上海科技大学
ShanghaiTech University

CS289: **Mobile Manipulation Fall 2024**

Sören Schwertfeger

ShanghaiTech University

Final

- Dec 3rd – next Tuesday!
  - 15:00 – 17:00 in 1D-106
- Content:
  - All lectures
    - Take a look at facts, algorithms, concepts
- You are allowed to bring **3** A4 sheets (so 6 pages) of info to the exams. You can write/ print anything on those sheets. On top of **every page** (so 6 times) there needs to be your **name (pinyin), student ID and ShanghaiTech email** address. We will check every cheat sheet before the exam and **confiscate** every sheet without name or with a name that is not yours.
- No electronics/ calculator/ smartwatch allowed

Project meetings

- Another meeting next week Thursday or Friday (see wechat!)
- Another meeting Monday or Tuesday Dec 16, 17
- Another meeting Monday or Tuesday January 6, 7

Project conclusion

- Due January 10 – could be extended till January 17 if needed – need to write me an email!
  - Project Report:
    - Similar to proposal
    - With nice results and proper quantitative evaluation
    - Make look like a scientific paper
    - Use bibtex!
    - Put into git (folder: doc/final)
      - Include everything that is needed to generate the report in the git!
        - So don't forget images/ the bib file

  - Project Demo:
    - Make an appointment with Prof. Schwertfeger to show the final demo of your project
    - Before: January 17 !

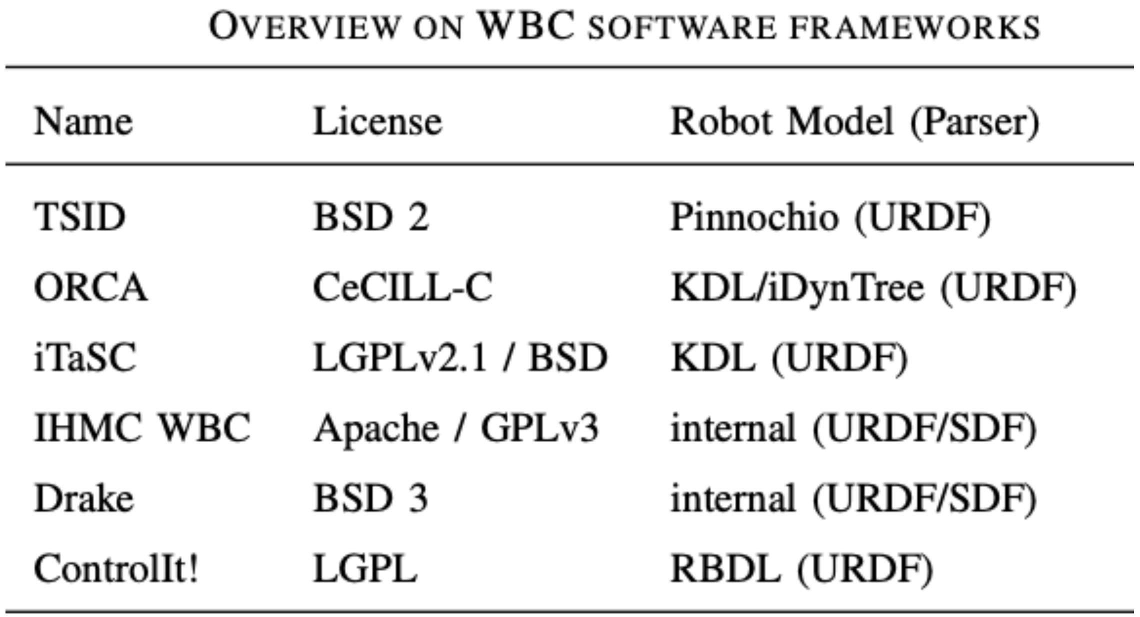
Project Webpage

- Write a text (word document) about your project for the general public – not too technical – not too many details
  - Some details can be written
  - Do not just copy the abstract/ intro from your final report – write a nice text for the general public!
- Provide a few images with captions (as images also extra files)
- Put into your group git (folder: doc/webpage)
- Prof. Schwertfeger will upload the data to the website – e.g. look at : https://robotics.shanghaitech.edu.cn/teaching/robotics2020 (hopefully up soon…)
- Also make a NICE video about your project. 4-8 minutes. Leave the video at good quality – size maybe 100 – 300 MB (MP4) – Prof. Schwertfeger will compress it to make a web version
  - Avoid showing other people; do not talk in the video; do not add music;
  - Add a title page: same info as on report
  - Add to your git folder

WHOLE-BODY CONTROL

Whole-Body Control

- Plan & control for combined motion of manipulator and mobile base
- Particular popular for legged, especially humanoid robots
  - Tree-like kinematic structure – no loops!
- Also needed for aerial, underwater, surface vehicles and space robots:
  - Manipulation forces move the mobile base!
- Ground vehicles: non-holonomic kinematics restricts possible motions -> difficult and unpopular
  - Alternative: holonomic ground robots!
- MPC popular
- Reinforcement Learning very popular


OVERVIEW ON WBC SOFTWARE FRAMEWORKS

Name	License	Robot Model (Parser)
TSID	BSD 2	Pinnochio (URDF)
ORCA	CeCILL-C	KDL/iDynTree (URDF)
iTaSC	LGPLv2.1 / BSD	KDL (URDF)
IHMC WBC	Apache / GPLv3	internal (URDF/SDF)
Drake	BSD 3	internal (URDF/SDF)
ControlIt!	LGPL	RBDL (URDF)

Mronga, D., Kumar, S., & Kirchner, F. (2022, May). Whole-body control of series-parallel hybrid robots. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 228-234). IEEE.

