GOSIM

GOSIM AI Paris 2025

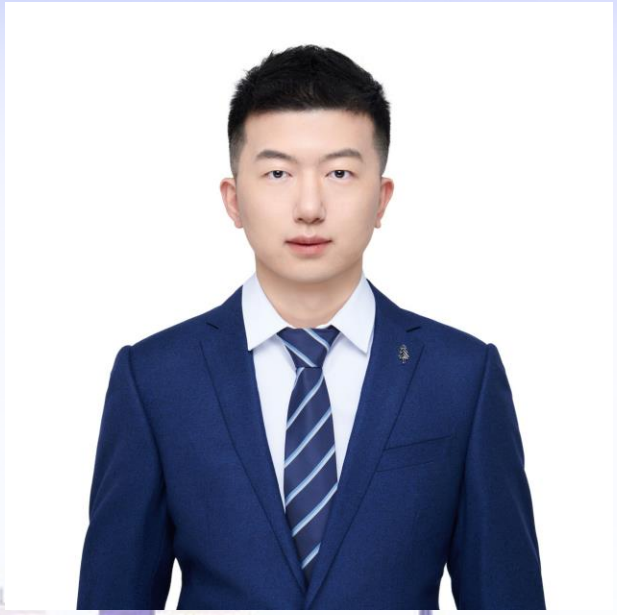
Code-as-Monitor: Constraint-aware Visual Programming for Reactive and Proactive Robotic Failure Detection

Cheng Chi

BAAI

GOSIM

GOSIM AI Paris 2025



Cheng Chi 迟程

Email: chicheng@baai.ac.cn

Homepage: chicheng123.github.io

- Researcher at Beijing Academy of Artificial Intelligence (BAAI)
- Focus on Embodiment AI
- Publish 20+ papers on AI top conferences and journals
- Google scholar citation 3700+

BAAI

GOSIM

GOSIM AI Paris 2025

Code-as-Monitor
Constraint-aware Visual Programming for Reactive and Proactive Robotic Failure Detection

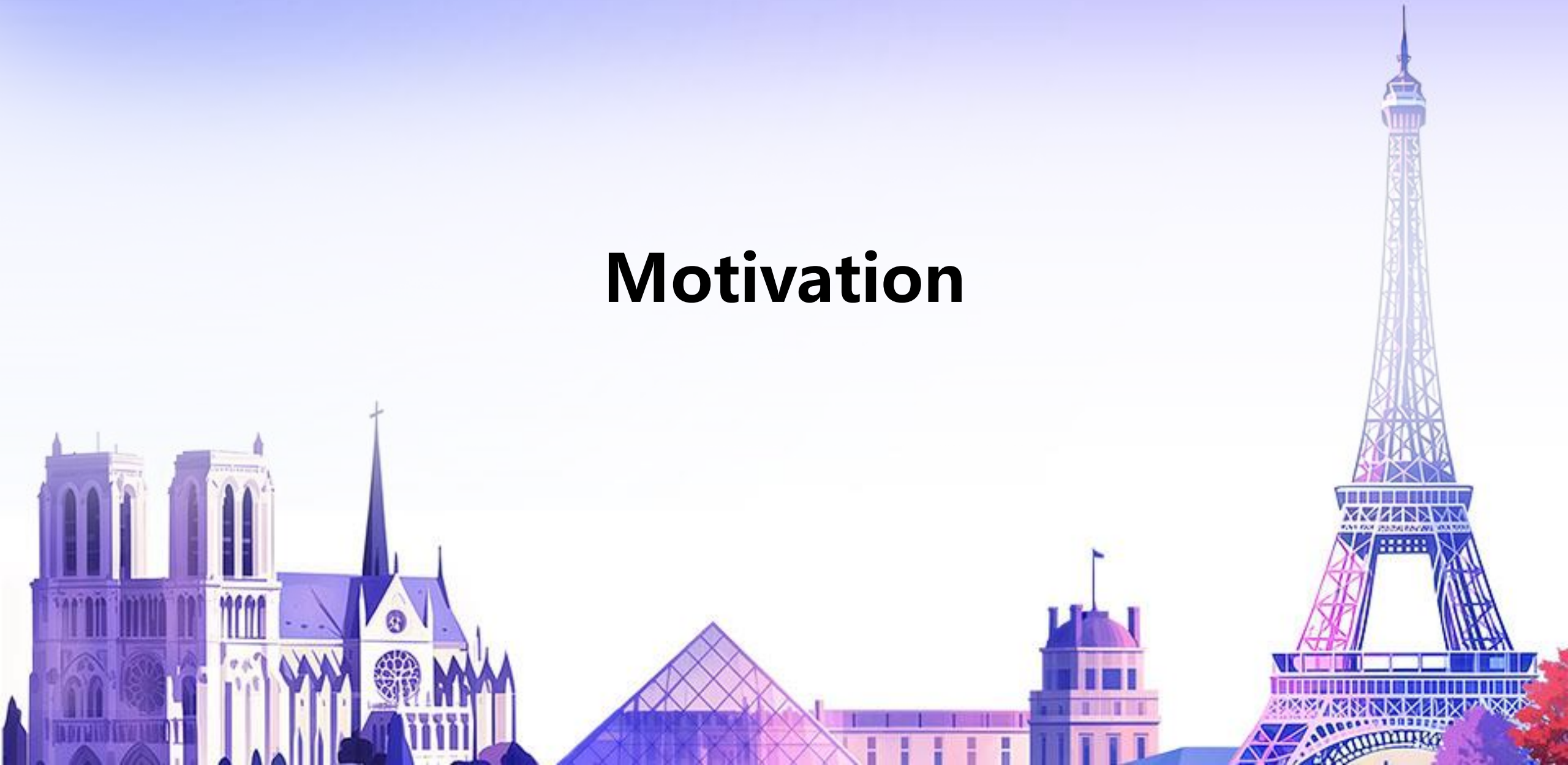
CVPR 2025

zhoues.github.io/Code-as-Monitor https://arxiv.org/pdf/2412.04455

Enshen Zhou*, Qi Su*, Cheng Chi*✉,
Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng✉, He Wang✉

* Equal Contribution ✉ Corresponding author

Motivation



Embodied AI

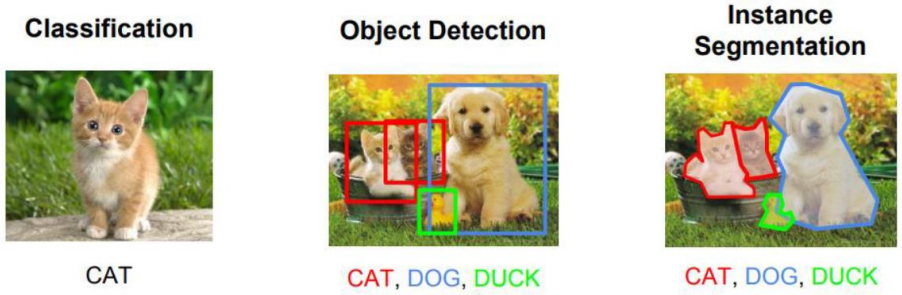
Internet AI: learn from web data

Classification Detection

Segmentation Tracking

Captioning Generation

.....

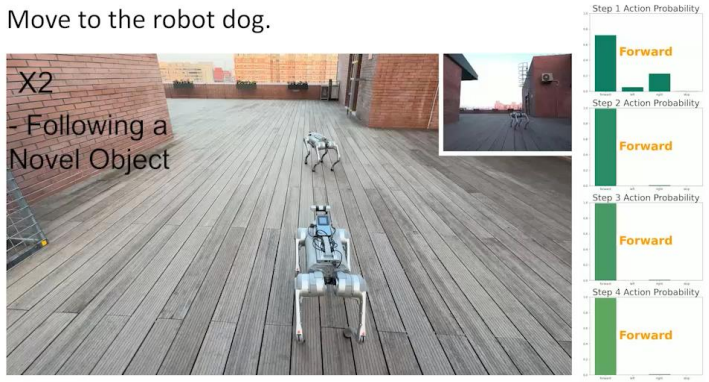


Embodied AI: learn from interaction with environment

Manipulation Navigation

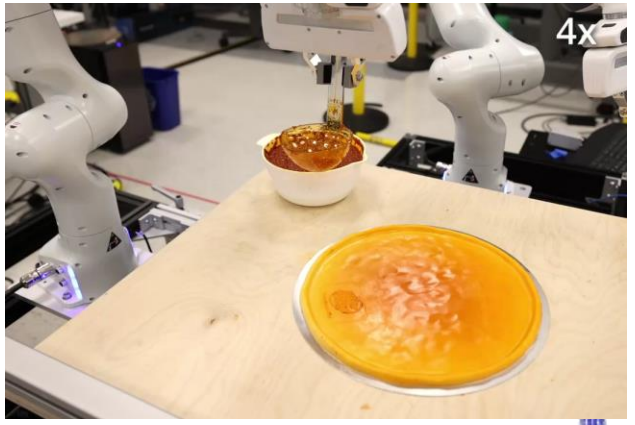
Mobile Manipulation

.....



