OVMM[1] and OK-Robot[2]

🗿 MOAI Lab / 2026.03.26

Kwangmo Yang

1) S. Yenamandra et al., "HomeRobot Open Vocab Mobile Manipulation Challenge," NeurIPS Competition Track, 2023.
2) Liu et al., "*OK-Robot: What Really Matters in Integrating Open-Knowledge Models*", 2024.

Outline

OVMM Introduction

OK-Robot Explanation

OK-Robot Discussion

Future Work

OK-Robot is a representative example solving OVMM task

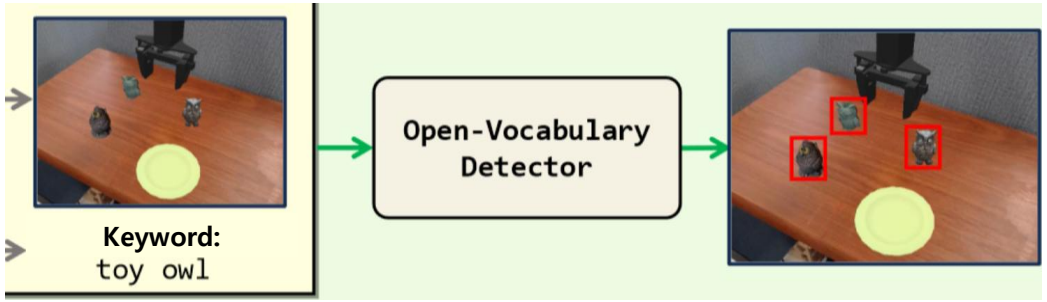
OVMM introduction

: Open-Vocabulary Mobile Manipulation

- **Open-Vocabulary**

- Open label space
- Deal with **unrestricted** natural language description in inference time
- Differs from Zero-shot or others in that Open-Vocabulary can **make user possible to refer to label of targets freely** in natural language form

e.g.)



- **Mobile Manipulation**

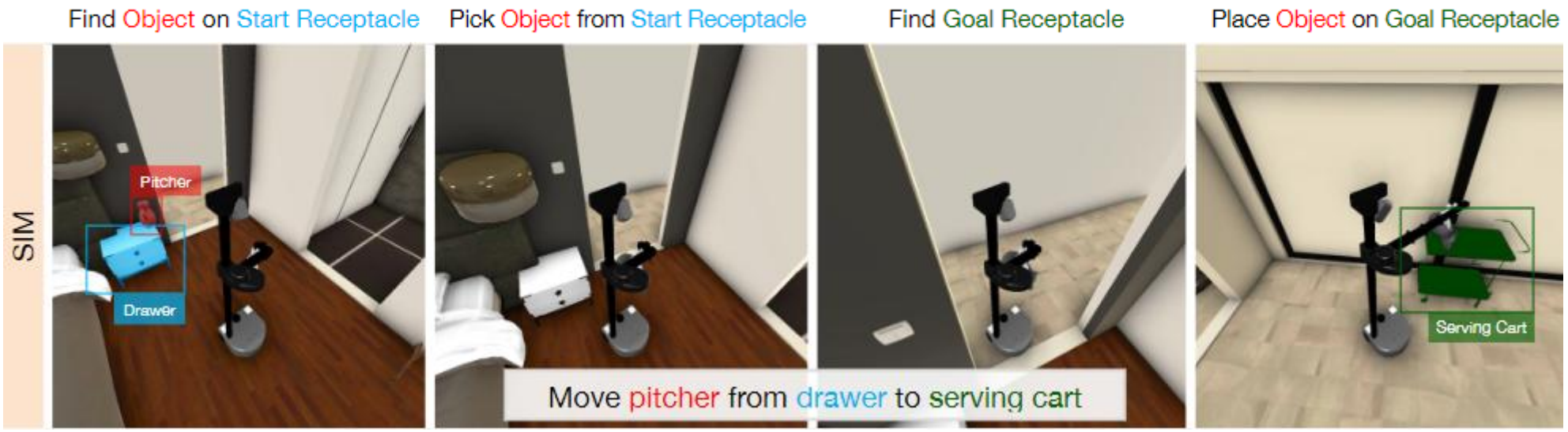
- Navigation & Manipulation



OVMM introduction

Task: A problem to Pick&Place any (unseen) object in unseen environment
→ command form of "Move *object* from *start_receptacle* to *goal_receptacle*"

(General) **process**: Exploration1 + Detection + Pick + Exploration2 + Place



Can be solved by Heuristic, RL, VLA, or combination of them etc…

OVMM introduction

Dataset composition (in simulation)

Scenes: 60 scenes in Habitat Synthetic Scenes Dataset (HSSD)

→ Train 38, Validation 12, Test 10

Objects: Curated from AI2Thor, Amazon-Berkeley Objects, Google Scanned Objects

→129 categories, 2,535 objects

Receptacles: Automatic labeling on HSSD receptacle objects, and remove invalid/inaccessible ones manually

→ 21 Categories



Figure 3: HSSD scenes.

trainig validation & test

	SC, SI	SC, UI	UC, UI	Total
Cats	85	64	44	129
Insts	1,363	748	424	2,535

Table 2: # of objects in the sim for each split of (S)een and (U)nseen (I)nstance and (C)ategory.

