traditional data collection and specialized training.
- Providing flexible and scalable solutions across diverse tasks and domains.

However, success depends on

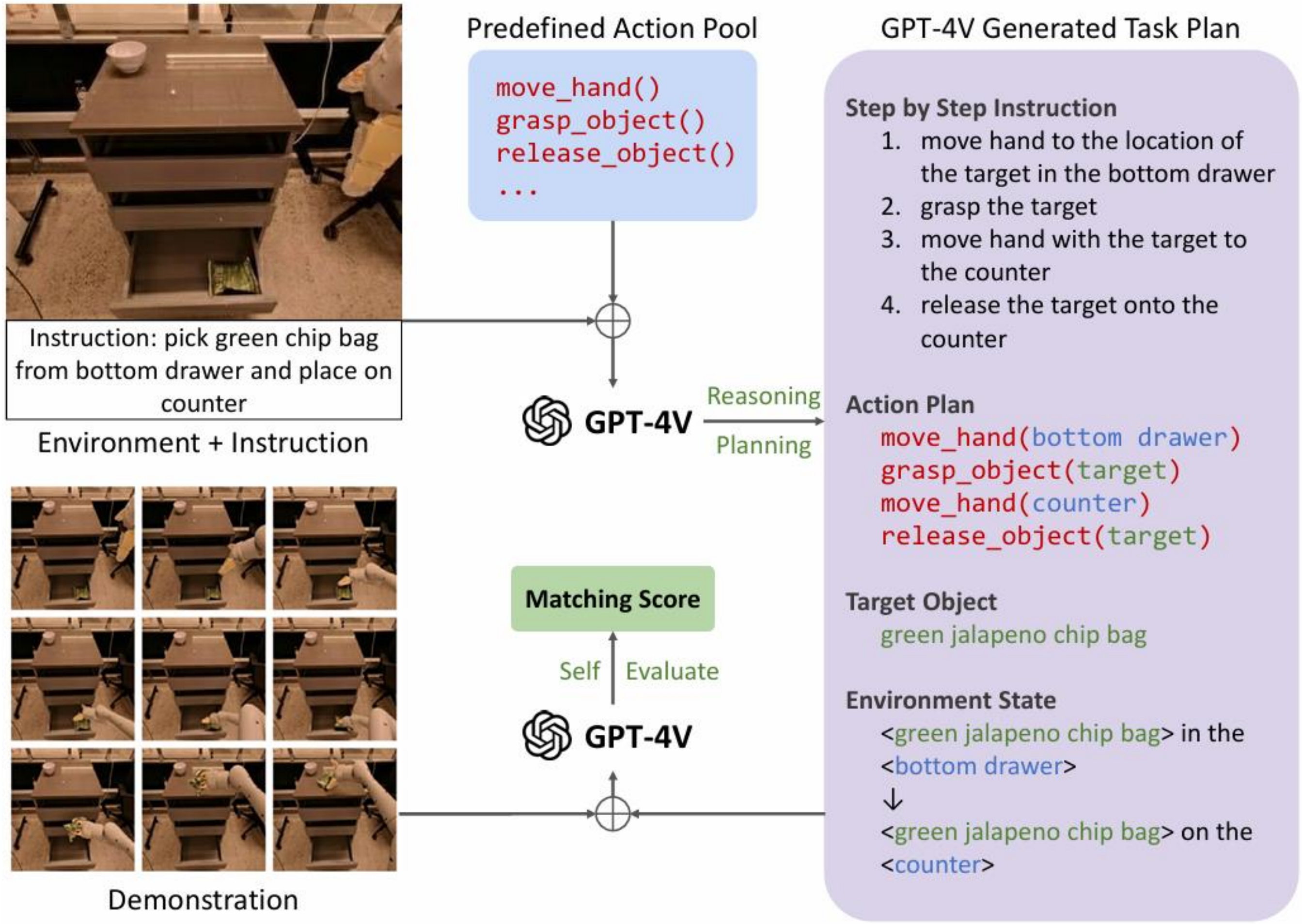
- Tailoring applications to specific tasks
- Addressing challenges such as system complexity and ethical concerns.

GPT -4V Empowered Embodied Task Planning

Overview

- Framework for robotic task planning using GPT-4V.
- Combines video demonstrations and natural language instructions.
- Evaluates manipulation and grasping tasks, critical in instruction-following robotics.