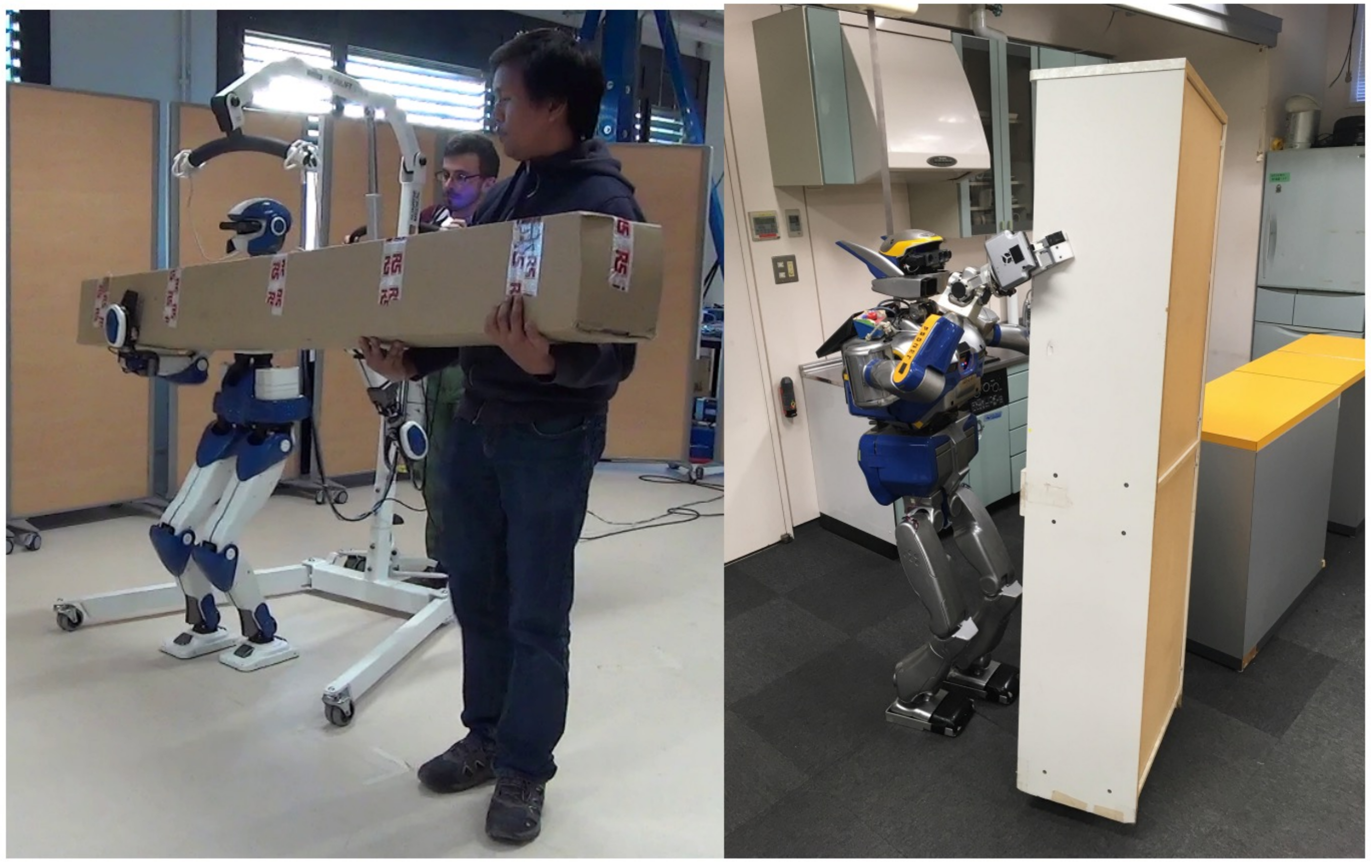
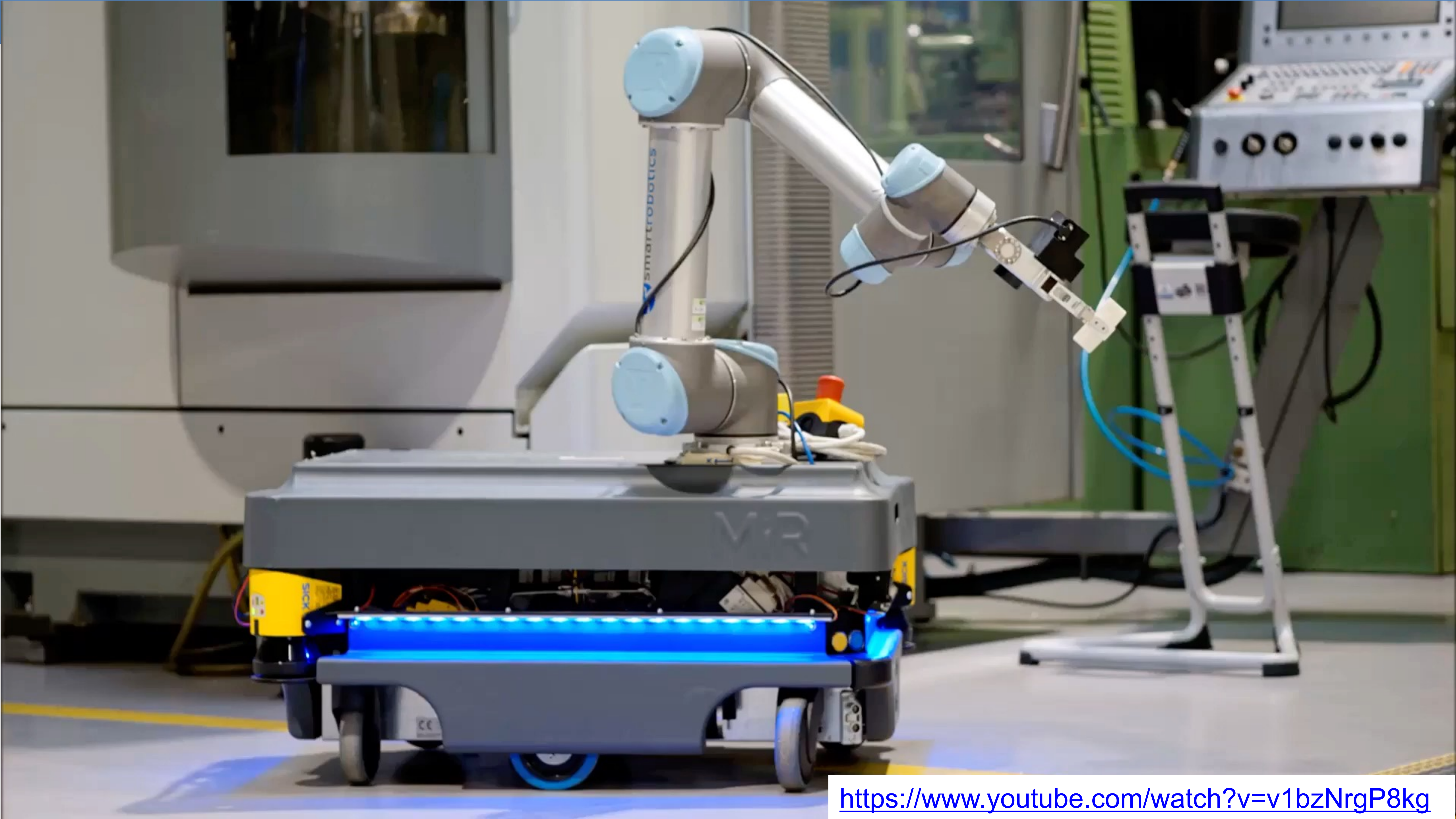
