

Use Cases of Visual Language Models (VLMs) in Robotics

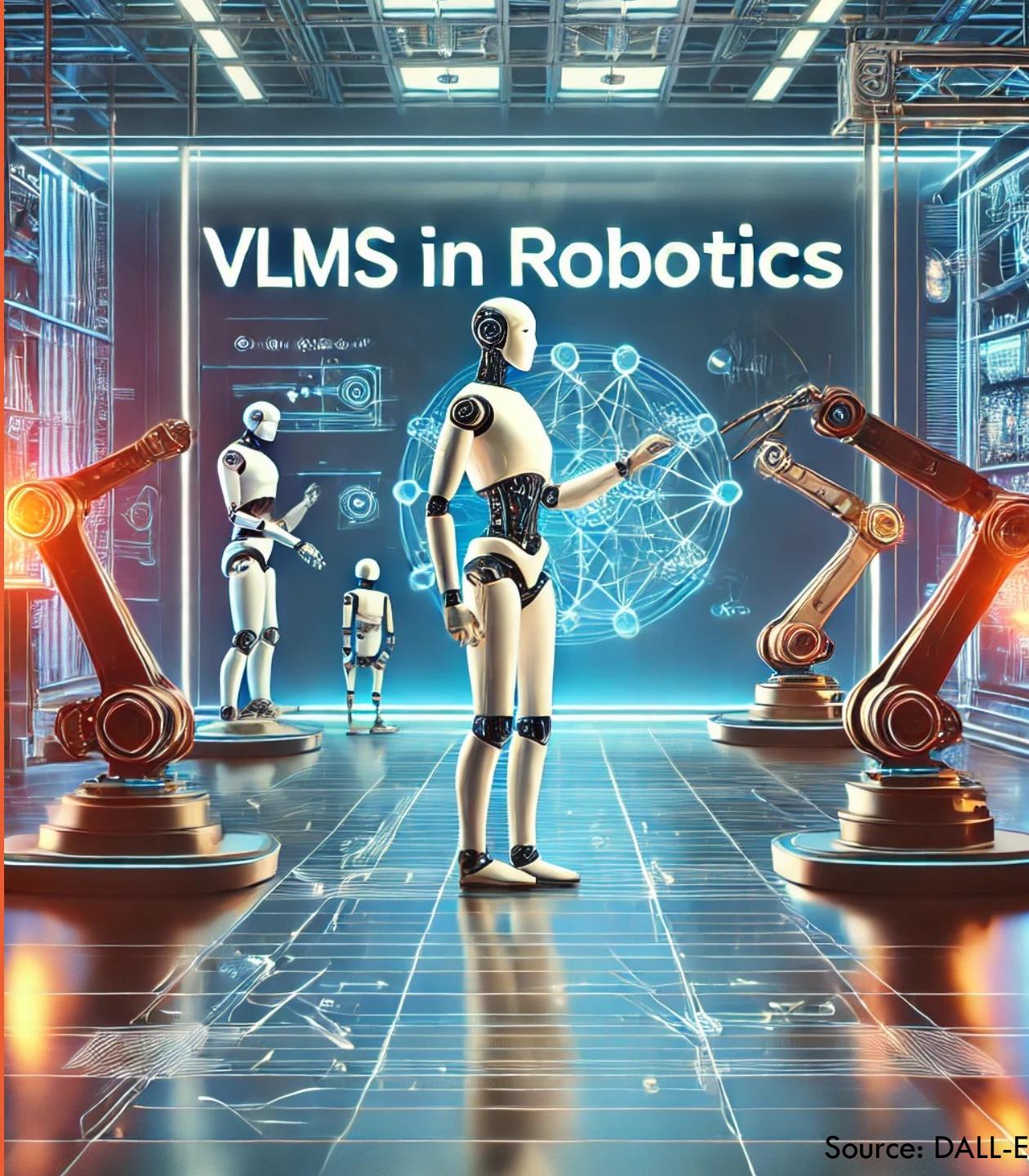
Zunzhi You

zyou5834@uni.sydney.edu.au

School of Computer Science



THE UNIVERSITY OF
SYDNEY



Source: DALL-E

Use Cases of VLMs in Robotics: Why and How

Why do we want to use VLMs in Robotics?

How can we use VLMs in Robotics?

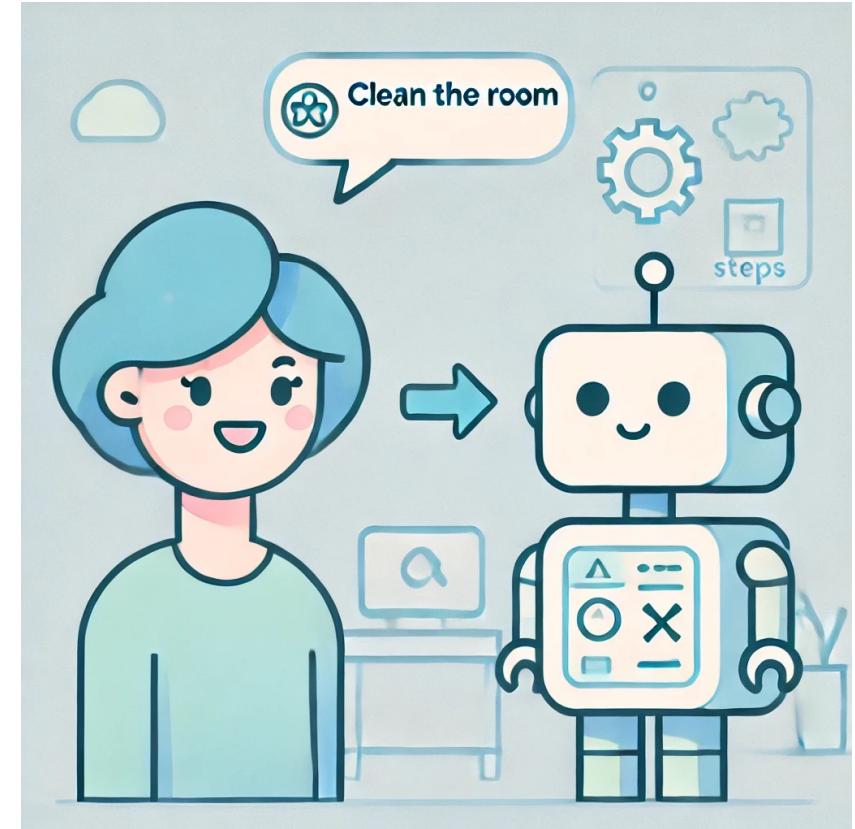
Any ideas?



THE UNIVERSITY OF
SYDNEY

Use Cases of Visual Language Models (VLMs) in Robotics

- High-level: VLMs for planning
 - PaLM-E
 - SpatialVLM
 - MANIPULATE-ANYTHING
- Mid-level: VLMs for structured output
 - RoboPoint
 - ReKep
- Low-level: VLMs for actions (VLA)
 - RT-2
 - RT-X
 - π_0



Source: DALL-E

PaLM-E: VLM for Planning

Mobile Manipulation



Human: Bring me the rice chips from the drawer. Robot: 1. Go to the drawers, 2. Open top drawer. I see 3. Pick the green rice chip bag from the drawer and place it on the counter.

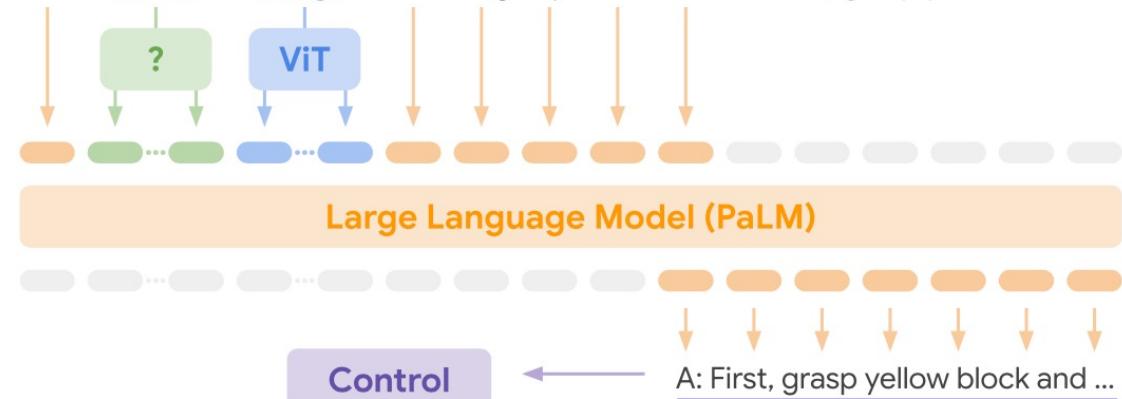
Visual Q&A, Captioning ...



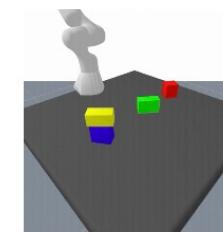
Given Q: What's in the image? Answer in emojis.
A: 🍎🍌🍇🍐🍑🥝🍒

PaLM-E: An Embodied Multimodal Language Model

Given **<emb>** ... **** Q: How to grasp blue block? A: First, grasp yellow block

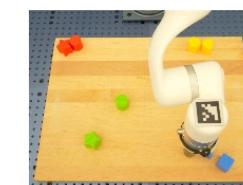


Task and Motion Planning



Given **<emb>** Q: How to grasp blue block?
A: First grasp yellow block and place it on the table, then grasp the blue block.

Tabletop Manipulation



Given Task: Sort colors into corners.
Step 1. Push the green star to the bottom left.
Step 2. Push the green circle to the green star.