GPT -4V Empowered Embodied Task Planning



Evaluation Setup

- **Datasets**: 40+ cases from 9 datasets (e.g., Google Open X-Embodiment Dataset).
- **Scenarios**: Kitchen pickups, tabletop rearrangements.
- **Inputs**: Video + natural language for generating action plans.
- **Process**
  - Natural language instructions and visual inputs processed by GPT-4V.
  - Step-by-step task plans generated.
  - Plans compared against ground truth for evaluation.

Prompt Design

Components of Prompts

1. **System Role Explanation**: Defines GPT-4V's task and role.
2. **Predefined Action Pool**: Provides action vocabulary.
3. **Example Output**: JSON format example ensures output consistency.
4. **Environment & Instruction**: Combines video frames and language instructions.
5. **Evaluation**: Self-assessment of task plans against ground truth

Prompt Example: System

You are an excellent robot task planner. Given a natural language instruction and information about the working environment, you break it down into a sequence of step-by-step instructions and corresponding robot actions.

Predefined Action Pool:
* `move_hand()`: Move the robot hand from one position to another with/without grasping an object.
* `grasp_object()`: Grab an object.
* `release_object()`: Release an object in the robot hand.
Necessary robot actions are defined as above. Note some actions can be done by `move_hand()` like open a drawer by pull back and close it by push forward. If necessary, add new actions to the pool to complete the task.

You generate the task plan and output it as JSON with three keys:
* `JSON{"task_cohesion"}`: A JSON containing information about the robot's actions that have been split up.
* `JSON{"environment_before"}`: The state of the environment before the actions.
* `JSON{"environment_after"}`: The state of the environment after the actions.

Three keys exist in `JSON{"task_cohesion"}`.
* `JSON{"task_cohesion"}{"task_sequence"}`: Contains a list of robot actions. Only the behaviors defined in the "Predefined Action Pool" will be used.
* `JSON{"task_cohesion"}{"step_instructions"}`: Contains a list of step-by-step instructions corresponding to the `JSON{"task_cohesion"}{"task_sequence"}`.
* `JSON{"task_cohesion"}{"object_name"}`: The name of the target object.

You should only return the JSON, without any explanation or notes.

Prompt example

USER

Instruction: <natural language instruction>.

Working Env: as shown in the given image. <image>

ASSISTANT

Task planning response.

USER

Robot task instruction: <natural language instruction>
Task plan generated by GPT-4: <task plan>
Frames sampled from a reference demonstration for the above robot task: <images>

Generate a description of the video.
Score the generated actions on a scale from 0 to 10, based on how well they match the demo video: a perfect match scores 10.
You should only return a JSON, which has three keys:

* `JSON{"demo_video_description"}`: The description of the reference demonstration video.
* `JSON{"matching_score"}`: The matching score.
* `JSON{"explanation"}`: The explanation of your score.

ASSISTANT

Evaluation response.

Experimental Results - Dataset Performance

Table of Matching Scores

Dataset	Description	Matching Score
RT-1 Robot Action	Robot picks, places, and moves 17 objects from the Google micro kitchens.	**9/10**
QT-Opt	Kuka robot picking objects in a bin.	**8/10**
Berkeley Bridge	Robot interacts with household environments, including kitchens and sinks.	**8.7/10**
TOTO Benchmark	Dataset includes trajectories of scooping and pouring tasks.	**8.5/10**
BC-Z	Picking, wiping, and placing tasks on diverse tabletop objects.	**7.7/10**
Berkeley Autolab UR5	Data consists of 4 robot manipulation tasks.	**8.3/10**
NYU VINN	Robot arm performs diverse manipulation tasks on a tabletop.	**9/10**
Freiburg Franka Play	Interacts with toy blocks, performs pick-and-place and stacking tasks.	**10/10**
USC Jaco Play	Performs pick-and-place tasks in a tabletop toy kitchen environment.	**9/10**
Overall	Aggregate performance across datasets.	**8.7/10**

RT-1 Robot Action

Environment + Instruction

pick green jalapeno chip bag from bottom drawer and place on counter

Prediction by GPT-4V

Step by Step Instruction

1. move hand to the location of the target in the bottom drawer
2. grasp the target
3. move hand with the target to the counter
4. release the target onto the counter

Action Plan

move_hand(bottom drawer)
grasp_object(target)
move_hand(counter)
release_object(target)