Challenges of OVMM

- Baseline results in OVMM paper

Simulation Results	Skill			Partial Success Rates			Overall Success Rate	Partial Success Metric
Perception	Navigation	Gaze	Place	FindObj	Pick	FindRec		
Ground Truth	Heuristic	None	Heuristic	54.1	48.5	31.5	5.1	34.8
	Heuristic	RL	RL	56.5	51.5	42.3	13.2	40.9
	RL	None	Heuristic	65.4	54.8	43.7	7.3	42.8
	RL	RL	RL	66.6	61.1	50.9	14.8	48.3
DETIC [27]	Heuristic	None	Heuristic	28.7	15.2	5.3	0.4	12.4
	Heuristic	RL	RL	29.4	13.2	5.8	0.5	12.2
	RL	None	Heuristic	21.9	11.5	6.0	0.6	10.0
	RL	RL	RL	21.7	10.2	6.2	0.4	9.6

- Fail rates of subtasks are multiplied
  - → Failure rates of Navigation, Visual model, action policy are accumulated
  - → Performance drop when use Open-vocabulary detector (Grount Truth vs DETIC)
  - → Manipulation (Pick, Place) is also matter

Not only the performance of each component itself, but also their combination is an important issue.

Challenges of OVMM

- Neurips 2023 HomeRobot OVMM challenge

HomeRobot OVMM Challenge Leaderboard (sorted by Success Rate)

Rank	Team	Success (Sim)	Success (Real)
1	UniTeam	0.108	0.5
2	Rulai	0.032	0.0
3	KuzHum (RL YOLO+DETIC (full tracker +)	0.024	0.0

- Even the first-ranked team recorded approximately 30% in average

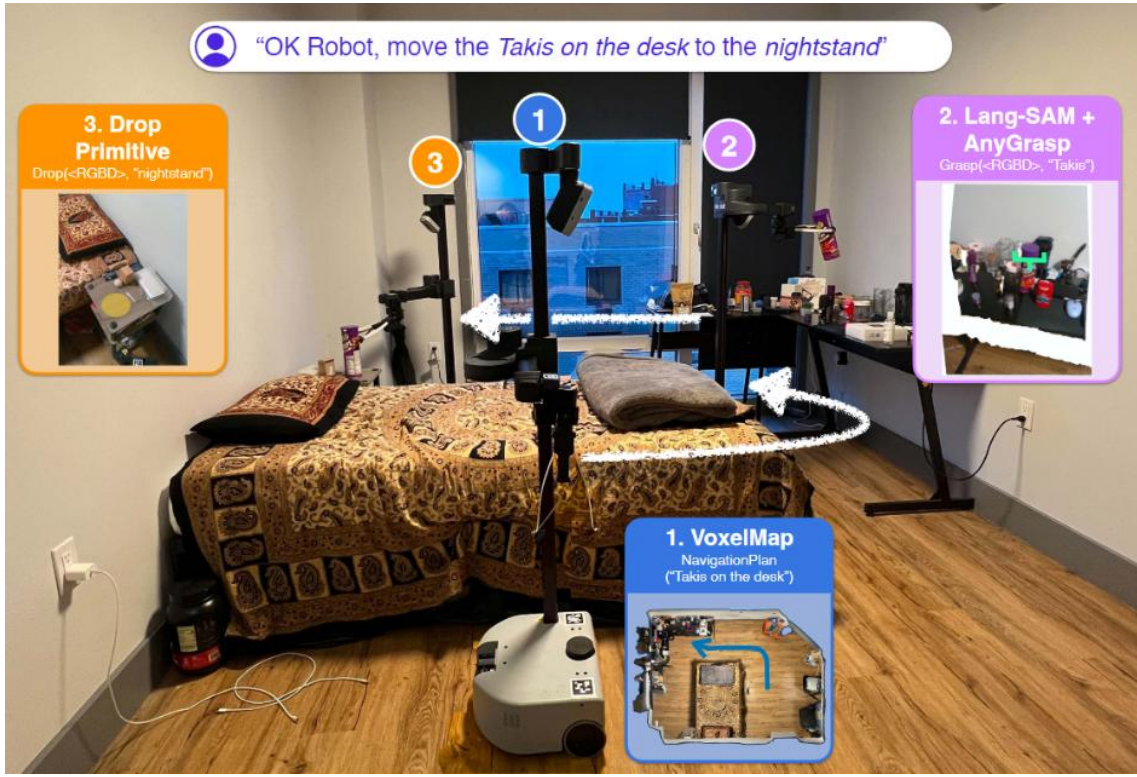
OK-Robot: A modular system for OVMM

- Recorded **58.5% Success rate** in normal house environment, and 82% in clear(uncluttered) environment on the real-world test
  → SOTA for OVMM task

- Utilized pre-trained models(CLIP, Lang-SAM, AnyGrasp)
  → No extra training!

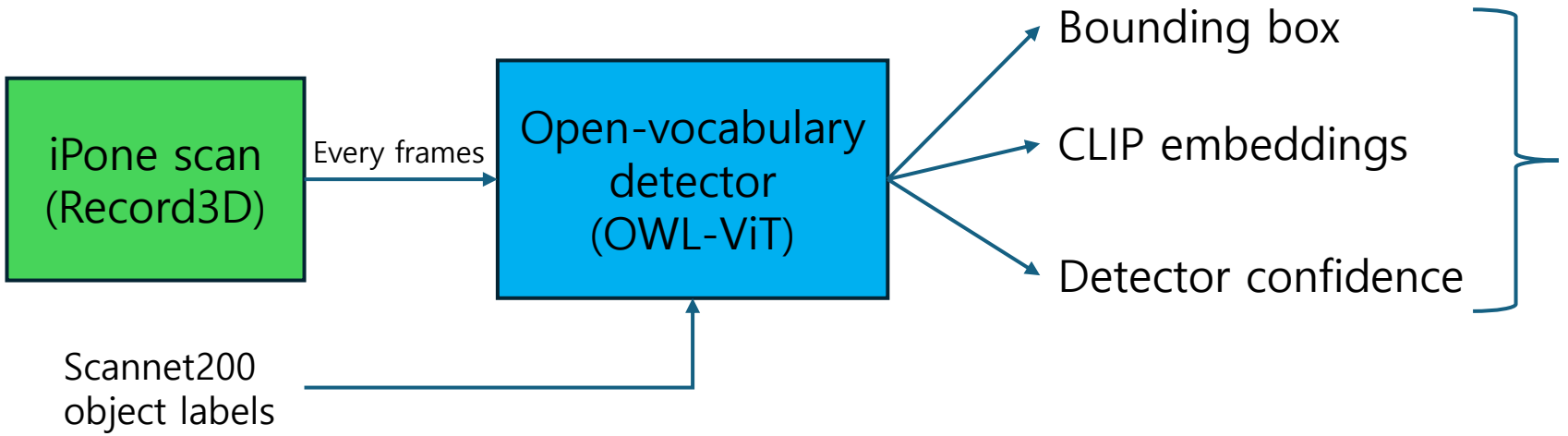
- Modular system
(**Navigation** module, **Grasping** module, **Dropping** heuristic)