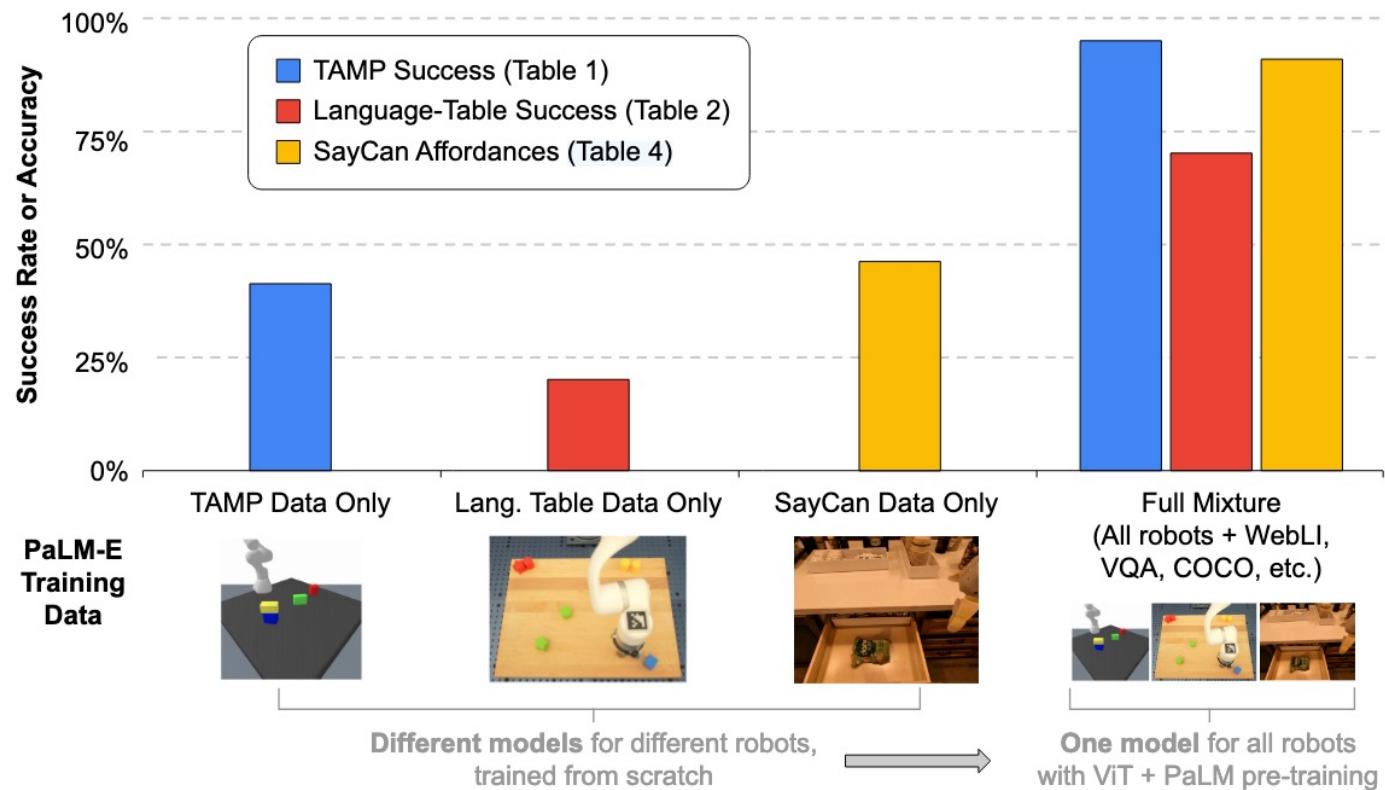
Language Only Tasks



Describe the following
A dog jumping over a hurdle at a dog show.

Q: Miami Beach borders which ocean? A: Atlantic. Q: What is 372×18 ? A: 6696. Q: Write a Haiku about embodied LLMs. A: Embodied language. Models learn to understand.
The world around them.

Large, diverse data boosts performance



Dataset in full mixture	Sampling frequency	%
Webli (Chen et al., 2022)	100	52.4
VQ ² A (Changpinyo et al., 2022)	25	13.1
VQG (Changpinyo et al., 2022)	10	5.2
CC3M (Sharma et al., 2018)	25	13.1
Object Aware (Piergiovanni et al., 2022)	10	5.2
OKVQA (Marino et al., 2019)	1	0.5
VQAv2 (Goyal et al., 2017)	1	0.5
COCO (Chen et al., 2015)	1	0.5
Wikipedia text	1	0.5
(robot) Mobile Manipulator, real	6	3.1
(robot) Language Table (Lynch et al., 2022), sim and real	8	4.2
(robot) TAMP, sim	3	1.6

SpatialIVLM: VLM for Reward Annotation

User



You are a cleaning robot that is 1 meter wide, now you are standing in a living room and see the image, you want to move from here to the door that leads to the backyard, do you think I can go through the path between the sofa and the table and chairs.



User



Which can is the closest to the yellow finger that is on the right most side of the picture, the blue, orange or the silver one?



GPT-4V*



As an AI, I'm unable to physically interact with environments, but I can provide some insights based on the image provided. ... From the image, we can make some visual estimations. Judging from the image, it appears that the sofa and the table are quite close to each other, and there may not be enough space to accommodate a 1-meter wide robot....

Spatial-VLM



[VLM Reasoning] How wide is the path between the sofa and the table and chairs?
[VLM Answer] 1.56m [Answer] Yes, the robot can go through the path between the sofa and the table and chairs since it is wider than the robot's width.

GPT-4V



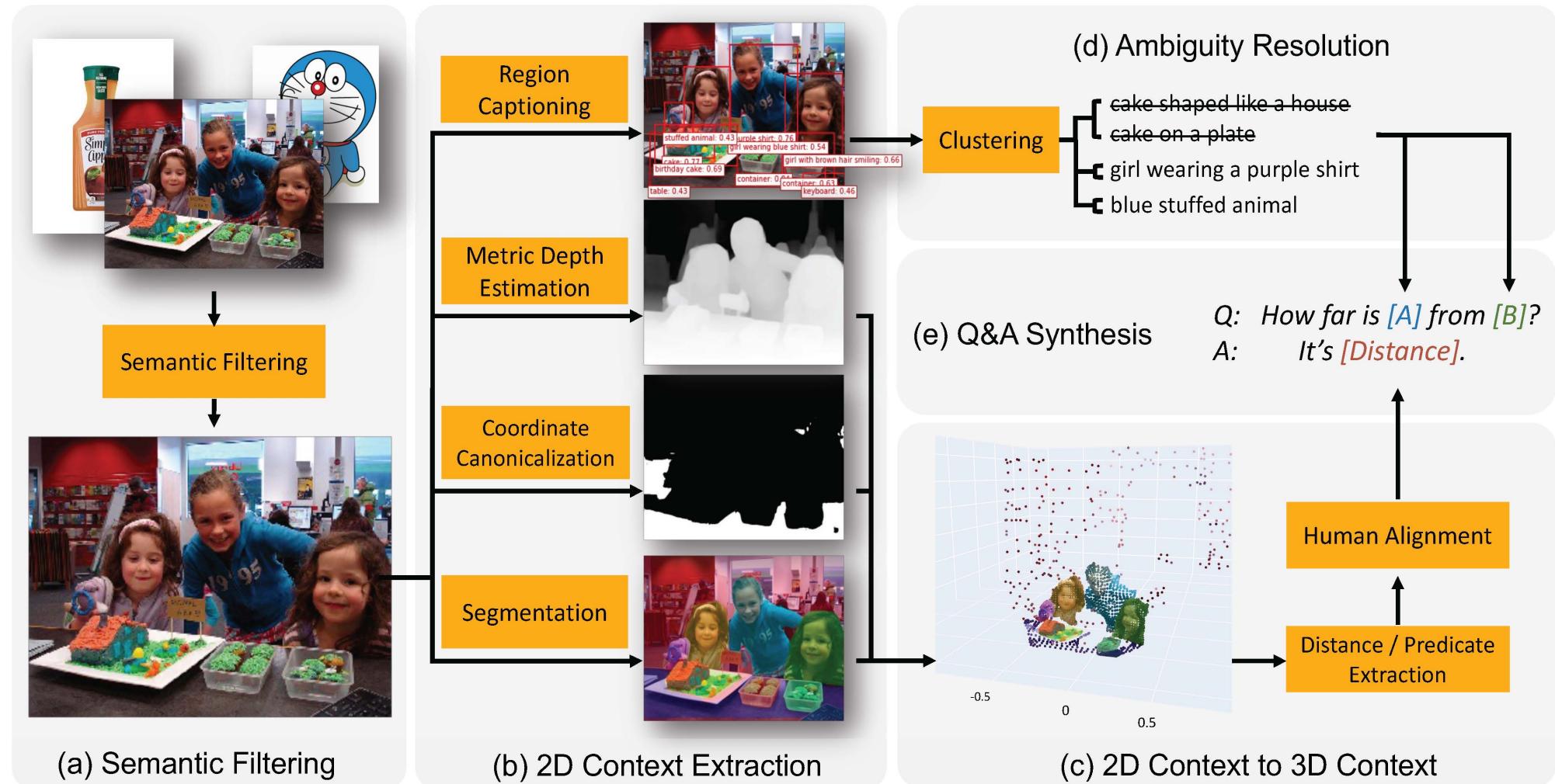
After estimating the distances by measuring the Euclidean distances between the yellow finger and each of the cans, it is clear that the silver can is the closest to the yellow finger with an approximate distance of 110 units. In comparison, the blue can is about 262 units away, and the orange can is approximately 233 units away from the yellow finger.

Spatial-VLM

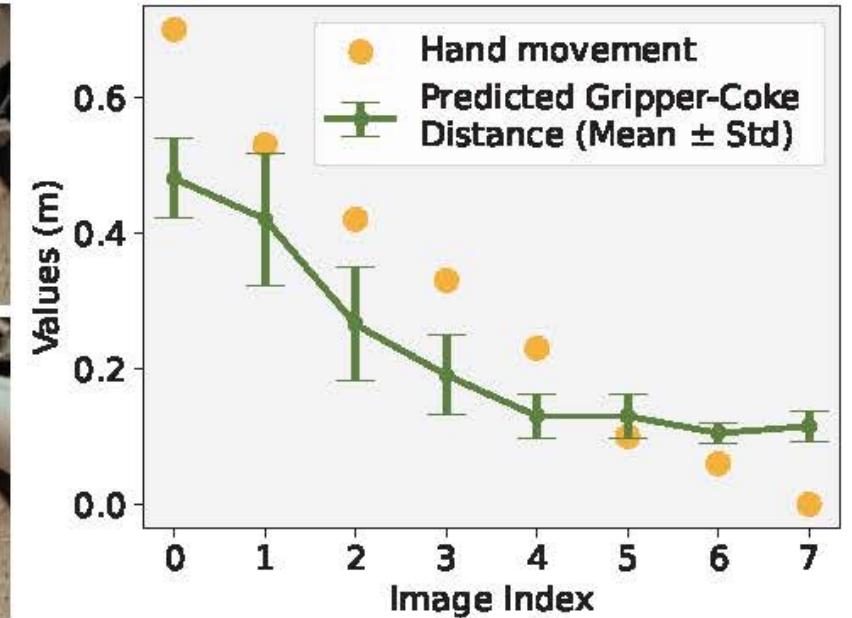
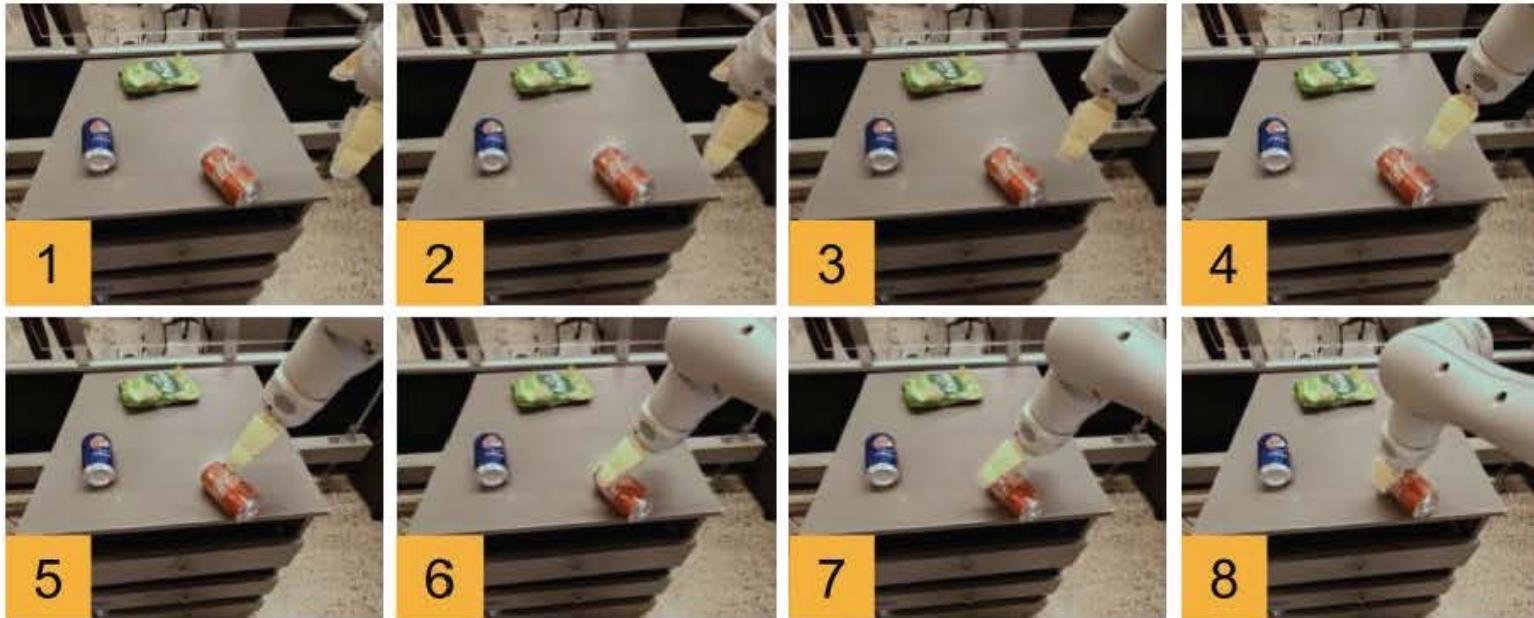