Uni-Navid

Inner monologue



Diffusion Policy

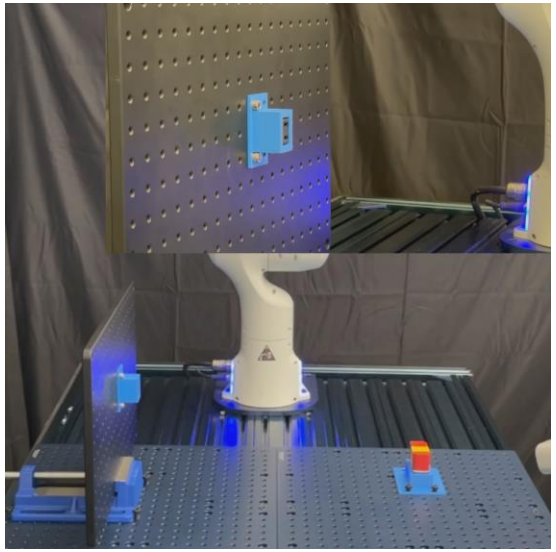
Zhang J, Wang K, Wang S, et al. Uni-NaVid: A Video-based Vision-Language-Action Model for Unifying Embodied Navigation Tasks

Huang W, Xia F, Xiao T, et al. Inner monologue: Embodied reasoning through planning with language models

Chi C, Xu Z, Feng S, et al. Diffusion policy: Visuomotor policy learning via action diffusion

Why do we need failure detection?

$\pi 0$



RVT-2

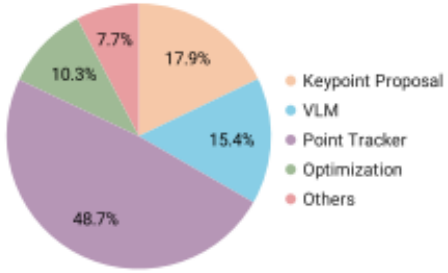


Figure 4: System error breakdown.

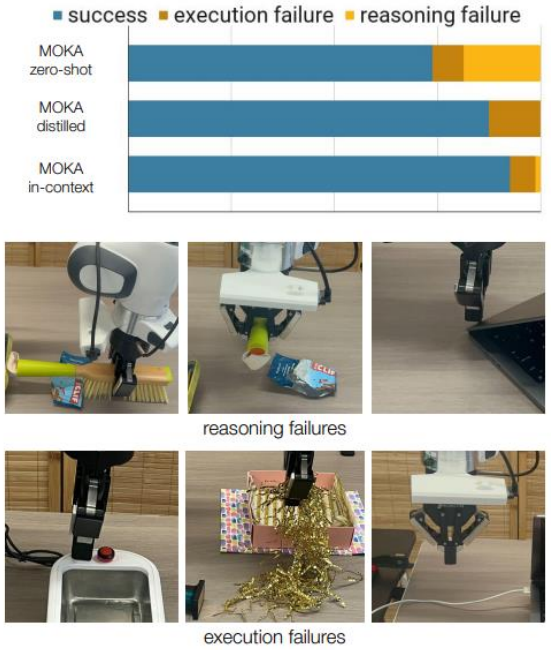
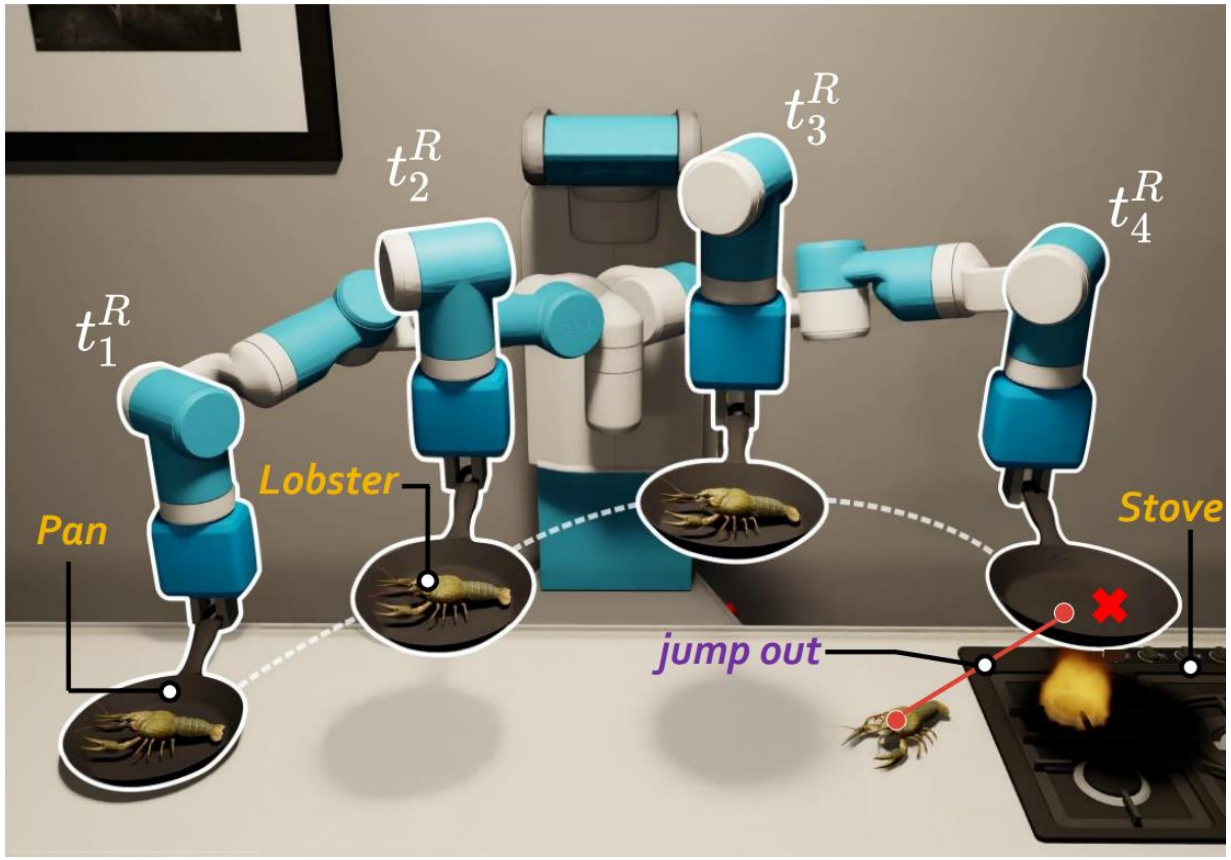


Fig. 4: **Failure analysis.** We break down the failure cases of the variants of our approach using zero-shot prediction, distilled policies, and in-context learning. We break down the failures into reasoning failures (caused by errors in the affordance prediction) and execution failures (caused by low-level motions). We demonstrate how policy distillation and in-context learning reduce failure cases.

- Performing long-term tasks in open world and complex environment, failures are difficult to completely avoid.
- Automatic detection and prevention of failures is critical, especially for closed-loop systems.

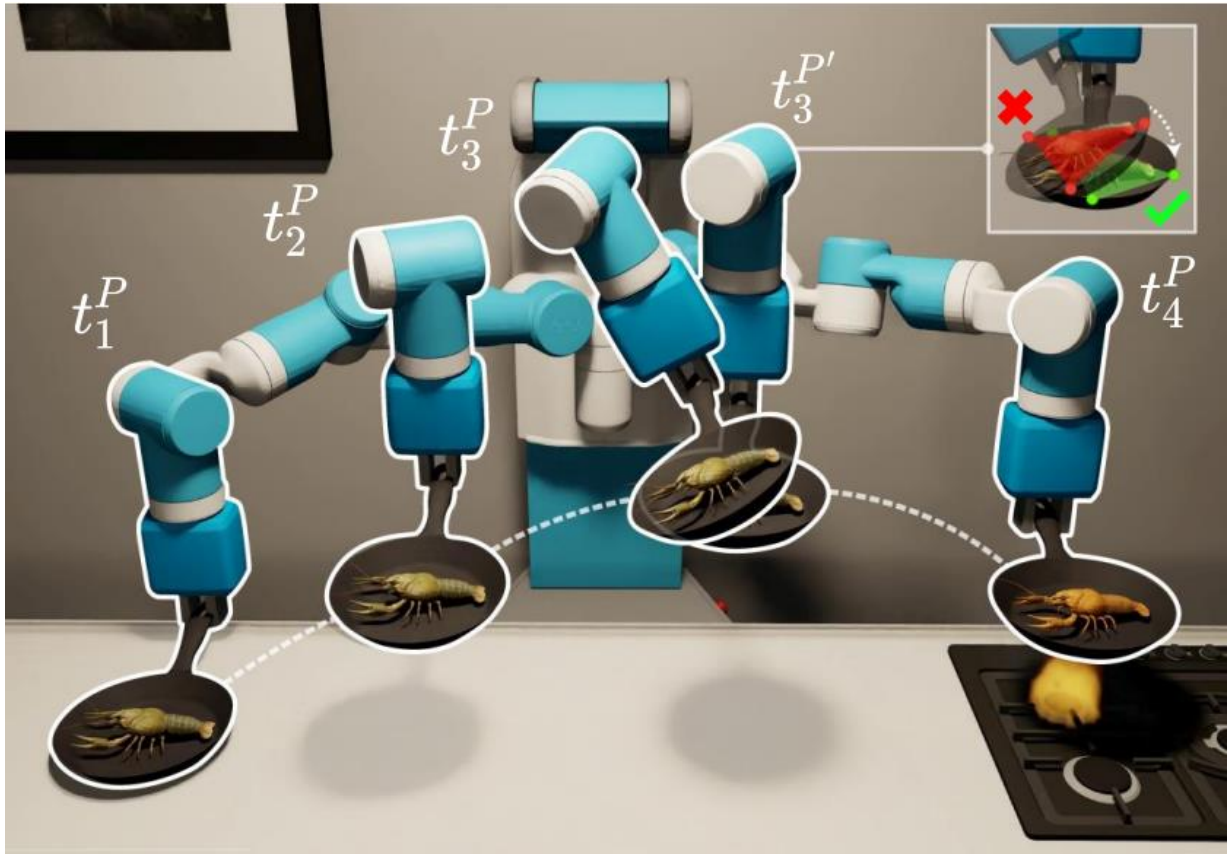
What should we expect from an ideal failure detection framework?