- Good example for explaining how robotic system can be composed

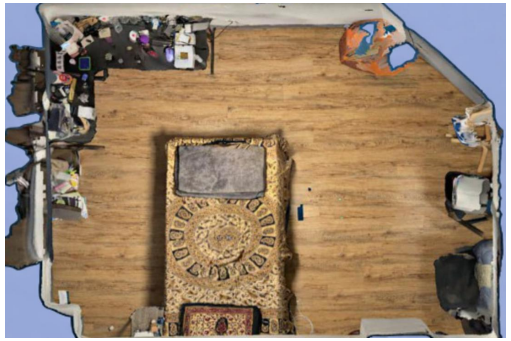


OK-Robot: 0. Setup

- **Manual scanning needed** (Record3D app with iPhone, taking video under 1 minute)
  → take sequence of posed RGB-D images



Voxel map



- 5cm resolution
- CLIP embeddings per voxel

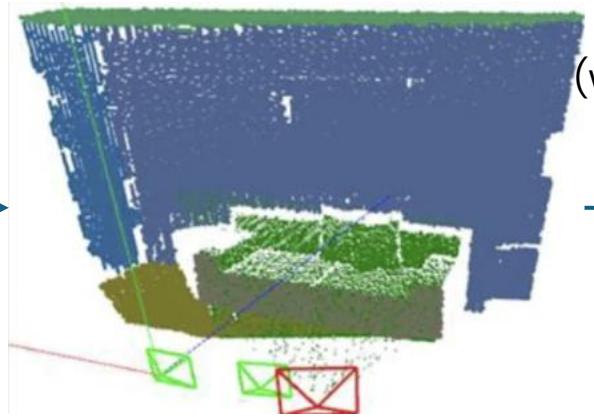
About 5 minutes

OK-Robot: 0. Setup

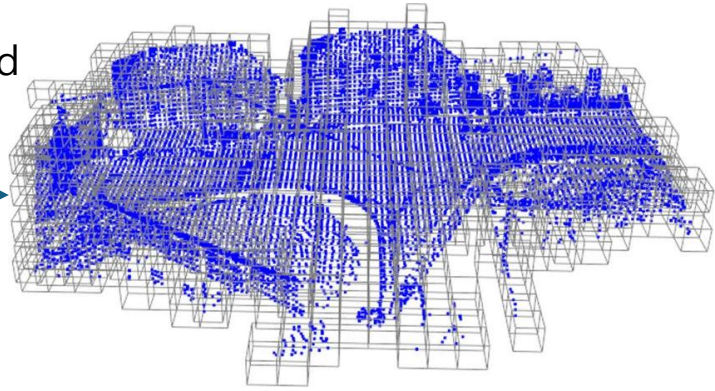
- From posed RGB-D images to Voxel map



Back-projection



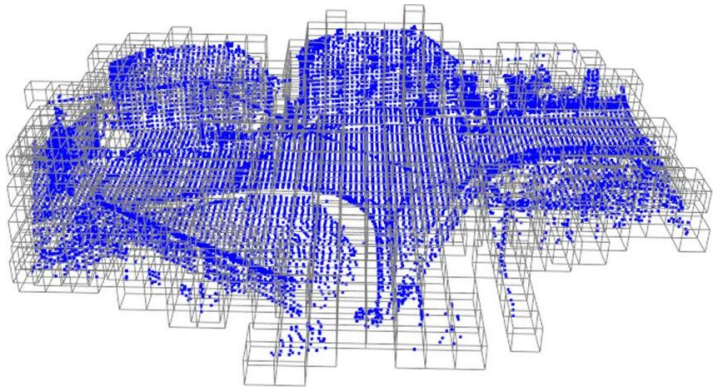
Voxelization
(w/ confidence-weighted average)



→ By voxelization, we can reduce computational cost and make the system more robust

OK-Robot: 0. Setup

- Querying Voxel map



- Eace voxels take semantic embeddings in CLIP space

- Query template:
  "Pick up A (from B) and drop it on/in C"

  - Locating A: use **Top-K method**

    1) Dot product voxel embeddings to A and B for each

    2) Top-10 for A, Top-50 for B

    3) Compute 10x50 L2 distance for (A, B) pairs
       and pick A with shortest distance

OK-Robot: 1. Navigation

- What needs to be considered during navigation:

1) Robot should be close enough to object
2) Robot needs enough space to move gripper for grasping object
3) Robot should not be caught on obstacles during Navigation & Manipulation