[VLM Reasoning] How far is the yellow finger from the blue can? [VLM Answer] 0.5m [VLM Reasoning] How far is the yellow finger from the orange can? [VLM Answer] 0.59m [VLM Reasoning] How far is the yellow finger from the silver can? [VLM Answer] 0.21m [Answer] The silver can is the closest to the yellow finger that is on the right most side of the picture.

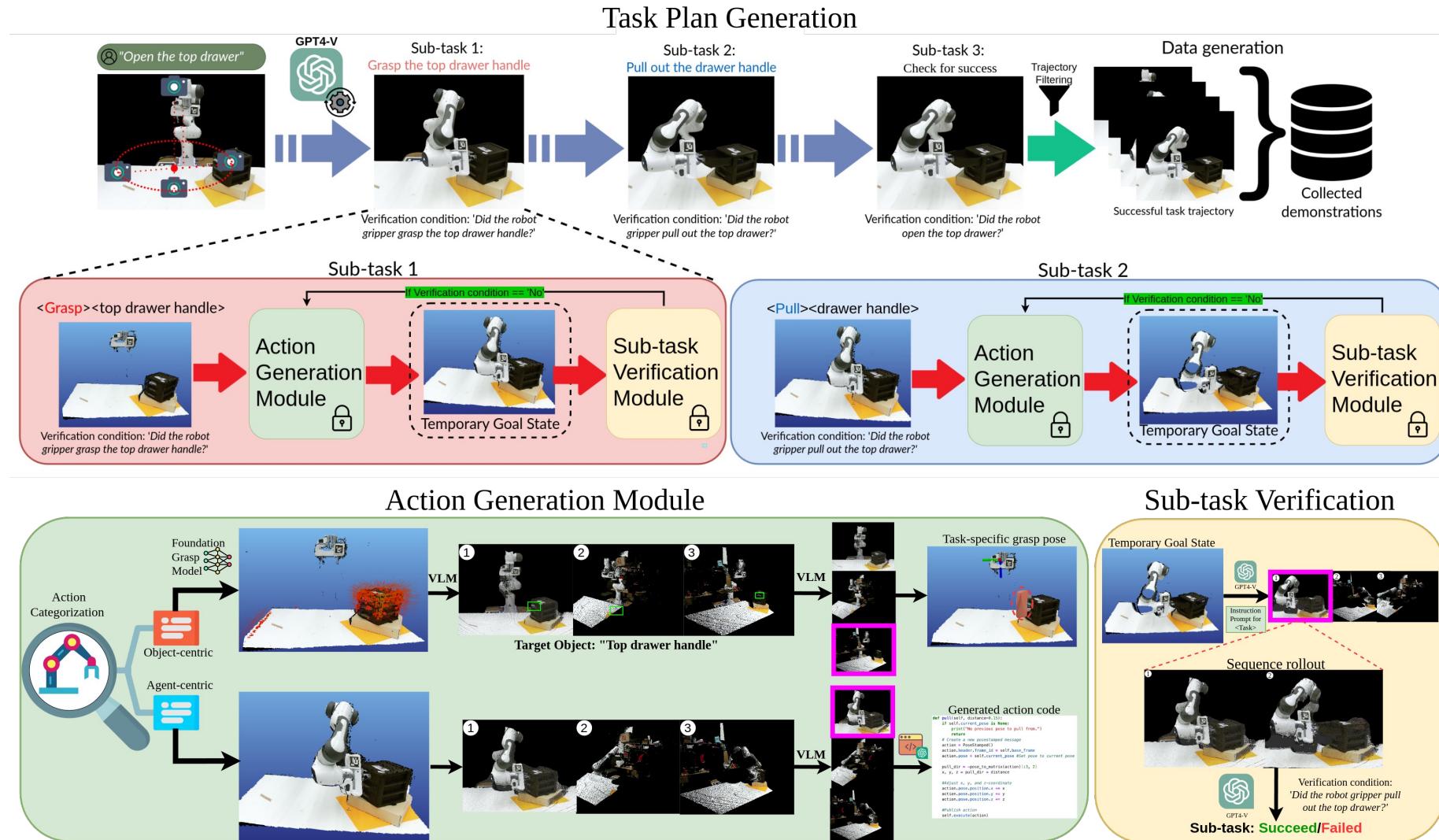
SpatialVLM: Automatic Data Synthesis



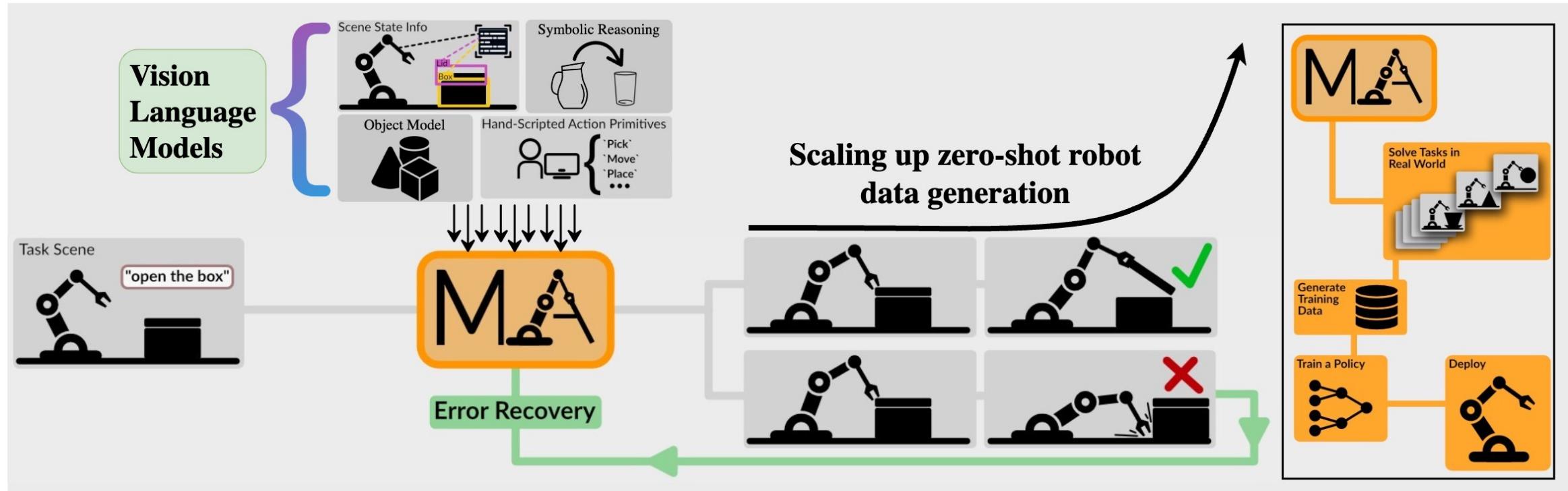
SpatialVLM: VLM for Reward Annotation



MANIPULATE-ANYTHING: VLM for Planning and Verification

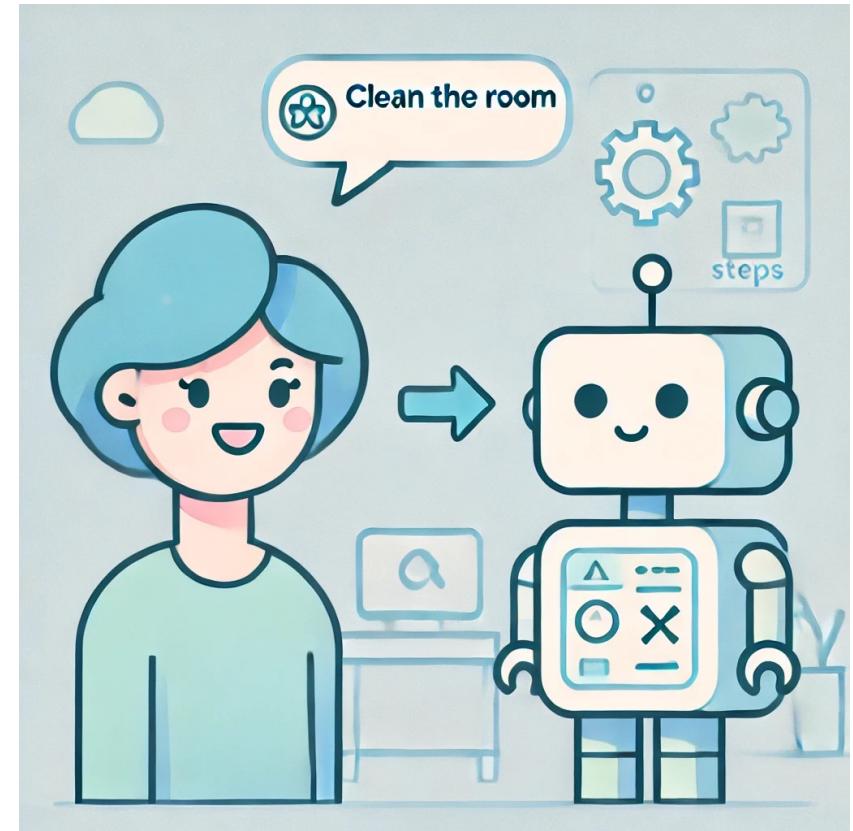


MANIPULATE-ANYTHING: VLM for Data Generation



Use Cases of Visual Language Models (VLMs) in Robotics

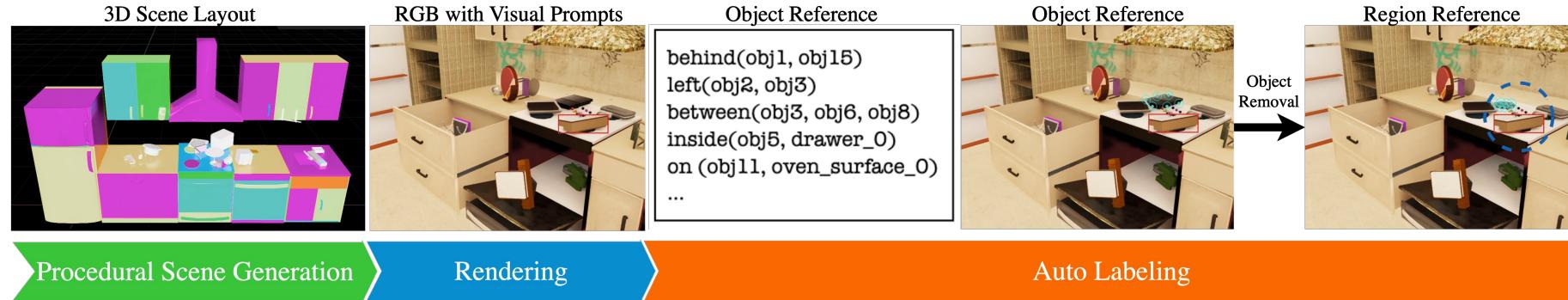
- High-level: VLMs for planning
 - PaLM-E
 - SpatialVLM
 - MANIPULATE-ANYTHING
- **Mid-level: VLMs for structured output**
 - RoboPoint
 - ReKep
- Low-level: VLMs for actions (VLA)
 - RT-2
 - RT-X
 - π_0