- Faced with a variety of failures, the framework can accurately identify them **after they occur**.
- This type is called **Reactive Failure Detection**

Q: Do we have to detect and recover after failures occur?

What should we expect from an ideal failure detection framework?



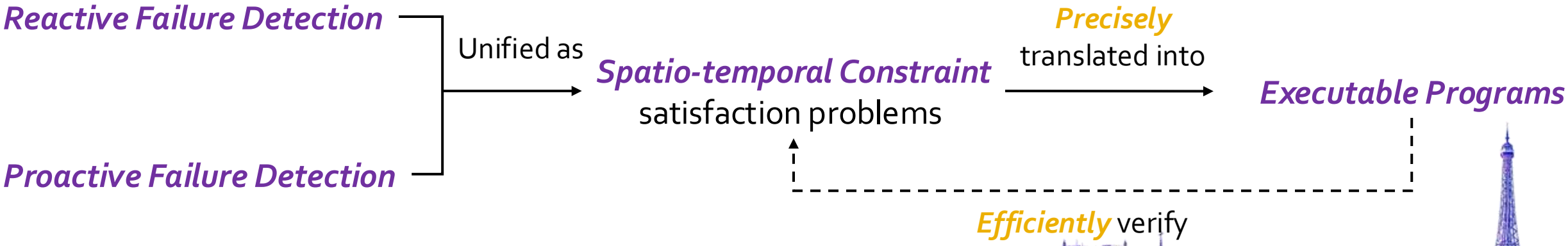
- Detect **foreseeable**, impending failures and prevent them **before they occur**.
- This type is called **Proactive Failure Detection**

New definition of failure detection

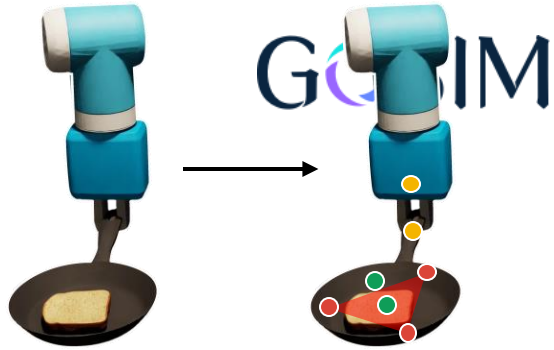
- Unify **Reactive** and **Proactive** failure detection into **spatio-temporal constraint satisfaction** problem
- **Reactive**: whether the required states of entities (such as robots, objects, etc.) in the environment are reached **at the end of the task (or subtask).**
- **Proactive:** whether all entities in the environment (such as robots, objects, etc.) maintain the required state **during the task (or subtask).**

Leverage VLM to pursue real-time and accuracy

- Using reasoning and generalization of **VLMs** to convert the **open set** failure detection into a set of spatiotemporal constraints.
- Using **codes** to **accurately and real-time** evaluate whether the entities maintain the required state during the process or reach the state in the end



Constraint Element



Reactive Failure Detection

Proactive Failure Detection

Unified as

Spatio-temporal Constraint Satisfaction problems

Abstract

Constraint Elements (Point, Line, Surface)

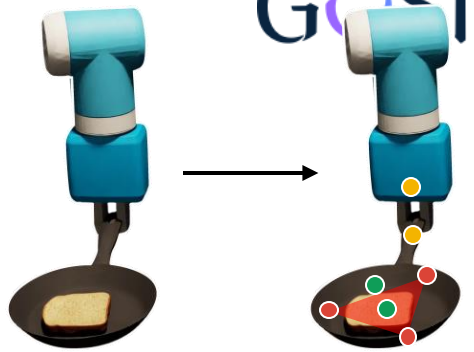
Precisely translated into

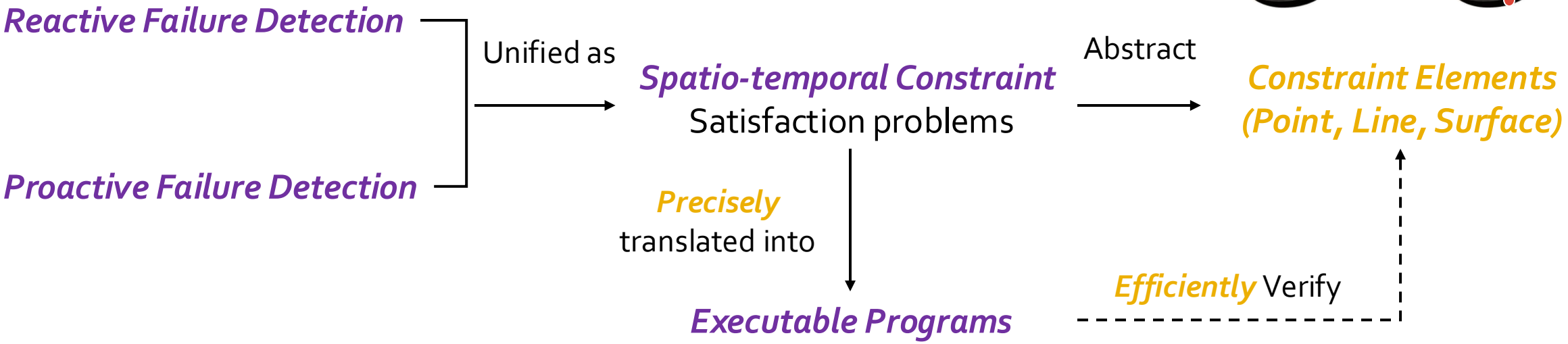
Executable Programs

Efficiently Verify

- Propose a new representation - constraint element
- Constraint elements abstract constraint-related entities or their parts into geometric elements (such as points, lines, and surfaces), which are composed of points in 3D space.

