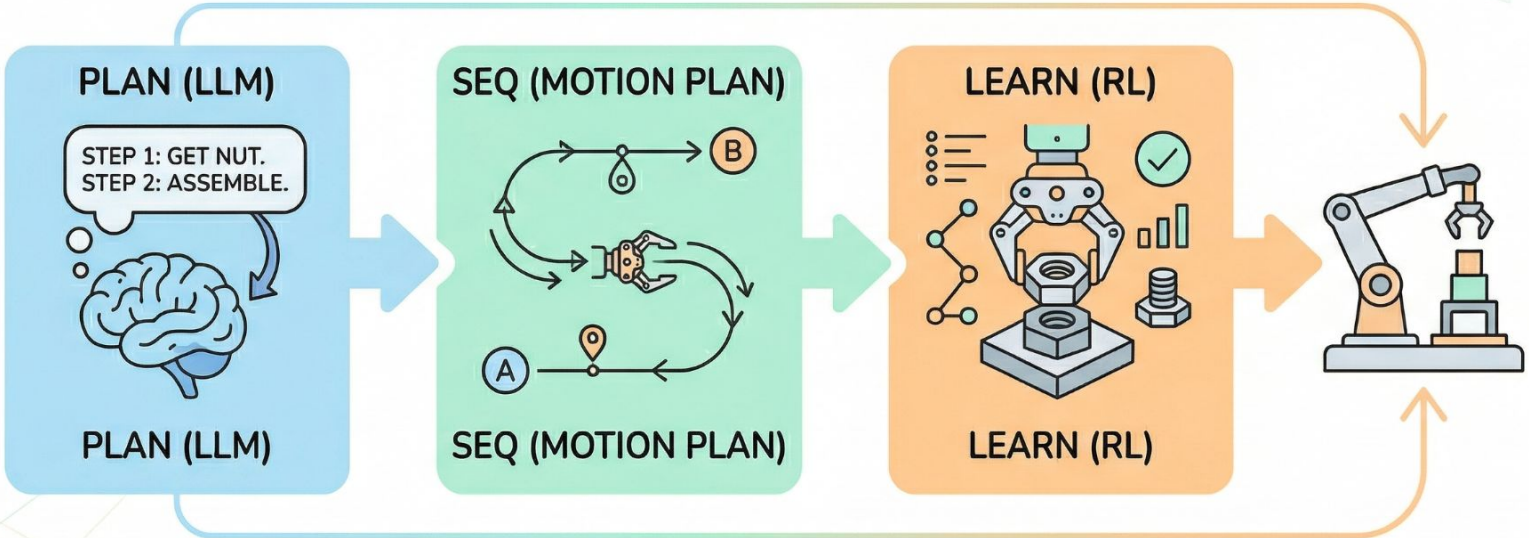
PLAN-SEQ-LEARN: LANGUAGE MODEL GUIDED RL FOR SOLVING LONG HORIZON ROBOTICS TASKS

Murtaza Dalal, Tarun Chiruvolu, Devendra Chaplot, Ruslan Salakhutdinov
ICLR 2024



Presenter: Wentao Cao, Chaol Tuan

CMU

MISTRAL AI

Plan-Seq-Learn

Language model guided RL for long-horizon robotics

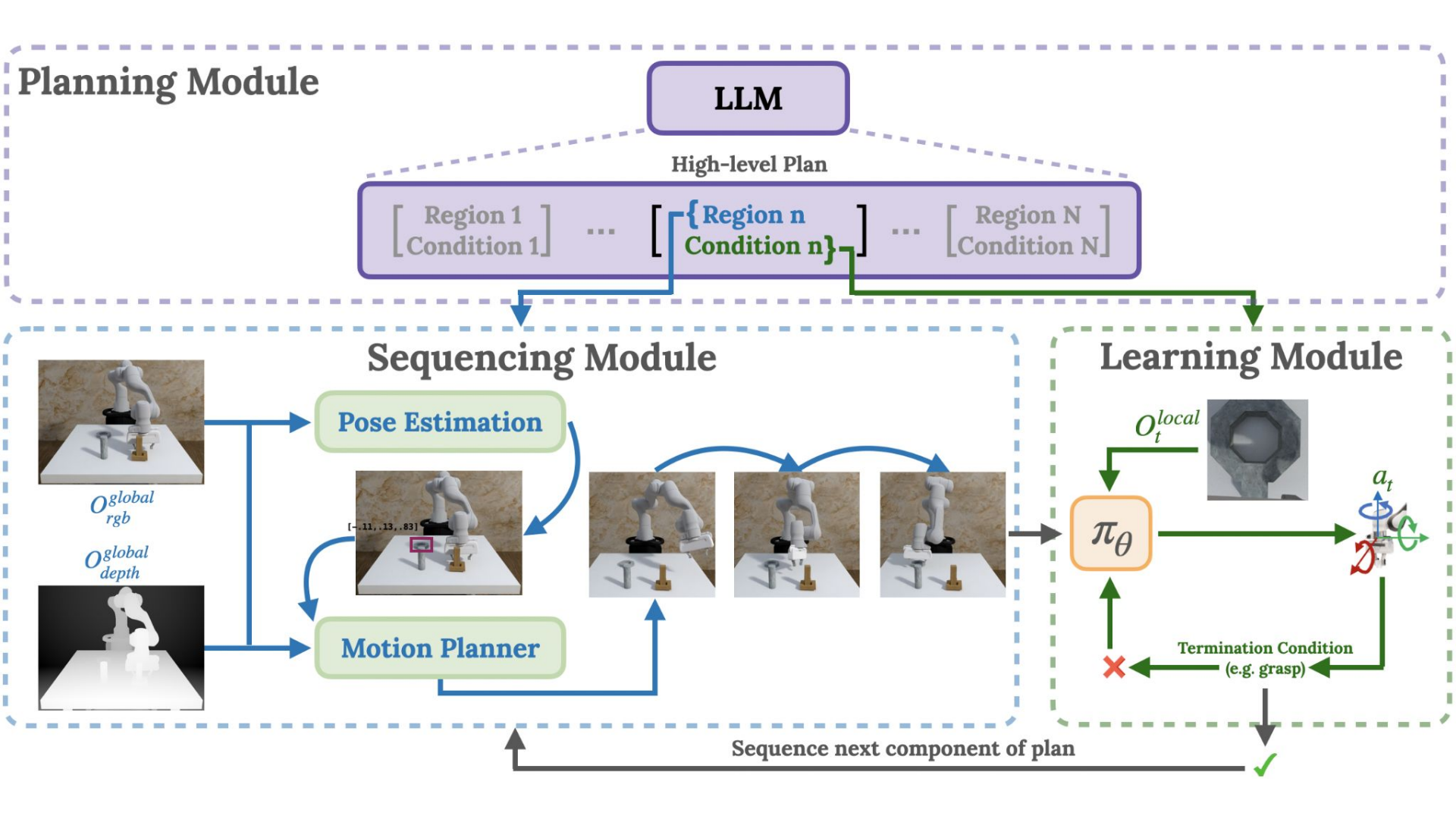
LLMs can reason but can't move. RL can move but can't reason over long horizons.
PSL bridges them with motion planning — no pre-built skills needed.