- Heuristic **navigation score** for locating **robot's position**
  → Computed on every voxels

$$s_1(\vec{x}) = ||\vec{x} - \vec{x_o}||$$

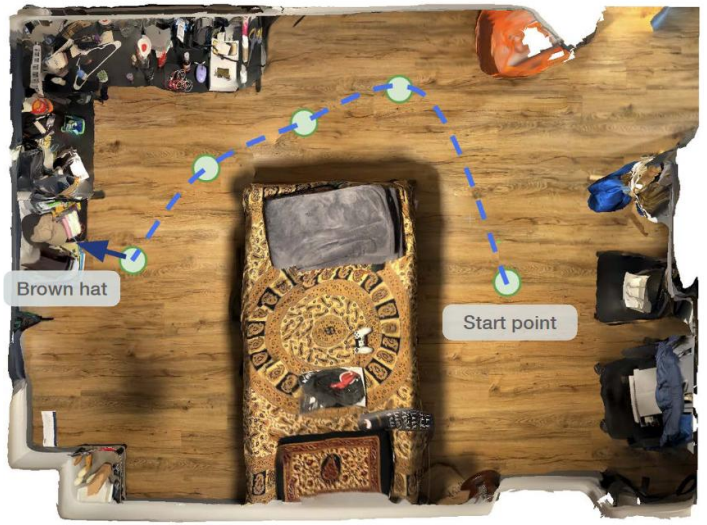
$$s_2(\vec{x}) = 40 - \min(||\vec{x} - \vec{x_o}||, 40)$$

$$s_3(\vec{x}) = \begin{cases} 1/||\vec{x} - \vec{x}_{obs}||, & \text{if } ||\vec{x} - \vec{x}_{obs}||_0 \leq 30 \\ 0, & \text{otherwise} \end{cases}$$

$$s(\vec{x}) = s_1(\vec{x}) + 8s_2(\vec{x}) + 8s_3(\vec{x})$$

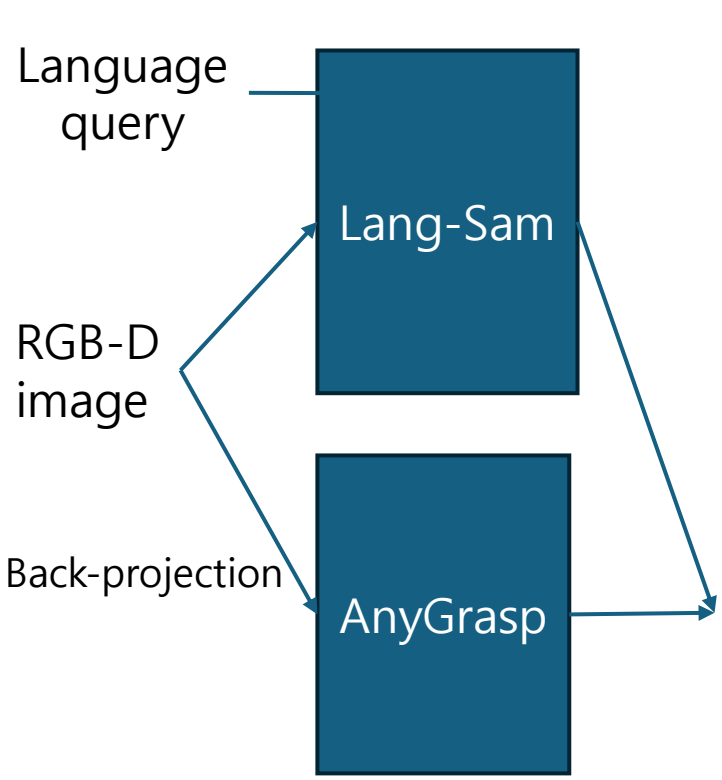
- **Navigating the path**

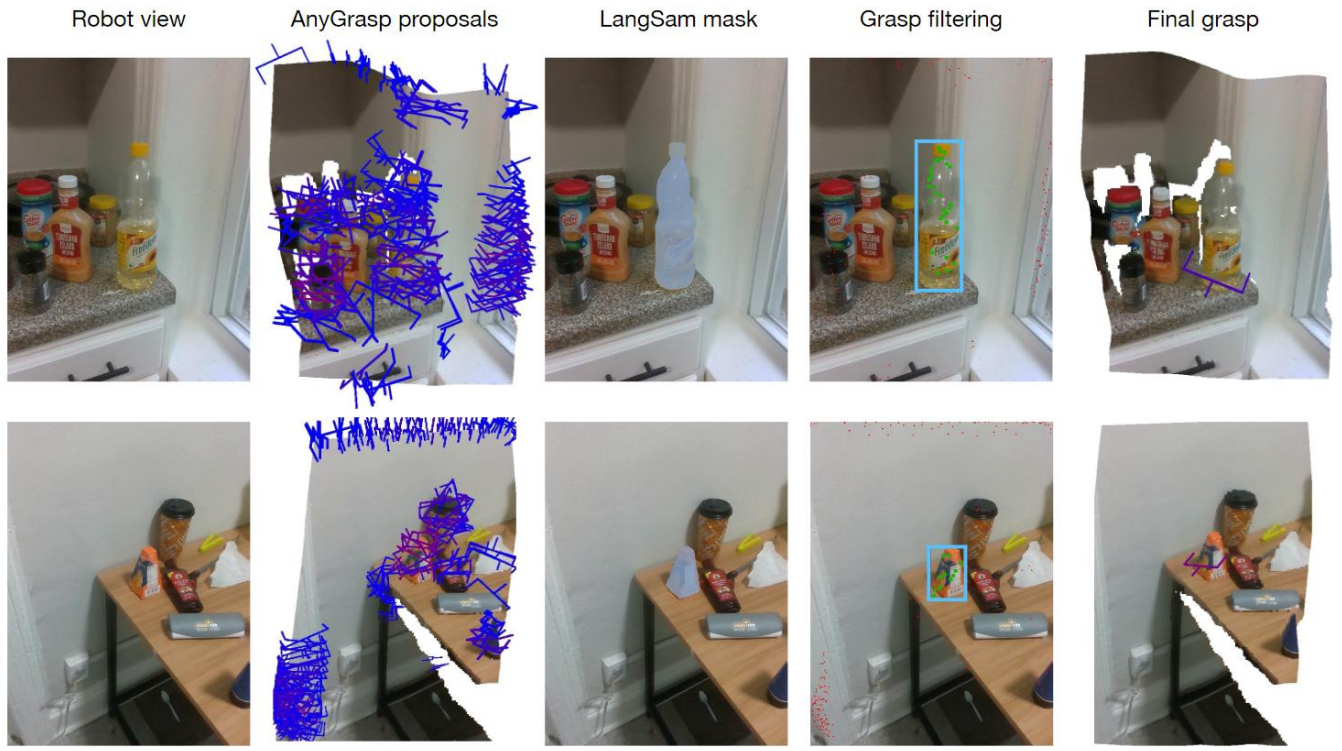
1. Make 2D 10cm x 10cm Obstacle map
2. Run A* algorithm by setting $s_3(\vec{x})$ as node cost



