VLMs in Embodied AI

Vlad Shakhuro

AIRI, FusionBrain.Robotics

Outline

AIRI

01

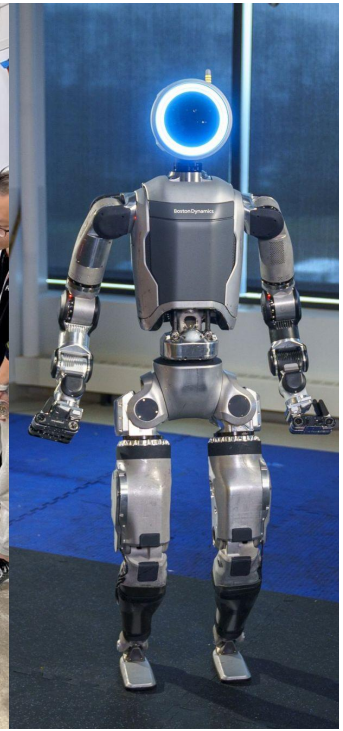
What is Embodied AI?

Embodied AI

Goal of EAI is to create intelligent agents (i.e. robots) with physical **embodiment** that can solve challenging tasks. Such agents should be able to:

- **Perceive:** see, listen, touch their environment using various sensors and extract meaningful information
- **Talk:** hold a natural dialog grounded in their environment
- **Reason:** consider and plan for the long-term consequences of their actions
- **Act:** navigate and interact with their environment to accomplish goals

General-purpose robots



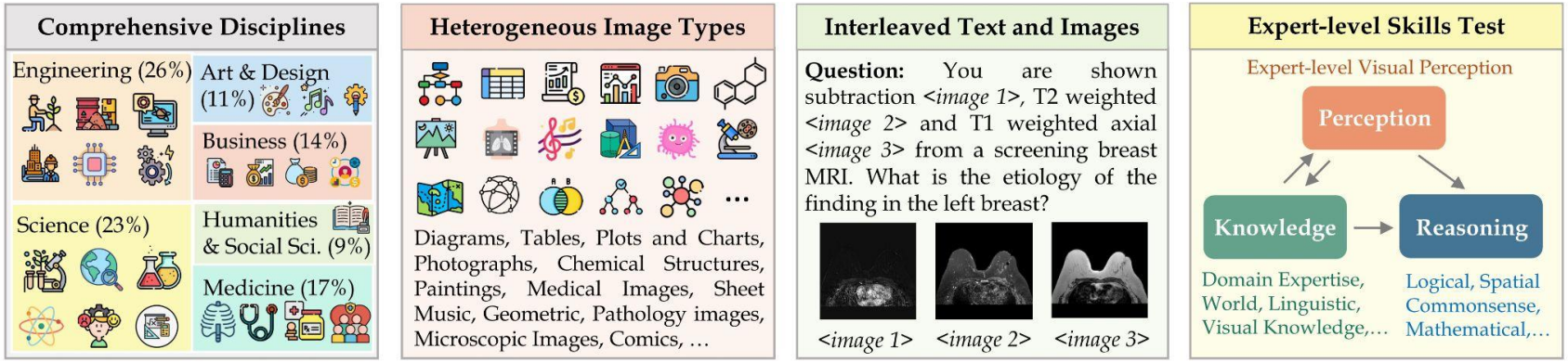
02

Evaluation: sim and real

Static benchmarks for CV/NLP

A lot of CV/NLP benchmarks are available for various tasks:

- → usually very fast evaluation
- → reproducible
- → cheap
- → have big domain shift in relation to robotics data
- → may be only a proxy metric/guidance for choosing baseline models



Yue et al. MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI. CVPR 2024

Arena for evaluating VLLMs

Lu et al. WildVision: Evaluating Vision-Language Models in the Wild with Human Preferences. arXiv:2406.11069

Simulator: static scans vs CAD

Scan-based env:

- → relatively fast to collect
- → realistic look
- → very fast to render
- → limited number of supported tasks

CAD-based env:

- → hard to prepare
- → realistic look requires a lot of effort
- → may be challenging to render
- → all tasks are supported

Szot et al. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. NeurIPS 2021.

Simulator: world model

World model is a special case of simulators. Usually these models allow to obtain sensor data conditioned on actions and additional input

Bruce et al. Genie: Generative Interactive Environments. arXiv:2402.15391
Zhu et al. IRASim: Learning Interactive Real-Robot Action Simulators. arXiv:2406.14540
Hu et al. GAIA-1: A Generative World Model for Autonomous Driving. arXiv:2309.17080

Real-world evaluation

→ the only evaluation that really matters
→ very slow
→ very expensive and technically complex
→ scales badly

Fu et al. Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation. arXiv:2401.02117

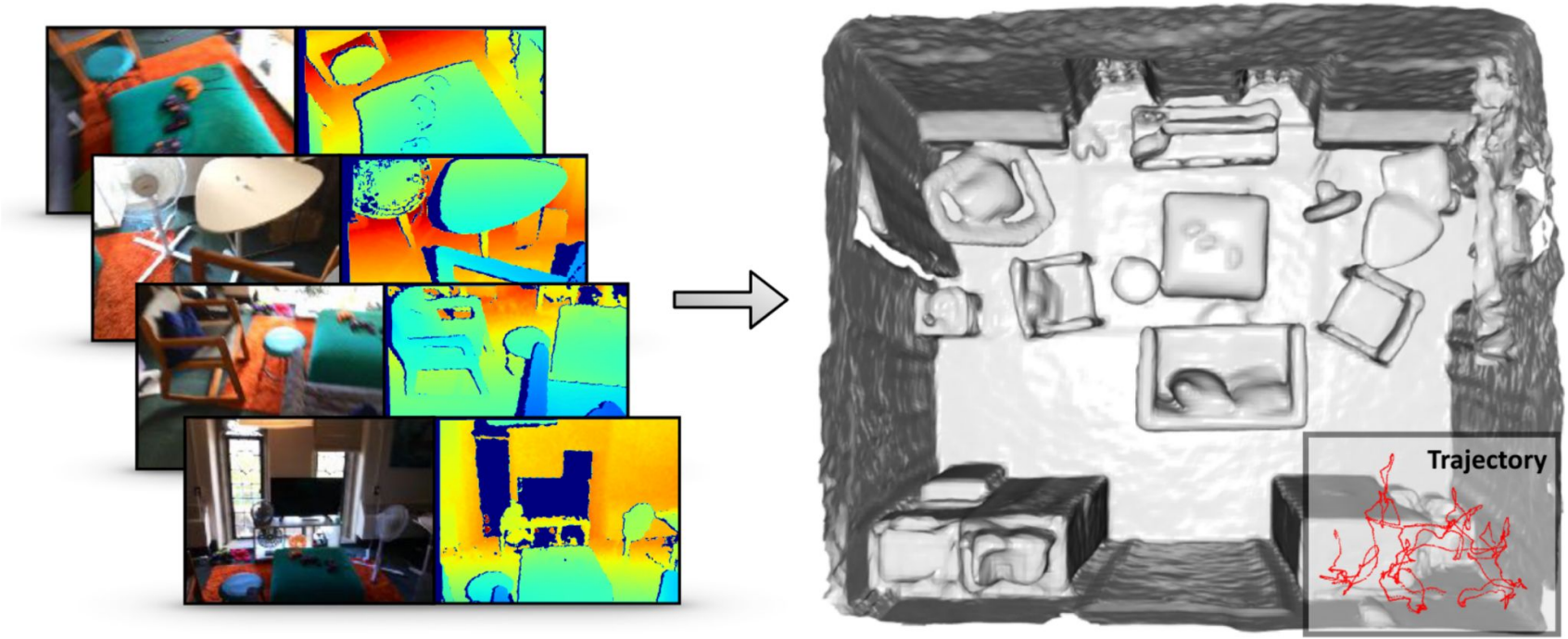
Comparison of evaluation methods

	Static benchmarks	Arena	End-to-end sim	End-to-end real
Relevance	Low	Medium	Medium	High
Safety	High	High	High	Low
Speed	High	Medium	Medium	Low
Cheapness	High	Medium	Medium	Low
Reproducibility	High	High	Medium	Low

03

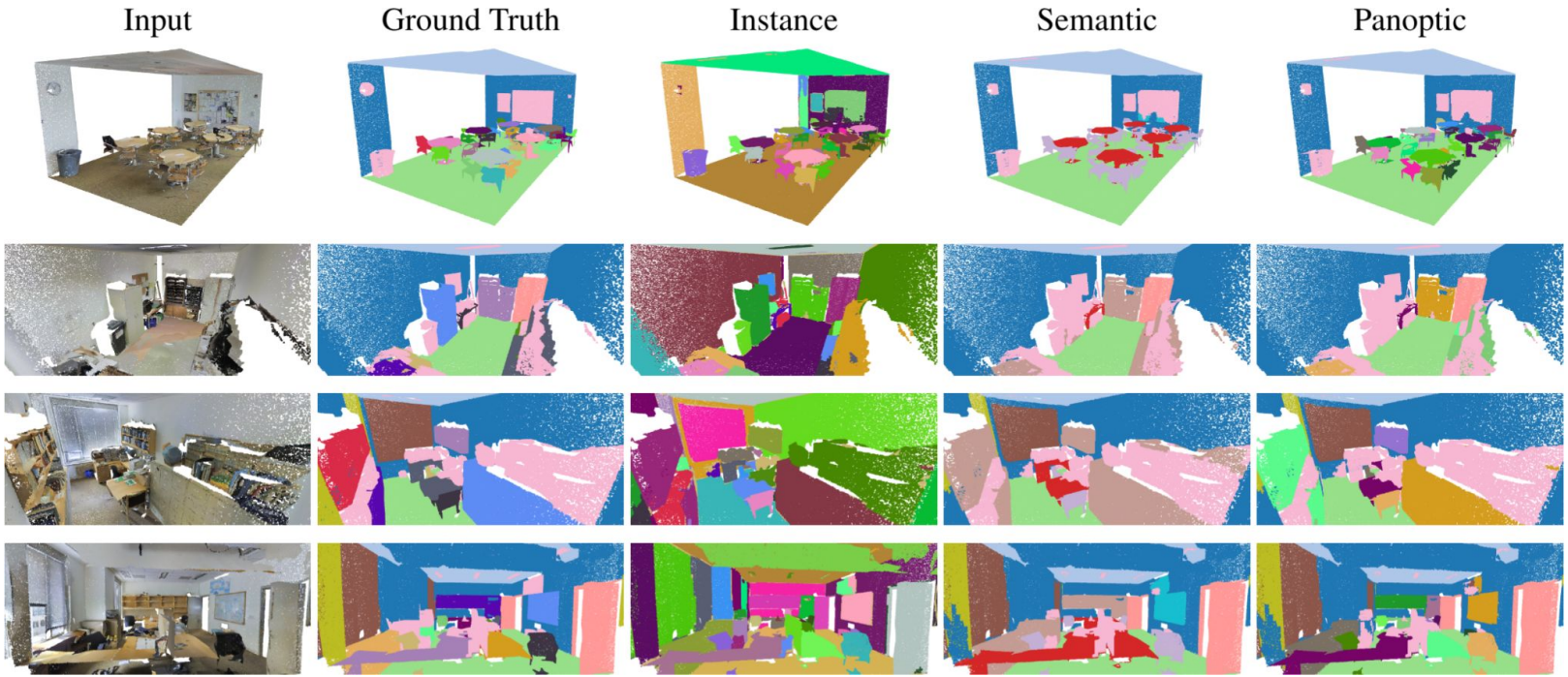
Understanding the world

3D reconstruction (SLAM)



3D segmentation

3D scene representation (i.e. point cloud) is segmented into a fixed set of classes



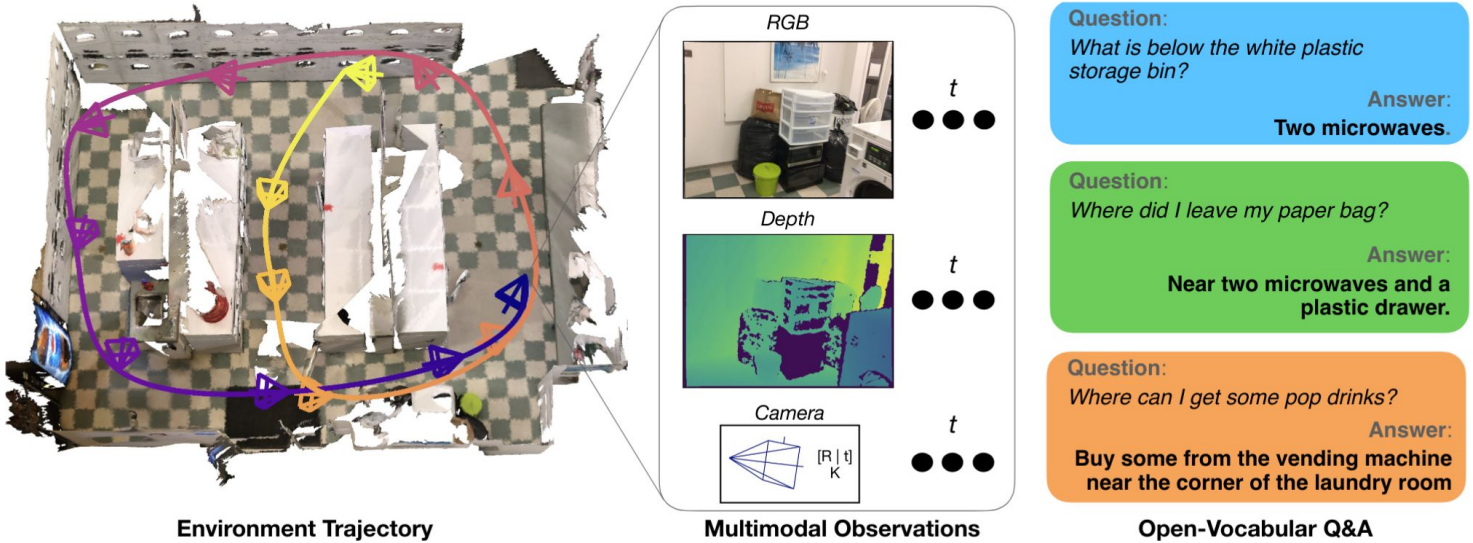
Kolodiazhnyi et al. OneFormer3D: One Transformer for Unified Point Cloud Segmentation. CVPR 2024

Embodied question answering

Agent has to answer questions about environment. Two scenarios are possible:

- → pre-recorded video stream
- → fully interactive mode: agent can freely explore environment

Evaluation of answers is done with GPT-4 or via human evaluation



Majumdar et al. OpenEQA: Embodied Question Answering in the Era of Foundation Models. CVPR 2024.