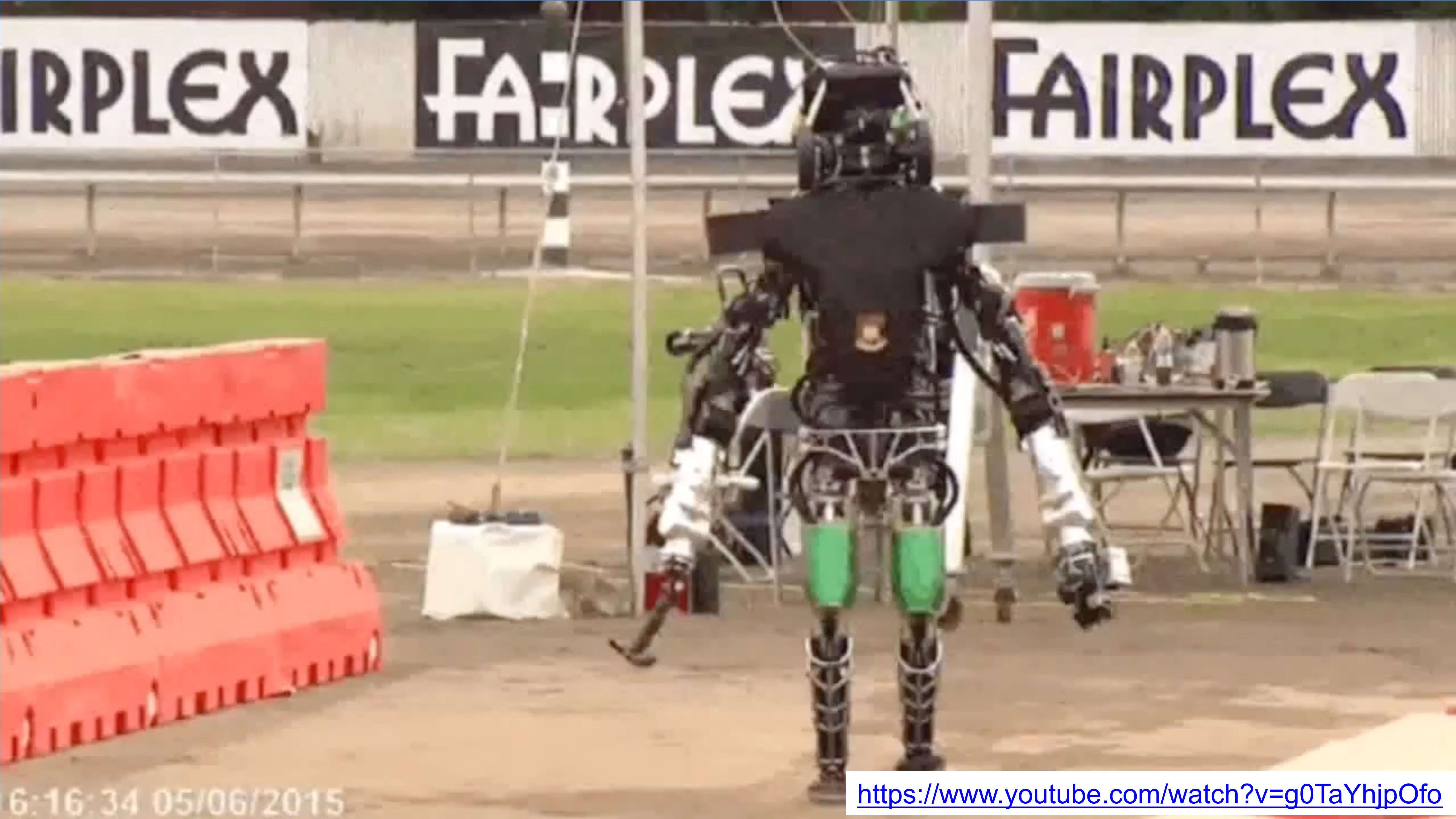