Advantages of Constraint Elements

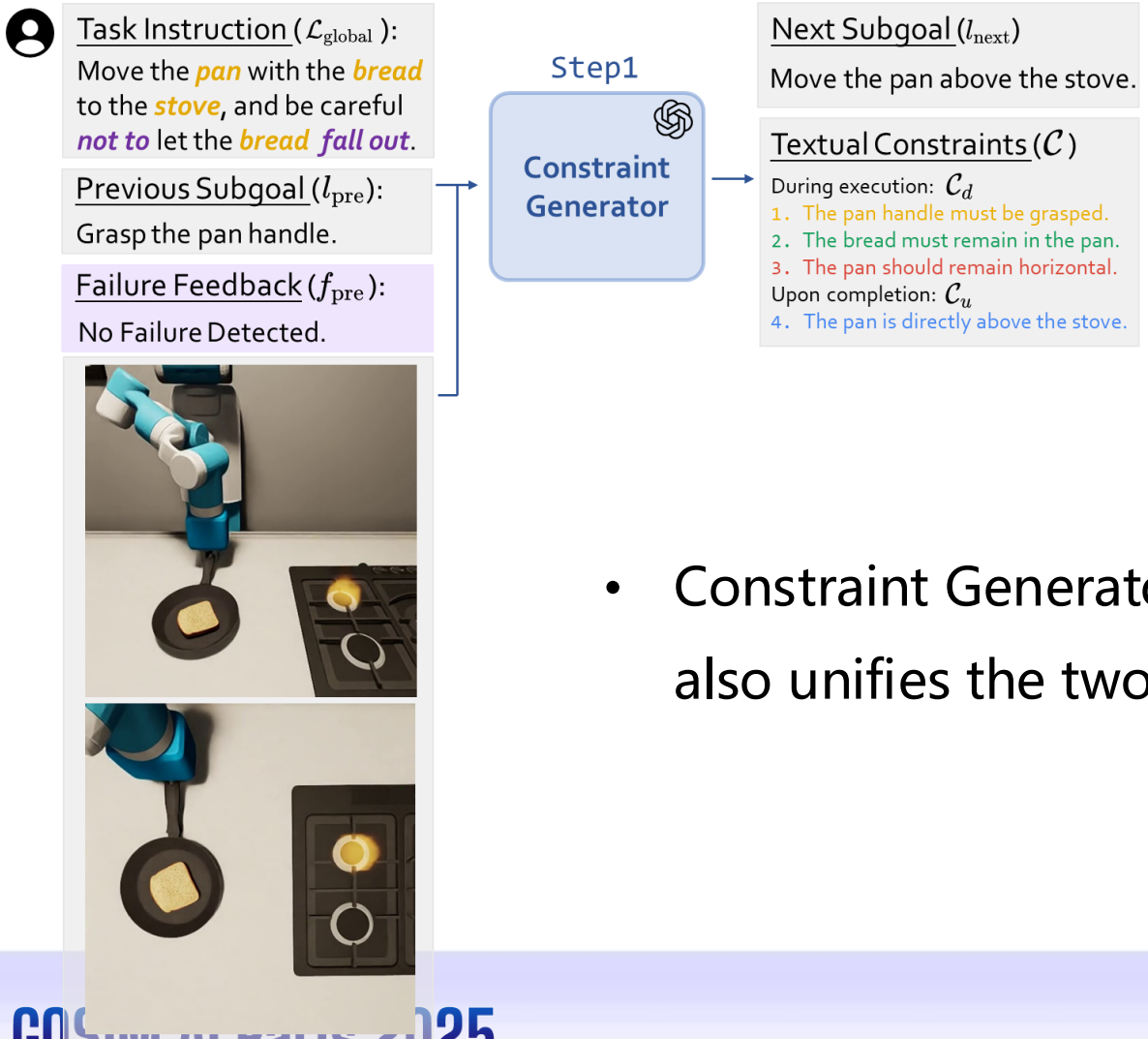




- 3D points are convenient as code input for calculation.
- Abstract constraint-related entities and remove irrelevant geometric and visual elements.
- Real-time monitoring can be achieved by simply tracking constraint elements and calculating their spatiotemporal relationships.

The Whole Framework

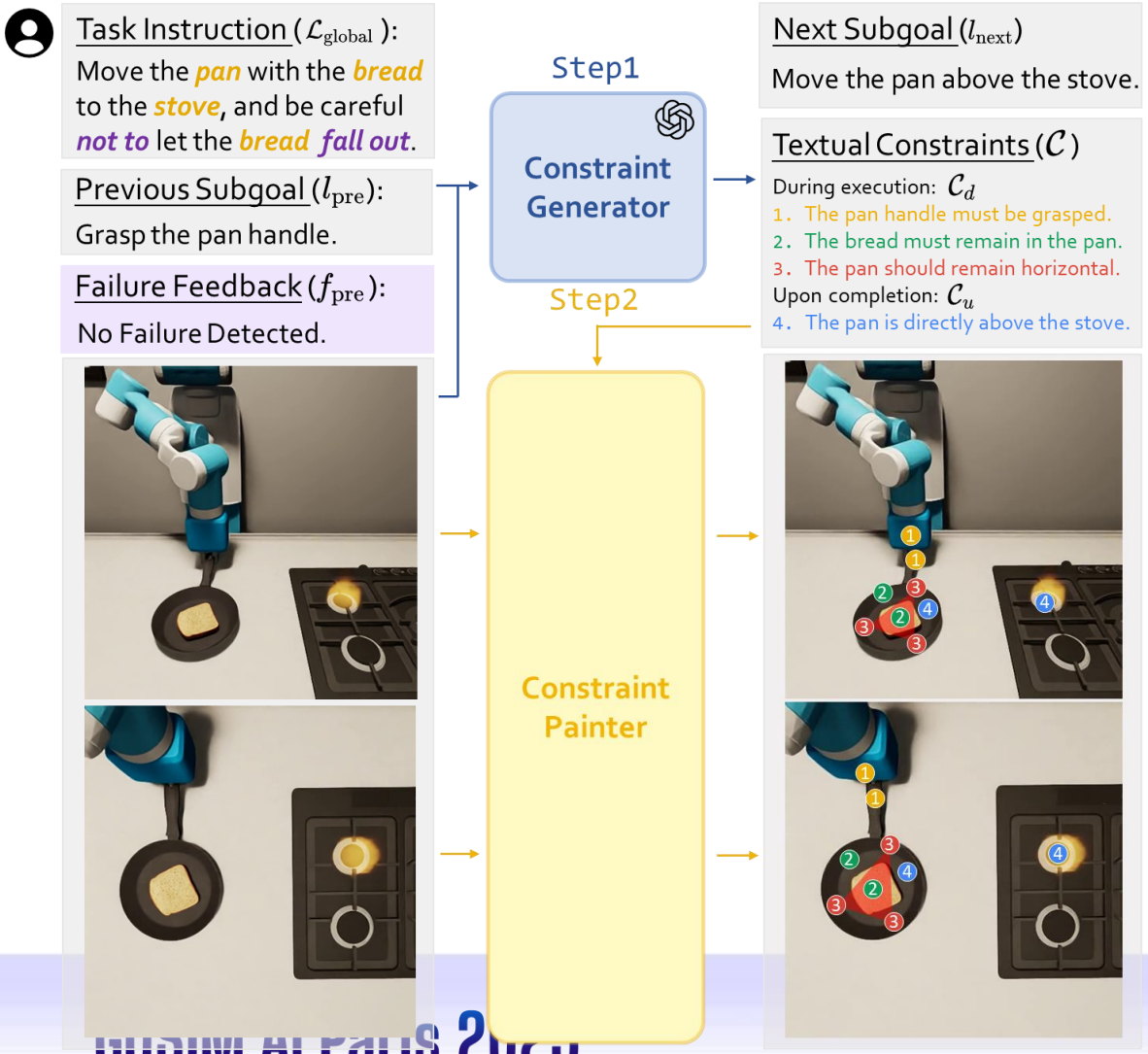
Step 1: Constraint Generation



$$l_{\text{next}}, \mathcal{C}_d, \mathcal{C}_u = \mathcal{F}_{\text{VLM}}(\mathcal{O}, \mathcal{L}_{\text{global}}, l_{\text{pre}}, f_{\text{pre}})$$

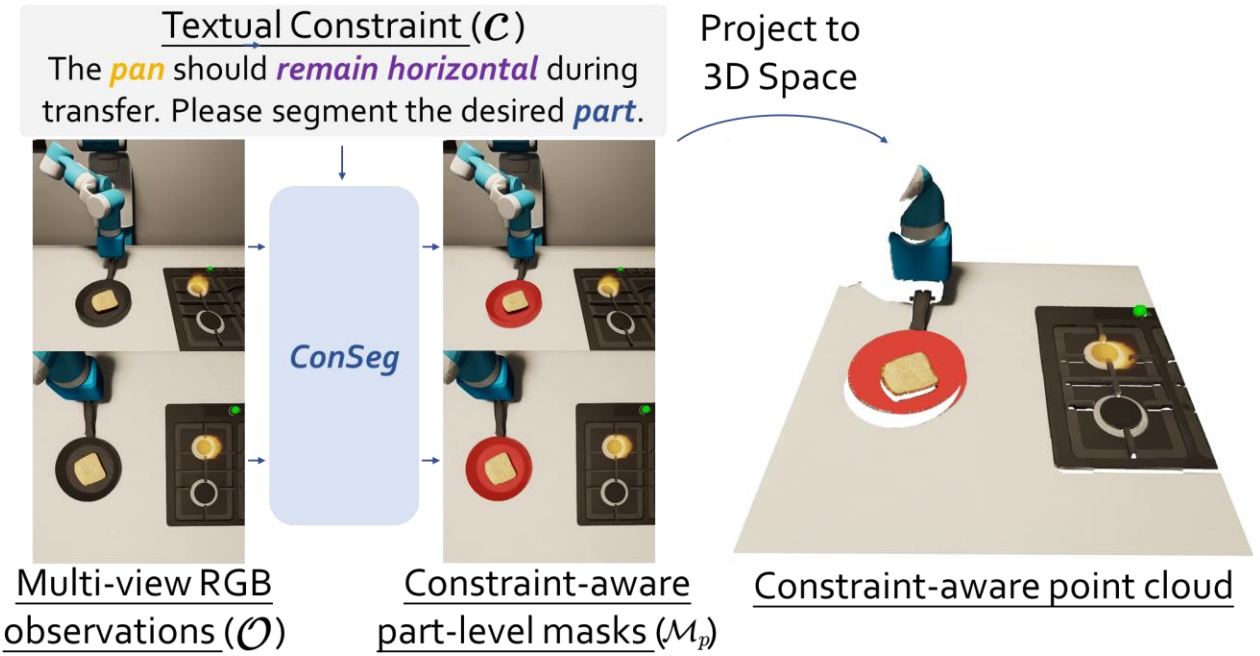
- Constraint Generator not only decomposes long-term tasks, but also unifies the two types of failure detection.