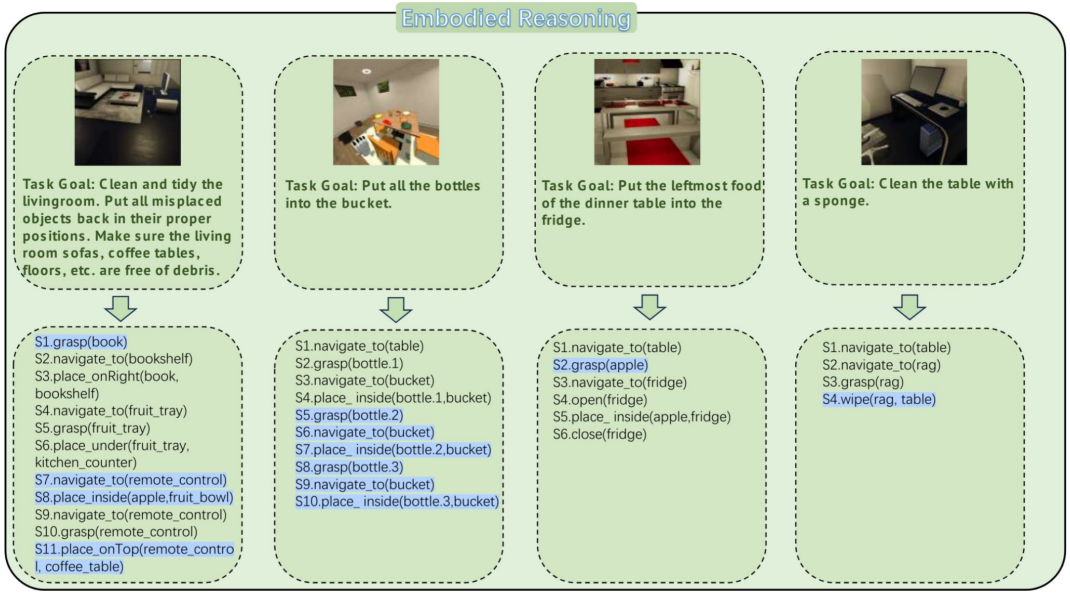
04

Planning

Planning with ground truth

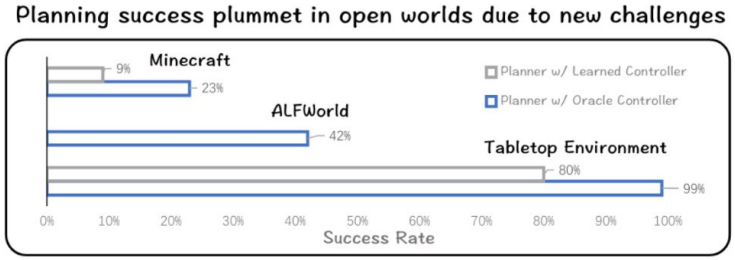
Planning is **decomposing** high-level task into sequence of sub-tasks. Simplest case of planning is planning with ground truth:

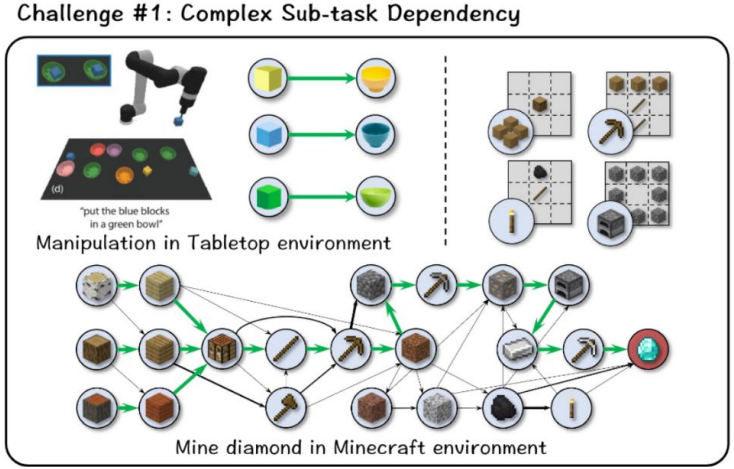
- → obtaining ground truth is laborious
- → quality may be measured by comparing sequences (accuracy, LCS length or GPT score)

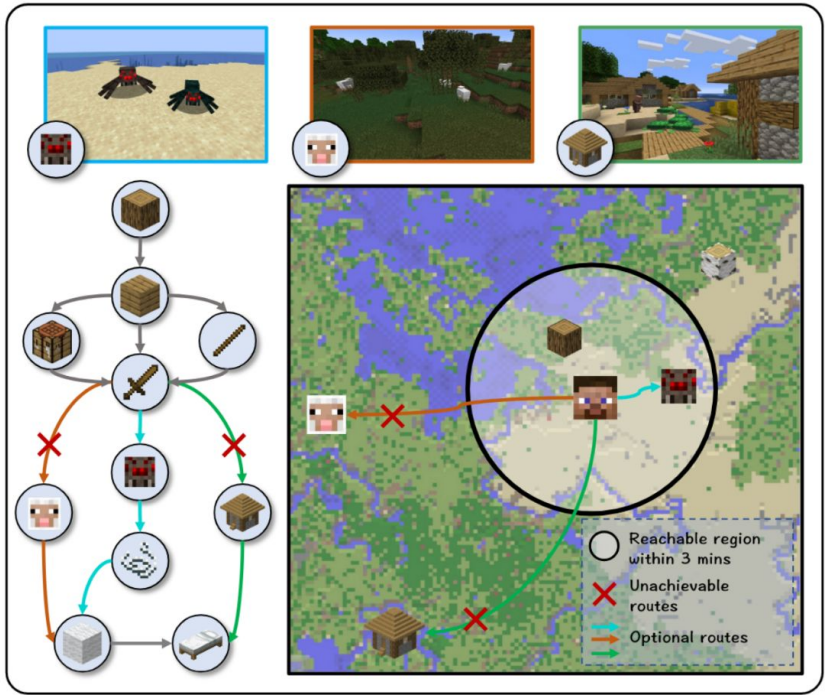


Zhang et al. MFE-ETP: A Comprehensive Evaluation Benchmark for Multi-modal Foundation Models on Embodied Task Planning. arXiv:2407.05047

Open-world planning







Wang et al. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. NeurIPS 2023

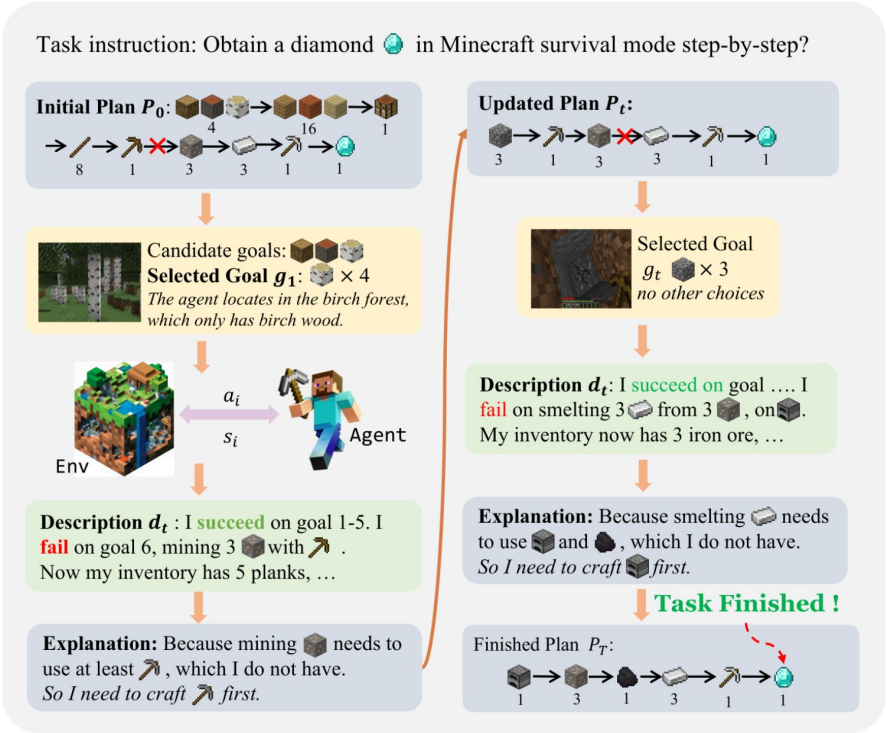
Open-world planning

Two challenges arise in open-world planning:

- → complex sub-task dependency that may be unknown in advance
- → sub-task feasibility depends on the (complex/unobservable) state of the world

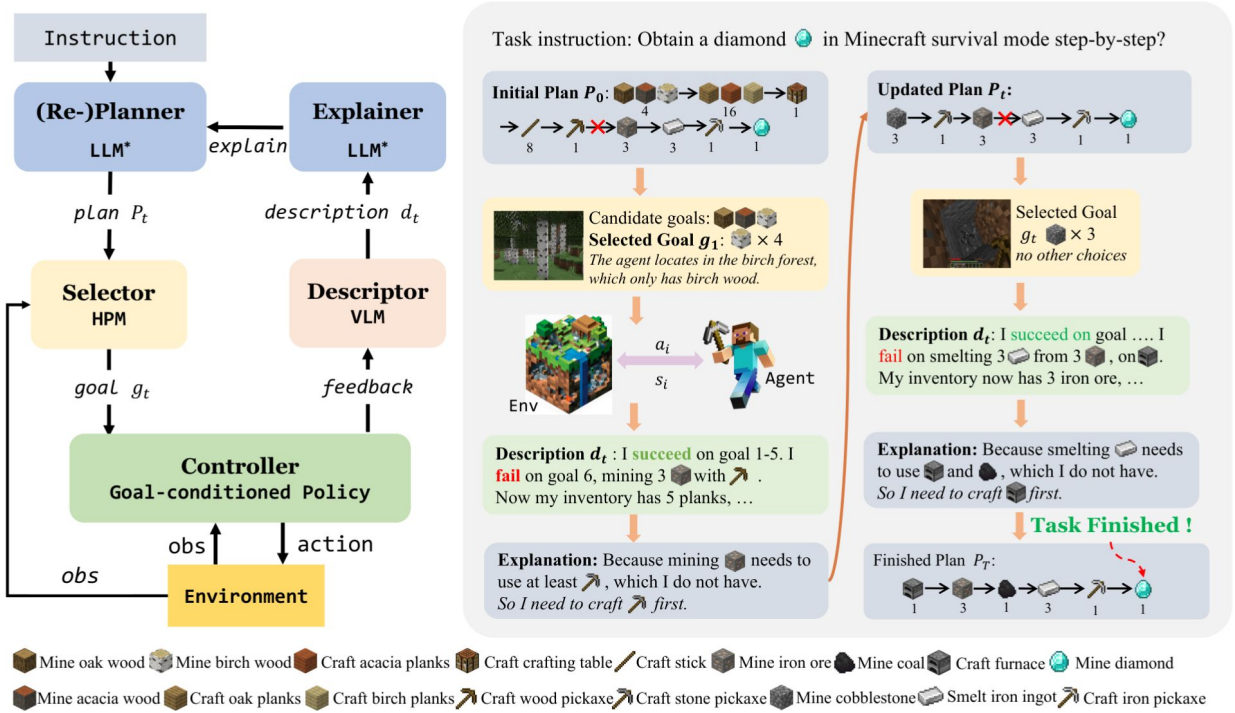
Therefore:

- → we can't benchmark planning on fixed ground truth sequences, running in simulator and measuring **success rate (SR) is the only option**
- → the planning module has to be adaptable and be able to **modify the plan**



Wang et al. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. NeurIPS 2023

Describe, Explain, Plan and Select (DEPS)

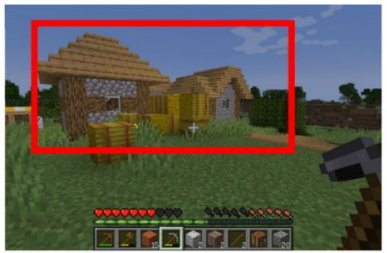


Describe, Explain, Plan and Select (DEPS)

Goal: Meat*3
Candidate Skill: Kill `Sheep` *OR* `Cow` *OR* Pig
Selection: Kill Sheep
Explanation: Meet sheep first.