Fig. 3: (left) HRP-4 holding a large box with a human while walking (Agravante et al. (2019)) (right) HRP-2 pivoting a furniture (Murooka et al. (2017)).

Stasse, O., & Righetti, L. (2020). Whole-body manipulation. *Encyclopedia of Robotics*, 1-9.

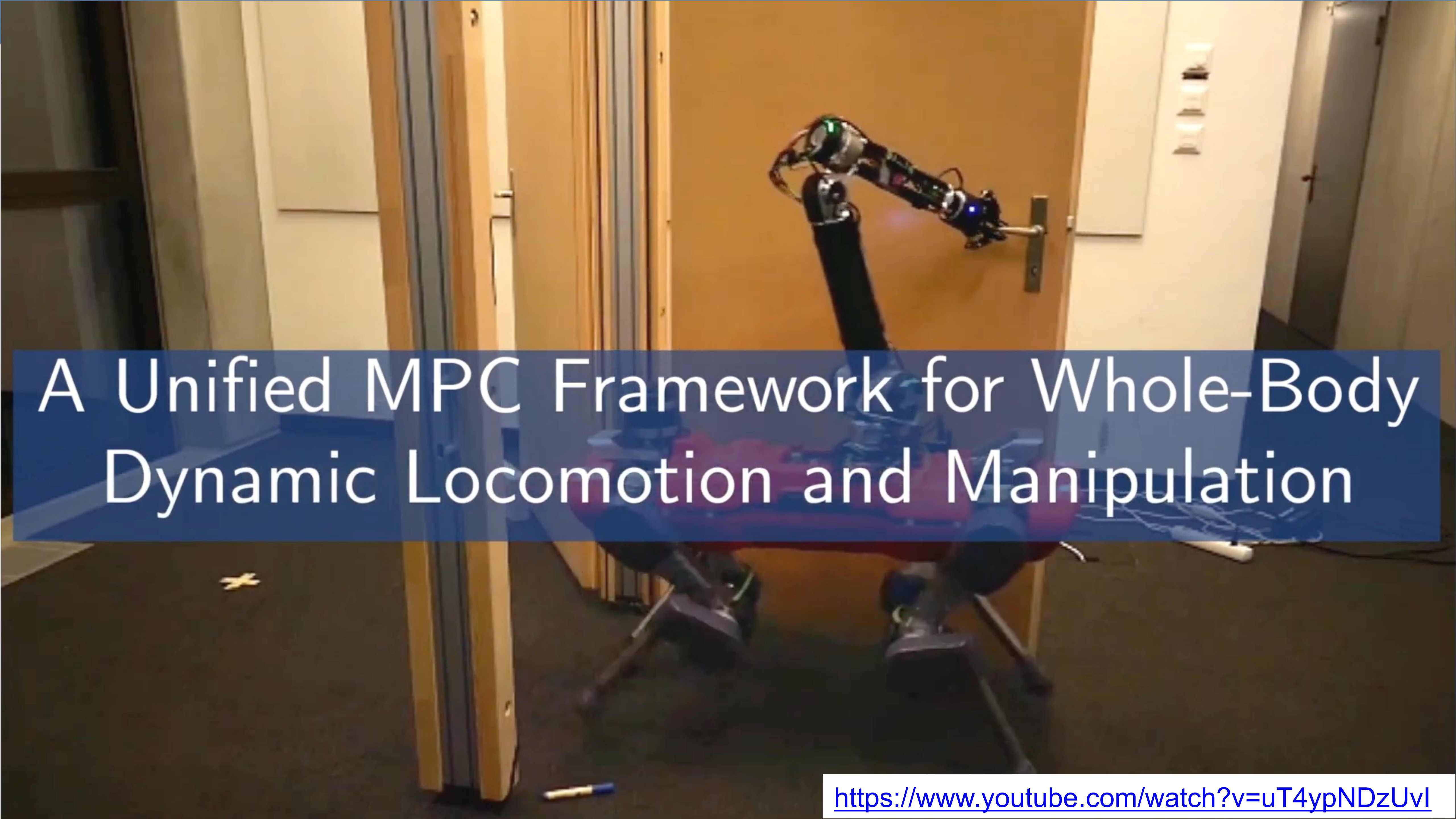


https://www.youtube.com/watch?v=v1bzNrgP8kg



https://www.youtube.com/watch?v=g0TaYhjpOfo

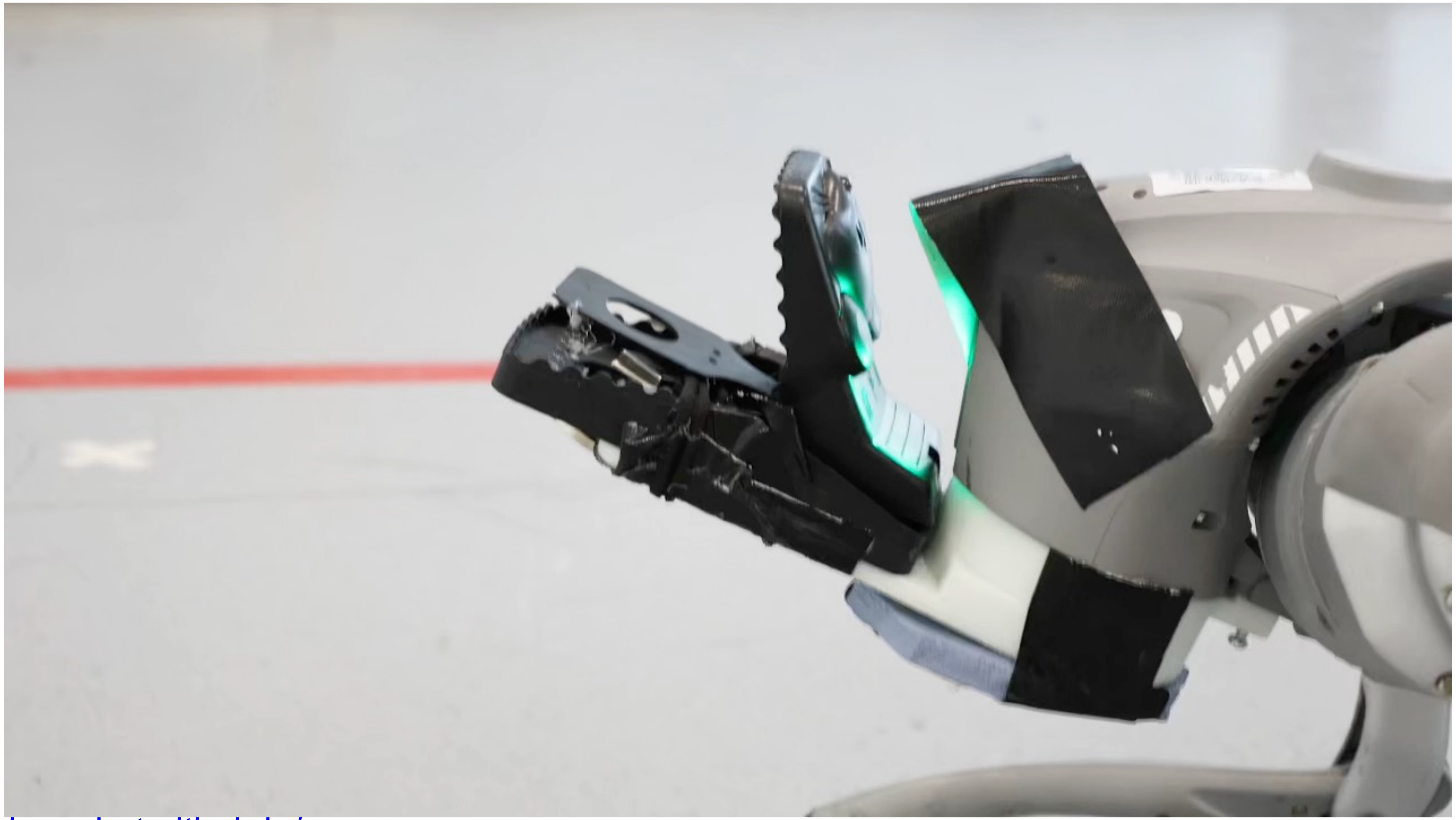
https://www.youtube.com/watch?v=QXS1gBGc23A



A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation

https://www.youtube.com/watch?v=uT4ypNDzUvI

Playful DoggyBot: Learning Agile and Precise Quadrupedal Locomotion



https://playful-doggybot.github.io/

ROBOTIC LEARNING

Overview of Learning Approaches in Robotics

- **Goal:** To explore various learning paradigms that enable robots to perform tasks autonomously.
- **Categories:**
  - Model-Free vs. Model-Based Learning
  - Supervised vs. Unsupervised Learning
  - Passive vs. Active Learning
  - Reinforcement Learning (RL)
  - Imitation Learning
  - End-to-End Deep Learning
  - Actor-Critic Learning
  - Evolutionary Algorithms
  - Transfer Learning
  - Self-Supervised Learning
  - Few-Shot and Zero-Shot Learning
  - Multi-Agent Learning
  - Curriculum Learning
  - LLM
  - Foundation Models
  - Other types of “learning”

Next few slides
With help from ChatGPT :D

Model-Based vs Model-Free Learning

Model-Based Learning:

- Involves learning a model of the environment or dynamics (e.g., using physics or system dynamics).
- Robot can plan and predict actions based on this model.
- **Example:** Planning with a learned dynamics model in robotic control tasks.