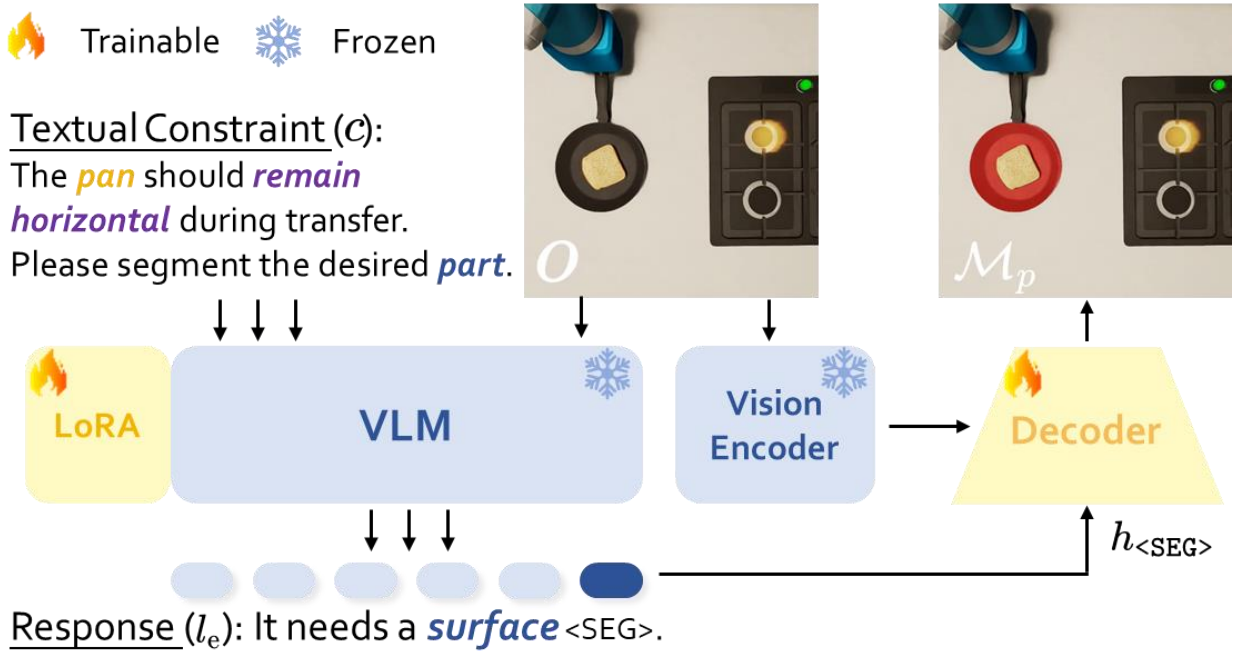
Step 2: Paint Constrain element



- Constraint Painter converts text constraints into constraint elements and paints them on multi-view images.

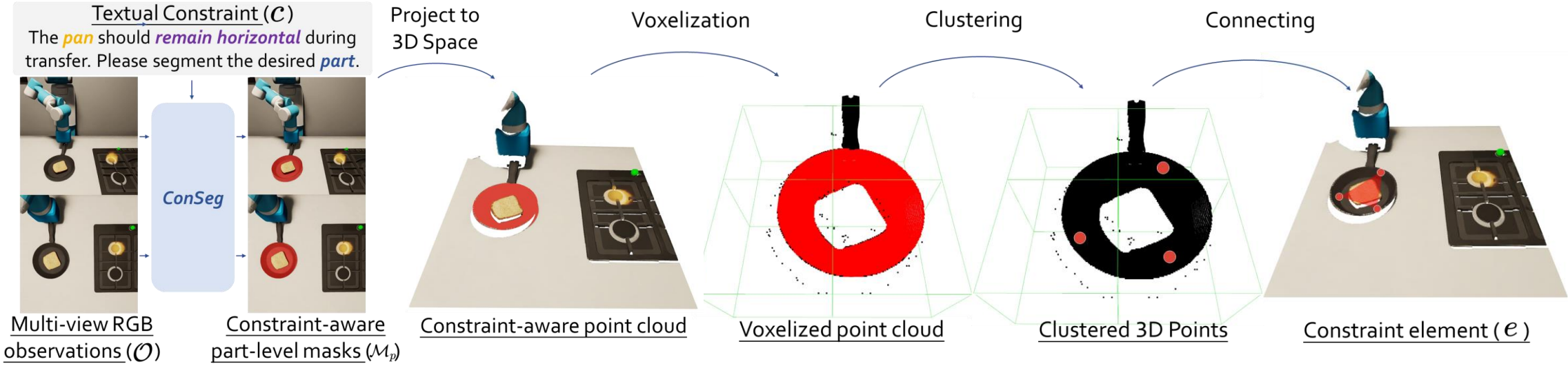
Constraint-based instance-level / part-level segmentation





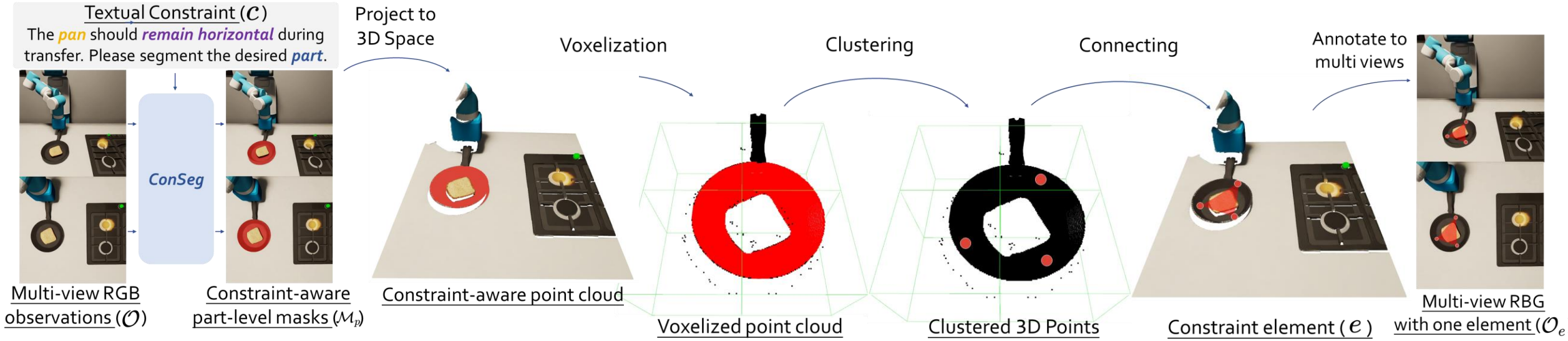
- ConSeg is a reasoning segmentation model that gives both the pixel-level segmentation results and the required text-level constraint types (for example, segmenting the pan surface and giving the "surface" type).

Voxelization, Clustering, and Wiring



- Point clouds are fused according to the camera internal and external parameters and depth information, and voxelized according to the constraint element type.
- Voxel grid is clustered and the point number is ensured according to the constraint element type
- All points are connected within the object to obtain the constraint element in 3D space.

Painting



- Project the constrained elements in 3D space back to the image view

Step 3: Monitor code generation

Task Instruction ($\mathcal{L}_{\text{global}}$):
Move the *pan* with the *bread* to the *stove*, and be careful *not to* let the *bread* *fall out*.

Previous Subgoal (l_{pre}):
Grasp the pan handle.

Failure Feedback (f_{pre}):
No Failure Detected.

Step1

Constraint Generator

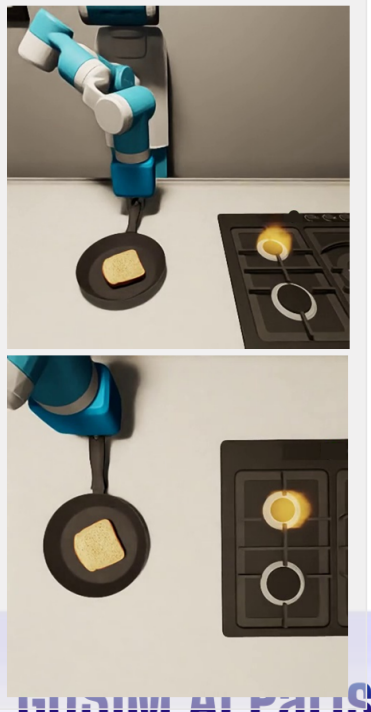
Next Subgoal (l_{next})
Move the pan above the stove.

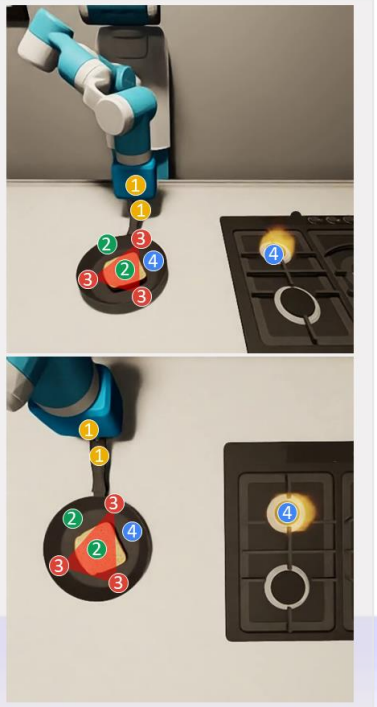
Textual Constraints ($\mathcal{C}$)
During execution: $\mathcal{C}_d$
1. The pan handle must be grasped.
2. The bread must remain in the pan.
3. The pan should remain horizontal.