Goal: Log*2
Candidate Skill: Chop Oak *OR* Birch *OR* Acacia Tree
Selection: Chop Acacia Tree
Explanation: Savanna biome only has Acacia tree.

Goal: Coal*1 AND Iron_Ore*1
Candidate Skill: Mine Coal *AND* Iron_Ore
Selection: Mine Iron_Ore
Explanation: Meet iron_ore first.



Goal: Survive in Night.
Candidate Skill: Sleep in bed *OR* Dig down.
Selection: Sleep_in_bed
Explanation: Village has beds.

Selector ranks candidate skills according to trained value/affordance function

Describe, Explain, Plan and Select (DEPS)

Meta	Name	Number	Example Task	Max. Steps	Initial Inventory	Given Tool
MT1	Basic	14	Make a wooden door.	3000	Empty	Axe
MT2	Tool (Simple)	12	Make a stone pickaxe.	3000	Empty	Axe
MT3	Hunt and Food	7	Cook the beef.	6000	Empty	Axe
MT4	Dig-Down	6	Mine coal.	3000	Empty	Axe
MT5	Equipment	9	Equip the leather helmet.	6000	Empty	Axe
MT6	Tool (Complex)	7	Make shears and bucket.	6000	Empty	Axe
MT7	IronStage	13	Obtain an iron sword.	6000	Empty	Axe
MT8	Challenge	1	Obtain a diamond!	12000	Empty	Axe

Methods	MT1	MT2	MT3	MT4	MT5	MT6	MT7	MT8	AVG
GPT[16, 32]	25.85±24.8	47.88±31.5	10.78±14.6	7.14±9.0	1.98±5.9	0.0±0.0	0.0±0.0	0.0±0.0	15.42
PP[42]	30.61±23.6	40.09±30.6	17.13±19.1	16.00±17.3	3.21±4.9	0.47±1.3	0.60±2.2	0.0±0.0	16.88
CoT[45]	40.24±30.8	55.21±26.8	6.82±11.6	4.76±8.2	1.73±5.2	0.0±0.0	0.0±0.0	0.0±0.0	18.89
IM[17]	46.89±31.4	53.73±20.8	3.64±6.9	18.41±17.4	4.57±7.4	0.64±1.7	1.02±2.5	0.0±0.0	21.64
CaP[20]	60.08±17.3	60.11±20.24	8.72±9.7	20.33± 21.0	2.84±4.6	0.63±1.3	0.60±2.2	0.0±0.0	25.77
DEP	75.70±10.4	66.13±13.4	45.69±16.2	43.35±20.2	15.93±13.9	5.71±3.7	4.60±7.1	0.50±0.5	**39.36**
DEPS	79.77±8.5	79.46±10.6	62.40±17.9	53.32±29.3	29.24±27.3	13.80±8.0	12.56±13.3	0.59±0.5	**48.56**

Describe, Explain, Plan and Select (DEPS)

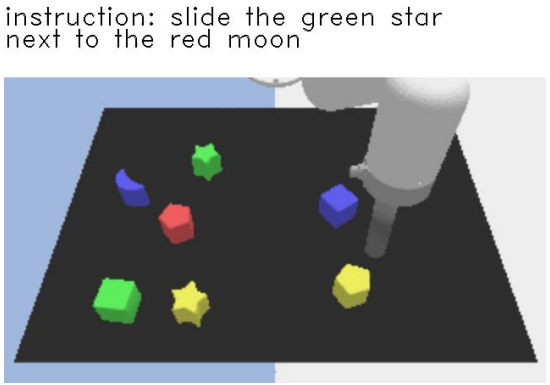
05

Acting: manipulation

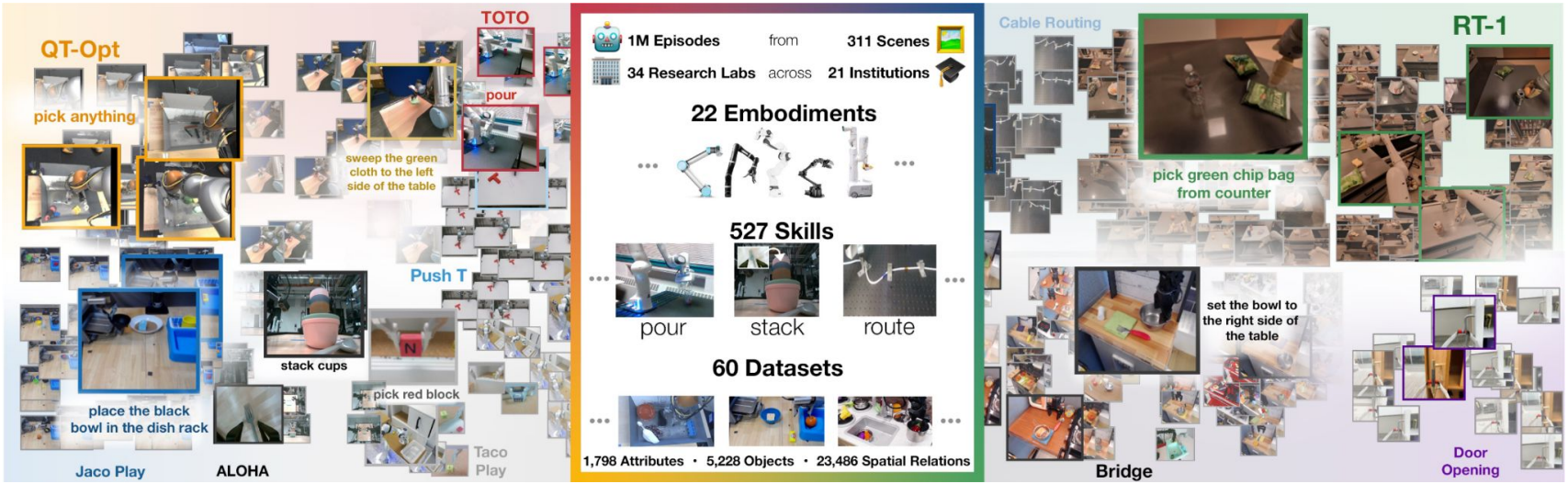
Manipulation

Given **language instruction** and **observations** from sensors, output **action/sequence of actions**. Modern notable approaches are based on:

- → hybrid models
- → diffusion models
- → transformers
- → VLLMs

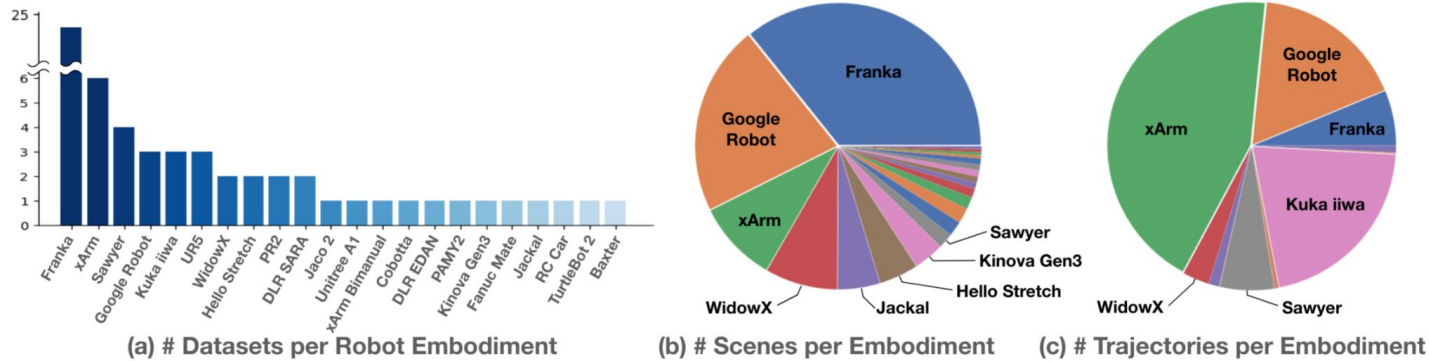


Towards ImageNet for manipulation: Open X-Embodiment

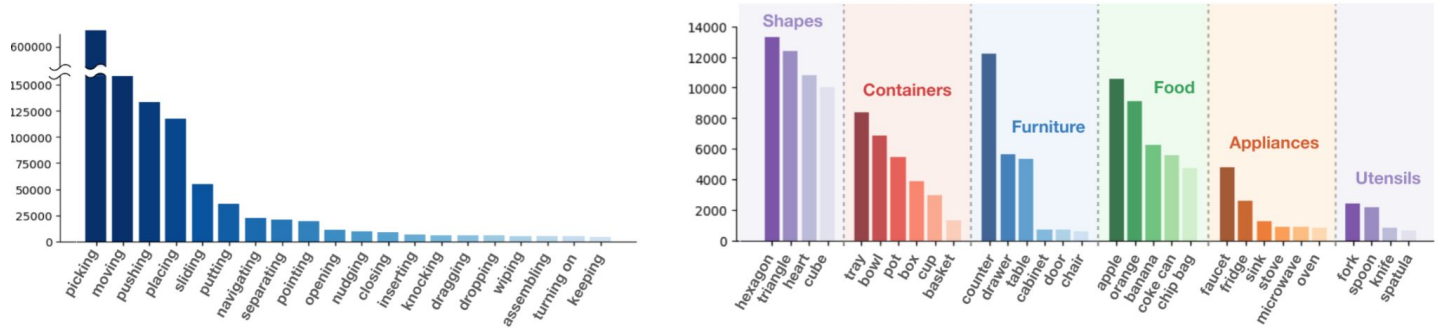


Open X-Embodiment Collaboration. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. arXiv:2310.08864

Open X-Embodiment

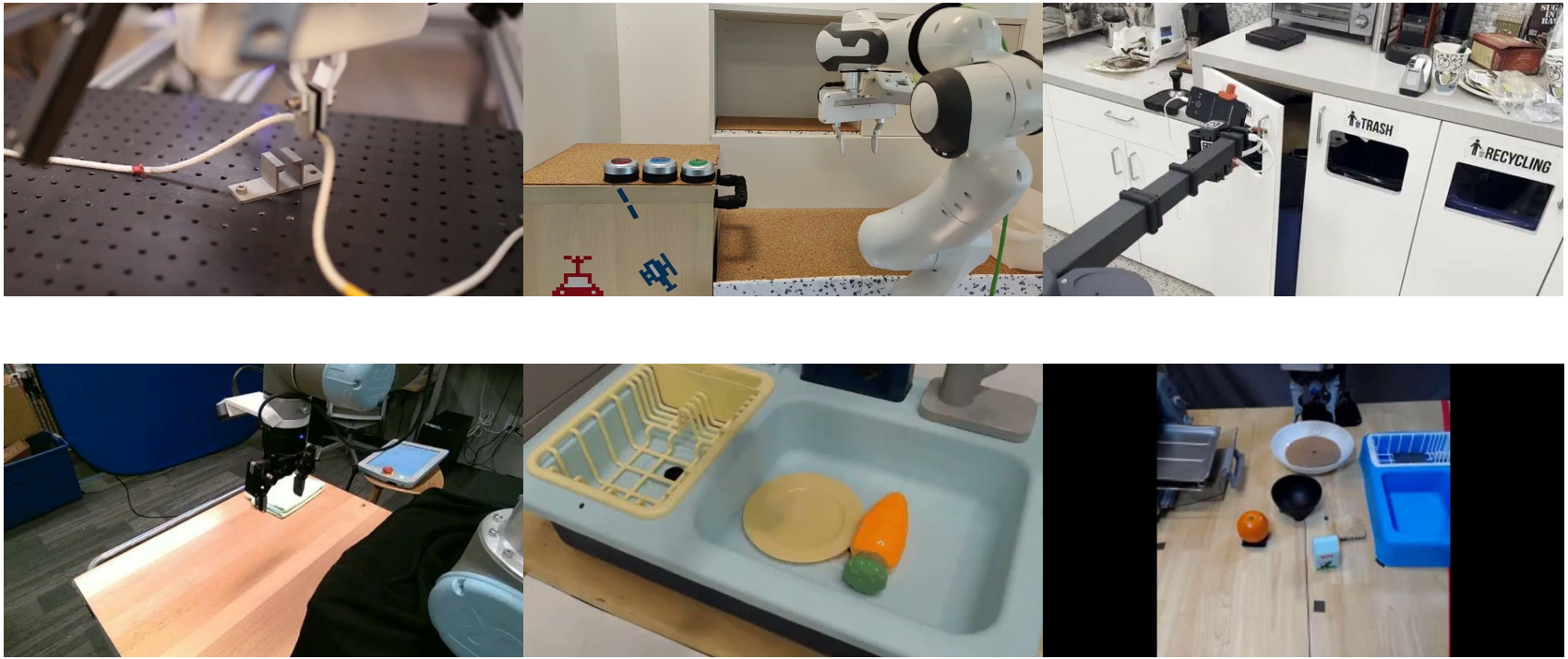


(a) # Datasets per Robot Embodiment (b) # Scenes per Embodiment (c) # Trajectories per Embodiment