OK-Robot: 2. Grasping

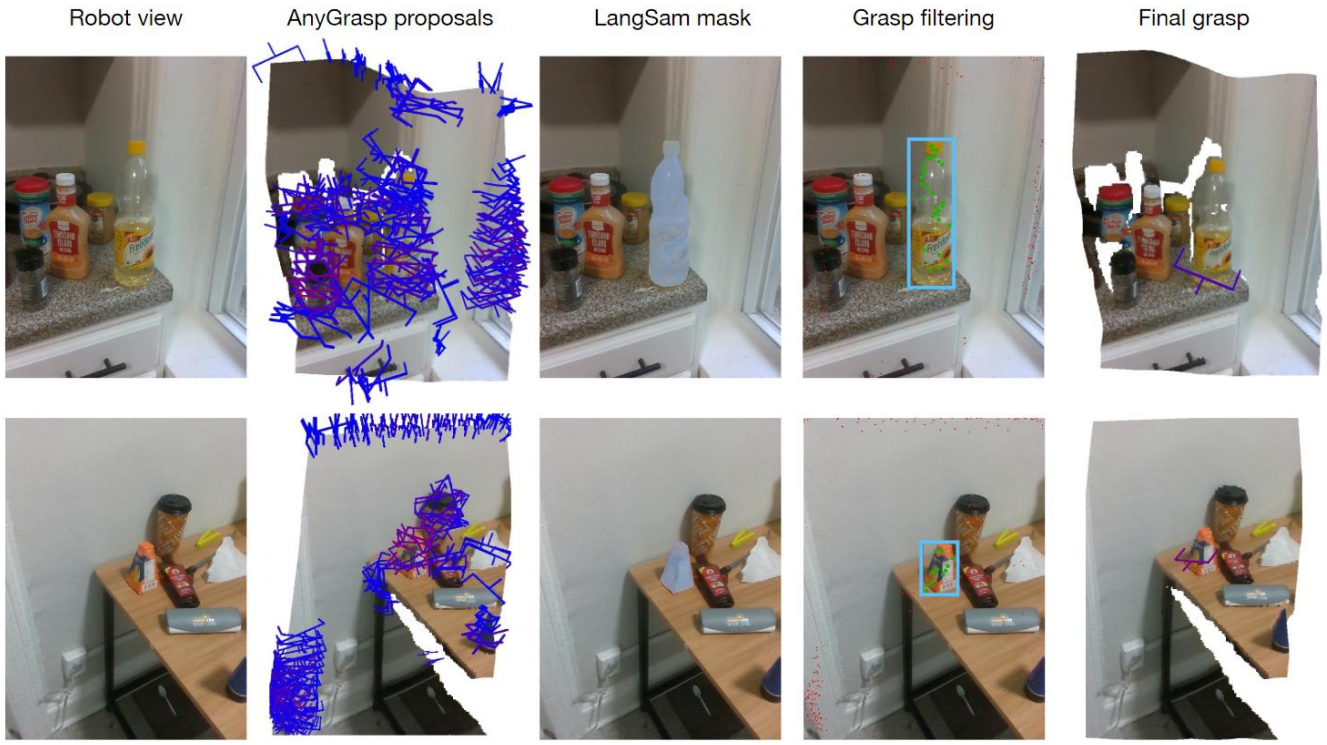
After the robot moves too in front of the object and taking RGB-D image from head camera,





→ Lang-Sam segments target object on the RGB image
→ AnyGrasp infers any graspable local region on point cloud scene, and outputs pose and grasp confidence

OK-Robot: 2. Grasping



- Final heuristic grasping score:

$$S - (\theta^4/10)$$

S: grasping confidence by AnyGrasp
θ: angle between grasp normal and floor normal