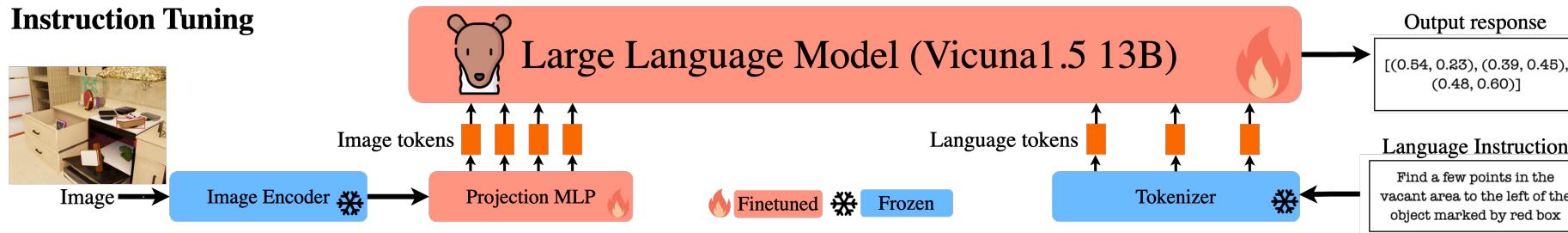
Source: DALL-E

RoboPoint: VLM for Action Points

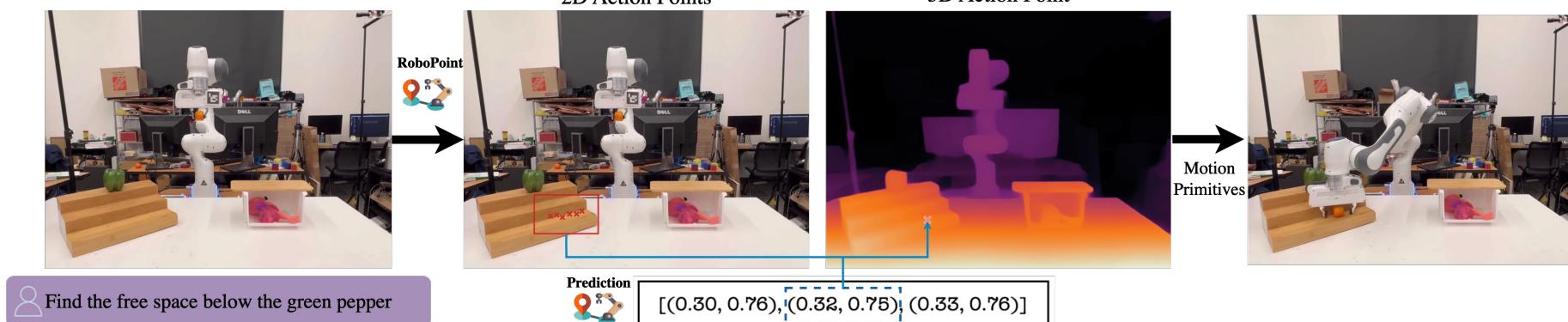
Data Generation



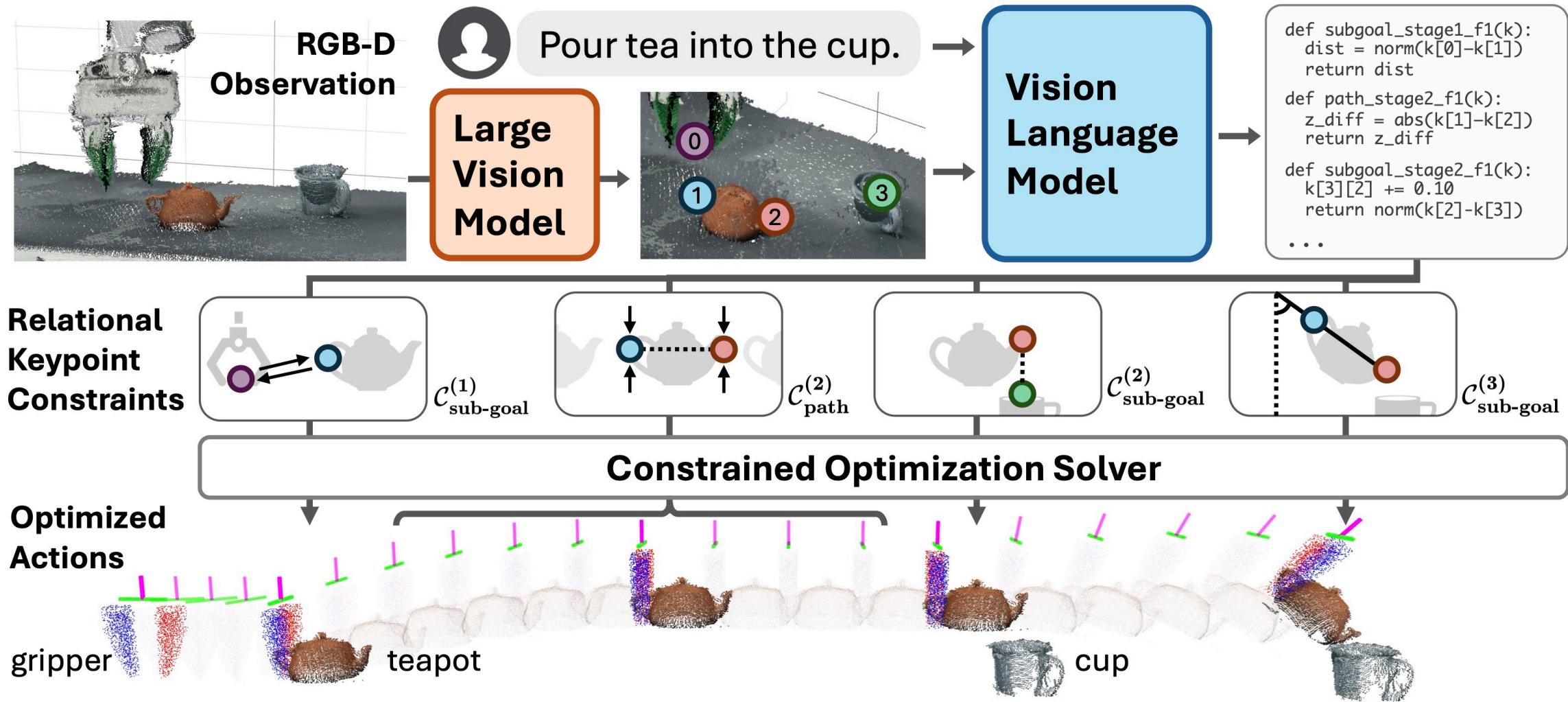
Instruction Tuning



Real-world Execution



ReKep: VLM for Keypoint Constraints



ReKep: VLM for Keypoint Constraints

Instructions

Suppose you are controlling a robot to perform manipulation tasks by writing constraint functions in Python.

The manipulation task is given as an image of the environment, overlayed with keypoints marked with their indices, along with a text instruction. The instruction starts with a parenthesis indicating whether the robot has a single arm or is bimanual. For each given task, please perform the following steps:

- Determine how many stages are involved in the task. Grasping must be an independent stage. Some examples:
 - "(single-arm) pouring tea from teapot":
 - 3 stages: "grasp teapot", "align teapot with cup opening", and "pour liquid"
 - "(single-arm) put red block on top of blue block":
 - 3 stages: "grasp red block", "align red block on top of blue block", and "release red block"
 - "(bimanual) fold sleeves to the center":
 - 2 stages: "left arm grasps left sleeve and right arm grasps right sleeve" and "both arms fold sleeves to the center"
 - "(bimanual) fold a jacket":
 - 3 stages: "left arm grasps left sleeve and right arm grasps right sleeve", "both arms fold sleeves to the center", and "grasp the neck with one arm (the other arm stays in place)", and "align the neck with the bottom"
- For each stage, write two kinds of constraints, "sub-goal constraints" and "path constraints". The "sub-goal constraints" are constraints that must be satisfied **at the end of the stage**, while the "path constraints" are constraints that must be satisfied **within the stage**. Some examples:
 - "(single-arm) pouring liquid from teapot":
 - "grasp teapot" stage:
 - sub-goal constraints: "align the end-effector with the teapot handle"
 - path constraints: None