25+ tasks

up to 10 stages

85%+ success

visual input

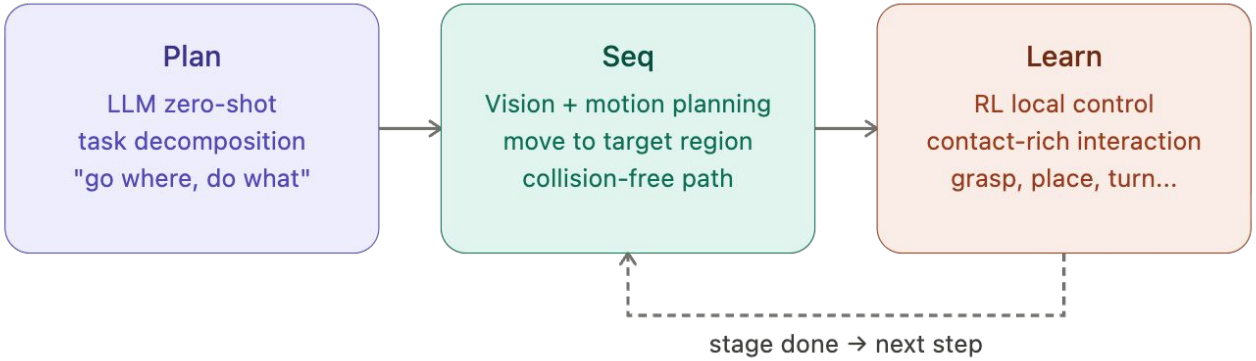

Planning Module
LLM
High-level Plan
Region 1
Condition 1
Region n
Condition n
Region N
Condition N
Sequencing Module
Pose Estimation
Motion Planner
Learning Module
Termination Condition
(e.g. grasp)
Sequence next component of plan

So how does PSL actually work?

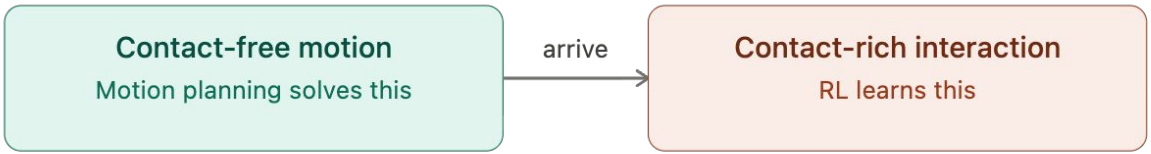
We'll walk through each module in detail.

PSL: three modules, one pipeline

Each module handles what it's best at



Why this decomposition works

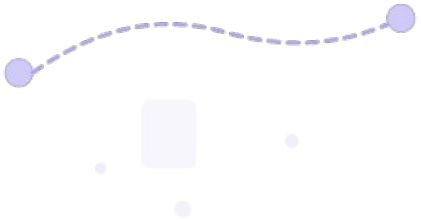


Result: RL only needs to learn short-horizon manipulation

No more exploring a 10-step task from scratch

The core insight: decompose by contact

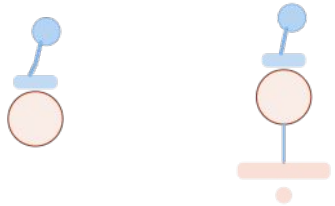
Contact-free motion



No object contact
Solved by AIT* planner

arrive

Contact-rich interaction



Continuous surface contact
Learned by RL online

Motion planning handles the "where"
SAM + depth + IK + AIT*

RL handles the "how"
DRQ-v2 + wrist camera

Given a task description, GPT-4 produces a structured plan zero-shot.
Each step is a (target region, termination condition) pair
that the Sequencing Module can directly interpret.

Only GPT-4 produced correct plans across all tasks.

Input to GPT-4

```
Stage termination conditions: (grasp, place)
Task: Gold nut on gold peg, silver nut on silver peg.
Give me a plan using only the conditions above.
Format: [("region", "condition"), ...]
```

GPT-4 output (temperature=0)

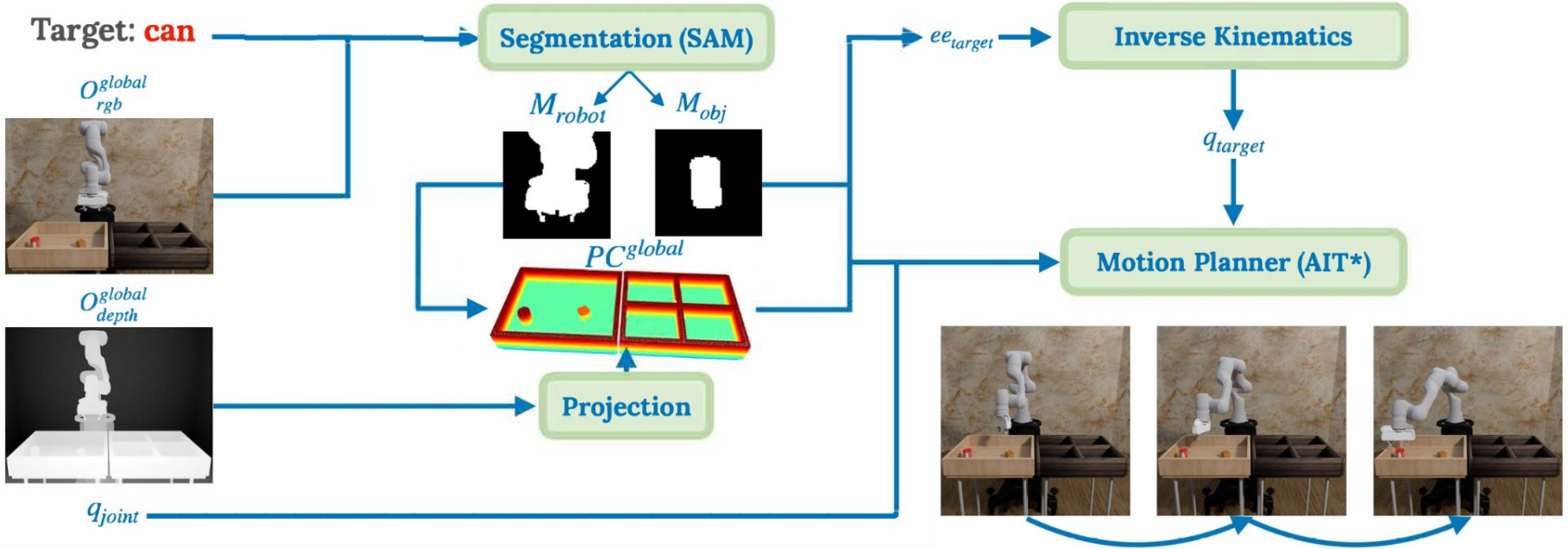
gold nut : grasp → gold peg : place → silver nut : grasp → silver peg : place