Model-Free Learning:

- Directly learns a mapping from states to actions or rewards without modeling the environment.
- **Example**: Q-learning or policy gradient methods in Reinforcement Learning.

Supervised vs. Unsupervised Learning

Supervised Learning:

- Learning from labeled data (input-output pairs).
- Requires large datasets and human supervision.
- **Example:** Image classification for object detection in robotics, such as recognizing "graspable" objects in a scene.

Unsupervised Learning:

- Learning from unlabeled data to find hidden patterns (e.g., clustering or representation learning).
- **Example:** A robot exploring its environment autonomously to cluster sensory data (e.g., LIDAR or visual data) into distinct regions like walls, furniture, or open spaces. This clustering can later help the robot map its environment for navigation.

Passive vs Active Learning

Passive Learning:

- The robot learns from a fixed dataset (either labeled or unlabeled).
- **Example:** Supervised learning with a fixed dataset.

Active Learning:

- The robot queries the environment for more informative data based on its current knowledge or uncertainty.
- **Example:** A robot selects which objects to interact with in order to maximize learning.

End-to-End Deep Learning in Robotics

- **Definition:** Learning a direct mapping from raw input (e.g., images, sensory data) to the output (e.g., control commands).
- **Example:** A robot controlling a gripper using only camera images.
- **Advantages:** Simplifies the pipeline by learning a direct mapping.
- **Challenges:** Requires large amounts of labeled data.

Reinforcement Learning (RL)

- **Definition:** An agent learns to take actions in an environment to maximize cumulative reward over time.
- **Key Components:** States, actions, rewards, policy.
- **Example:** Training a robot to navigate using trial-and-error.
- **Types:**
  - **Model-Free:** Methods like Q-learning, policy gradients.
  - **Model-Based:** Use of learned models to simulate and plan actions.

Imitation Learning

- **Definition:** Robots learn by observing and imitating human demonstrations or expert behaviors.

- **Approaches:**
  - **Behavior Cloning:** Supervised learning from demonstrations.
  - **Inverse Reinforcement Learning (IRL):** Learning the underlying reward function from expert demonstrations.

- **Example:** Teaching a robot to grasp objects by mimicking human actions.

Actor-Critic Learning

- **Definition:** A type of Reinforcement Learning where two models (actor and critic) are trained simultaneously.
  - **Actor:** Decides which action to take based on the current state.
  - **Critic:** Evaluates the chosen action based on the reward.
- **Advantages:** More stable training by using both policy and value functions.
- **Example:** Robotic manipulation tasks requiring fine-grained control.

Evolutionary Algorithms in Robotics

- **Definition:** Optimization methods inspired by natural selection, such as genetic algorithms or neuroevolution.
- **Example:** Optimizing robot locomotion for uneven terrains or designing neural network architectures for control.
- **Advantages:** Effective for tasks where gradient-based optimization struggles or is infeasible.
- **Challenges:** Computationally expensive and may require many iterations.

Transfer Learning in Robotics

- **Definition:** Leveraging knowledge from one domain/task to accelerate learning in another.
- **Example:** Transferring knowledge from simulation-trained robots to real-world environments.
- **Advantages:** Reduces training time and reliance on large datasets.
- **Challenges:** Ensuring the transfer is effective despite domain differences.

Self-Supervised Learning in Robotics

- **Definition:** Robots generate their own training signals from raw, unlabeled data.
- **Example:** Learning to predict the next sensory input (e.g., next video frame) for tasks like navigation or manipulation.
- **Advantages:** Removes dependence on human-labeled data, making learning scalable.
- **Challenges:** Designing effective self-supervised objectives.

Few-Shot and Zero-Shot Learning in Robotics

- **Definition:** Learning to generalize with few or no examples of the target task.
- **Example:** Teaching a robot to recognize and manipulate a novel object with only one image or no prior direct training.
- **Advantages:** Crucial for real-world scalability.
- **Challenges:** Relies heavily on robust pre-trained models.

Multi-Agent Learning in Robotics

- **Definition:** Learning strategies for robots that interact and collaborate or compete in shared environments.
- **Example:** Swarms of drones coordinating for mapping or delivery tasks.
- **Advantages:** Enables cooperative behaviors in teams of robots.
- **Challenges:** Complex interactions and scalability.

Curriculum Learning in Robotics

- **Definition:** Gradually increasing the complexity of tasks during training.
- **Example:** Training a robot arm to first stack one block, then multiple blocks, before solving general stacking problems.
- **Advantages:** Stabilizes and accelerates learning.
- **Challenges:** Designing a proper curriculum and transitions between tasks.

LLM for Robotics

- **Definition:** LLMs are AI systems trained on massive text corpora to process, understand, and generate human-like text.
- **Key Capabilities in Robotics:**
  - **Natural Language Understanding:** Interpreting commands and queries.
  - **Knowledge Integration:** Retrieving and applying knowledge to tasks (e.g., assembly instructions).
  - **Reasoning and Task Decomposition:** Breaking down complex instructions into actionable steps.
- **Advantages:**
  - Provides high-level reasoning and task planning.
  - Reduces the need for detailed programming in language-based tasks.
  - Can handle diverse instructions using pre-trained knowledge

How LLMs Are Used in Robotics

- **Applications:**
  - **Human-Robot Interaction:** Robots can interpret and execute natural language instructions (e.g., “Bring me a cup of water”).
  - **Task Planning:** Combining linguistic reasoning with real-world task execution.
  - **Multi-Modal Integration:** Enhancing decision-making by linking text, vision, and sensory inputs.
- **Challenges:**
  - Ensuring grounding in physical environments (e.g., interpreting "left" in a spatial context).
  - Real-time response constraints due to the size of models.
  - Domain-specific fine-tuning for robotics applications.

Robotics Foundation Models

- **Definition:** Large-scale AI models pre-trained on diverse, multi-modal datasets (e.g., text, images, videos).
- **Core Characteristics:**
  - **Versatile Pre-training:** Serve as a base for fine-tuning on specific tasks.
  - **Multi-Modal Understanding:** Integrate text, vision, and other sensory inputs for broader applicability.
- **Key Advantages for Robotics:**
  - Generalize across multiple tasks with minimal retraining.
  - Simplify the training pipeline by leveraging shared representations.
  - Adaptable to new tasks without extensive data collection.