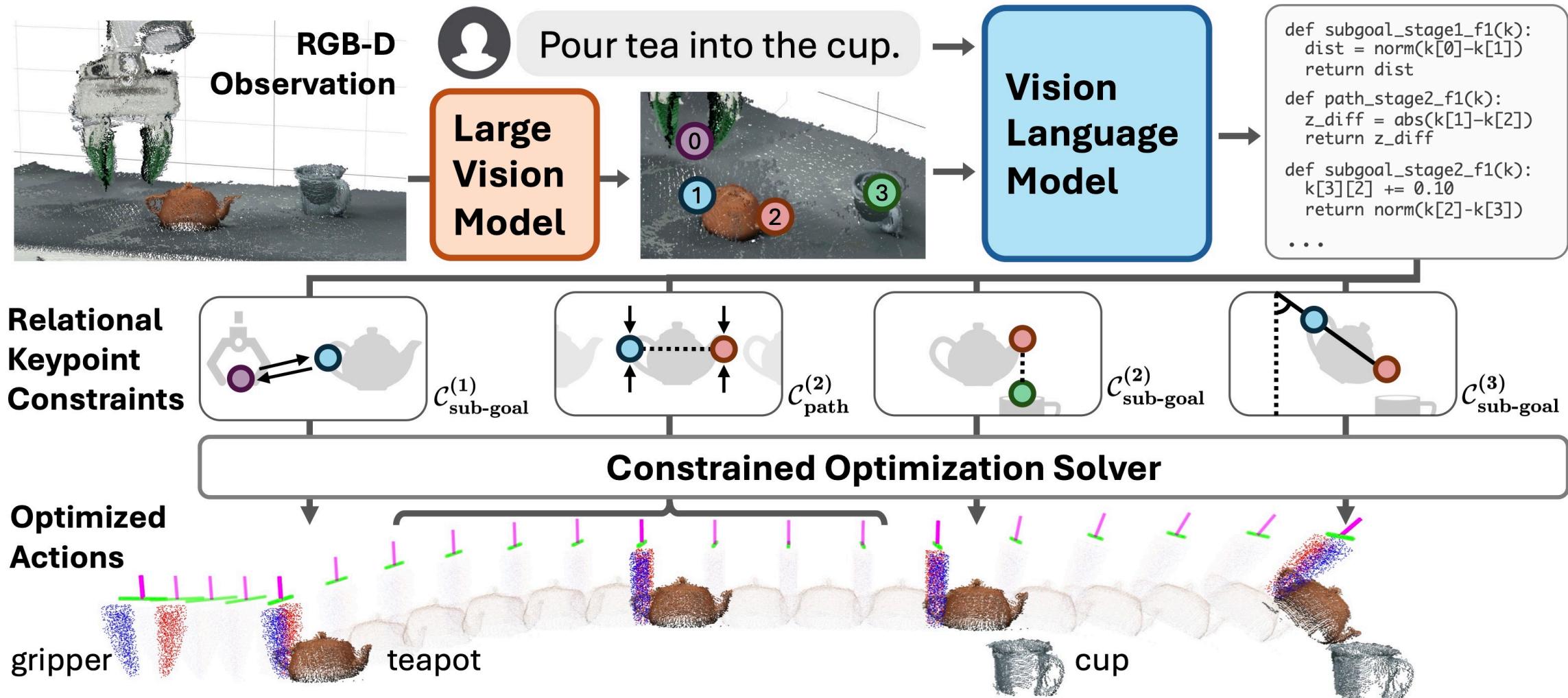
ReKep: VLM for Keypoint Constraints

- "align teapot with cup opening" stage:
 - sub-goal constraints: "the teapot spout needs to be 10cm above the cup opening"
 - path constraints: "robot is grasping the teapot", and "the teapot must stay upright to avoid spilling"
- "pour liquid" stage:
 - sub-goal constraints: "the teapot spout needs to be 5cm above the cup opening", "the teapot spout must be tilted to pour liquid"
 - path constraints: "the teapot spout is directly above the cup opening"
- "(bimanual) fold sleeves to the center":
 - "left arm grasps left sleeve and right arm grasps right sleeve" stage:
 - sub-goal constraints: "left arm grasps left sleeve", "right arm grasps right sleeve"
 - path constraints: None
 - "both arms fold sleeves to the center" stage:
 - sub-goal constraints: "left sleeve aligns with the center", "right sleeve aligns with the center"
 - path constraints: None

Note:

- Each constraint takes a dummy end-effector point and a set of keypoints as input and returns a numerical cost, where the constraint is satisfied if the cost is smaller than or equal to zero.
- For each stage, you may write 0 or more sub-goal constraints and 0 or more path constraints.
- Avoid using "if" statements in your constraints.
- Avoid using path constraints when manipulating deformable objects (e.g., clothing, towels).
- You do not need to consider collision avoidance. Focus on what is necessary to complete the task.
- Inputs to the constraints are as follows:
 - `end_effector`: np.array of shape `(3,)` representing the end-effector position.
 - `keypoints`: np.array of shape `(K, 3)` representing the keypoint positions.
- Inside of each function, you may use native Python functions and NumPy functions.
- For grasping stage, you should only write one sub-goal constraint that associates the end-effector with a keypoint. No path constraints are needed.
- For non-grasping stage, you should not refer to the end-effector position.
- In order to move a keypoint, its associated object must be grasped in one of the previous stages.
- The robot can only grasp one object at a time.
- Grasping must be an independent stage from other stages.

ReKep: VLM for Keypoint Constraints



ReKep: VLM for Keypoint Constraints

- Grasping must be an independent stage from other stages.
- You may use two keypoints to form a vector, which can be used to specify a rotation (by specifying the angle between the vector and a fixed axis).
- You may use multiple keypoints to specify a surface or volume.
- You may also use the center of multiple keypoints to specify a position.
- A single folding action should consist of two stages: one grasp and one place.

Structure your output in a single python code block as follows for single-arm robot:

```
'''python

# Your explanation of how many stages are involved in the task and what each stage is about.
# ...

num_stages = ?

### stage 1 sub-goal constraints (if any)
def stage1_subgoal_constraint1(end_effector, keypoints):
    """Put your explanation here."""
    ...
    return cost
# Add more sub-goal constraints if needed

### stage 1 path constraints (if any)
def stage1_path_constraint1(end_effector, keypoints):
    """Put your explanation here."""
    ...
    return cost
# Add more path constraints if needed

# repeat for more stages
...'''
```

ReKep: VLM for Keypoint Constraints

```
Structure your output in a single python code block as follows for bimanual robot:  
```python  

Your explanation of how many stages are involved in the task and what each stage is about.
...

num_stages = ?

left-arm stage 1 sub-goal constraints (if any)
def left_stage1_subgoal_constraint1(end_effector, keypoints):
 """Put your explanation here."""
 ...
 return cost

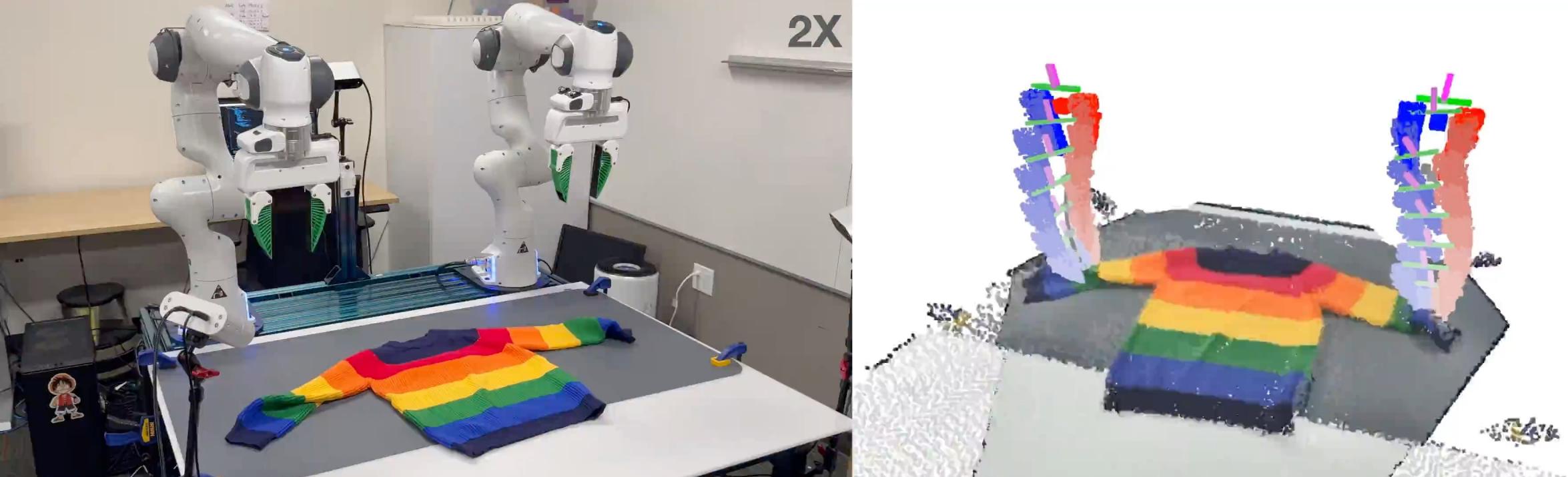
right-arm stage 1 sub-goal constraints (if any)
def right_stage1_subgoal_constraint1(end_effector, keypoints):
 """Put your explanation here."""
 ...
 return cost
Add more sub-goal constraints if needed

left stage 1 path constraints (if any)
def left_stage1_path_constraint1(end_effector, keypoints):
 """Put your explanation here."""
 ...
 return cost
right stage 1 path constraints (if any)
def right_stage1_path_constraint1(end_effector, keypoints):
 """Put your explanation here."""
 ...
 return cost
Add more path constraints if needed

repeat for more stages
...