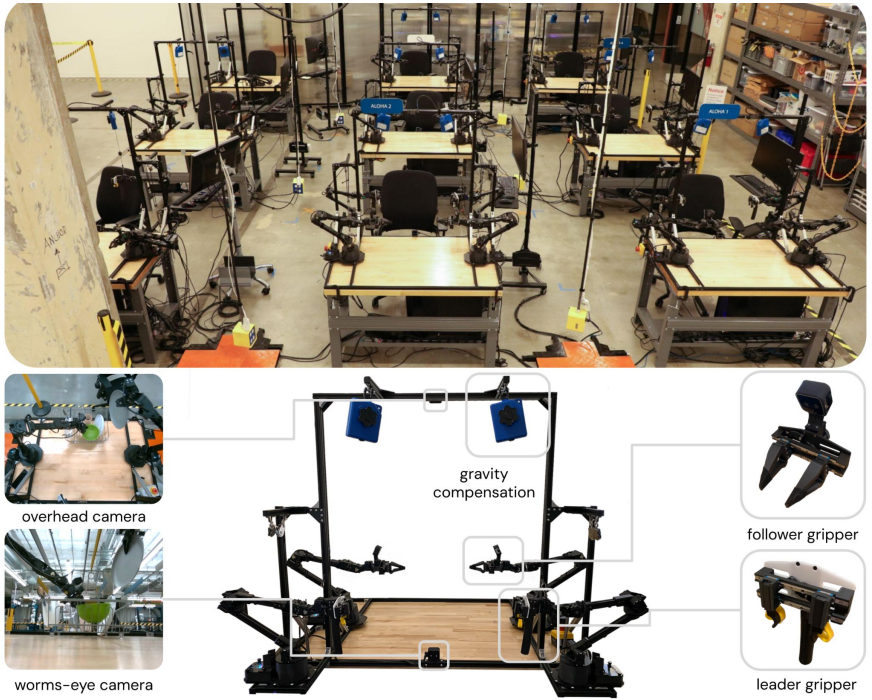


(d) Common Dataset Skills (e) Common Dataset Objects

Open X-Embodiment



Collecting real data: ALOHA



Aldaco et al. ALOHA 2: An Enhanced Low-Cost Hardware for Bimanual Teleoperation

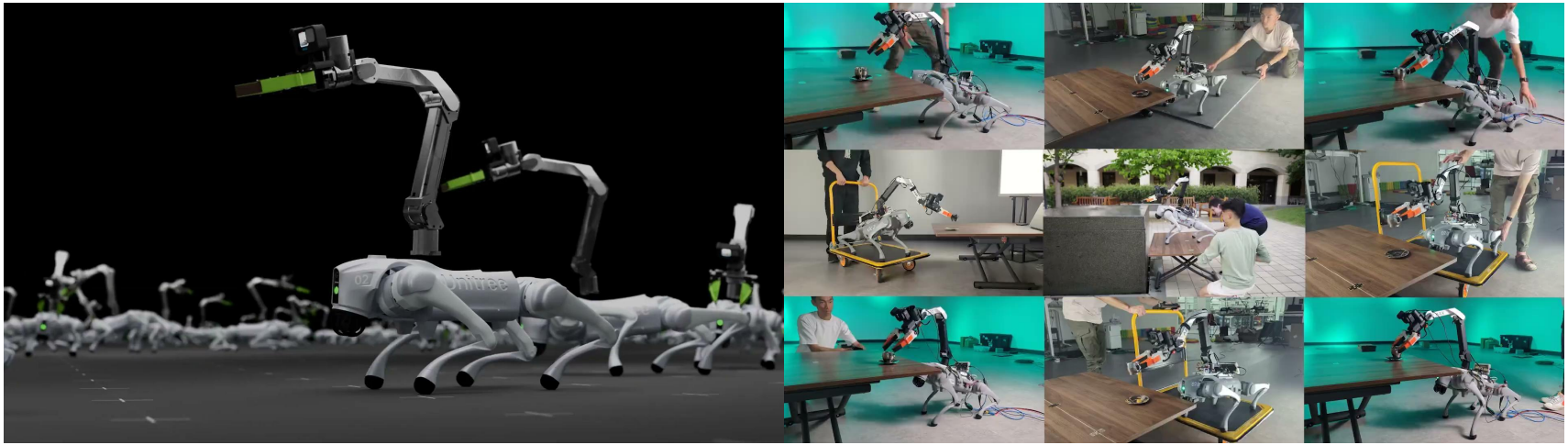
Collecting real data: ALOHA

Fu et al. Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation. arXiv:2401.02117

Collecting real data: UMI

Chi et al. Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-The-Wild Robots. RSS 2024

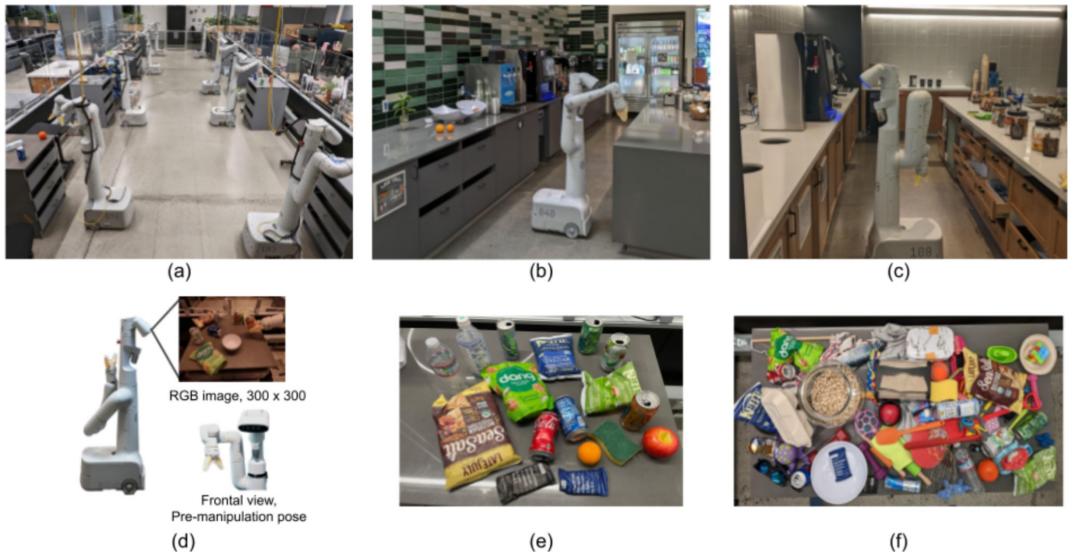
Adapting real data for movement: UMI on Legs



Ha et al. UMI on Legs: Making Manipulation Policies Mobile with Manipulation-Centric Whole-body Controllers. arXiv:2407.10353

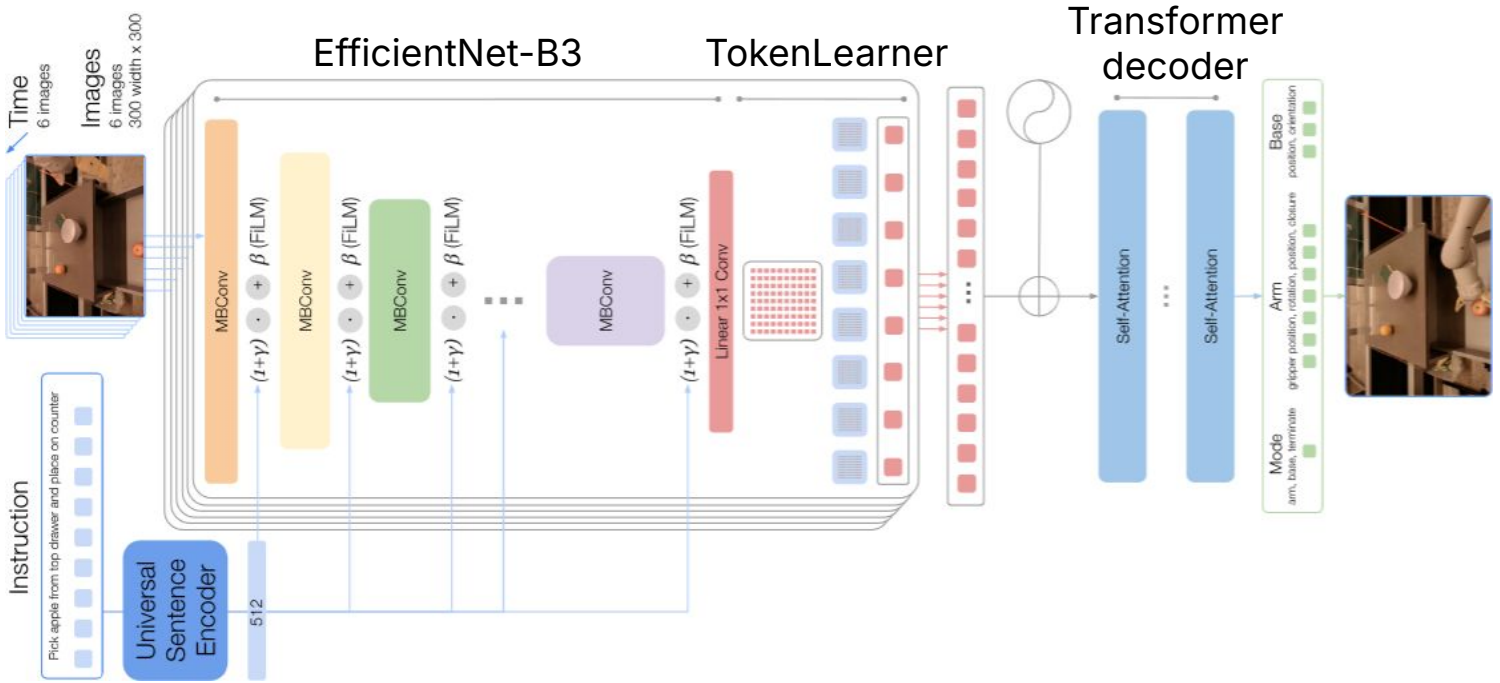
Hybrid policy: RT-1

Training data: 130k episodes, 700 tasks, collected with 13 robots over 17 months. Inference time is 100ms, overall system works at 3 Hz



Brohan et al. RT-1: Robotics Transformer for Real-World Control at Scale. arXiv:2212.06817

Hybrid policy: RT-1



Hybrid policy: RT-1

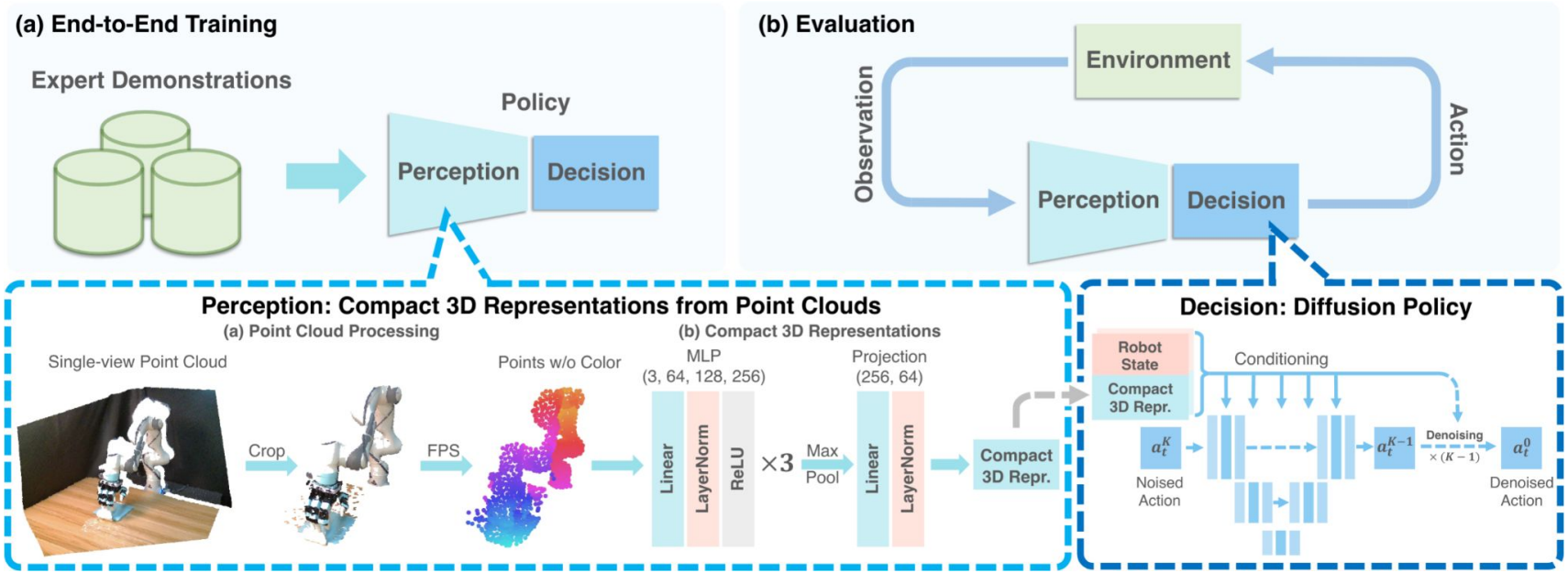
Key results:

- → RT-1 (35M parameters) successfully performs large number of instructions and generalizes to new tasks and environments
- → Adding synthetic data for unseen tasks improves success rate
- → RT-1 may be used together with SayCan to solve long-horizon tasks

Skill	Count	Description	Example Instruction
Pick `Object`	130	Lift the object off the surface	pick iced tea can
Move `Object` Near `Object`	337	Move the first object near the second	move pepsi can near rxbar blueberry
Place `Object` Upright	8	Place an elongated object upright	place water bottle upright
Knock `Object` Over	8	Knock an elongated object over	knock redbull can over
Open `Drawer`	3	Open any of the cabinet drawers	open the top drawer
Close `Drawer`	3	Close any of the cabinet drawers	close the middle drawer
Place `Object` into `Receptacle`	84	Place an object into a receptacle	place brown chip bag into white bowl
Pick `Object` from `Receptacle` and Place on the Counter	162	Pick an object up from a location and then place it on the counter	pick green jalapeno chip bag from paper bowl and place on counter
Section 6.3 and 6.4 tasks	9	Skills trained for realistic, long instructions	open the large glass jar of pistachios pull napkin out of dispenser grab scooper
Total	744		

Diffusion-based policies

Capture well multimodal nature of high-dimensional action distribution
Scalable for high dim. output (sequence of actions)



Ze et al. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. RSS 2024

Transformer-based policies: Octo

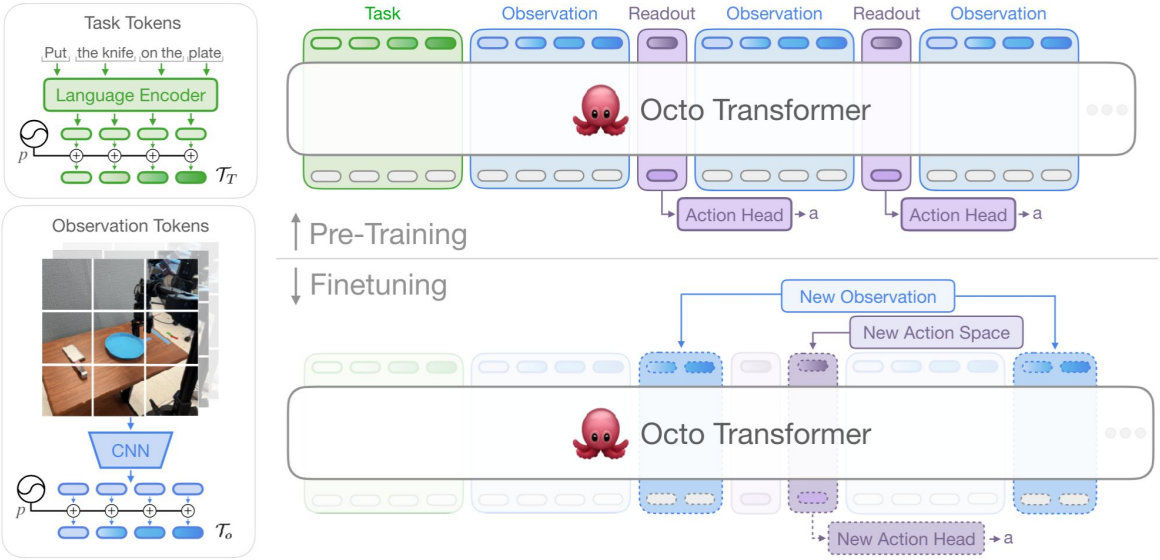


Fig. 2: **Model architecture. Left**: Octo tokenizes task descriptions (green) and input observations (blue) using a pretrained language model and a lightweight CNN, respectively. **Top**: The transformer backbone processes the sequence of task and observation tokens and produces readout tokens (purple) that get passed to output heads to produce actions. **Bottom**: The block-wise attention structure of the transformer allows us to add or remove inputs and outputs during finetuning: for example, we can add new observations (blue, dashed) or action spaces (purple, dashed) without modifying any pretrained parameters.

3 components:
tokenizers, **transformer**, **action head**

t5-base (111M) for text tokeniz.
Octo-Small (27M) ≈ ViT-S
Octo-Base (93M) ≈ ViT-B

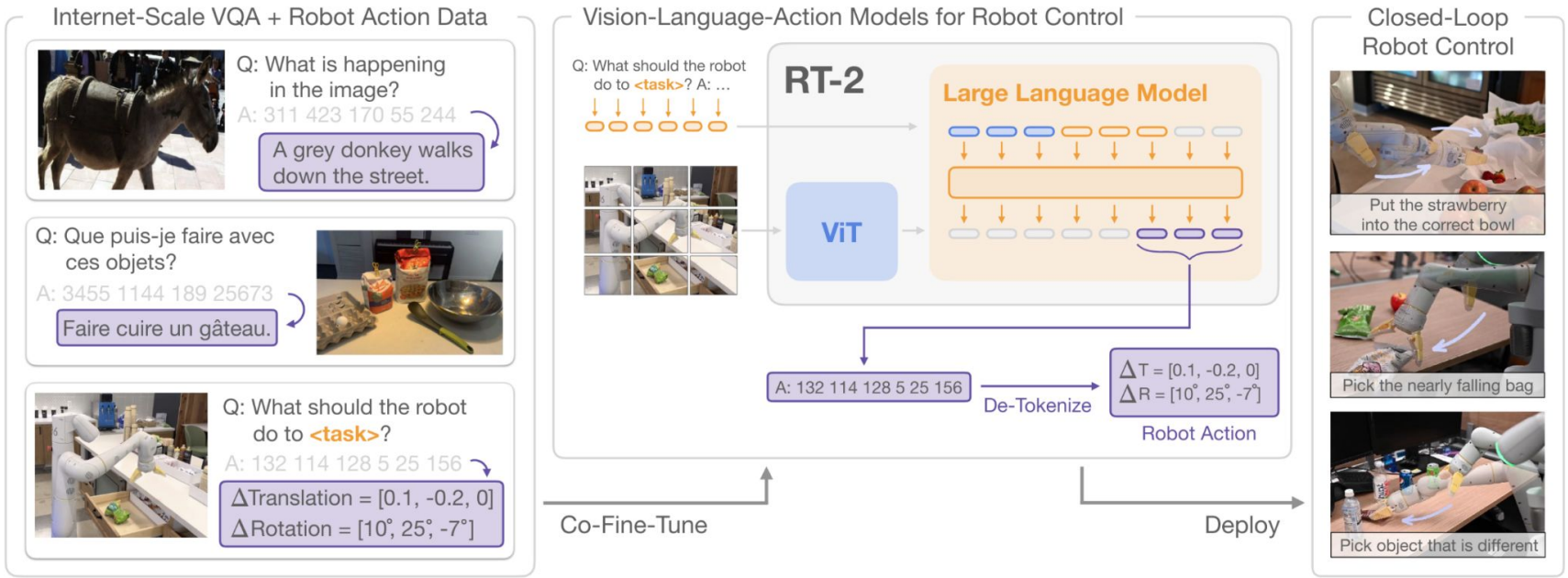
Pretrained on 128 TPUv4
(≈ 200 A100) for 14 hours,
much longer than on ImageNet

May be finetuned
on 3090 in 4 hours

Best starting point for training
own manipulation policies

VLLMs as policies: RT-2

Aim to utilize general knowledge obtained from web data

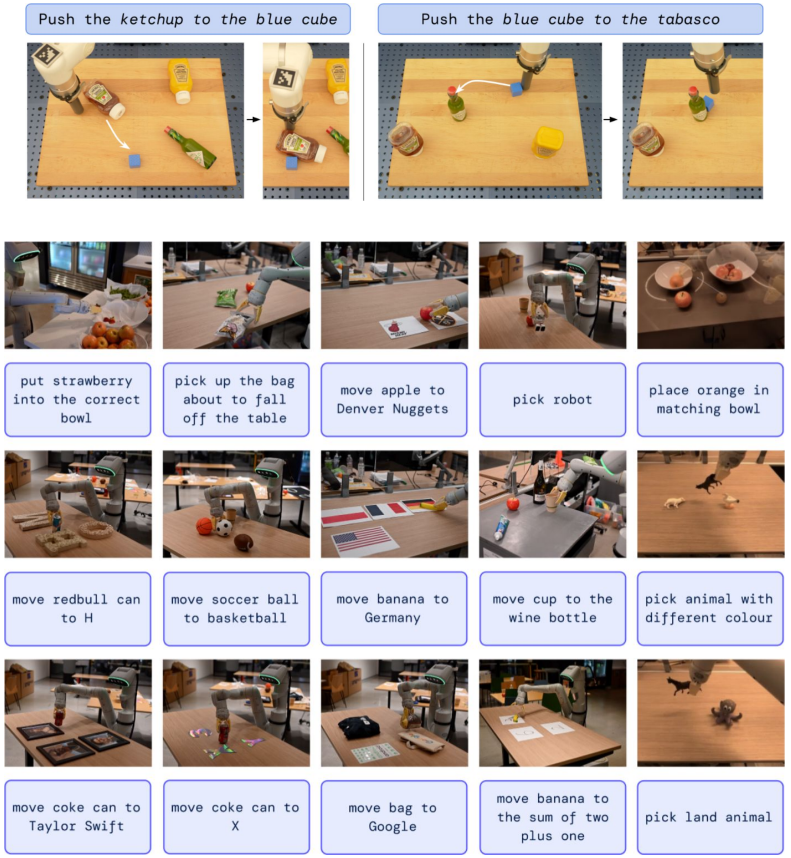


Brohan et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. arXiv:2307.15818

RT-2

Key idea: reuse knowledge from LLM
Results:

- → better generalization to unseen objects and environments
- → symbol understanding:
  move apple to 3
- → reasoning:
  move apple to cup with same color,
  move X near the sum of two plus one,
  mueve la manzana al vaso verde
- → human recognition:
  move the coke can to the person with glasses