→ Prefer horizontal grasp

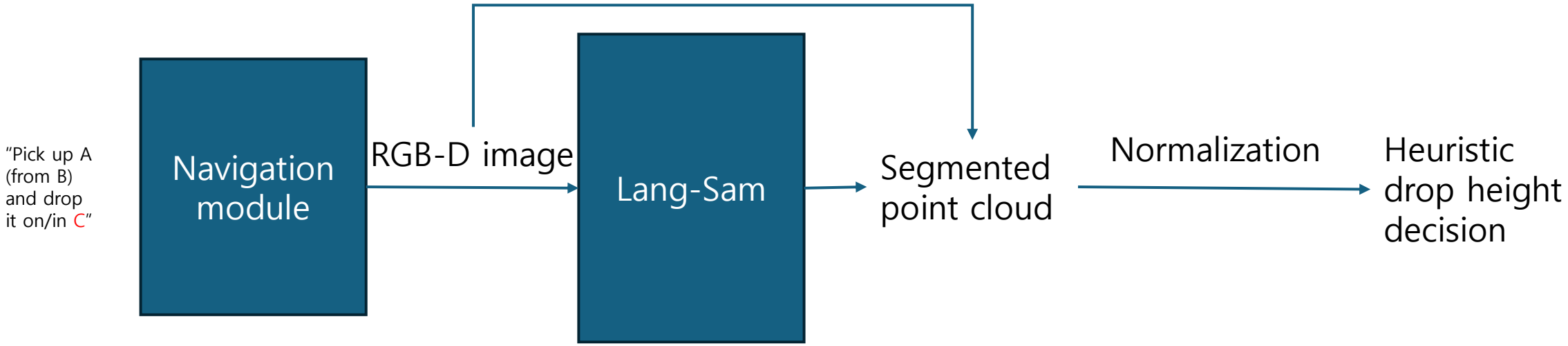
- Grasping execution trajectory

$$\langle \vec{p} - 0.2\vec{a},\ \vec{p} - 0.08\vec{a},\ \vec{p} - 0.04\vec{a},\ \vec{p} \rangle$$

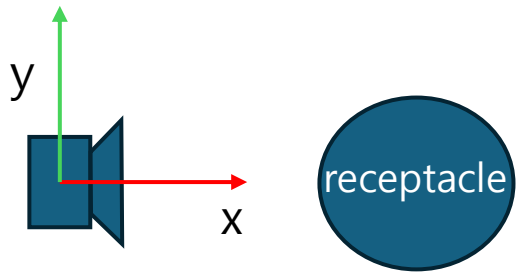
$\vec{p}$: grasp point
$\vec{a}$: approach vector by AnyGrasp

→ pre-defined, progressly smaller action

OK-Robot: 3. Drop



- Normalization



(0, 0, 0) at robot camera's x, y and floor z

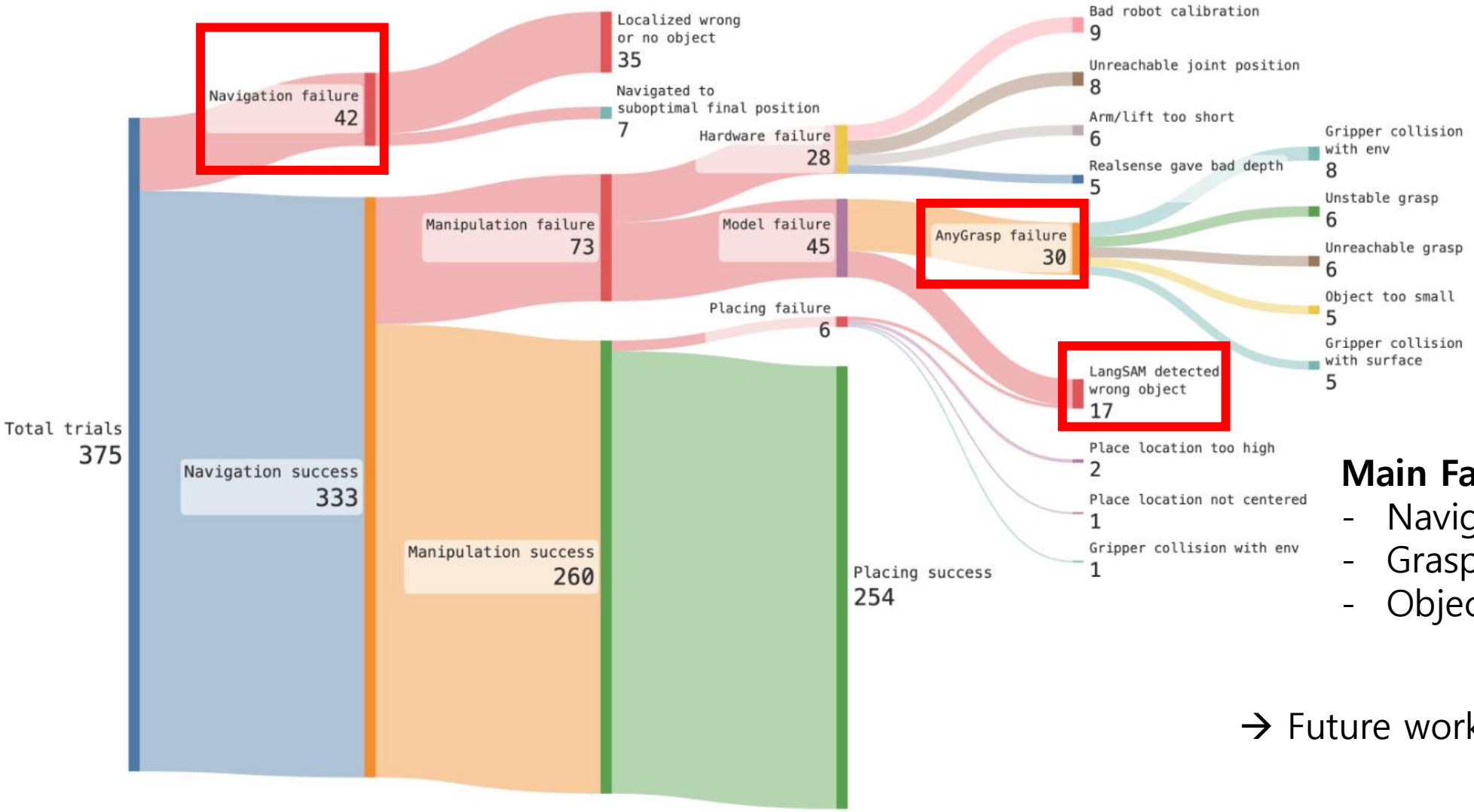
- Drop height decision

$$z_{max} = 0.2 + \max\{z \mid (x, y, z) \in P_a; 0 \leq x \leq x_m; |y - y_m| < 0.1\}$$