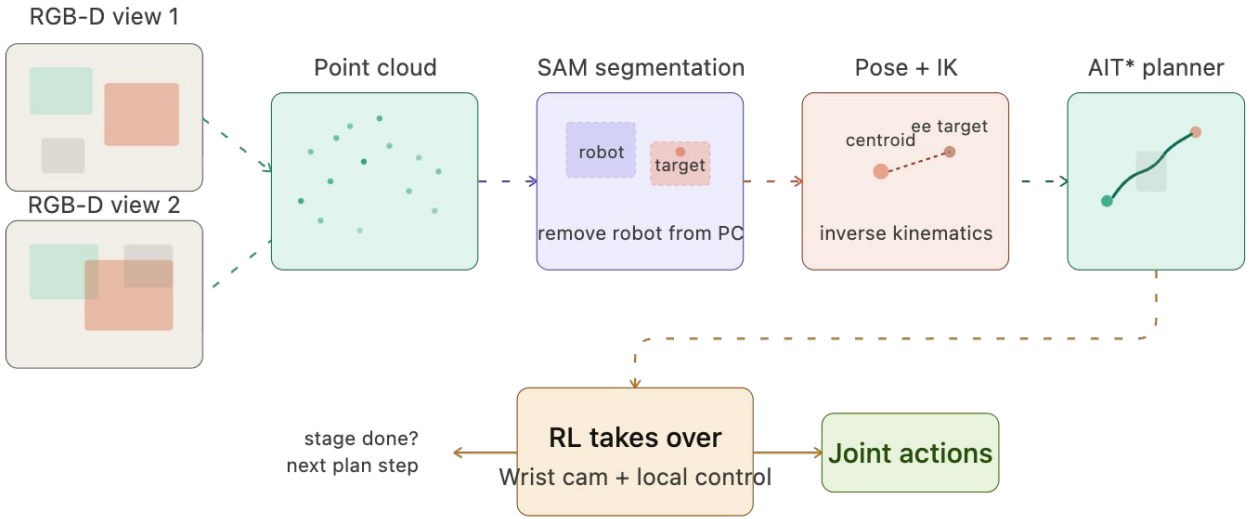
Only GPT-4 produced correct plans across all tasks. LLaMA and GPT-3 failed on multi-step reasoning. Each step is a (target region, termination condition) pair — directly interpretable by the Sequencing Module. Hallucinated objects or conditions are automatically filtered out in a post-processing step.

The connective tissue between language and control.
SAM segments the target object, depth gives 3D position,
IK converts to joint angles, AIT* plans a collision-free path.

Entirely vision-based. No simulator access. No ground-truth poses.

Sequencing Module





Sequencing module: from language label to collision-free robot trajectory
All vision-based. No simulator access. No ground-truth poses.

1. Dual RGB-D input
2. Point cloud construction
3. SAM segmentation
4. Pose estimation + inverse kinematics
5. AIT* motion planning
6. RL handoff

RL handoff

The robot arrives near the target. Control transfers to the RL policy, which uses only local wrist camera images. It has 25 steps max to complete the interaction (grasp/place/turn). Stage termination conditions trigger the next cycle.

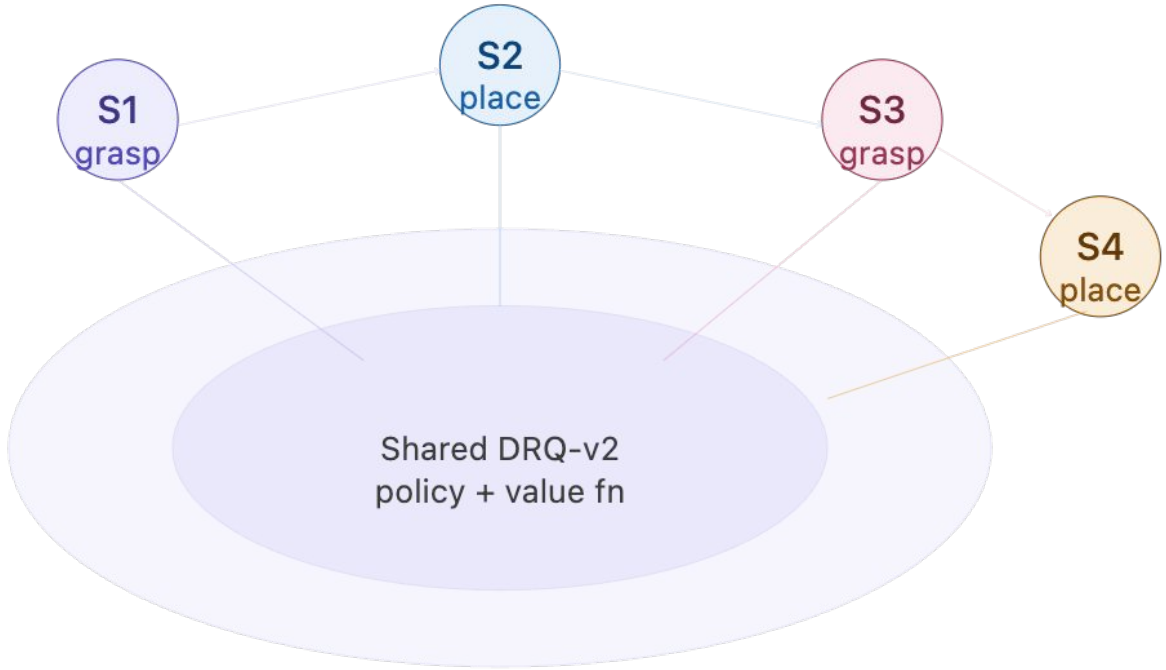
Once near the target, RL takes over for contact-rich interaction.
Three design choices make learning efficient:

Shared policy

Wrist camera

Stage termination

One policy serves all stages



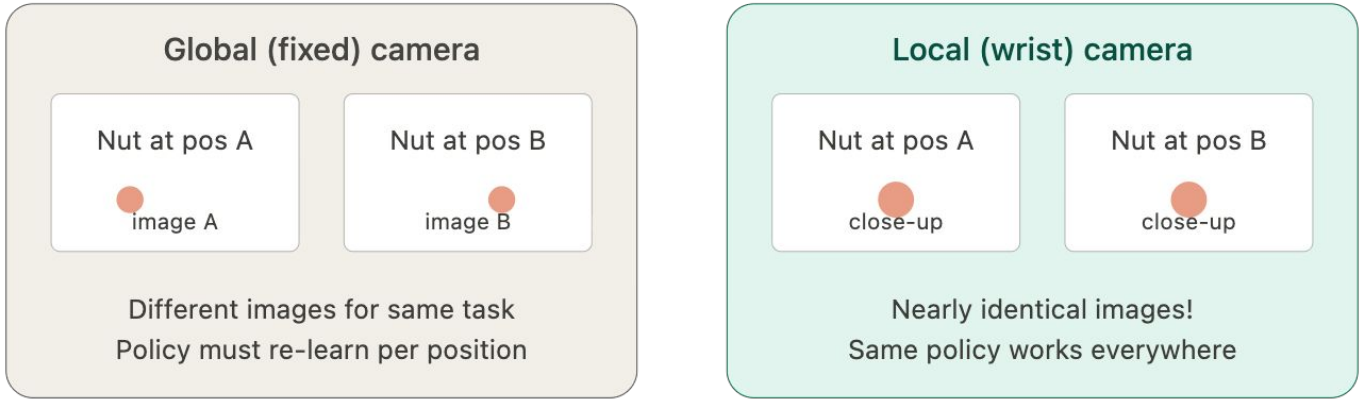
1 reward function

Future-aware V(s)

Adapts to Seq errors

Why local observation is the secret weapon

Wrist camera vs. fixed camera — 4x faster learning, enables policy chaining



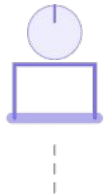
Policy chaining across stages

Wrist camera: chain works (100%) | Fixed camera: chain fails (0%)

Fixed views go out-of-distribution as the scene changes between stages

Stage termination: vision-based detection

Grasp



z increased
+ gripper caging

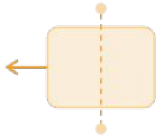
Place

Object near target
pose threshold

Turn

Knob principal axis
angle from vertical

Push/Open



Displacement or
handle angle

The curriculum effect

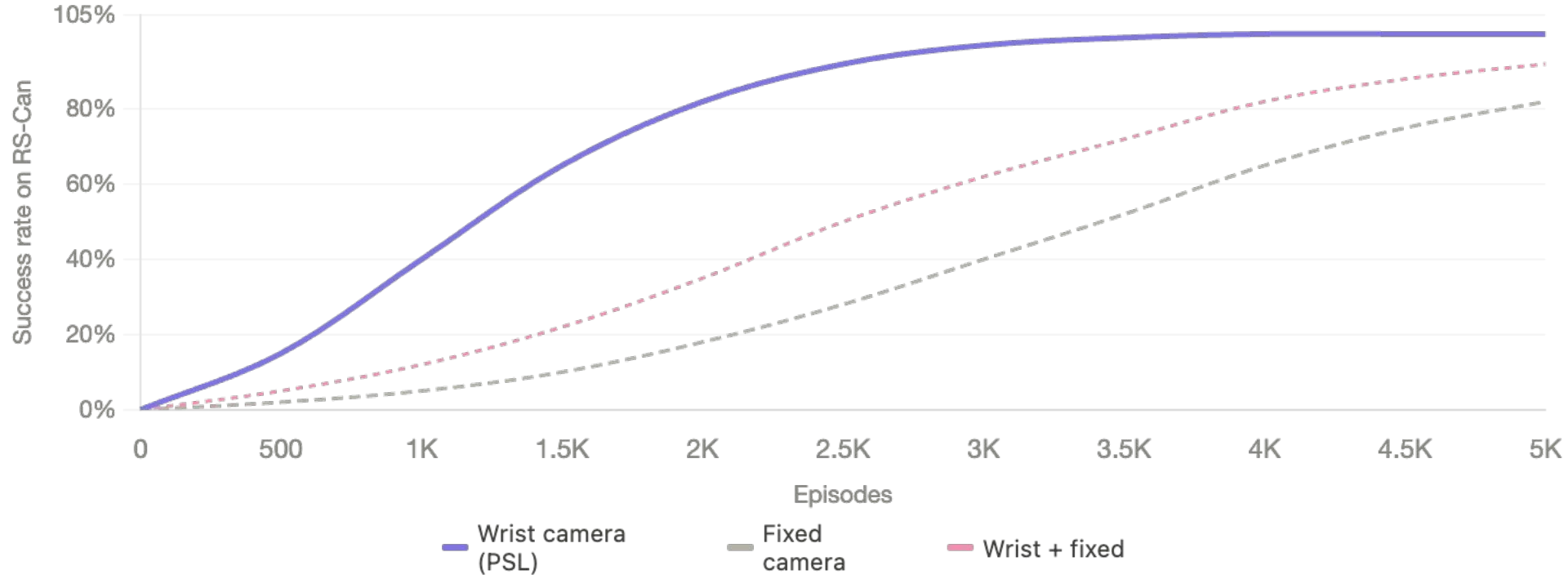
69%
Timeout only

→

100%
With checks

+31% on RS-Can
Prevents dithering behavior

All checks use vision (SAM + depth). Same conditions across all benchmarks.
Advance only when current stage truly succeeds = implicit curriculum.