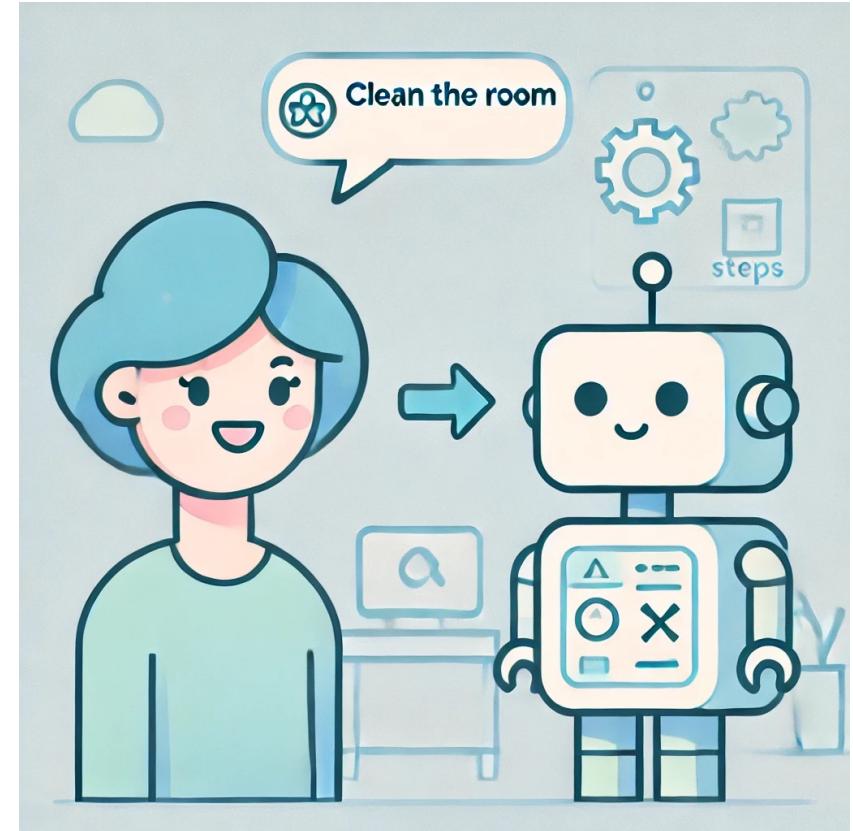
Query
Query Task: "[INSTRUCTION]"
Query Image: [IMAGE WITH KEYPOINTS]
```

# ReKep: VLM for Keypoint Constraints



# Use Cases of Visual Language Models (VLMs) in Robotics

- High-level: VLMs for planning
  - PaLM-E
  - SpatialVLM
  - MANIPULATE-ANYTHING
- Mid-level: VLMs for structured output
  - RoboPoint
  - ReKep
- Low-level: VLMs for actions (VLA)
  - RT-2
  - RT-X
  - $\pi_0$



Source: DALL-E

# RT-2: VLM for Actions (VLA)

Internet-Scale VQA + Robot Action Data

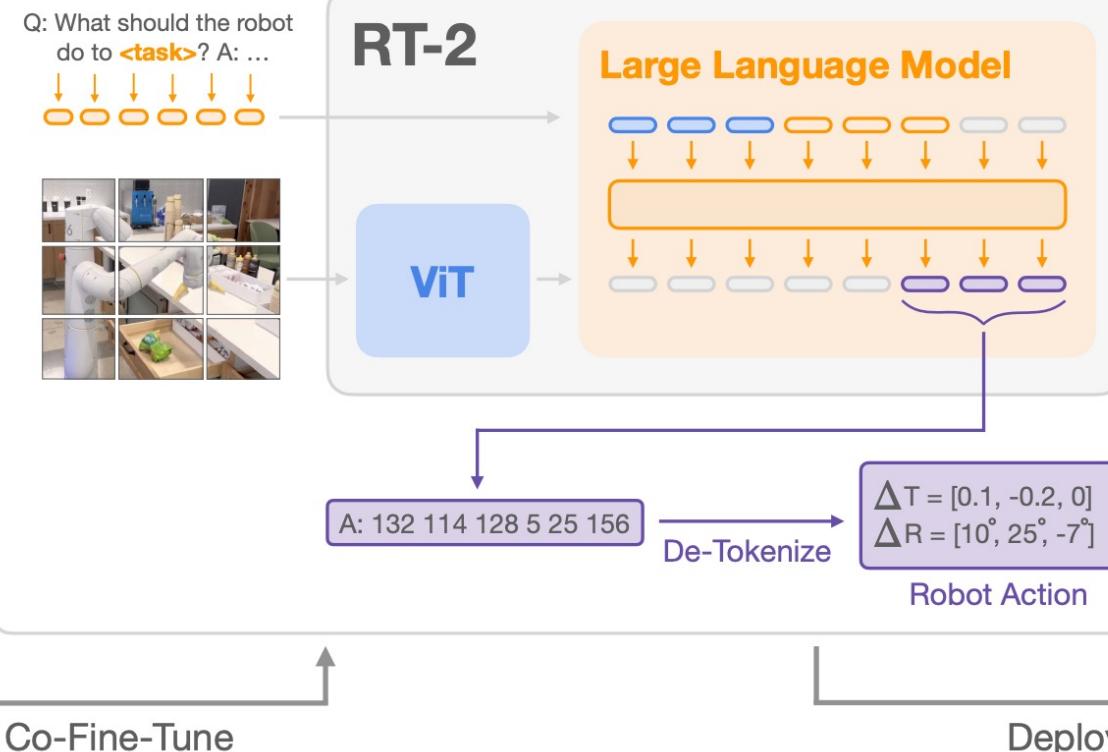


Q: What is happening in the image?  
A: 311 423 170 55 244  
A grey donkey walks down the street.

Q: Que puis-je faire avec ces objets?  
A: 3455 1144 189 25673  
Faire cuire un gâteau.

Q: What should the robot do to <task>?  
A: 132 114 128 5 25 156  
 $\Delta T = [0.1, -0.2, 0]$   
 $\Delta R = [10^\circ, 25^\circ, -7^\circ]$

Vision-Language-Action Models for Robot Control



Closed-Loop Robot Control



## RT-2: Action as a Language



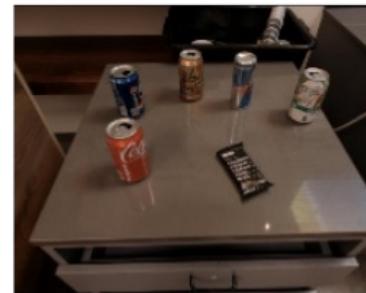
Prompt:  
Given <img> Instruction:  
Bring me a drink.  
Prediction:  
Plan: pick 7up can.  
Action: 1 143 129 123 145  
114 115 127



Prompt:  
Given <img> Instruction:  
Move all the objects together.  
Prediction:  
Plan: move green can near green rice chip bag.  
Action: 1 128 126 127 135  
123 119 127



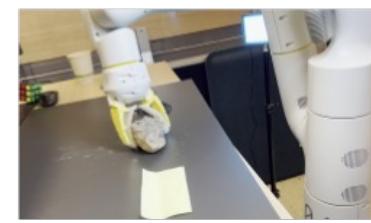
Prompt:  
Given <img> Instruction:  
Pick the object that is different from all other objects  
Prediction:  
Plan: pick rxbar chocolate. Action: 1 128  
129 125 131 125 128 127



Prompt:  
Given <img> Instruction:  
Move the green objects together.  
Prediction:  
Plan: move green can near green rice chip bag.  
Action: 1 130 129 121 131  
127 128 127



Prompt:  
Given <img> I need to hammer a nail, what object from the scene might be useful?  
Prediction:  
Rocks. Action: 1 129 138  
122 132 135 106 127

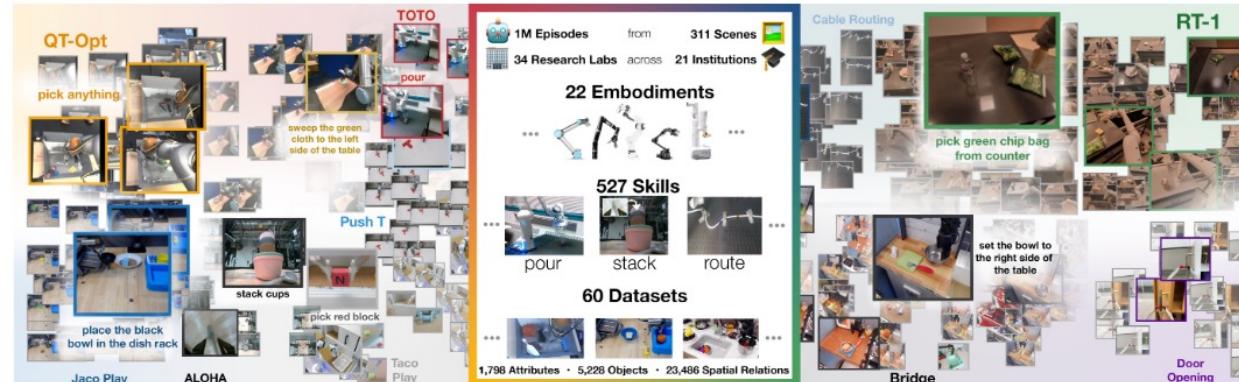


# Open X-Embodiment: Robotic Learning Datasets and RT-X Models

Open X-Embodiment Collaboration<sup>0</sup>

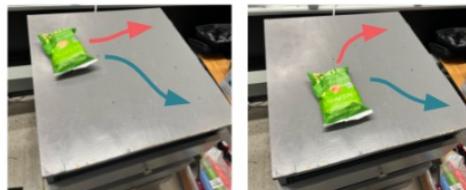
[robotics-transformer-x.github.io](https://robotics-transformer-x.github.io)

Abby O'Neill<sup>34</sup>, Abdul Rehman<sup>37</sup>, Abhinav Gupta<sup>4</sup>, Abhiram Maddukuri<sup>45</sup>, Abhishek Gupta<sup>46</sup>, Abhishek Padalkar<sup>10</sup>, Abraham Lee<sup>34</sup>, Acorn Pooley<sup>11</sup>, Agrim Gupta<sup>28</sup>, Ajay Mandelkar<sup>22</sup>, Ajinkya Jain<sup>15</sup>, Albert Tung<sup>28</sup>, Alex Bewley<sup>11</sup>, Alex Herzog<sup>11</sup>, Alex Irpan<sup>11</sup>, Alexander Khazatsky<sup>28</sup>, Anant Rai<sup>23</sup>, Anchit Gupta<sup>19</sup>, Andrew Wang<sup>34</sup>, Andrey Kolobov<sup>20</sup>, Anikait Singh<sup>11,34</sup>, Animesh Garg<sup>9</sup>, Aniruddha Kembhavi<sup>1</sup>, Annie Xie<sup>28</sup>, Anthony Brohan<sup>11</sup>, Antonin Raffin<sup>10</sup>, Archit Sharma<sup>28</sup>, Arefeh Yavary<sup>35</sup>, Arhan Jain<sup>46</sup>, Ashwin Balakrishna<sup>32</sup>, Ayzaan Wahid<sup>11</sup>, Ben Burgess-Limerick<sup>25</sup>, Beomjoon Kim<sup>17</sup>, Bernhard Schölkopf<sup>18</sup>, Blake Wulfe<sup>32</sup>, Brian Ichter<sup>11</sup>, Cewu Lu<sup>27,8</sup>, Charles Xu<sup>34</sup>, Charlotte Le<sup>34</sup>, Chelsea Finn<sup>11,28</sup>, Chen Wang<sup>28</sup>, Chengfen Xu<sup>34</sup>, Cheng Chi<sup>5,28</sup>, Chenguang Huang<sup>38</sup>, Christine Chan<sup>11</sup>, Christopher Agia<sup>28</sup>, Chuer Pan<sup>28</sup>, Chuyuan Fu<sup>11</sup>, Coline Devin<sup>11</sup>, Danfei Xu<sup>9</sup>, Daniel Morton<sup>28</sup>, Danny Driess<sup>11</sup>, Daphne Chen<sup>46</sup>, Deepak Pathak<sup>4</sup>, Dhruv Shah<sup>34</sup>, Dieter Bühler<sup>18</sup>, Dinesh Jayaraman<sup>42</sup>, Dmitry Kalashnikov<sup>11</sup>, Dorsa Sadigh<sup>11</sup>, Edward Johns<sup>14</sup>, Ethan Foster<sup>28</sup>, Fangchen