Upon completion: $\mathcal{C}_u$
4. The pan is directly above the stove.

Step2

Constraint Painter





Step3

Constraint Monitor

Monitor Code

```
def constraint_monitor(end_effector, element_position, is_finished):
    points_yellow, point_green, surface_red, point_blue = element_position
    if not is_finished:
        v1, v2 = surface_red[-1][1] - surface_red[-1][0], surface_red[-1][2] - surface_red[-1][0]
        normal_vector = np.cross(v1, v2)
        normal_vector = normal_vector / np.linalg.norm(normal_vector)
        angle = np.degrees(np.arccos(np.dot(normal_vector, np.array([0, 0, 1]))))
        if angle > tol_angle:
            return False, f"The pan surface is not horizontal, as the angle between the pan
                            surface and the horizontal plane is {angle} degrees."
    …
    return True, "No Failure Detected."
```

Multi-view RGB-D Observations ($\mathcal{O}$)

Multi-view RGB Observations with all constraint elements ($\mathcal{O}_{\mathcal{E}}$)

Step 4: Constraint element tracking and calculation

Task Instruction ($\mathcal{L}_{\text{global}}$):
Move the *pan* with the *bread* to the *stove*, and be careful *not to* let the *bread fall out*.

Previous Subgoal (l_{pre}):
Grasp the pan handle.

Failure Feedback (f_{pre}):
No Failure Detected.

Step1

Constraint Generator

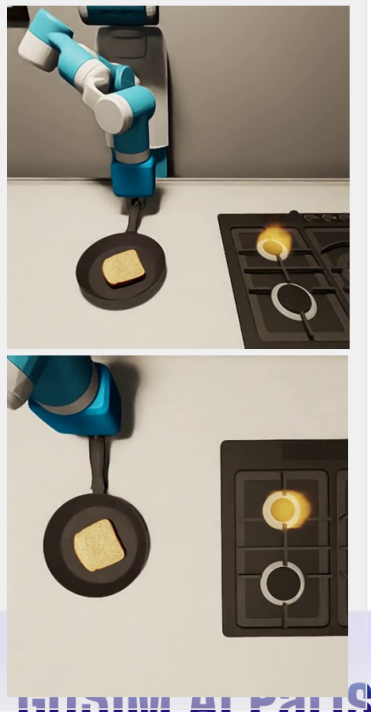
Next Subgoal (l_{next})
Move the pan above the stove.

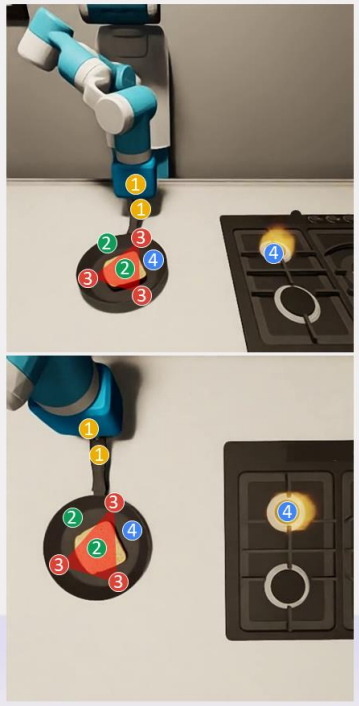
Textual Constraints ($\mathcal{C}$)
During execution: $\mathcal{C}_d$
1. The pan handle must be grasped.
2. The bread must remain in the pan.
3. The pan should remain horizontal.

Upon completion: $\mathcal{C}_u$
4. The pan is directly above the stove.

Step2

Constraint Painter



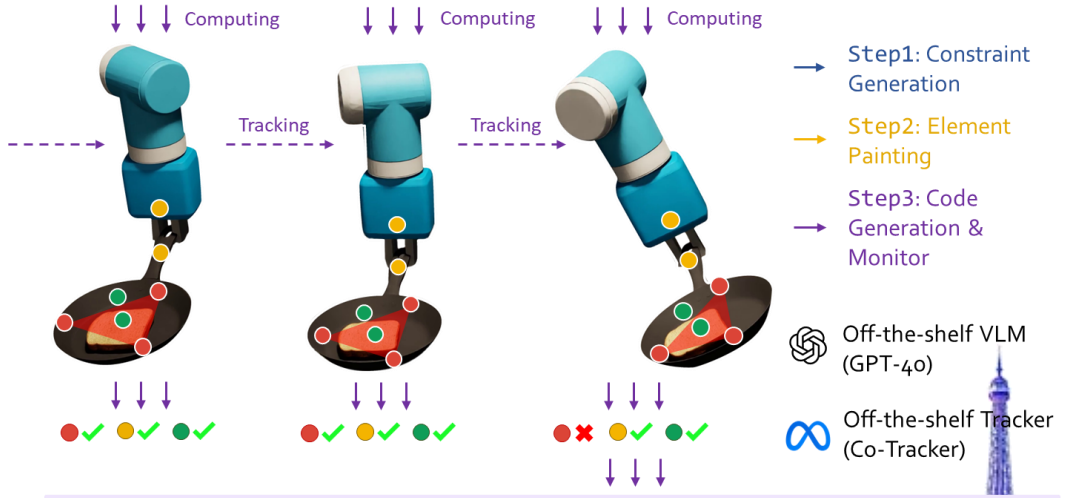


Step3

Constraint Monitor

Monitor Code

```
def constraint_monitor(end_effector, element_position, is_finished):
    points_yellow, point_green, surface_red, point_blue = element_position
    if not is_finished:
        v1, v2 = surface_red[-1][1] - surface_red[-1][0], surface_red[-1][2] - surface_red[-1][0]
        normal_vector = np.cross(v1, v2)
        normal_vector = normal_vector / np.linalg.norm(normal_vector)
        angle = np.degrees(np.arccos(np.dot(normal_vector, np.array([0, 0, 1]))))
        if angle > tol_angle:
            return False, f"The pan surface is not horizontal, as the angle between the pan
                            surface and the horizontal plane is {angle} degrees."
    …
    return True, "No Failure Detected."
```

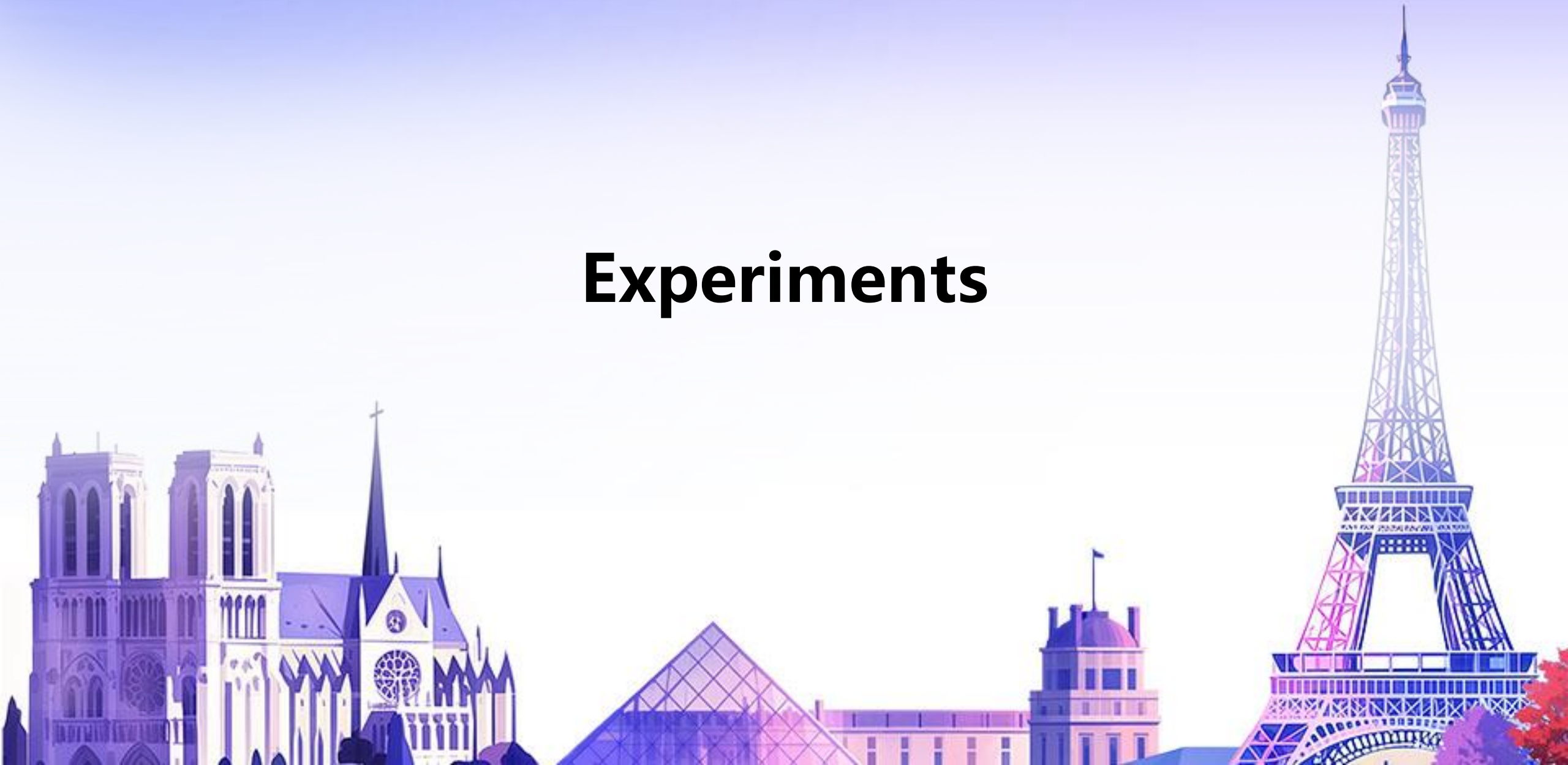


Failure Feedback (f_{pre})
Detect failure! The *pan* surface is *not horizontal*, as the angle between the pan surface and the horizontal plane is 25 degrees. Please use this feedback to *re-plan*.