RT-2 simple chain of thought

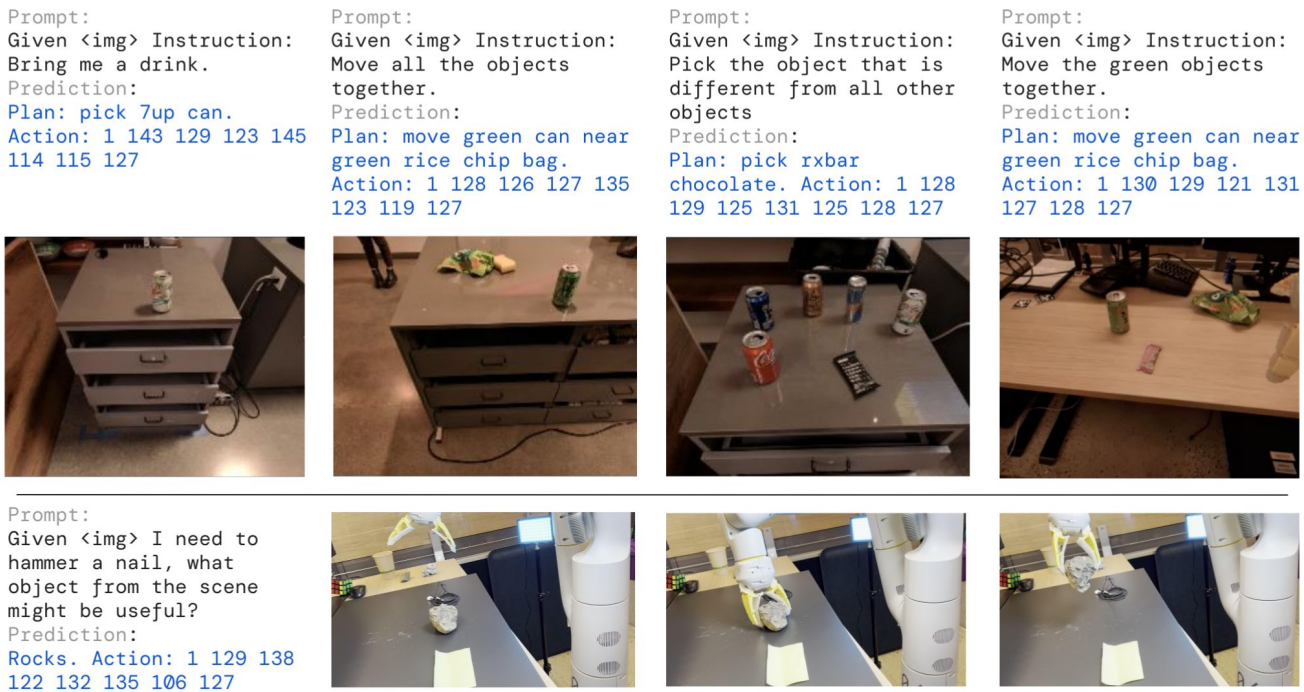
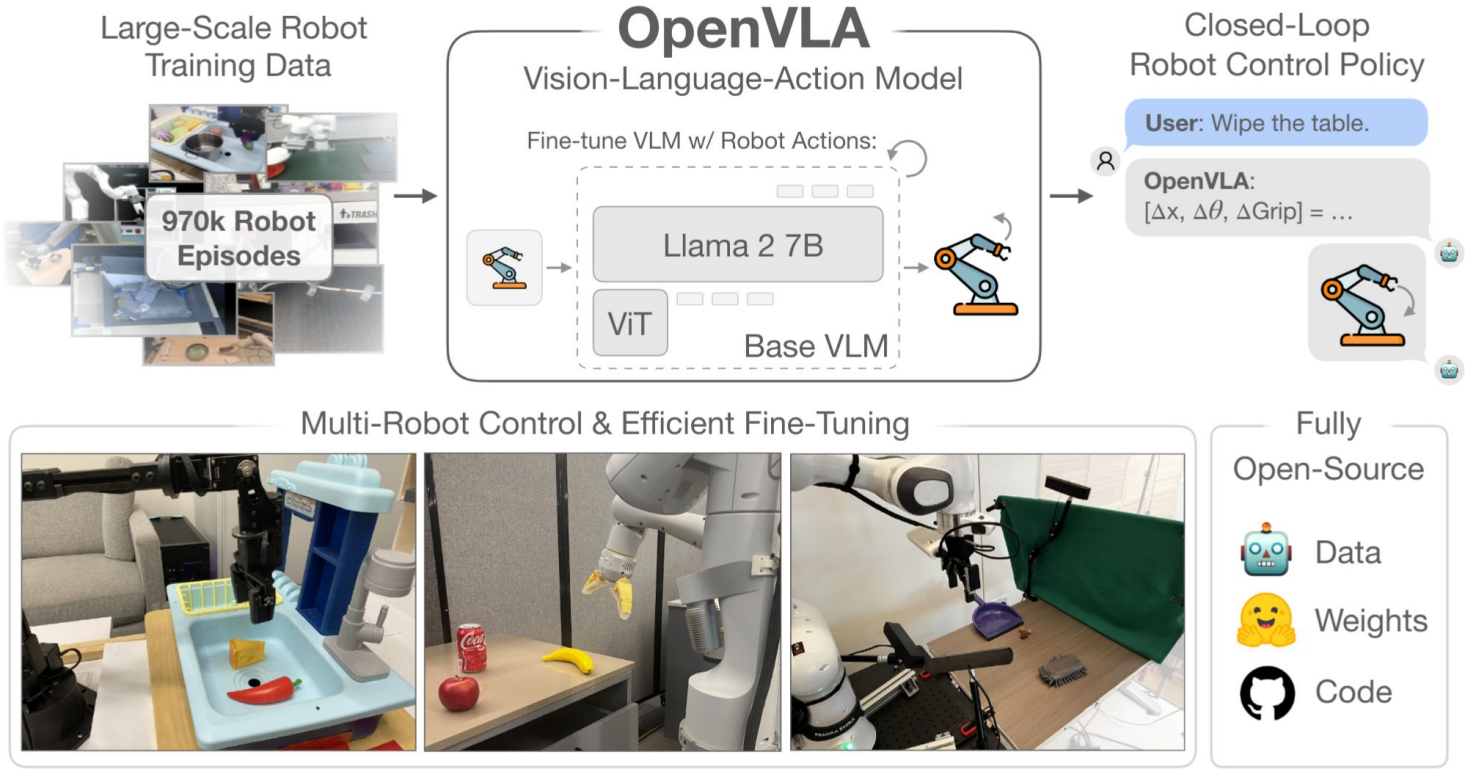


Figure 7 | Rollouts of RT-2 with chain-of-thought reasoning, where RT-2 generates both a plan and an action.

VLLMs as policies: OpenVLA



Kim et al. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246

OpenVLA

Pretrained on 64 A100 for 14 days, finetuned on 8 A100 for 5-15 hours
Runs at 6Hz on 4090 in bfloat16 format

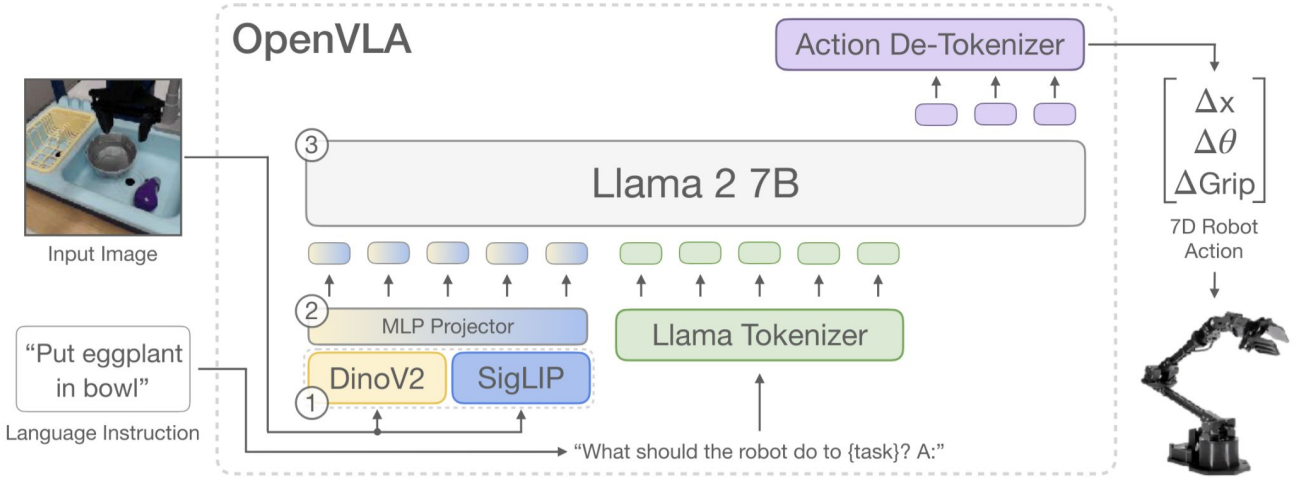


Figure 2: **OpenVLA model architecture.** Given an image observation and a language instruction, the model predicts 7-dimensional robot control actions. The architecture consists of three key components: (1) a **vision encoder** that concatenates Dino V2 [25] and SigLIP [77] features, (2) a **projector** that maps visual features to the language embedding space, and (3) the **LLM backbone**, a Llama 2 7B-parameter large language model [10].

OpenVLA

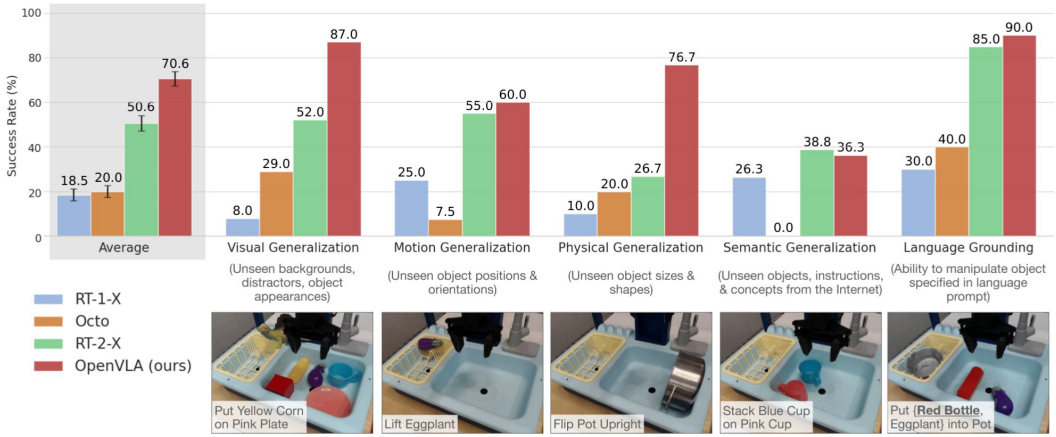


Figure 3: **BridgeData V2 WidowX robot evaluation tasks and results.** We evaluate OpenVLA and prior state-of-the-art generalist robot policies on a comprehensive suite of tasks covering several axes of generalization, as well as tasks that specifically assess language conditioning ability. OpenVLA achieves highest overall performance and even outperforms closed-source model RT-2-X in all categories except for semantic generalization. Average success rates ± StdErr are computed across 170 total rollouts per approach. See Table 4 for detailed results.

Table 1: **Parameter-efficient fine-tuning evaluation.** LoRA fine-tuning [26] achieves the best performance-compute trade-off, matching full fine-tuning performance while training only 1.4% of the model parameters. Mean success rate ± StdErr is computed across 33 rollouts per approach on select Franka-Tabletop tasks.
*: Sharded across 2 GPUs with FSDP [75].

Strategy	Success Rate	Train Params ($\times 10^6$)	VRAM (batch 16)
Full FT	**69.7 ± 7.2 %**	7,188.1	163.3 GB*
Last layer only	30.3 ± 6.1 %	465.1	51.4 GB
Frozen vision	47.0 ± 6.9 %	6,760.4	156.2 GB*
Sandwich	62.1 ± 7.9 %	914.2	64.0 GB
LoRA, rank=32	**68.2 ± 7.5%**	**97.6**	**59.7 GB**
rank=64	**68.2 ± 7.8%**	195.2	60.5 GB

8x speedup when training with LoRA (but we got only 3x)

OpenVLA

Future research directions:

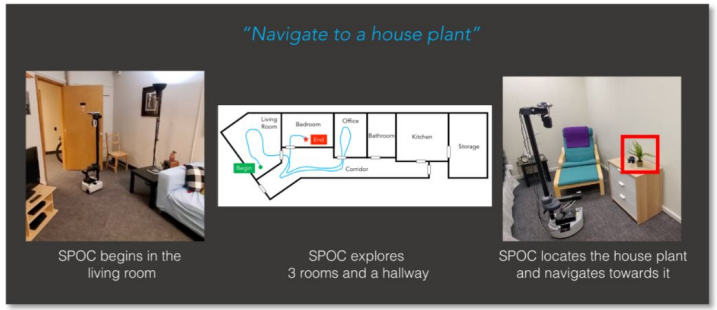
- → VLA model with multi-image/videos and depth observations
- → Real-time inference (50Hz). After int4-quantization the model runs only at 3Hz
- → Performance improvement (now SR < 90%)
- → Co-training for VQA and action prediction is to be explored

06

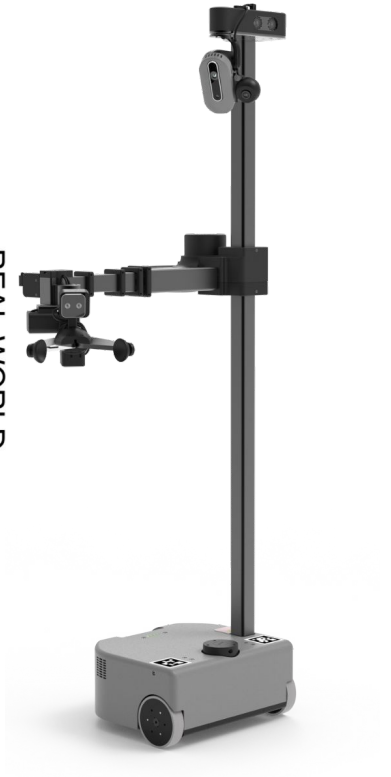
Acting: navigation

SPOC

SIMULATOR



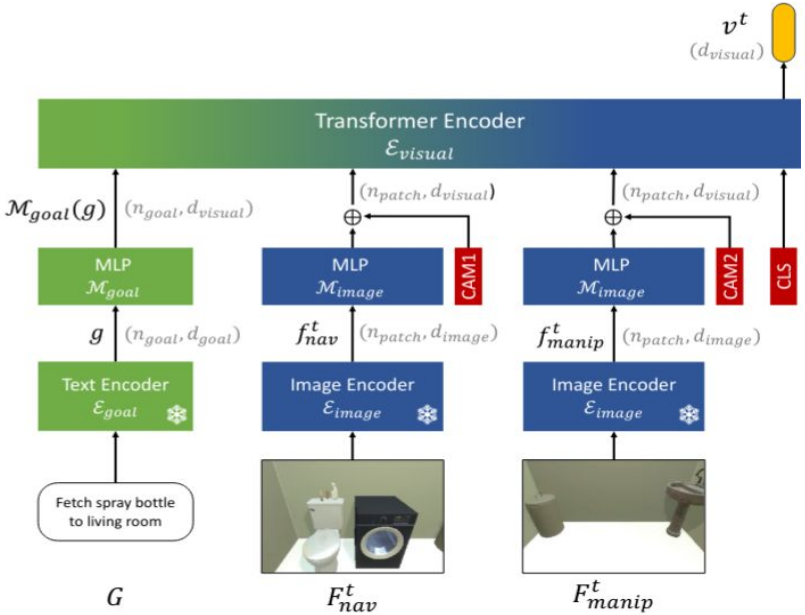
REAL WORLD

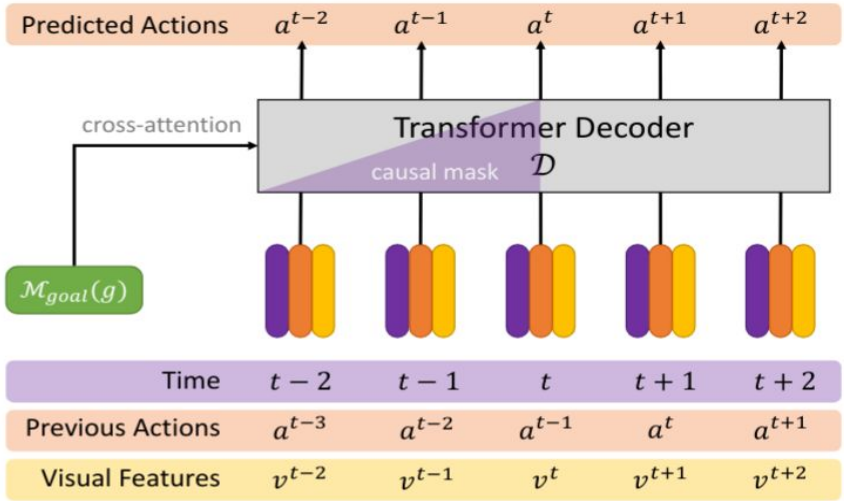


Ehsani et al. SPOC: Imitating Shortest Paths in Simulation Enables Effective Navigation and Manipulation in the Real World. CVPR 2024