P_a: Normalized point cloud

x_m, y_m: median position of receptacle

OK-Robot: Evaluation - Failure cases

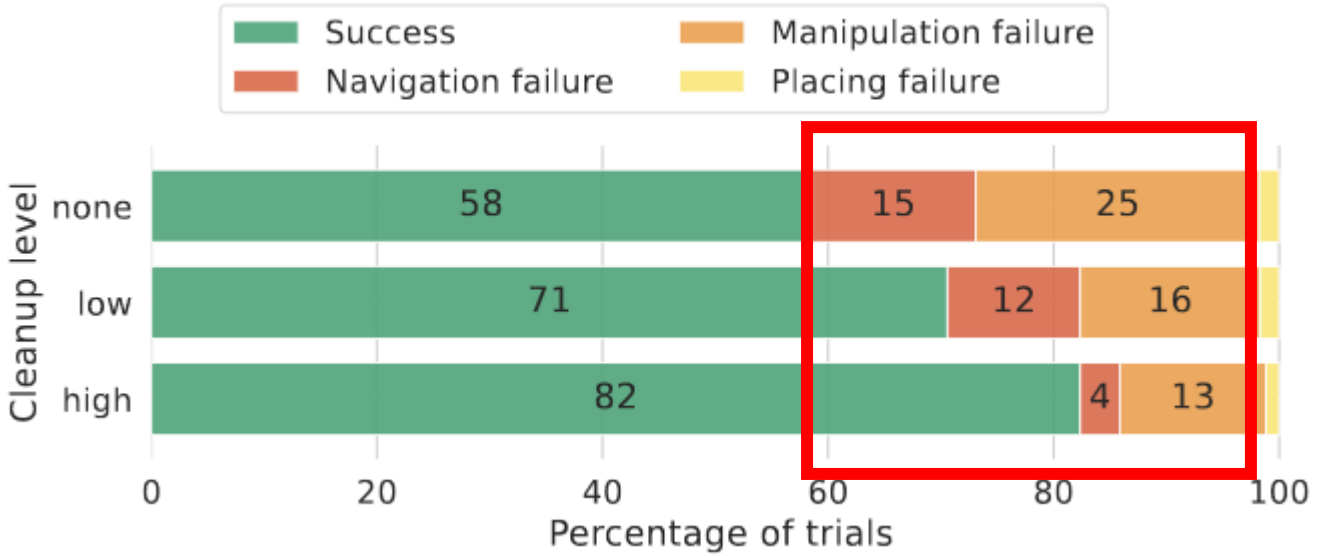


Main Failure:

- Navigation (or Voxel map)
- Grasp
- Object perception

→ Future works need to reduce them

OK-Robot: Evaluation – effect of clutter



→ Navigation failure rate significantly decreases, because well-scanned and obstacles located sparsely

→ Manipulation failure rate decreases as well, because no obstacles nearby during manipulation

→ Placing failure rate rarely decreases, because it is heuristic algorithm

OK-Robot: Discussion

- **Textual ambiguity**



❌ Orange soda can
✅ Metallic golden beverage can



❌ Grey eye glass box
✅ Grey eyeglass box



❌ Green zandu balm
✅ Green zandu balm container



❌ McDonalds french fries container
✅ Fast-food french fries container



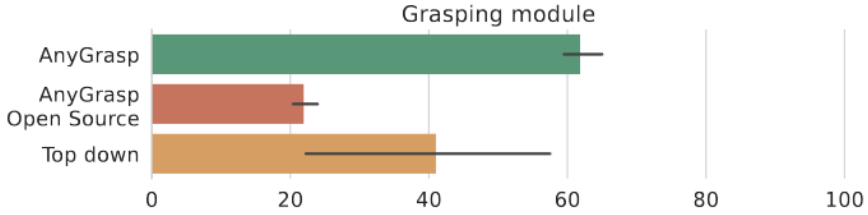
❌ Red insecticide
✅ Red spray on brown shelf



❌ Brown bandage roll
✅ Brown medical bandage

→ Similar semantic meanings,
but different results in navigation
(matching on Voxel map)