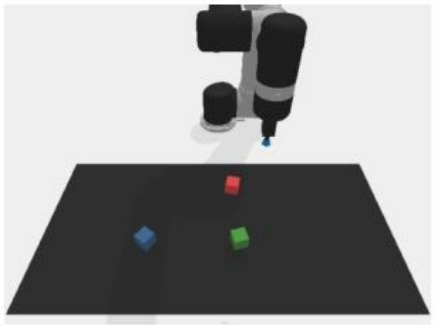
Multi-view RGB-D Observations ($\mathcal{O}$)

Multi-view RGB Observations with all constraint elements ($\mathcal{O}_{\mathcal{E}}$)

Experiments



Simulator / Real-world Setting



Front View

Top View

(a) CLIPort

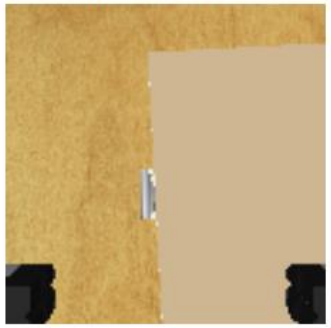
Front View

Top View

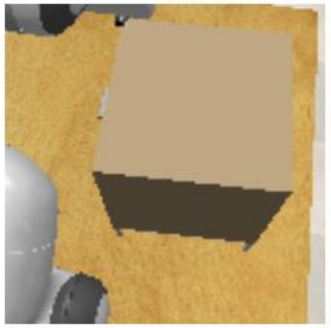
(b) Omnigibson

Front View

Left Shoulder View

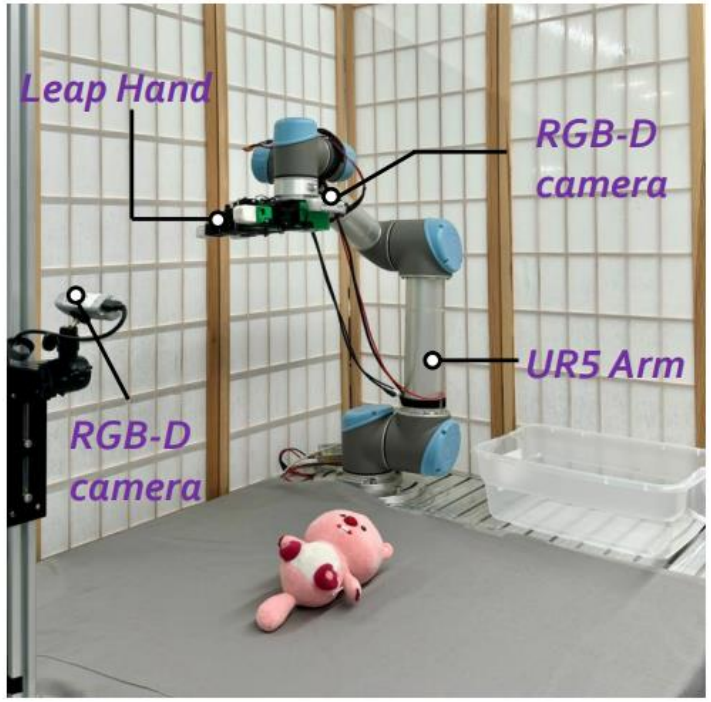


Wrist View

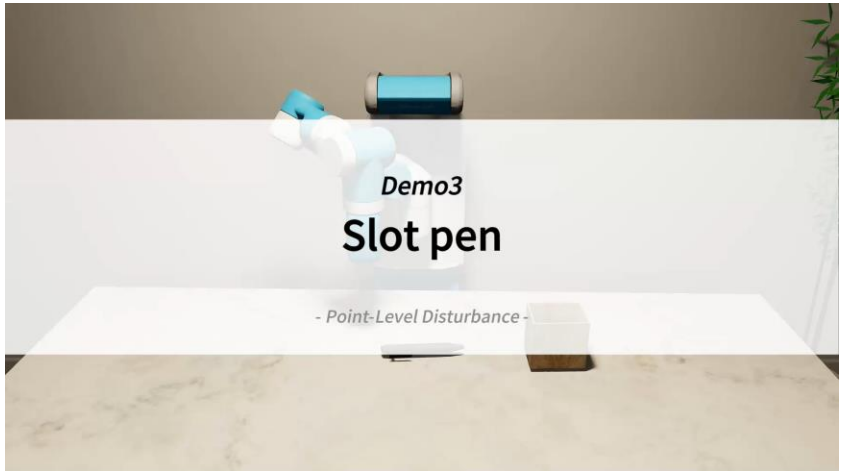


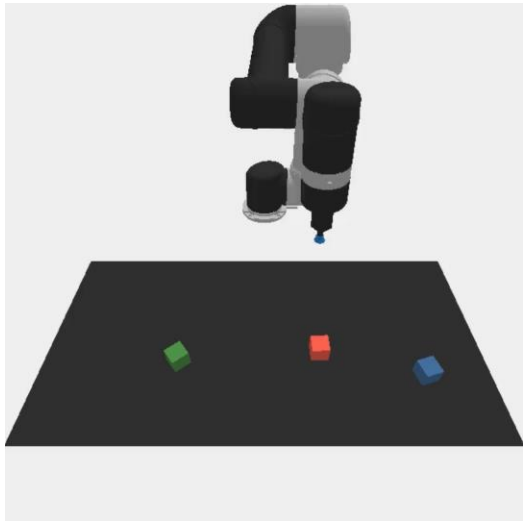
Right Shoulder View

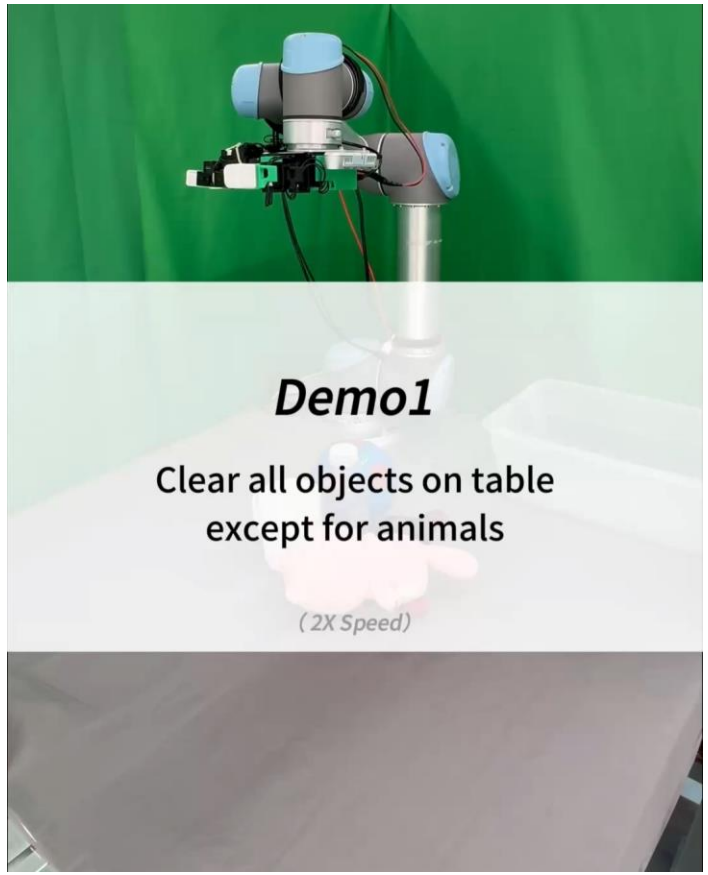
(c) RLBench

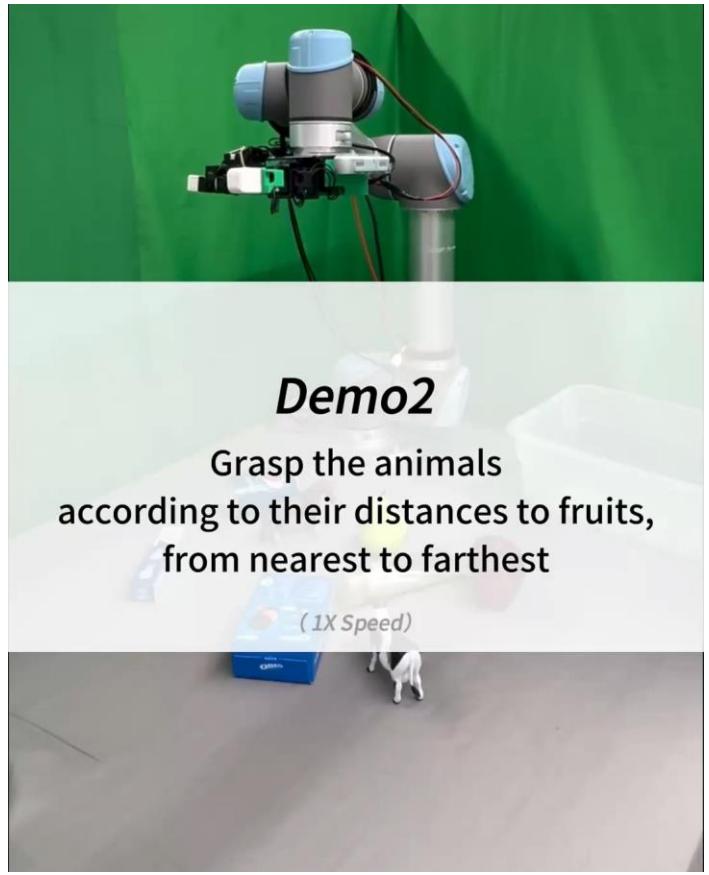


(d) Real-world Setting


Demo3
Slot pen
- Point-Level Disturbance -




Demo1
Clear all objects on table
except for animals
(2X Speed)


Demo2
Grasp the animals