How Foundation Models Empower Robotics

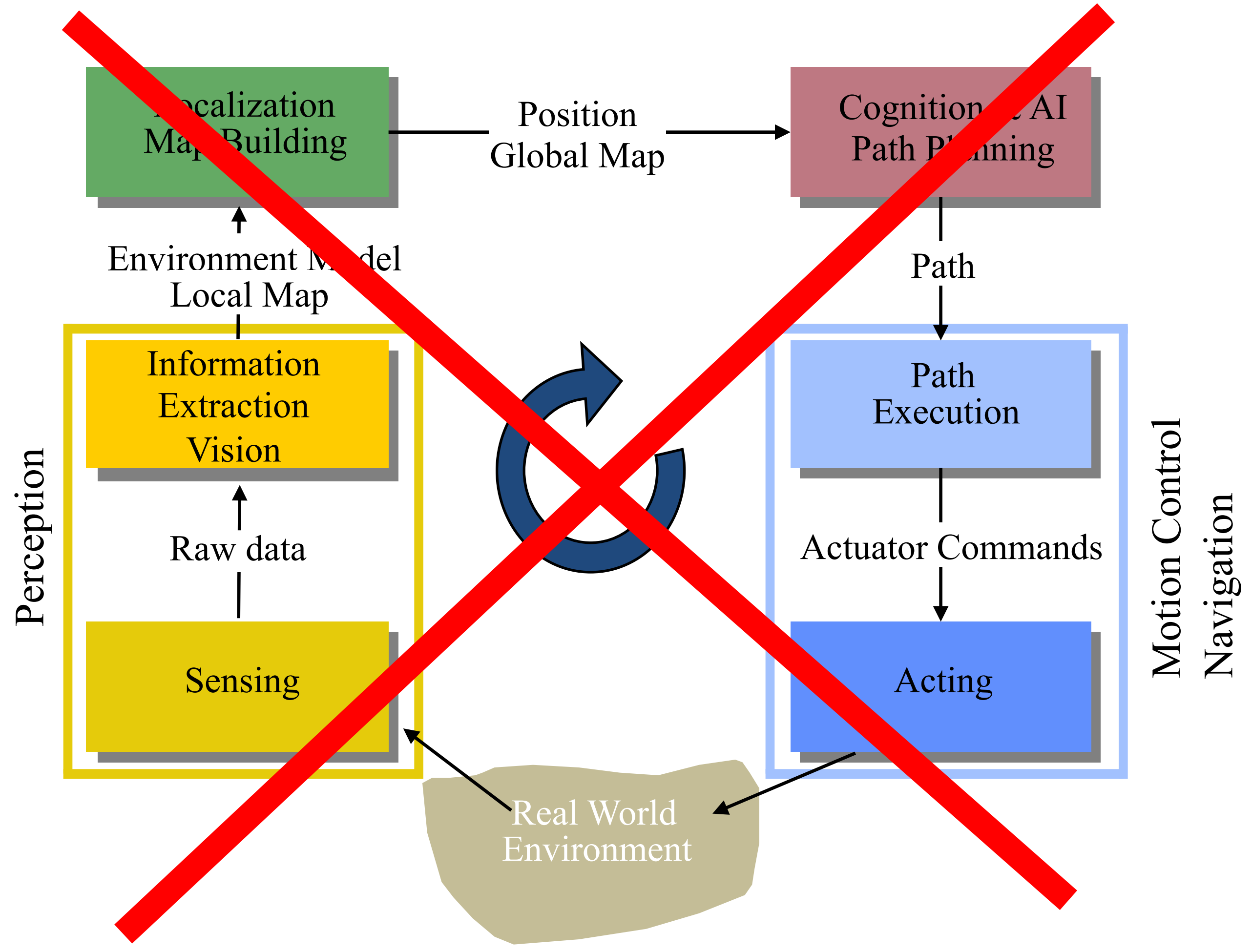
- **Applications:**
  - **Perception:** Models like CLIP interpret visual data for scene understanding.
  - **Control:** Leveraging shared representations for motion planning and actuation.
  - **Task Generalization:** Performing varied tasks without task-specific training.
  - **Simulation-to-Real Transfer:** Reducing the gap between simulated and real-world performance.
- **Challenges:**
  - High computational costs for pre-training and fine-tuning.
  - Limited grounding in physical dynamics without additional modeling.
  - Potential biases from pre-training on non-robotic data.

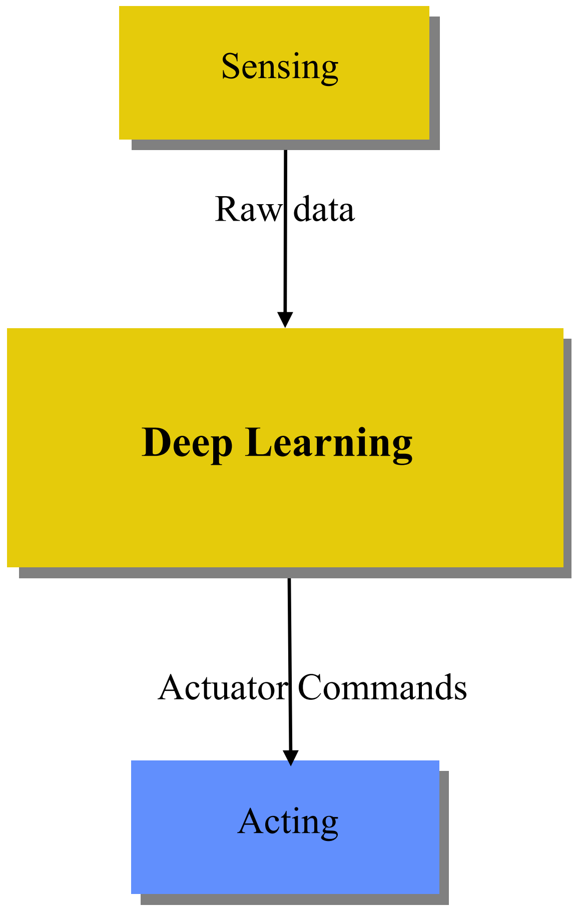
Other types of learning

- Term “learning” also used in different contexts in robotics, e.g.
- SLAM: the robot “learns” the map (by SLAM algorithm)
- Adaptive Control Learning
- Motion Planning
- Meta-learning: learn to learn
- Lifelong Learning (Continuous Learning)
- …