Target Object: green jalapeno chip bag

Environment State

<green jalapeno chip bag> in the <bottom drawer>
↓
<green jalapeno chip bag> on the <counter>

Matching Score: 10/10

RT-1 Robot Action

Environment + Instruction

pick apple from white bowl

Prediction by GPT-4V

Step by Step Instruction

1. Move robot hand to the position above the apple in the white bowl
2. Grasp the apple
3. Move robot hand away from the bowl with the apple grasped

Action Plan

move_hand(above the apple)
grasp_object(apple)
move_hand(away from the bowl)

Target Object: apple

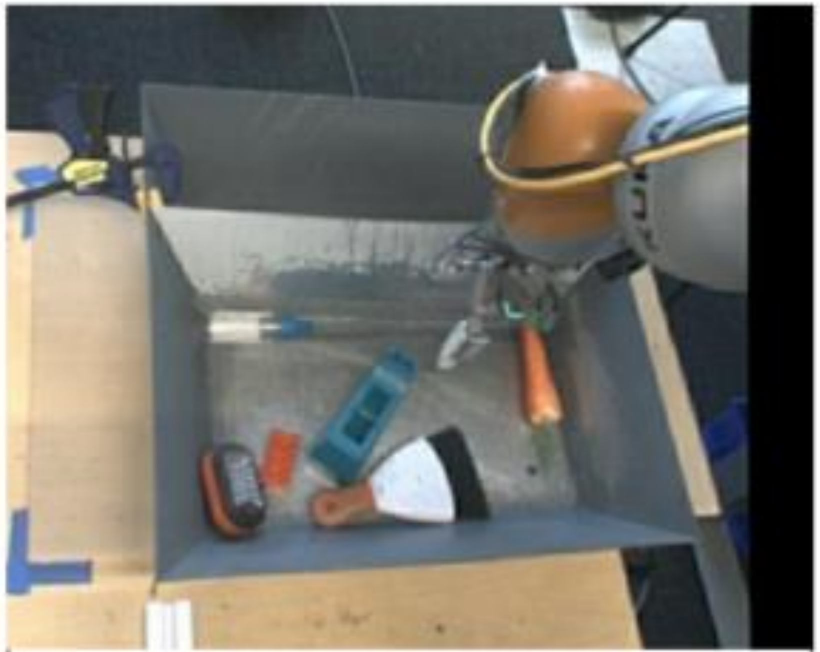
Environment State

<apple> in the <white bowl on the table>
↓
<apple> grasped by the robot hand

Matching Score: 8/10

QT-Opt

Environment + Instruction



pick anything

Prediction by GPT-4V

Step by Step Instruction

1. Move the robot hand above an object to pick it
2. Grasp the chosen object
3. Move the hand to a different location with the object grasped

Action Plan

move_hand(above an object)
grasp_object(any object)
move_hand(different location)

Target Object: any object

Environment State

A work area containing <various objects>
↓
<an object> moves it to a <different location>

Matching Score: 8/10

Berkeley Bridge

Environment + Instruction

Move the silver pot from in front of the red can, to next to the blue towel at the front edge of the table.

Prediction by GPT-4V

Step by Step Instruction

1. Grasp the silver pot in front of the red can
2. Move the grasped silver pot to the location next to the blue towel at the front edge of the table
3. Release the silver pot next to the blue towel at the front edge of the table

Action Plan

grasp_object(silver pot)
move_hand(next to the blue towel at the front edge of the table)
release_object(silver pot)