Liu<sup>34</sup>, Federico Ceola<sup>16</sup>, Fei Xia<sup>11</sup>, Feiyu Zhao<sup>13</sup>, Felipe Vieira Frumeri<sup>20</sup>, Freely Stulp<sup>10</sup>, Gaoyue Zhou<sup>23</sup>, Gaurav S. Sukhatme<sup>43</sup>, Gautam Salhotra<sup>43,15</sup>, Ge Yan<sup>36</sup>, Gilbert Feng<sup>34</sup>, Giulio Schiavi<sup>7</sup>, Glen Berseth<sup>41,21</sup>, Gregory Kahn<sup>34</sup>, Guangwen Yang<sup>33</sup>, Guanzhi Wang<sup>3,22</sup>, Hao Su<sup>36</sup>, Hao-Shu Fang<sup>27</sup>, Haochen Shi<sup>28</sup>, Henghui Bao<sup>43</sup>, Heni Ben Amor<sup>2</sup>, Henrik I Christensen<sup>36</sup>, Hiroki Furuta<sup>31</sup>, Homanga Bharadhwaj<sup>4,19</sup>, Homer Walke<sup>34</sup>, Hongjia Fang<sup>27</sup>, Huy Ha<sup>5,28</sup>, Igor Mordatch<sup>11</sup>, Ilija Radosavovic<sup>34</sup>, Isabel Leal<sup>11</sup>, Jacky Liang<sup>11</sup>, Jad Abou-Chakra<sup>25</sup>, Jachyung Kim<sup>17</sup>, Jaimyn Drake<sup>34</sup>, Jan Peters<sup>29</sup>, Jan Schneider<sup>18</sup>, Jasmine Hsu<sup>11</sup>, Jay Vakil<sup>19</sup>, Jeannette Bohg<sup>28</sup>, Jeffrey Bingham<sup>11</sup>, Jeffrey Wu<sup>34</sup>, Jensen Gao<sup>28</sup>, Jiaheng Hu<sup>30</sup>, Jiajun Wu<sup>28</sup>, Jialin Wu<sup>12</sup>, Jiankai Sun<sup>28</sup>, Jianlan Luo<sup>34</sup>, Jiayuan Gu<sup>36</sup>, Jie Tan<sup>11</sup>, Jihoon Oh<sup>31</sup>, Jimmy Wu<sup>24</sup>, Jingpei Lu<sup>36</sup>, Jingyun Yang<sup>28</sup>, Jitendra Malik<sup>34</sup>, João Silvério<sup>10</sup>, Joey Hejna<sup>28</sup>, Jonathan Booher<sup>28</sup>, Jonathan Tompson<sup>11</sup>, Jonathan Yang<sup>28</sup>, Jordi Salvador<sup>1</sup>, Joseph J. Lim<sup>17</sup>, Junhyek Han<sup>17</sup>, Kaiyuan Wang<sup>36</sup>, Kanishka Rao<sup>11</sup>, Karl Pertsch<sup>34,28</sup>, Karol Hausman<sup>11</sup>, Keegan Go<sup>15</sup>, Keerthana Gopalakrishnan<sup>11</sup>, Ken Goldberg<sup>34</sup>, Kendra Byrne<sup>11</sup>, Kenneth Oslund<sup>11</sup>, Kento Kawaharazuka<sup>31</sup>, Kevin Black<sup>34</sup>, Kevin Lin<sup>28</sup>, Kevin Zhang<sup>4</sup>, Kiana Ehsani<sup>1</sup>, Kiran Lekkala<sup>43</sup>, Kirsty Ellis<sup>41</sup>, Krishan Rana<sup>25</sup>, Krishnan Srinivasan<sup>28</sup>, Kuan Fang<sup>34</sup>, Kunal Pratap Singh<sup>6</sup>, Kuo-Hao Zeng<sup>1</sup>, Kyle Hatch<sup>32</sup>, Kyle Hsu<sup>28</sup>, Laurent Itti<sup>43</sup>, Lawrence Yunliang Chen<sup>34</sup>, Lerrel Pinto<sup>23</sup>, Li Fei-Fei<sup>28</sup>, Liam Tan<sup>34</sup>, Linxi "Jim" Fan<sup>22</sup>, Lionel Ott<sup>7</sup>, Lisa Lee<sup>11</sup>, Luca Wehns<sup>1</sup>, Magnum Chen<sup>13</sup>, Marion Leperf<sup>28</sup>, Marius Memmel<sup>46</sup>, Masayoshi Tomizuka<sup>34</sup>, Masha Itkina<sup>32</sup>, Mateo Guaman Castro<sup>46</sup>, Max Spero<sup>28</sup>, Maximilian Du<sup>28</sup>, Michael Ahn<sup>11</sup>, Michael C. Yip<sup>36</sup>, Mingtong Zhang<sup>39</sup>, Mingyu Ding<sup>34</sup>, Minho Heo<sup>17</sup>, Mohan Kumar Sriram<sup>4</sup>, Mohit Sharma<sup>4</sup>, Moo Jin Kim<sup>28</sup>, Naoaki Kanazawa<sup>31</sup>, Nicklas Hansen<sup>36</sup>, Nicolas Heess<sup>11</sup>, Nikhil J Joshi<sup>11</sup>, Niko Suenderhauf<sup>25</sup>, Ning Liu<sup>13</sup>, Norman Di Palo<sup>14</sup>, Nur Muhammad Mahi Shafullah<sup>23</sup>, Oier Mees<sup>38</sup>, Oliver Kroemer<sup>4</sup>, Osbert Bastani<sup>42</sup>, Pannag R Sanketi<sup>11</sup>, Patrick "Tree" Miller<sup>32</sup>, Patrick Yin<sup>46</sup>, Paul Wohlhart<sup>11</sup>, Peng Xu<sup>11</sup>, Peter David Fagan<sup>37</sup>, Peter Mitrano<sup>40</sup>, Pierre Sermanet<sup>11</sup>, Pieter Abbeel<sup>34</sup>, Priya Sundaresan<sup>28</sup>, Qiuyu Chen<sup>46</sup>, Quan Vuong<sup>11</sup>, Rafael Rafailev<sup>11,28</sup>, Ran Tian<sup>34</sup>, Ria Doshi<sup>34</sup>, Roberto Mart'in-Mart'in<sup>30</sup>, Rohan Bajjal<sup>46</sup>, Rosario Scalise<sup>46</sup>, Rose Hendrix<sup>1</sup>, Roy Lin<sup>34</sup>, Runjia Qian<sup>13</sup>, Ruohan