REINFORCEMENT LEARNING

End-to-End Deep Learning



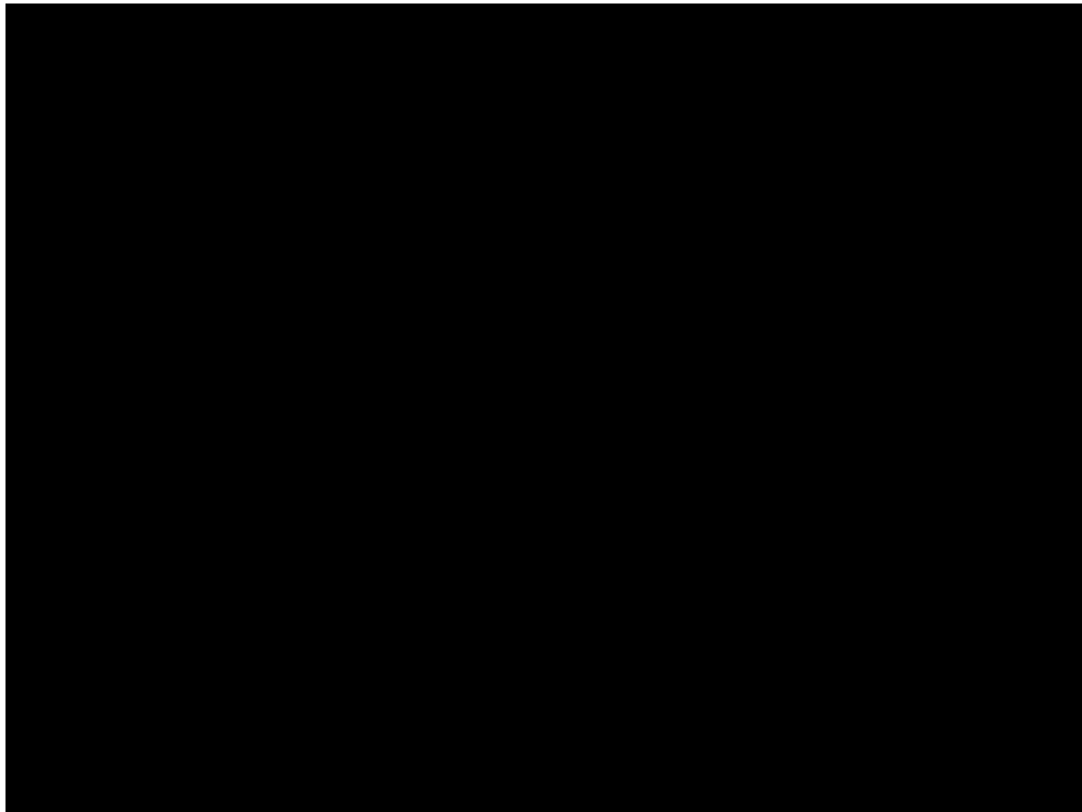


End-to-End Deep Reinforcement Learning

- From sensors to actuation: one layered or recurrent neural network! =>
  - NOT classical general control scheme (Perception, SLAM, Cognition & Planning, Navigation)

- Needs reward signal: sparse, noisy, delayed!
- Take time into account: input frames are related!

- Gained interest 2013 again with:
  - Deep Mind (google) playing ATARI 2600 games
  - Video: Breakout
  - Learned 7 games
  - Surpasses human expert in 3



https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

What is Reinforcement Learning?

- Learning from interaction with an environment to achieve some long-term goal that is related to the state of the environment
- The goal is defined by reward signal, which must be maximized
- Agent must be able to partially/fully sense the environment state and take actions to influence the environment state
- The state is typically described with a feature-vector

Material adapted from
Karan Kathpalia
https://www.cs.princeton.edu/courses/archive/spring17/cos598F/lectures/RL.pptx

Exploration versus Exploitation

- We want a reinforcement learning agent to earn lots of reward
- The agent must prefer past actions that have been found to be effective at producing reward
- The agent must exploit what it already knows to obtain reward
- The agent must select untested actions to discover reward-producing actions
- The agent must explore actions to make better action selections in the future
- Trade-off between exploration and exploitation

Reinforcement Learning Systems

- Reinforcement learning systems have 4 main elements:
  - Policy
  - Reward signal
  - Value function
  - Optional model of the environment

Policy

- A policy is a mapping from the perceived states of the environment to actions to be taken when in those states
- A reinforcement learning agent uses a policy to select actions given the current environment state

Reward Signal

- The reward signal defines the goal
- On each time step, the environment sends a single number called the reward to the reinforcement learning agent
- The agent’s objective is to maximize the total reward that it receives over the long run
- The reward signal is used to alter the policy

Value Function (1)

- The reward signal indicates what is good in the short run while the value function indicates what is good in the long run
- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting in that state
- Compute the value using the states that are likely to follow the current state and the rewards available in those states
- Future rewards may be time-discounted with a factor in the interval [0, 1]

Value Function (2)

- Use the values to make and evaluate decisions
- Action choices are made based on value judgements
- Prefer actions that bring about states of highest value instead of highest reward
- Rewards are given directly by the environment
- Values must continually be re-estimated from the sequence of observations that an agent makes over its lifetime

Model-free versus Model-based

- A model of the environment allows inferences to be made about how the environment will behave
- Example: Given a state and an action to be taken while in that state, the model could predict the next state and the next reward
- Models are used for planning, which means deciding on a course of action by considering possible future situations before they are experienced
- Model-based methods use models and planning. Think of this as modelling the dynamics
- Model-free methods learn exclusively from trial-and-error (i.e. no modelling of the environment)
- Followoing: model-free methods

On-policy versus Off-policy

- An on-policy agent learns only about the policy that it is executing
- An off-policy agent learns about a policy or policies different from the one that it is executing

Credit Assignment Problem

- Given a sequence of states and actions, and the final sum of time-discounted future rewards, how do we infer which actions were effective at producing lots of reward and which actions were not effective?

- How do we assign credit for the observed rewards given a sequence of actions over time?