4x
Wrist vs. fixed speed

100%
Wrist: 5-stage chain

0%
Fixed: 5-stage chain

Baselines: what PSL competes against

Learning methods

E2E (DRQ-v2)

SOTA model-free visual RL.
Learns everything from scratch.

RAPS

Hierarchical RL with engineered
action primitives (pick, place...).

MoPA-RL

RL decides when and where
to call motion planner.

Planning methods

TAMP

Classical task + motion planning.
Privileged simulator access.

SayCan

LLM plans over pre-built skills.
Open-loop, cascading failures.

PSL's advantage

Combines LLM planning (SayCan) + motion planning (TAMP) + online RL (E2E) without pre-built skills (RAPS)
First method to integrate all three paradigms into one framework

	Planning	Sequencing	Control	Online?
E2E	None	None	RL (DRQ-v2)	○
RAPS	None	None	Primitives	○
TAMP	Task planner	Motion plan	Primitives	○
SayCan	LLM	Skill selection	Pre-trained	○
PSL	LLM	Vision + MP	RL (online)	○

Purple = has capability

Four benchmarks, 25+ tasks, up to 10 stages

Robosuite

Lift · Door · Can Milk Bread · NutAsm 4 stages · Cereal Milk · CanBr

10 tasks, realistic OSC control
Contact-rich + pick-place

Kitchen

Micro · Slide · Light · Burn · Kettle · MS-10 10 stages!

12 tasks, sparse rewards
Up to 10 stages chained

Meta-World

Hammer · Assem. · Disasm · BinPick

4 tasks, shaped rewards, EE control

Obstructed suite

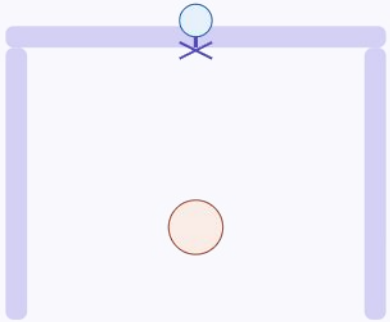
3 tasks, obstacles in workspace

Circle size = relative task complexity. PSL achieves 85%+ on every task.

	PSL	SayCan	TAMP	E2E	RAPS
Single-stage					
RS-Lift / Door	100	--	--	70	21
K-Microwave	100	--	--	12	14
K-Burner / Light	100	--	--	60	30
OS tasks (avg)	100	--	--	7	--
Two-stage					
RS PickPlace (avg)	100	87	94	22	0
RS NutRound/Sq	98	42	38	4	0
MW tasks (avg)	100	83	67	47	43
Multi-stage (3-10)					
RS 4-stage (avg)	90	53	54	0	0
RS NutAssembly	96	23	20	0	0
K-MS-3	100	100	100	0	89
K-MS-5 / 7 / 10	100	0	0	0	0

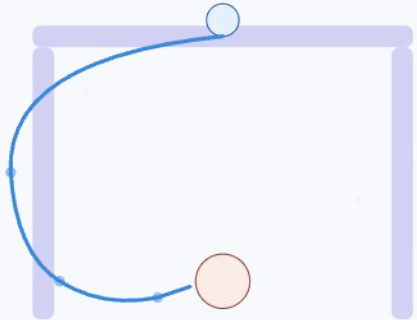
99	55	53	20	20
PSL	SayCan	TAMP	E2E	RAPS

E2E 0%



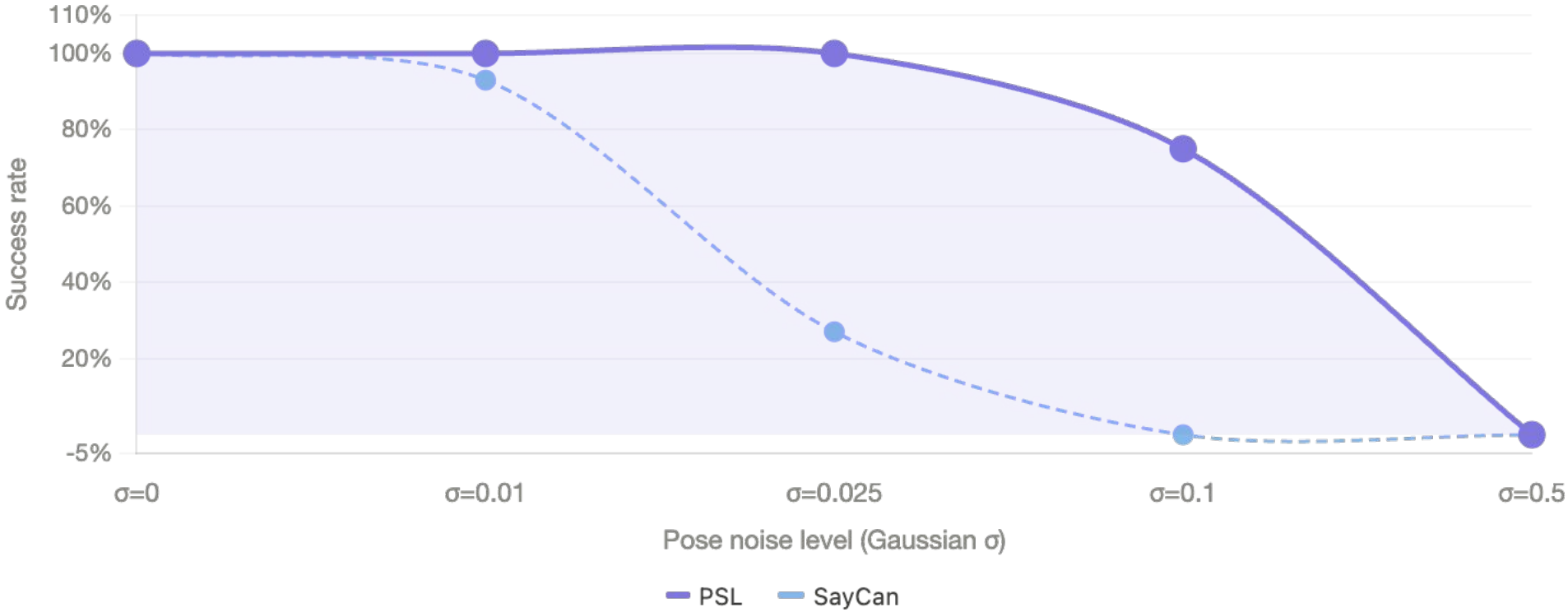
Arm starts above the box walls.
Direct path downward collides with the rim.
No mechanism to plan around obstacles.
Spends all episodes failing to reach the object.