according to their distances to fruits,
from nearest to farthest
(1X Speed)

Generalizability of ConSeg model

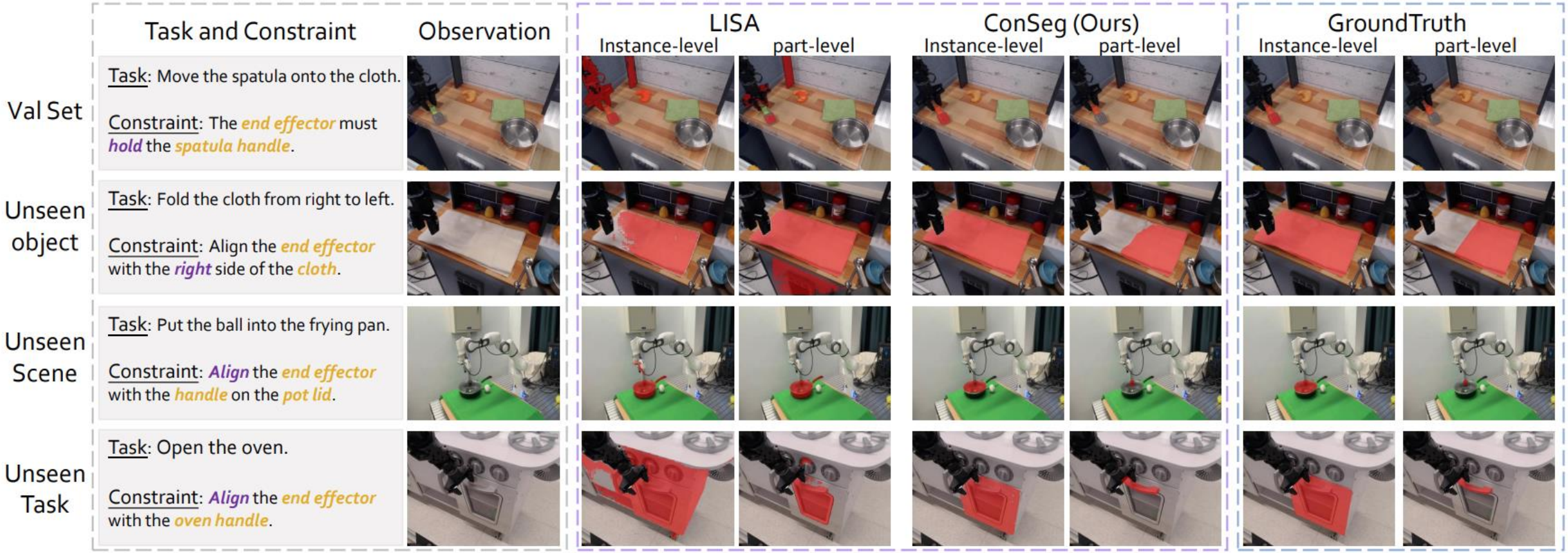
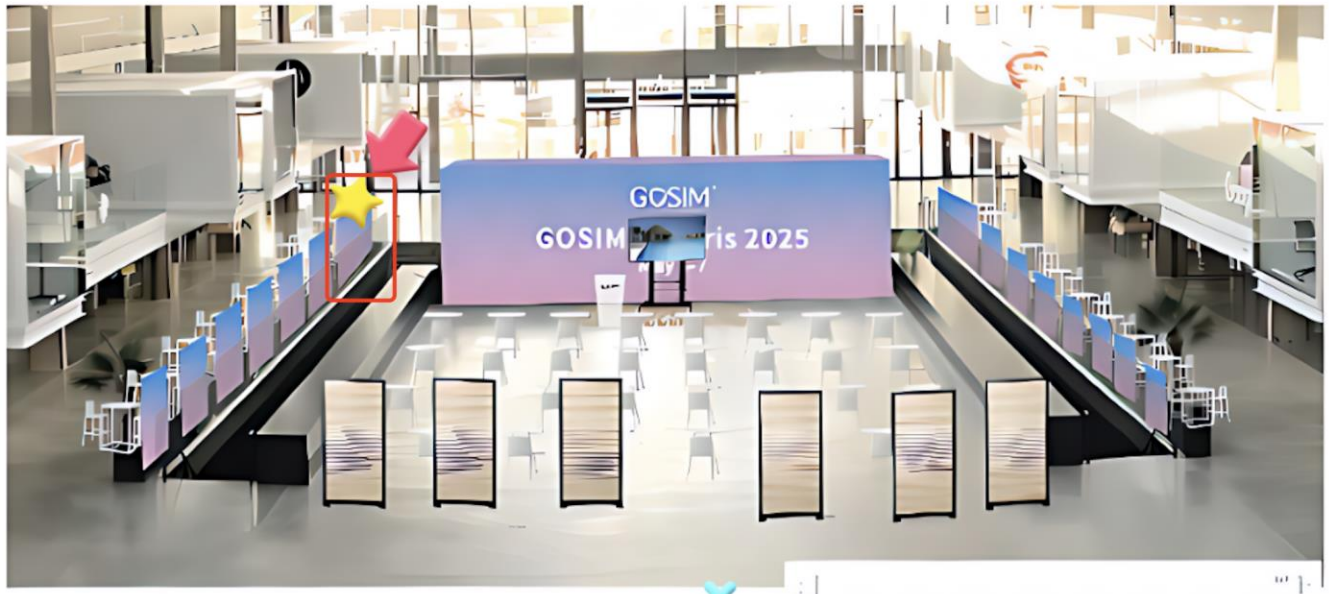


Figure 6. Visual comparison between our *ConSeg* and LISA [29] at instance and part level. The red masks are the segmentation results.

Welcome to our Booth!

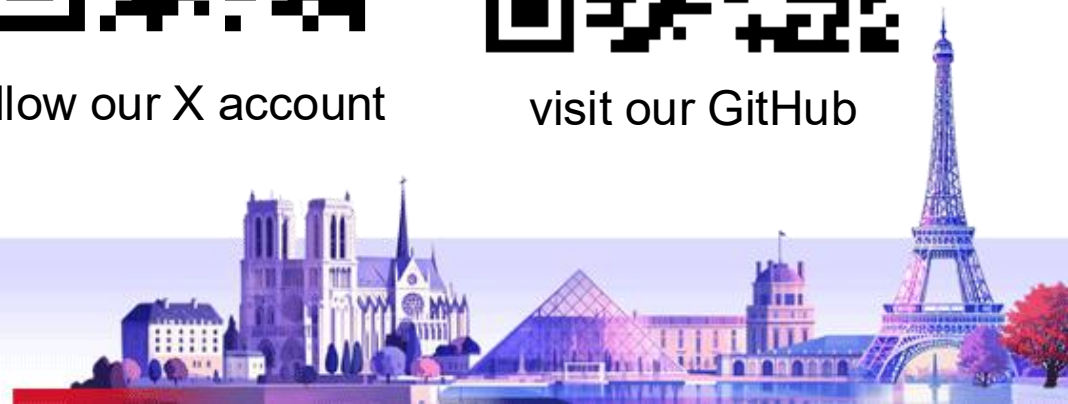


1st Floor, Open Platform Area
Entering the main gate, the first booth on the right side (next to the GOSIM main display board)

Visit us at **BAAI Booth** (with a shining Star in the left picture)

follow our X account

visit our GitHub



Q & A

Code-as-Monitor
Constraint-aware Visual Programming for
Reactive and Proactive Robotic Failure Detection

zhoues.github.io/Code-as-Monitor https://arxiv.org/pdf/2412.04455

GOSIM

GOSIM AI Paris 2025

THANK YOU