SPOC

Discr. actions: move base (±20cm), rotate base (±6°,±30°), move arm (x, z) (±2cm, ±10cm)





SPOC

Data:

- AI2THOR simulator
- 40k household objects from Objaverse dataset
- Use ProcTHOR to procedurally generate 200k houses (1-8 rooms)

Trajectories:

- Navigation: go to target using approximation of shortest path
- Manipulation: navigate to object, then iteratively minimize distance between robot and object
- Room visitation: calculate center of house, then navigate to all rooms via shortest paths

SPOC benchmarks

CHORES:

- navigation
- object recognition
- object manipulation
- exploration

CHORES Nav:

- open voc. instruction following
- object affordance
- scene understanding
- object comparison

Task	Description & Example
ObjNav	Locate an object category: "find a mug"
PickUp	Pick up a specified object in agent line of sight: "pick up a mug"
Fetch	Find and pick up an object: "locate a mug and pick up that mug"
RoomVisit	Traverse the house. "Visit every room in this 5-room house. Indicate when you have seen a new room and when you are done."

Table 1. CHORES tasks.

Task	Target Description & Example
ObjNav	Object's category: "vase"
ObjNavAfford	Object's possible uses: "a container that can best be used for holding fresh flowers decoratively"
ObjNavLocalRef	Object's nearby objects: "a vase near a tennis racket and a basketball"
ObjNavRelAttr	Object category comparative attribute: "the smallest vase in the bedroom"
ObjNavRoom	Object's room type: "vase in the living room"
ObjNavDesc	Open vocab instance description: "the brown vase painted orange with a bird on the side"
RoomNav	Type of room: "bedroom"

Table 2. CHORESNav tasks. The full task specification also includes a navigation verb, such as "Search for a vase".

SPOC

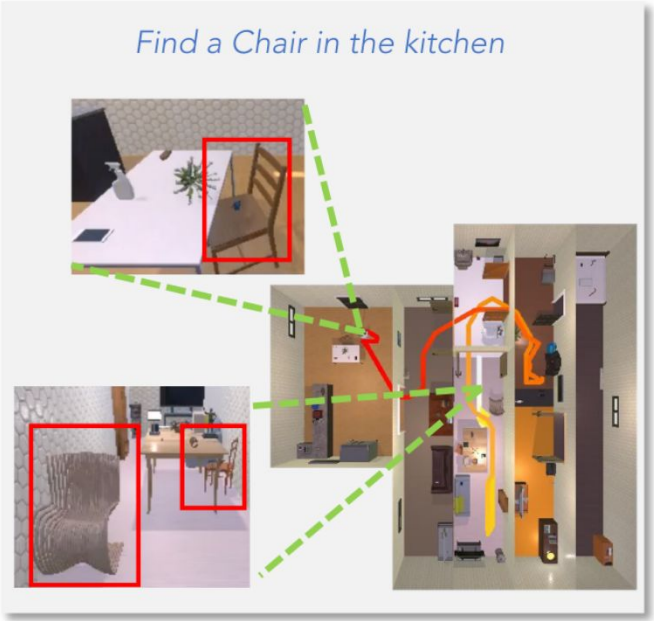
Key results:

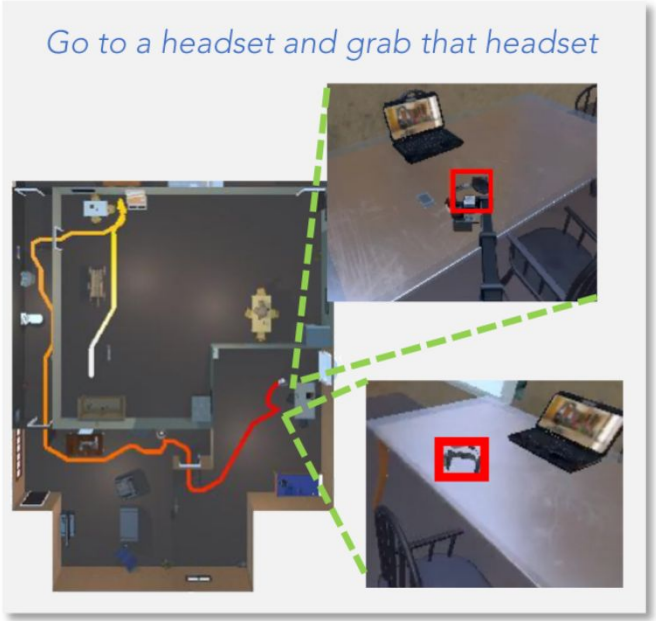
- → Imitation learning outperforms RL in speed (3500 FPS vs 175 FPS) and quality
- → Architecture scales well to multiple tasks
- → Object detection means a lot
- → Transformer encoders and decoders boost quality a lot
- → Long horizon tasks require long context windows (100 timesteps here vs 6 timesteps in RT-1)
- → SPOC generalizes to real world

Benchmark	Model	Training	ObjNav			PickUp			Fetch			RoomVisit			Avg
			Success	SEL	%Rooms	Success	SEL	%Rooms	Success	SEL	%Rooms	Success	SEL	%Rooms	Success
Chores -𝕊	EmbSigLIP* [38]	Single-task RL	36.5	24.5	42.2	71.9	52.9	30.3	0.0	0.0	50.5	16.5	11.9	44.6	31.2
	SPOC-1-task	Single-task IL	57.0	46.2	51.5	84.2	81.0	30.3	15.1	12.6	48.1	43.7	40.4	81.2	50.0
	SPOC	Multi-task IL	55.0	42.2	56.3	90.1	86.9	30.3	14.0	10.5	49.3	40.5	35.7	81.1	49.9
	SPOC w/ GT Det	Multi-task IL	85.0	61.4	58.7	91.2	87.9	30.3	47.3	35.6	61.6	36.7	33.7	79.3	65.0
Chores -𝕃	SPOC	Multi-task IL	33.7	25.1	53.7	75.1	69.1	31.5	10.6	8.1	42.9	35.0	33.2	77.8	38.6
	SPOC w/ GT Det	Multi-task IL	83.9	58.0	64.0	78.0	75.7	31.5	48.6	38.3	60.0	42.0	39.1	83.1	63.1

Model	ObjNav	PickUp	Fetch	RoomVisit	Average
SPOC	50.0	46.7 (66.7)	11.1 (33.3)	50.0	39.5
SPOC w/ Detic	83.3	46.7 (86.7)	44.4 (44.4)	50.0	56.1

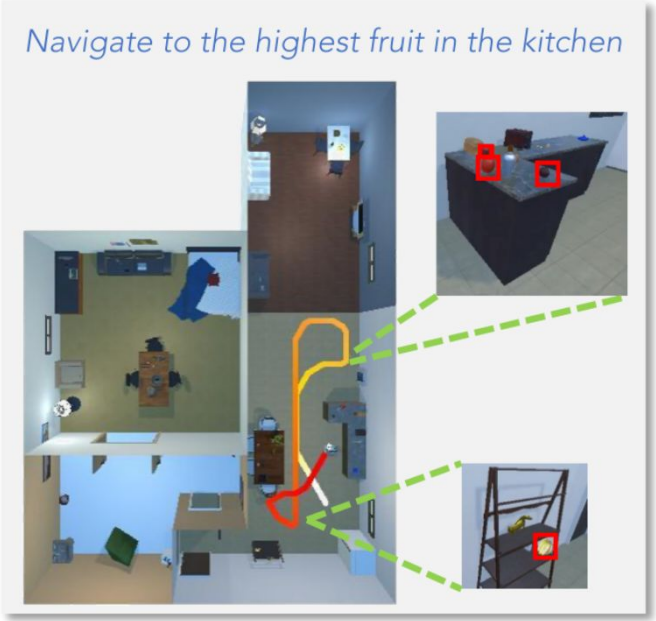
SPOC

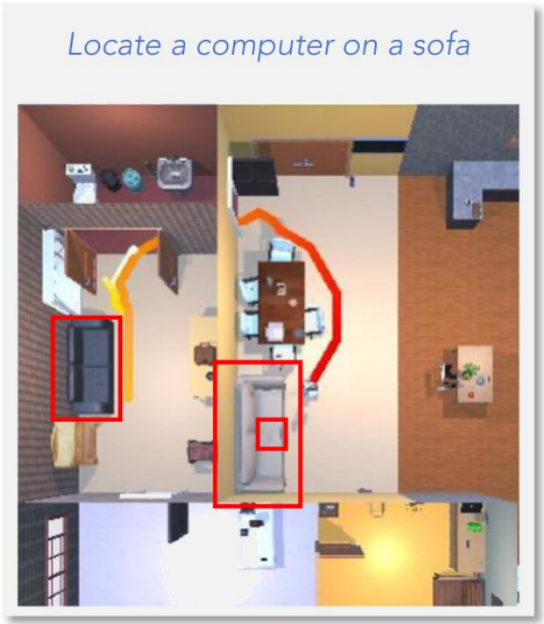




Left: skipping all chairs to find one in the kitchen
Right: repositioning itself to find a location where headset is reachable

SPOC



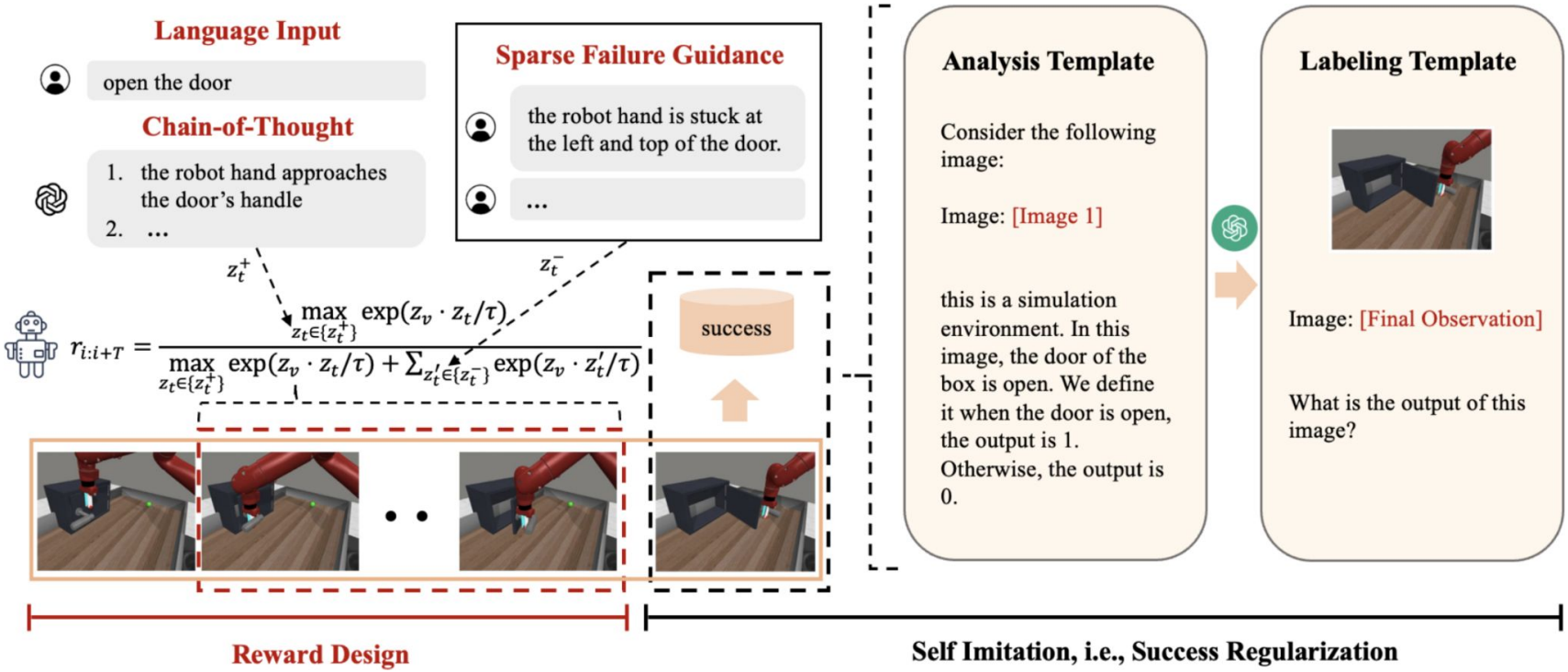


Left: looking for all fruits in the kitchen and then navigating to highest
Right: looking for a sofa which has a laptop on it