PSL 100%

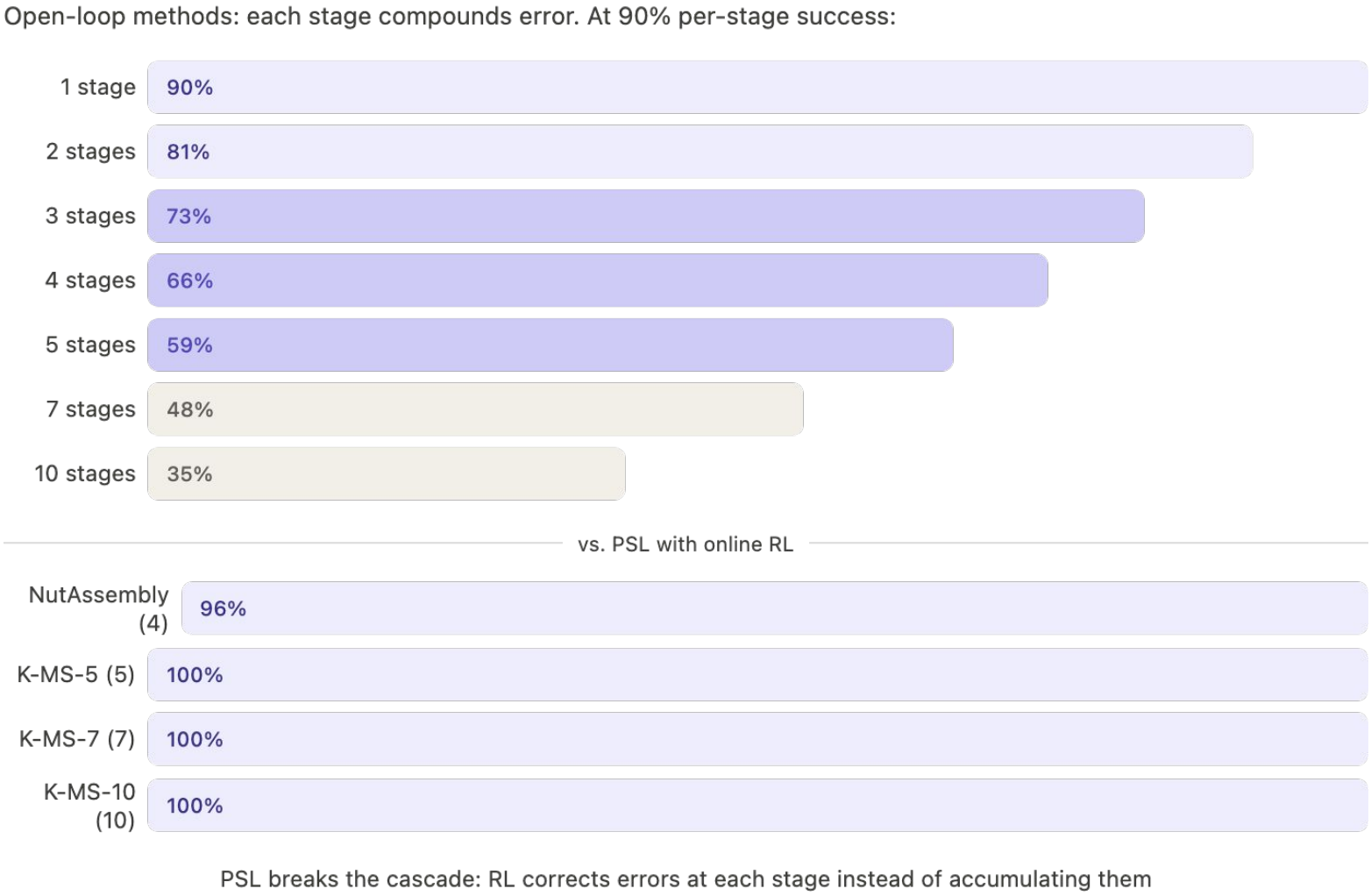


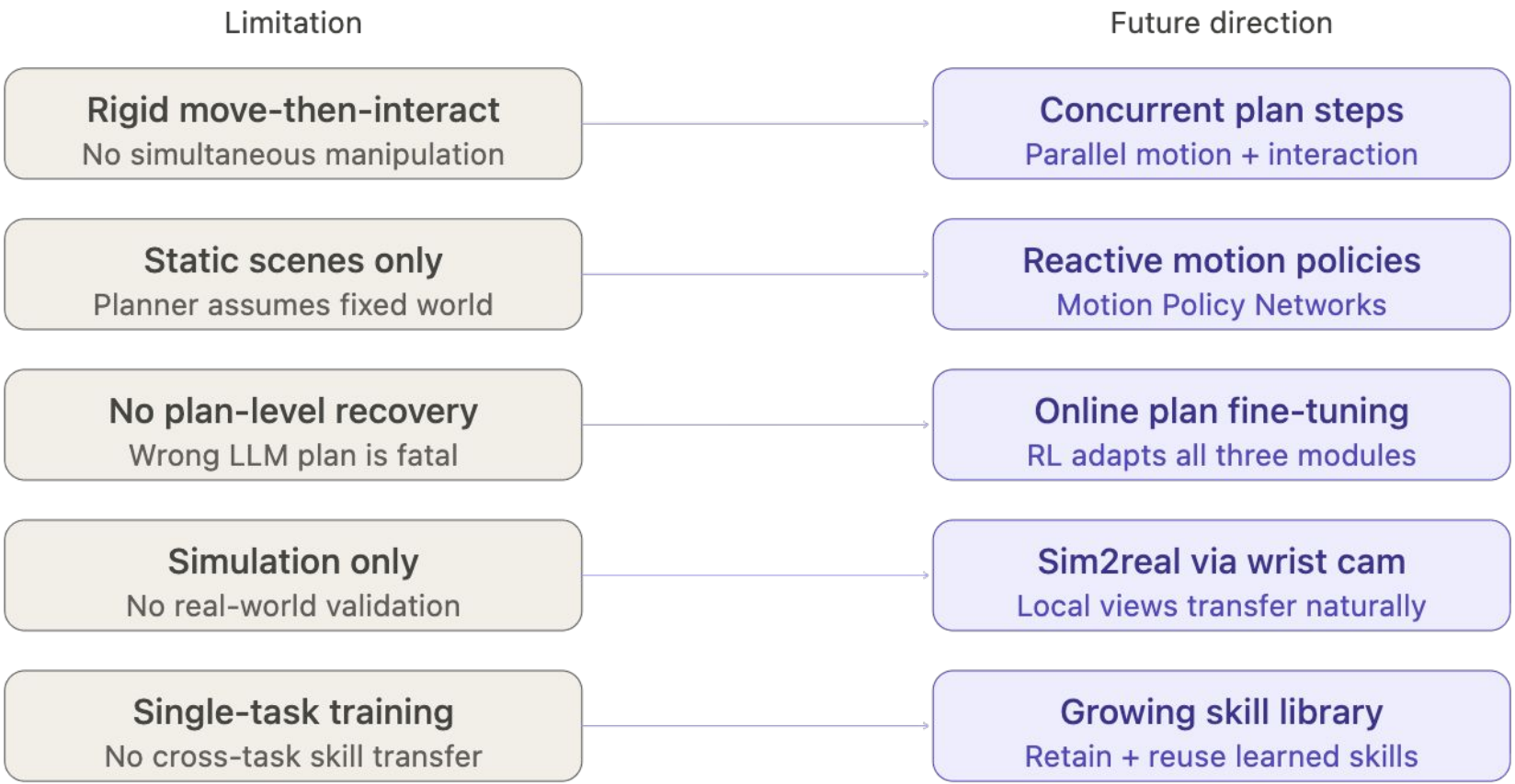
SAM segments the scene into point cloud.
AIT* computes a collision-free trajectory around the walls to reach the target object.
RL only needs to learn the short grasp.