- **Limitation of grasping module**

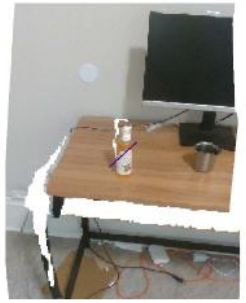




Object is transparent, so pointcloud is imperfect, so grasp is imperfect



Top-down grasp on tall counter is unreachable



Diagonal grasp on a cylindrical object is unstable

→ Low performance of grasp model itself, and other reasons coexist



Fine grasps on small objects are vulnerable to calibration errors



Grasps on round objects are unstable when not perfectly diametrical



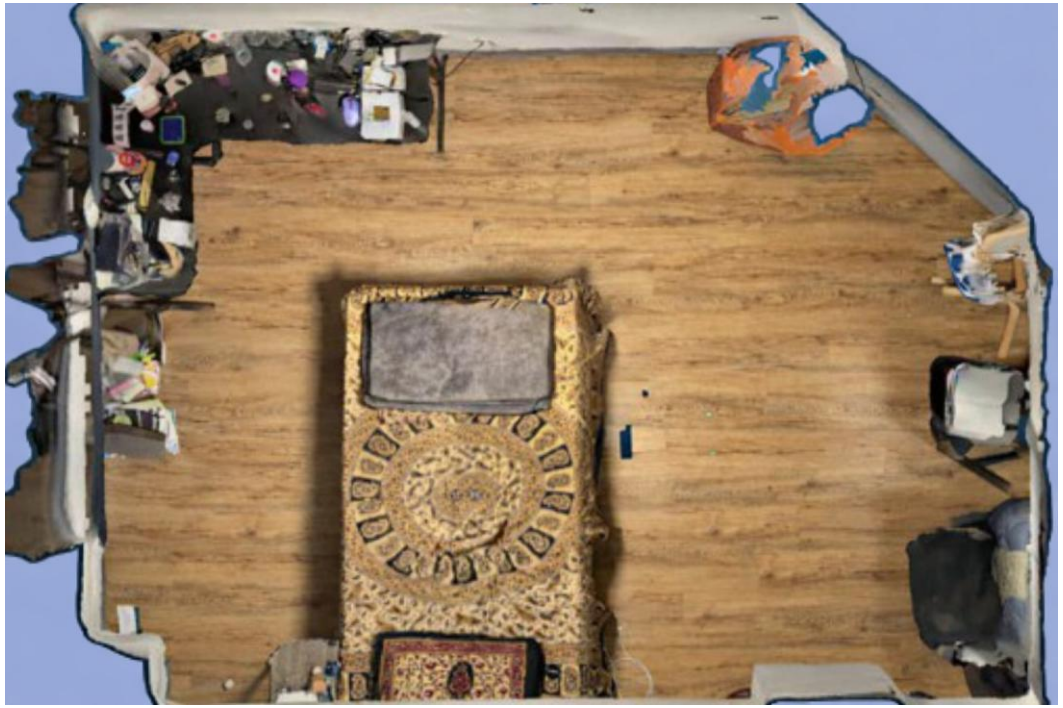
Grasps on flat objects collide with env if not perfectly executed

→ Especially, struggled with **flat objects** because of static grasp trajectory and two-fingered gripper

OK-Robot: Discussion

The biggest problem

- **Static map representation**



1) Scanning is noisy and not precise

2) Once scanned, and unable to change position of objects
   → User should scan again whenever something changed!

3) Semantic embeddings on Voxel map are limited to Scannet200 labels
   → Open-vocabulary, but embeddings are closed
   → Worsen textual ambiguity problem

Examples of Scannet200 labels: ['shower head', 'spray', 'inhaler', 'guitar case', 'plunger', 'range hood', 'toilet paper dispenser', 'adapter', 'soy sauce', 'pipe', 'bottle', 'door', 'scale', 'paper towel', 'paper towel roll', 'stove', 'mailbox', 'scissors', 'tape', 'bathroom stall', 'chopsticks', 'case of water bottles', 'hand sanitizer', 'laptop']

OK-Robot: Discussion

Problems if use OK-Robot as practical household robot:

In real house environment,

1) The objects' position changes frequently, sometime even missed

2) Actions after mistake/failure are not required, and need to try again (recover the mistake)
   → OK-Robot system is linear chain state machine, which is unable to stop during process

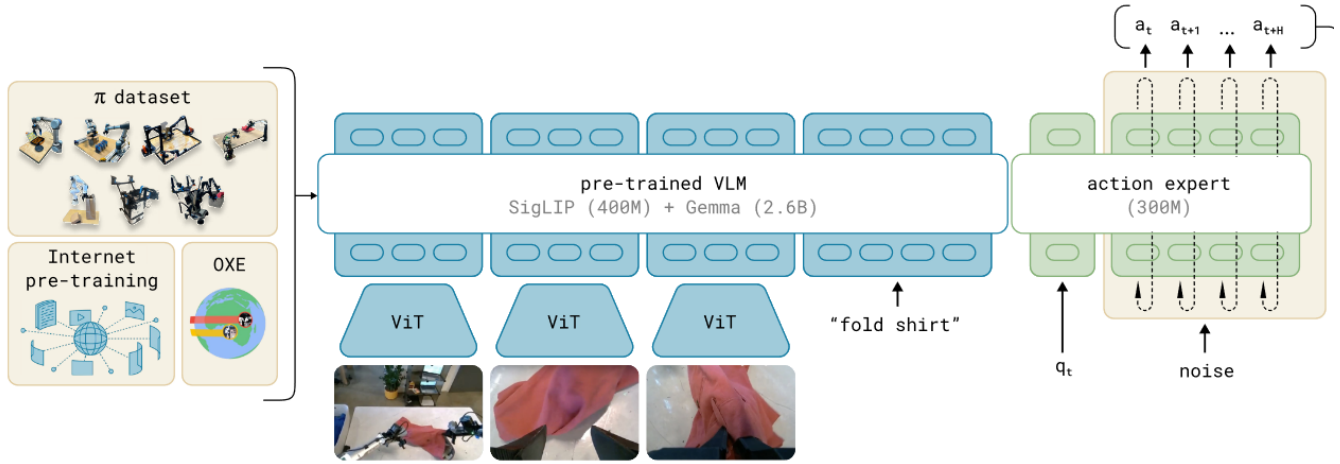
3) Similar objects in the same category coexist
   → Similar objects of parents and siblings can exist in the same room or house
   → OVMM is a task where no similar distractor exists, requiring only category-level perception
   → For personalization, instance-level perception is needed

Future Works

1) Static Map issue
→ Robot needs to move around on its own and detect object on real-time
→ Needs to be REAL Open-vocabulary (require precise visual-text embedding alignment)

2) Manipulation & Recovery issue
→ Employ end-to-end action expert structured VLAs such as $\pi_{0.6}$, GR00T…
→ Flexible grasp planning, and recover from the mistakes, try again

3) Similar distractors & personalization issue
→ Employ instance-level perception methods such as VAP



Main failure cases (Navigation, Manipulation, Perception) might be resolved!