07

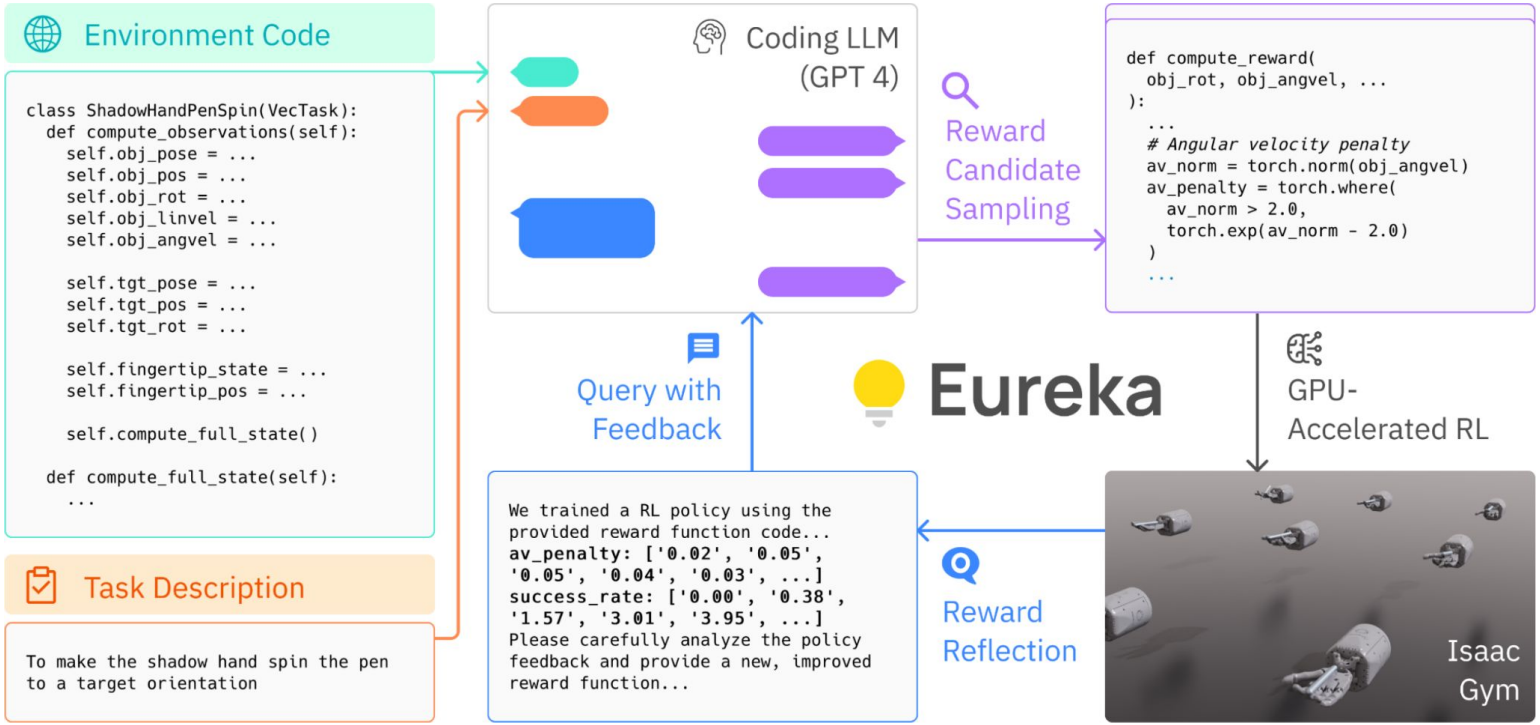
Reward function

RoboCoT



Zhang et al. Learning Manipulation Skills through Robot Chain-of-Thought with Sparse Failure Guidance. arXiv:2405.13573

Eureka



Ma et al. Eureka: Human-Level Reward Design via Coding Large Language Models. ICLR 2024

Eureka

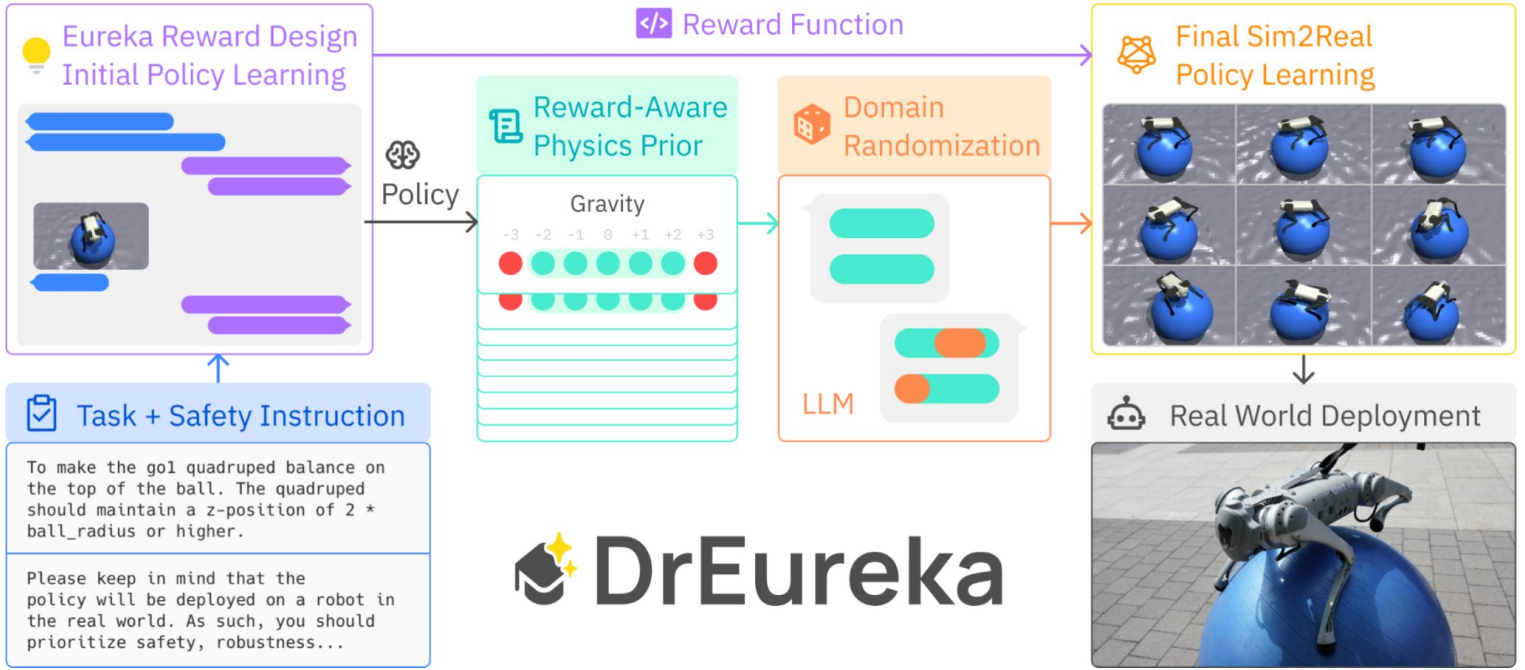
```
def compute_reward(object_rot, goal_rot, object_angvel, object_pos, fingertip_pos):
    # Rotation reward
    rot_diff = torch.abs(torch.sum(object_rot * goal_rot, dim=1) - 1) / 2
-   rotation_reward_temp = 20.0
+   rotation_reward_temp = 30.0                                          Changing hyperparameter
    rotation_reward = torch.exp(-rotation_reward_temp * rot_diff)

    # Distance reward
+   min_distance_temp = 10.0
    min_distance = torch.min(torch.norm(fingertip_pos - object_pos[:, None], dim=2), dim=1).values
-   distance_reward = min_distance
+   uncapped_distance_reward = torch.exp(-min_distance_temp * min_distance)
+   distance_reward = torch.clamp(uncapped_distance_reward, 0.0, 1.0)    Changing functional form

-   total_reward = rotation_reward + distance_reward
+   # Angular velocity penalty                                           Adding new component
+   angvel_norm = torch.norm(object_angvel, dim=1)
+   angvel_threshold = 0.5
+   angvel_penalty_temp = 5.0
+   angular_velocity_penalty = torch.where(angvel_norm > angvel_threshold,
+       torch.exp(-angvel_penalty_temp * (angvel_norm - angvel_threshold)), torch.zeros_like(angvel_norm))
+
+   total_reward = 0.5 * rotation_reward + 0.3 * distance_reward - 0.2 * angular_velocity_penalty

    reward_components = {
        "rotation_reward": rotation_reward,
        "distance_reward": distance_reward,
+       "angular_velocity_penalty": angular_velocity_penalty,
    }

    return total_reward, reward_components
```

Eureka

Key results of the method:

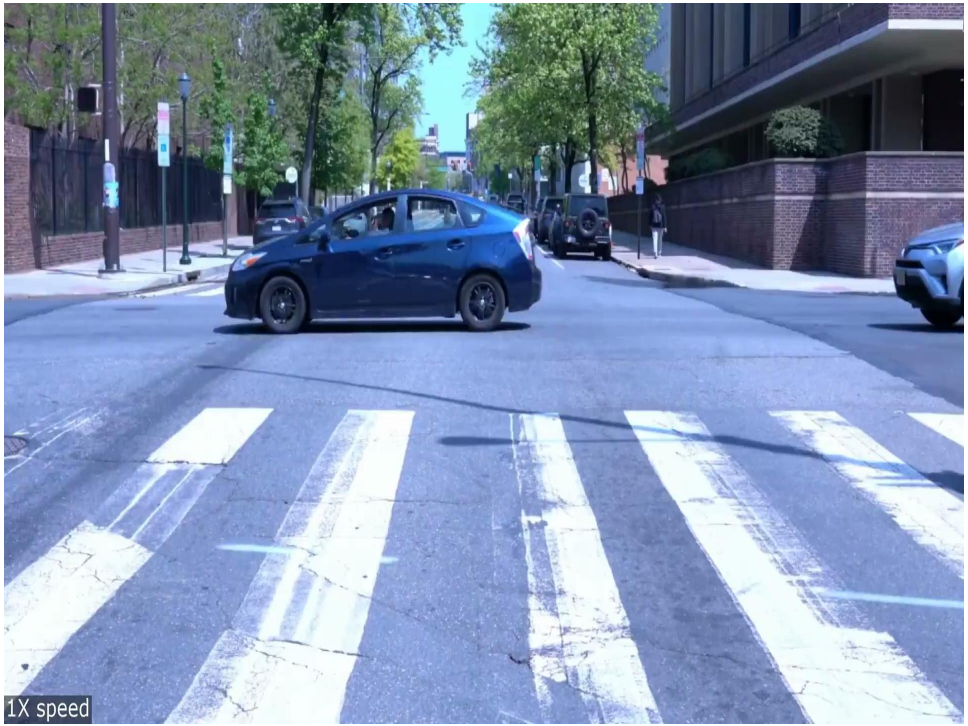
- → outperforms human reward on a wide range of environments
- → consistently improves over time
- → generates novel rewards compared to human rewards
- → improves from human feedback

DrEureka



Ma et al. DrEureka: Language Model Guided Sim-To-Real Transfer. RSS 2024

DrEureka



08

Open questions

(Some) Open questions

- → Unified and effective evaluation
- → Sim-to-real gap
- → Pre-training fundamental models for robotics
- → Efficient collection and usage of human demonstration data
- → High inference time of foundation models
- → Long-horizon task planning
- → Life-long learning
- → Ensuring robustness and safety of deployed models

Conclusion

- Embodied AI is a research area at the intersection of NLP, CV and RL
- Embodied agent has an **embodiment** and **AI software**
- EAI models enable robot to **perceive** the world, **talk**, **reason** and **act**
- Evaluation of EAI models in general is a very challenging task
- Despite a decade of the rapid progress in NLP and CV, EAI systems (**understanding the world, planning and acting**) are in the beginning of their development