PSL 100% on all three obstructed tasks. Motion planning handles avoidance implicitly.

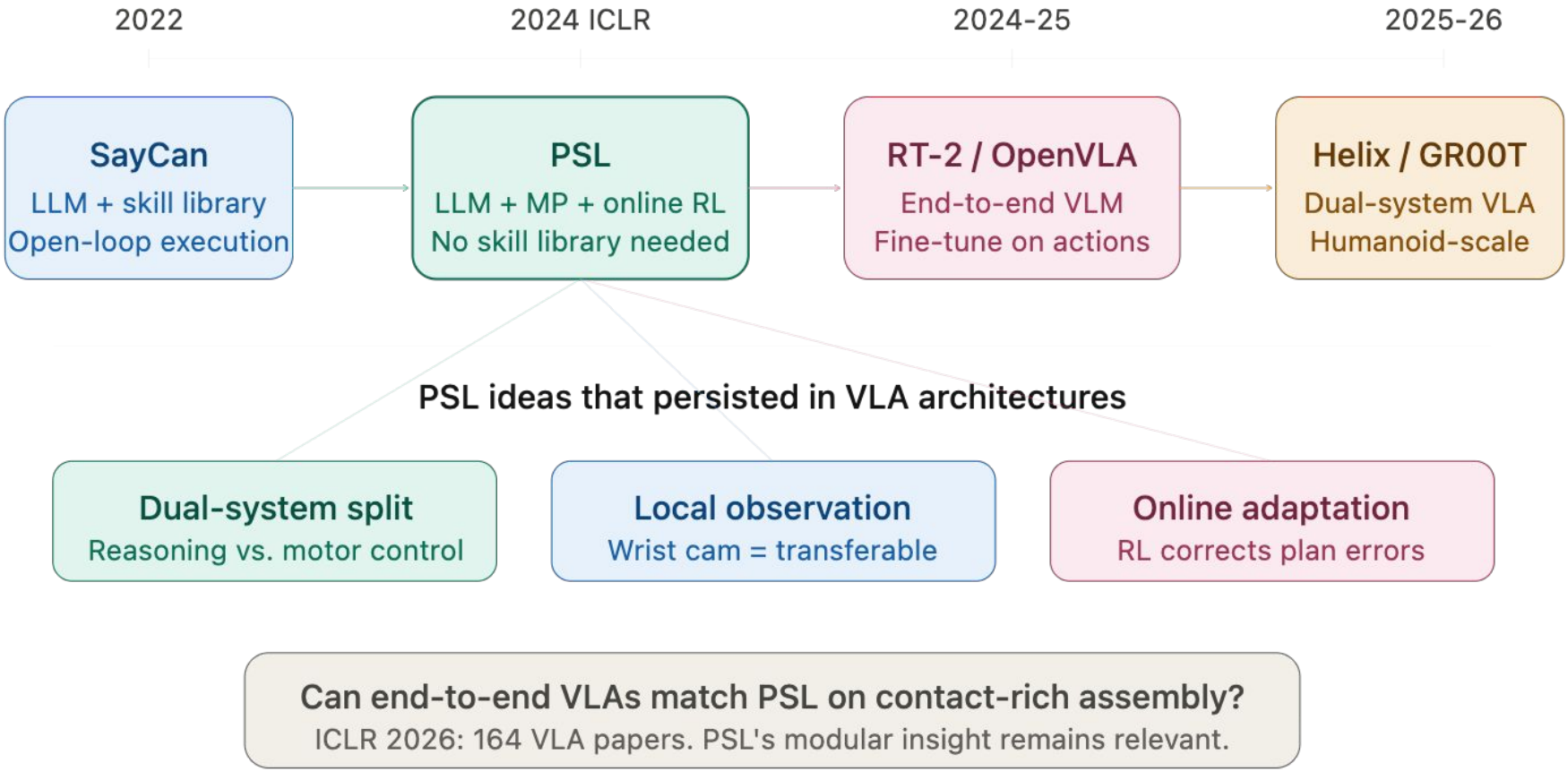


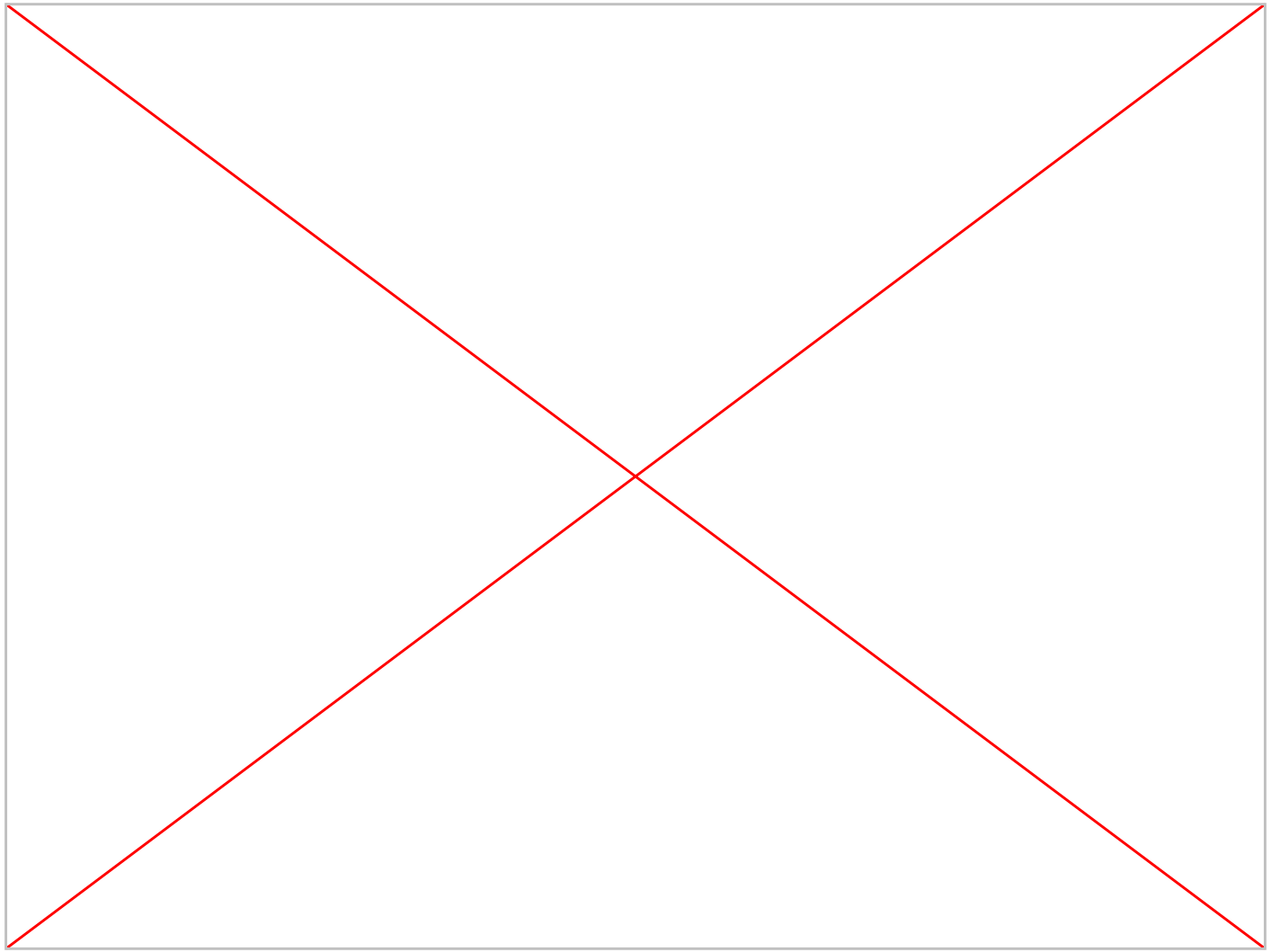
PSL tolerates 10x more pose noise than SayCan (open-loop).
Online RL compensates for imprecise initialization.


Open-loop methods: each stage compounds error. At 90% per-stage success:
1 stage
90%
2 stages
81%
3 stages
73%
4 stages
66%
5 stages
59%
7 stages
48%
10 stages
35%
vs. PSL with online RL
NutAssembly (4)
96%
K-MS-5 (5)
100%
K-MS-7 (7)
100%
K-MS-10 (10)
100%
PSL breaks the cascade: RL corrects errors at each stage instead of accumulating them



PSL's modular design enables each component to be upgraded independently


2022
2024 ICLR
2024-25
2025-26
SayCan
LLM + skill library
Open-loop execution
PSL
LLM + MP + online RL
No skill library needed
RT-2 / OpenVLA
End-to-end VLM
Fine-tune on actions
Helix / GR00T
Dual-system VLA
Humanoid-scale
PSL ideas that persisted in VLA architectures
Dual-system split
Reasoning vs. motor control
Local observation
Wrist cam = transferable
Online adaptation
RL corrects plan errors
Can end-to-end VLAs match PSL on contact-rich assembly?
ICLR 2026: 164 VLA papers. PSL's modular insight remains relevant.



THANK YOU!

Q&A SESSION

D PSL IMPLEMENTATION DETAILS