Target Object: silver pot

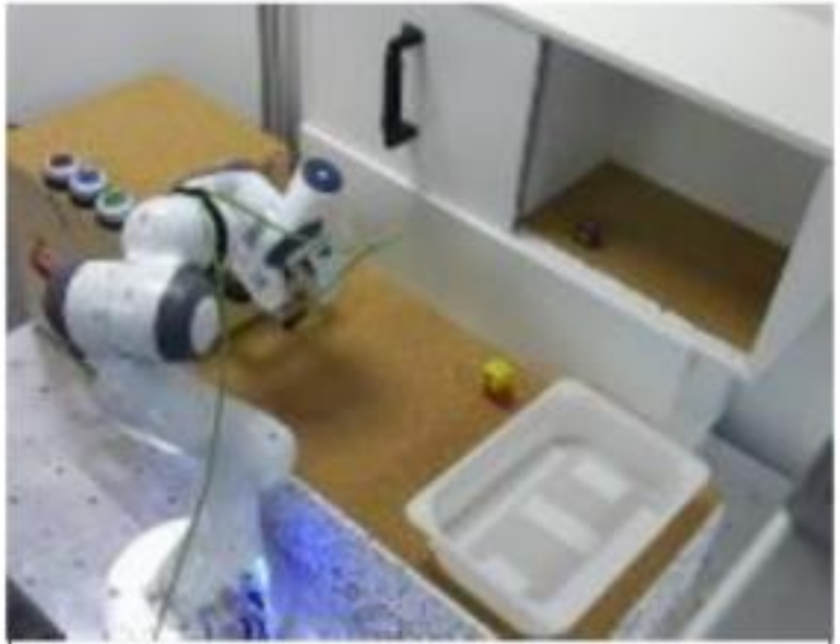
Environment State

<silver pot> in front of < the red can>
↓
<silver pot> next to < the blue towel at the front edge of the table>

Matching Score: 9/10

Freiburg Franka Play

Environment + Instruction



open the drawer

Prediction by GPT-4V

Step by Step Instruction

1. Move the robot hand to the handle of the drawer and pull it open

Action Plan

move_hand(the handle of the drawer)
move_hand(right end)

Target Object: drawer

Environment State

<drawer> closed
↓
<drawer> open

Matching Score: 10/10

USC Jaco Play

Environment + Instruction

place the gray bowl in the oven

Prediction by GPT-4V

Step by Step Instruction

1. Move the robot hand to the location of the gray bowl
2. Grasp the gray bowl
3. Move the robot hand with the gray bowl to the oven
4. Release the gray bowl in the oven

Action Plan

move_hand(location of the gray bowl)
grasp_object(gray bowl)
move_hand(to the oven)
release_object(gray bowl)

Target Object: gray bowl

Environment State

<gray bowl> on the <floor>
↓
<gray bowl> inside <the oven>

Matching Score: 8/10

BC-Z

Environment + Instruction

place the eraser on the white sponge

Prediction by GPT-4V

Step by Step Instruction

1. Move the robot hand to the position of the eraser
2. Grasp the eraser
3. Move the robot hand to the position above the white sponge
4. Release the eraser onto the white sponge

Action Plan

move_hand(to the apple)
grasp_object(apple)
move_hand(to the tray)
release_object(into the tray)

Target Object: eraser

Environment State

<eraser> <on the surface away from the white sponge><white_sponge>
<positioned on the surface>
↓
<eraser> <placed on the white sponge><white_sponge> <unchanged>

Matching Score: 5/10

TOTO Benchmark

Environment + Instruction

pour

Prediction by GPT-4V

Step by Step Instruction

1. Move the hand over the red cup
2. Grasp the bottle
3. Tilt the hand to pour the contents from the red cup
4. Return the red cup to the upright position and release it

Action Plan

move_hand(align with the bottle)
grasp_object(bottle)
move_hand(above the cup)
release_object(bottle)

Target Object: red cup

Environment State

<red cup> <on the table, upright, presumably with contents>
↓
<red cup> <on the table, empty>

Matching Score: 8/10

Key challenges

1. Homogeneous Plans
    - Lacks detailed, robust designs for complex environments.
2. Prompt Complexity
    - Requires carefully crafted, lengthy prompts.
3. Execution Constraints
    - Limited by predefined actions, reducing flexibility.
4. Closed-source & Delays
    - GPT-4V API limitations hinder real-time applications.

Future Opportunities

Precision Agriculture

- Automation of labor-intensive tasks (e.g., fruit picking, crop phenotyping).

Healthcare

- Enhancing robot-assisted surgeries and screenings.

Brain-Computer Interfaces (BCIs)

- Aligning brain signals with language for self-planning and control.

Conclusion

Key Takeaways

- LLMs like GPT-4V showcase impressive capabilities in enhancing robotic intelligence through reasoning, language understanding, and multimodal processing.
- While promising, significant challenges persist, including model transparency, real-world applicability, and safety concerns.

Path Forward

- Address limitations through rigorous research in:
    - Testing, training, and policy adaptation.
    - Ethical oversight and safer model architectures.
- Embrace **sim-to-real development** to streamline intelligent robot design and deployment.

Final Thought

- The integration of LLMs and robotics is a transformative frontier that demands interdisciplinary collaboration to unlock its full potential responsibly.

Thank you