Zhang<sup>28</sup>, Russell Mendonca<sup>4</sup>, Rutav Shah<sup>30</sup>, Ryan Hoque<sup>34</sup>, Ryan Julian<sup>11</sup>, Samuel Bustamante<sup>10</sup>, Sean Kirmani<sup>11</sup>, Sergey Levine<sup>11,34</sup>, Shan Lin<sup>36</sup>, Sherry Moore<sup>11</sup>, Shikhar Bahl<sup>4</sup>, Shivin Dass<sup>43,30</sup>, Shubham Sonawani<sup>2</sup>, Shubham Tulsiani<sup>4</sup>, Shuran Song<sup>5</sup>, Sichun Xu<sup>11</sup>, Siddhant Haldar<sup>23</sup>, Siddharth Karamcheti<sup>28</sup>, Simeon Adebola<sup>34</sup>, Simon Guist<sup>18</sup>, Soroush Nasirany<sup>30</sup>, Stefan Schaal<sup>15</sup>, Stefan Welker<sup>11</sup>, Stephen Tian<sup>28</sup>, Subramanian Ramamoorthy<sup>37</sup>, Sudeep Dasari<sup>4</sup>, Suneel Belkhale<sup>28</sup>, Sungjae Park<sup>17</sup>, Suraj Nair<sup>32</sup>, Suvir Mirchandani<sup>28</sup>, Takayuki Osa<sup>31</sup>, Tanmay Gupta<sup>1</sup>, Tatsuya Harada<sup>31,26</sup>, Tatsuya Matsushima<sup>31</sup>, Ted Xiao<sup>11</sup>, Thomas Kollar<sup>32</sup>, Tian Yu<sup>11</sup>, Tianli Ding<sup>11</sup>, Todor Davchev<sup>11</sup>, Tony Z. Zhao<sup>28</sup>, Travis Armstrong<sup>11</sup>, Trevor Darrell<sup>34</sup>, Trinity Chung<sup>34</sup>, Vidhi Jain<sup>11,4</sup>, Vikash Kumar<sup>4</sup>, Vincent Vanhoucke<sup>11</sup>, Wei Zhan<sup>34</sup>, Wenxuan Zhou<sup>11,4</sup>, Wolfram Burgard<sup>44</sup>, Xi Chen<sup>11</sup>, Xiangyu Chen<sup>13</sup>, Xiaolong Wang<sup>36</sup>, Xinghao Zhu<sup>34</sup>, Xinyang Geng<sup>34</sup>, Xiyuan Liu<sup>13</sup>, Xu Liangwei<sup>13</sup>, Xuanlin Li<sup>36</sup>, Yansong Pang<sup>11</sup>, Yao Lu<sup>11</sup>, Yecheng Jason Ma<sup>42</sup>, Yejin Kim<sup>1</sup>, Yevgen Chebotar<sup>11</sup>, Yifan Zhou<sup>2</sup>, Yifeng Zhu<sup>30</sup>, Yilin Wu<sup>4</sup>, Ying Xu<sup>11</sup>, Yixuan Wang<sup>39</sup>, Yonatan Bisk<sup>4</sup>, Yongqiang Dou<sup>33</sup>, Yoonyoung Cho<sup>17</sup>, Youngwoon Lee<sup>34</sup>, Yuchen Cui<sup>28</sup>, Yue Cao<sup>13</sup>, Yuch-Hua Wu<sup>36</sup>, Yujin Tang<sup>11,31</sup>, Yuke Zhu<sup>30</sup>, Yunchu Zhang<sup>46</sup>, Yunfan Jiang<sup>28</sup>, Yunshuang Li<sup>42</sup>, Yunzhu Li<sup>39</sup>, Yusuke Iwasawa<sup>31</sup>, Yutaka Matsuo<sup>31</sup>, Zehan Ma<sup>34</sup>, Zhuo Xu<sup>11</sup>, Zichen Jeff Cui<sup>23</sup>, Zichen Zhang<sup>1</sup>, Zipeng Fu<sup>28</sup>, Zipeng Lin<sup>34</sup>

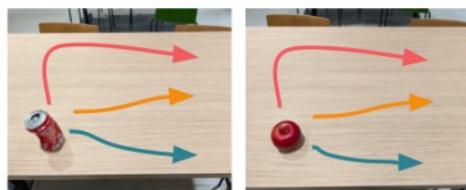


# RT-X: Training RT-2 with Large Diverse Robotic Learning Dataset

**(a) Absolute Motion**  
*move the chip bag to the top / bottom right of the counter*

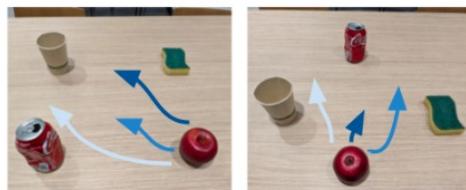


*move to top right / right / bottom right*

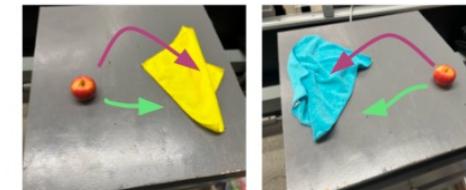


**(b) Object-Relative Motion**

*move apple between coke and cup / coke and sponge / cup and sponge*



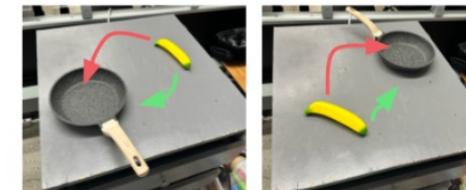
**(c) Preposition Alters Behavior**  
*put apple on cloth / move apple near cloth*



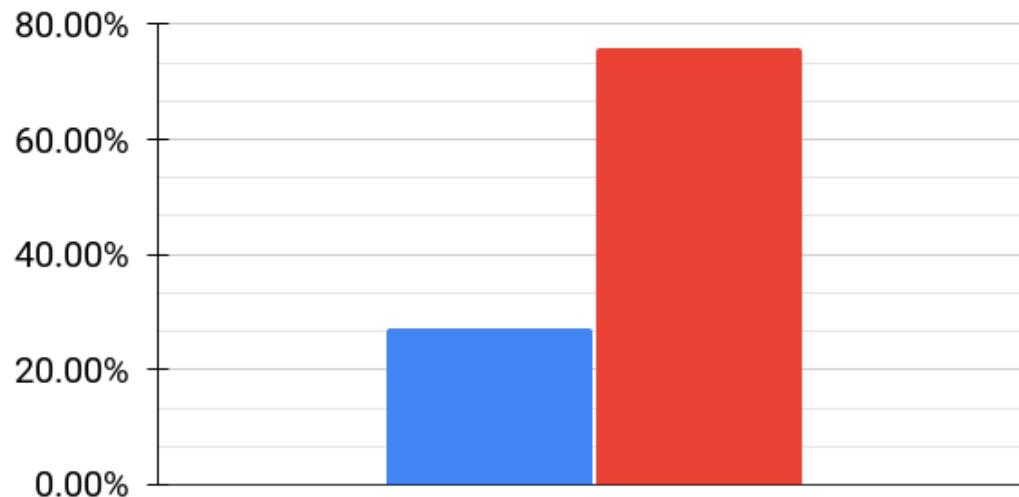
*put orange into the pot / move orange near pot*



*put banana on top of the pan / move banana near pan*

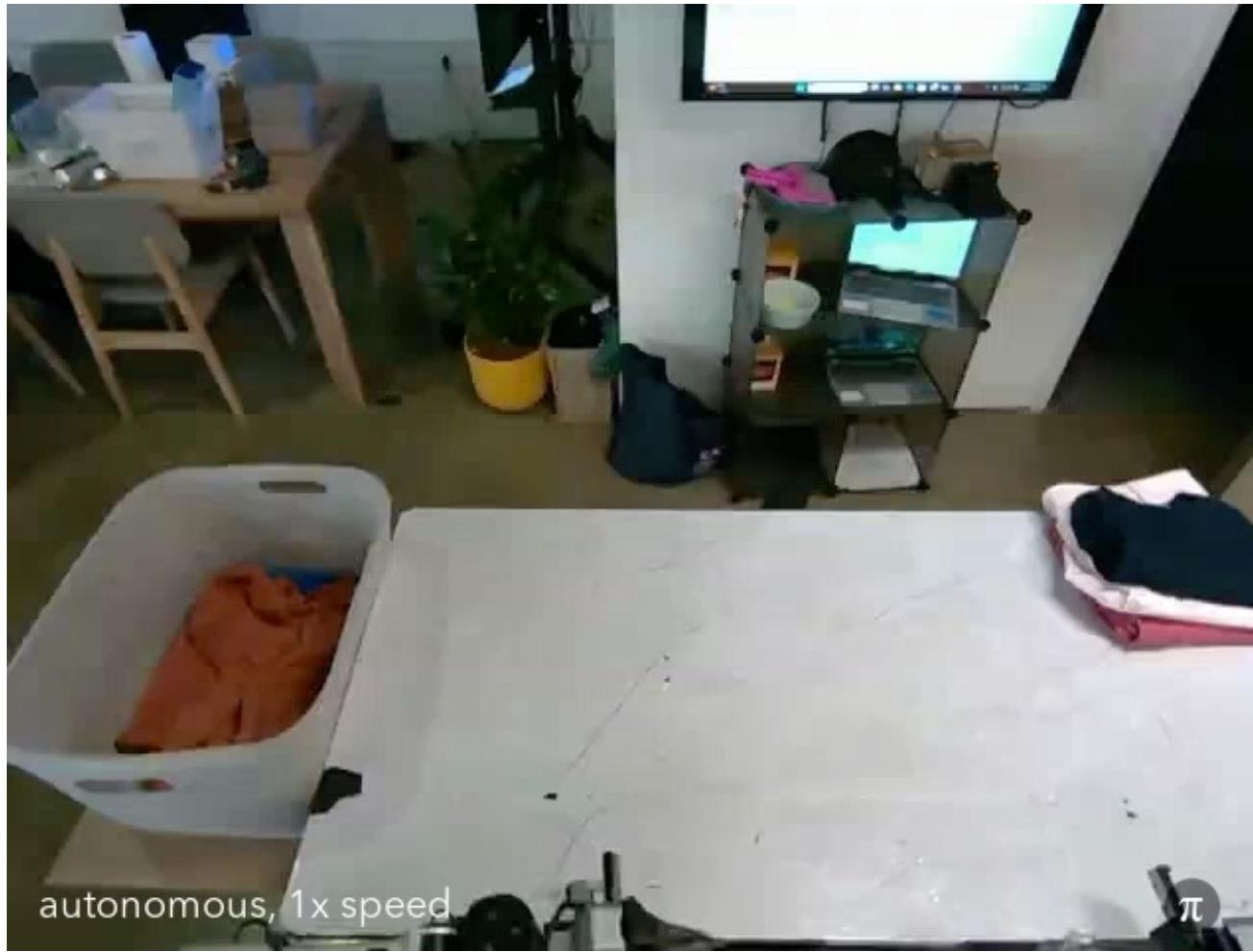


■ RT-2 ■ RT-2-X

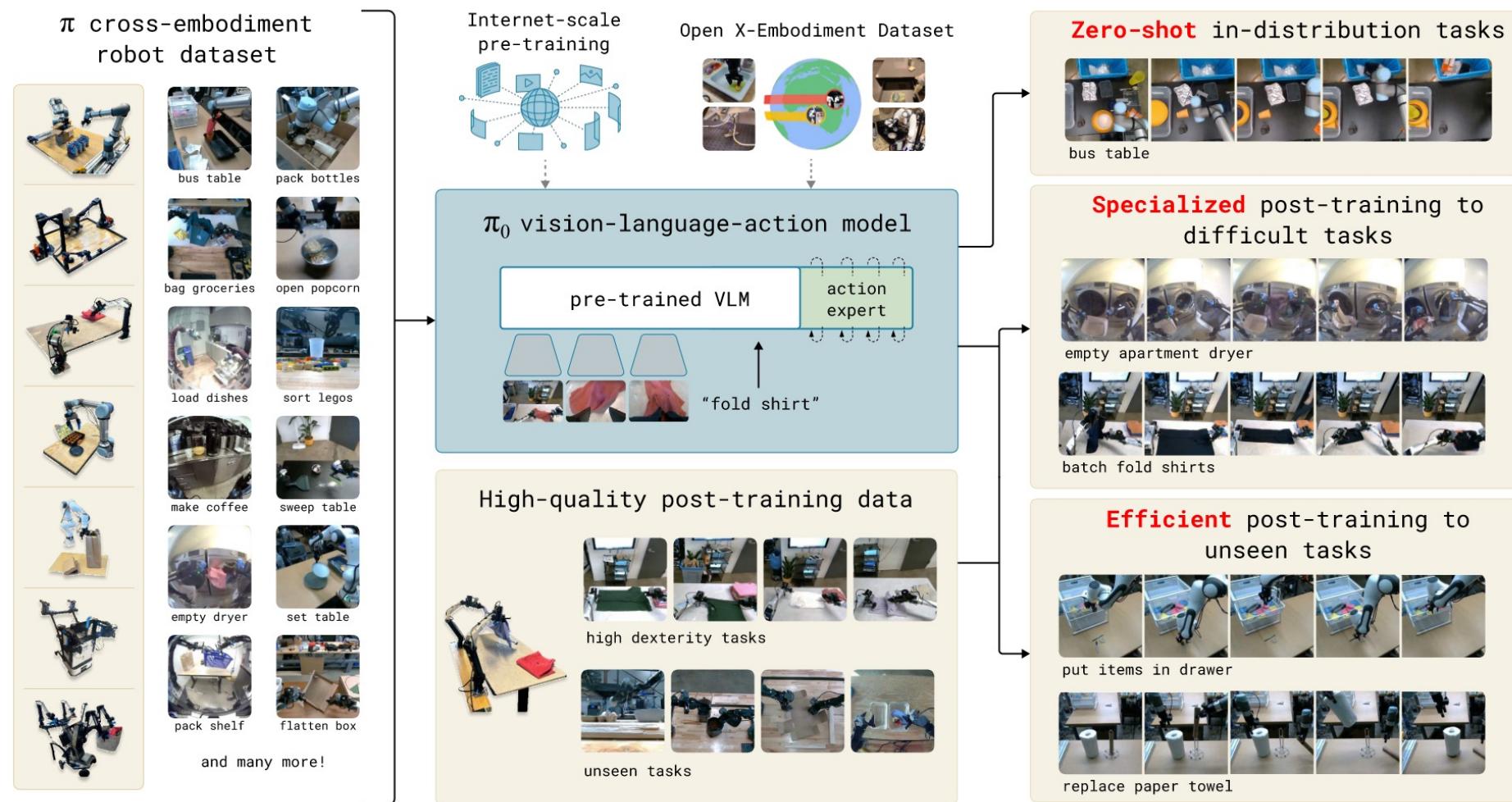


RT-2-X outperforms RT-2 by 3x in emergent skill evaluations

# $\pi_0$ : Current SOTA VLA



# $\pi_0$ : Current SOTA VLA

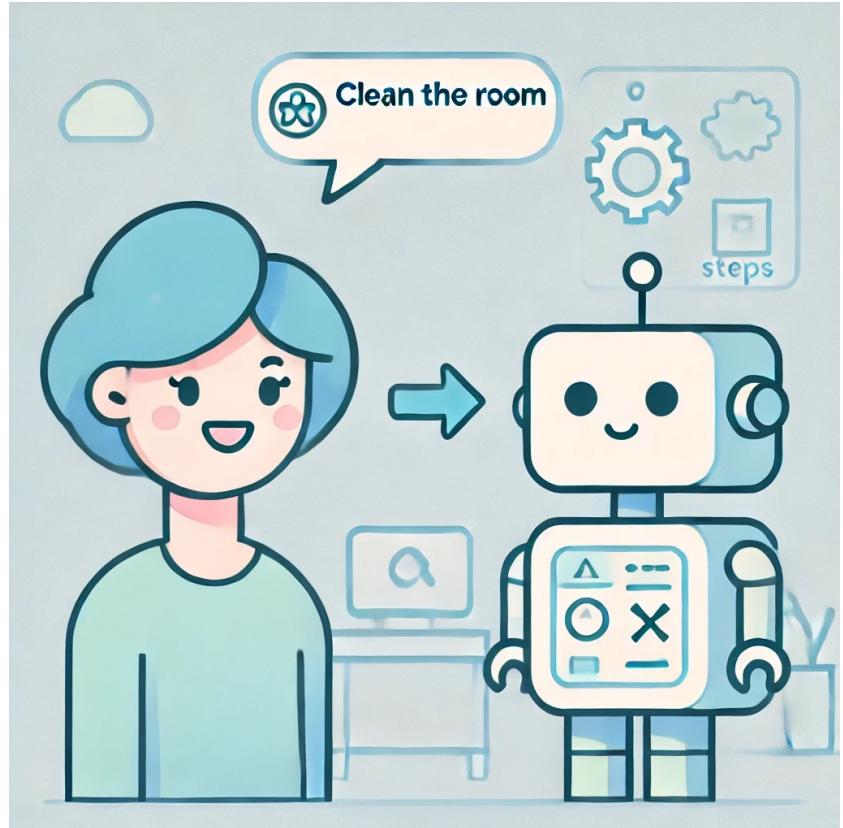


# Why using VLMs in Robotics

- The V(ision)
  - Visual grounding
- The L(anguage)
  - Natural language instructions
  - Reasoning
  - Explainability
- The M (Large Model):
  - General, versatile, flexible
    - Multiple tasks (“generalist”)
    - Multiple robots
  - Accurate
    - Data generation

## Q&A

- High-level: VLMs for planning
  - PaLM-E
  - SpatialVLM
  - MANIPULATE-ANYTHING
- Mid-level: VLMs for structured output
  - RoboPoint
  - ReKep
- Low-level: VLMs for actions (VLA)
  - RT-2
  - RT-X
  - $\pi\pi_0$



Source: DALL-E