Algorithm 2 PSL Implementation

Require: LLM, task description g_l, Motion Planner MP, low-level horizon H_l, segmentation model $\mathcal{S}$, RGB-D global cameras, RGB wrist camera, Camera Matrix K^{global}

initialize RL: π_θ, replay buffer $\mathcal{R}$
Planning Module
High-level plan $\mathcal{P} \leftarrow$ Prompt(LLM, g_l)
for episode $1 \ldots N$ **do**
  for $p \in \mathcal{P}$ **do**
Sequencing Module
    target region (t), termination condition $\leftarrow p$
    PC^{global} = Projection(O_1^{global}, O_2^{global}, K^{global})
    M_{robot}, M_{obj} = Segmentation(O_1^{global}, O_2^{global}, robot, object)
    PC^{robot}, PC^{object} = $M_{robot} * PC^{global}$, $M_{obj} * PC^{scene}$
    $PC^{scene} = PC^{global} - PC^{robot}$
    ee_{target} = mean(PC^{obj})
    q_{target} = IK(ee_{target})
    MotionPlan(MP, q_{target}, PC^{scene})
Learning Module
    for $i = 1, ..., h$ low-level steps **do**
      Get action $a_t \sim \pi_\theta(O_t^{local})$
      Get next state $O_{t+1}^{local} \sim p(|s_t, a_t)$.
      Store $(O_t^{local}, a_t, O_{t+1}^{local}, r)$ into $\mathcal{R}$
      Sample $(O_k^{local}, a_t, O_{k+1}^{local}, r) \sim \mathcal{R}$ ▷ k = random index
      update π_θ using RL
      if post-condition **then**
        break
      end if
    end for
  end for
end for