- Every reinforcement learning algorithm must address this problem

Reward Design

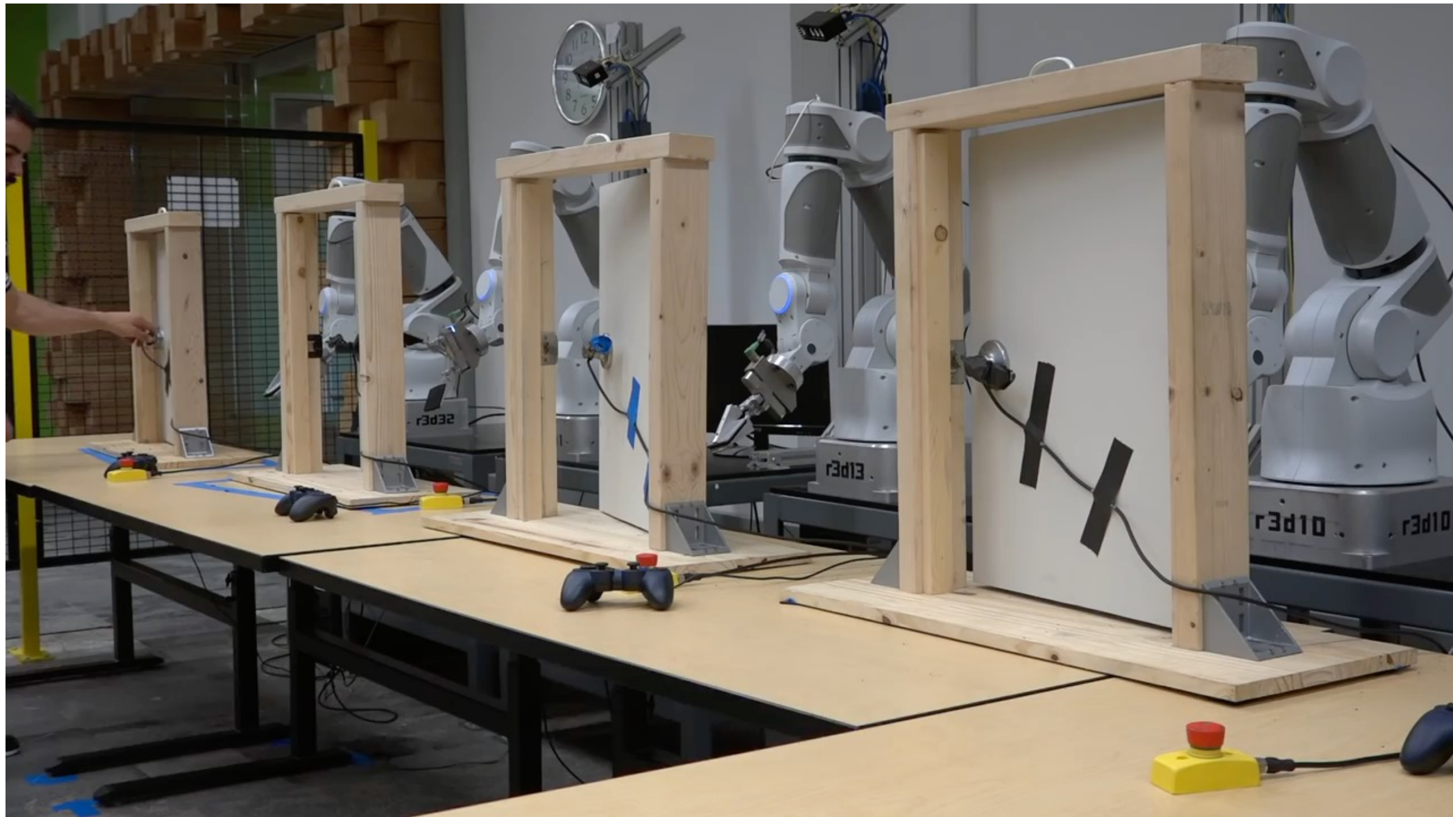
- We need rewards to guide the agent to achieve its goal
- Option 1: Hand-designed reward functions
- This is a black art
- Option 2: Learn rewards from demonstrations
- Instead of having a human expert tune a system to achieve the desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration

What is Deep Reinforcement Learning?

- Deep reinforcement learning is standard reinforcement learning where a deep neural network is used to approximate either a policy or a value function
- Deep neural networks require lots of real/simulated interaction with the environment to learn
- Lots of trials/interactions is possible in simulated environments
- We can easily parallelize the trials/interaction in simulated environments
- We cannot do this with robotics (no simulations) because action execution takes time, accidents/failures are expensive and there are safety concerns

Google Door Opening Project

- Learn to open doors using Reinforcement learning
  - Learning reward: opening the door
  - Much harder than purely digital learning: very slow iterations!
  - Simulation only helps a bit:
    real world much more complex
- Google and
  UC Berkeley Sergey Levine
- Google very secretive …



https://www.wired.com/2017/01/googles-go-playing-machine-opens-door-robots-learn/

RL Algorithms

• Finite Markov Decision Processes	MDP
• Temporal-Difference Learning	TD Learning
• State-Action-Reward-State-Action	SARSA TD Learning
• Q-learning: Off-policy TD Control	
• Deep Q-Networks	DQN

- Policy Gradient Methods
  - Actor-Critic Methods

- Asynchronous Reinforcement Learning

Slides by
Ziwen Zhuang

上海科技大学
ShanghaiTech University

Robot Learning

- Cognitive Intelligence
  - Manipulation
- Athletic Intelligence
  - Locomotion/Cont
    rol

立志成才报国裕民

Cognitive Intelligence

- Multi-modal transfer learning for grasping transparent and specular objects
- Learning Frine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)
- MimicPlay: Long-Horizon Imitation Learning by Watching Human Play
- RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools
- VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models
- ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization

Command by text

https://say-can.github.io
https://nlmap-saycan.github.io

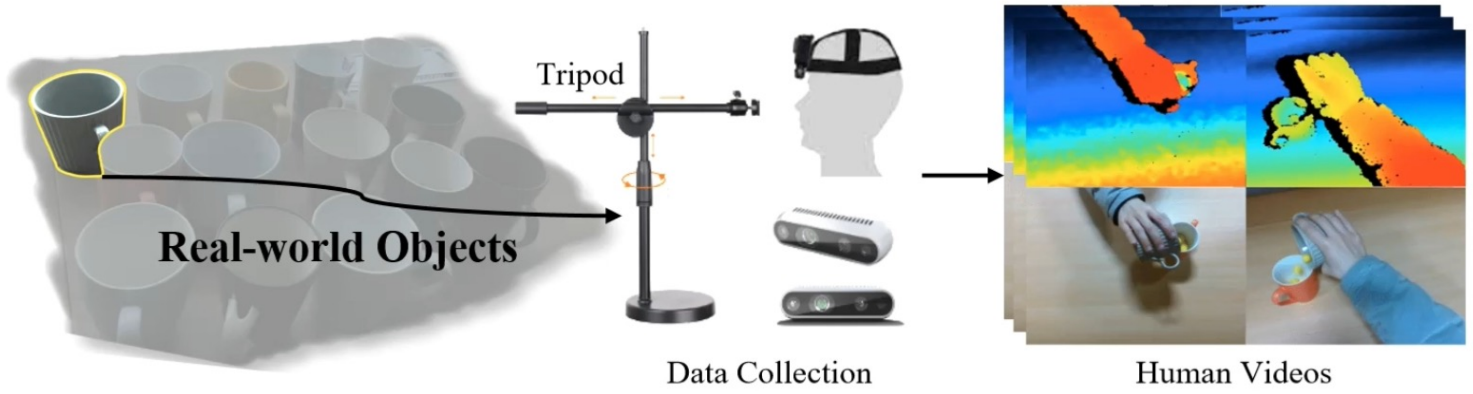
Do As I Can, Not As I Say: Grounding Language in Robotic Affordances

Customers want a robot that handles

all household tasks

and is
commanded by natural language

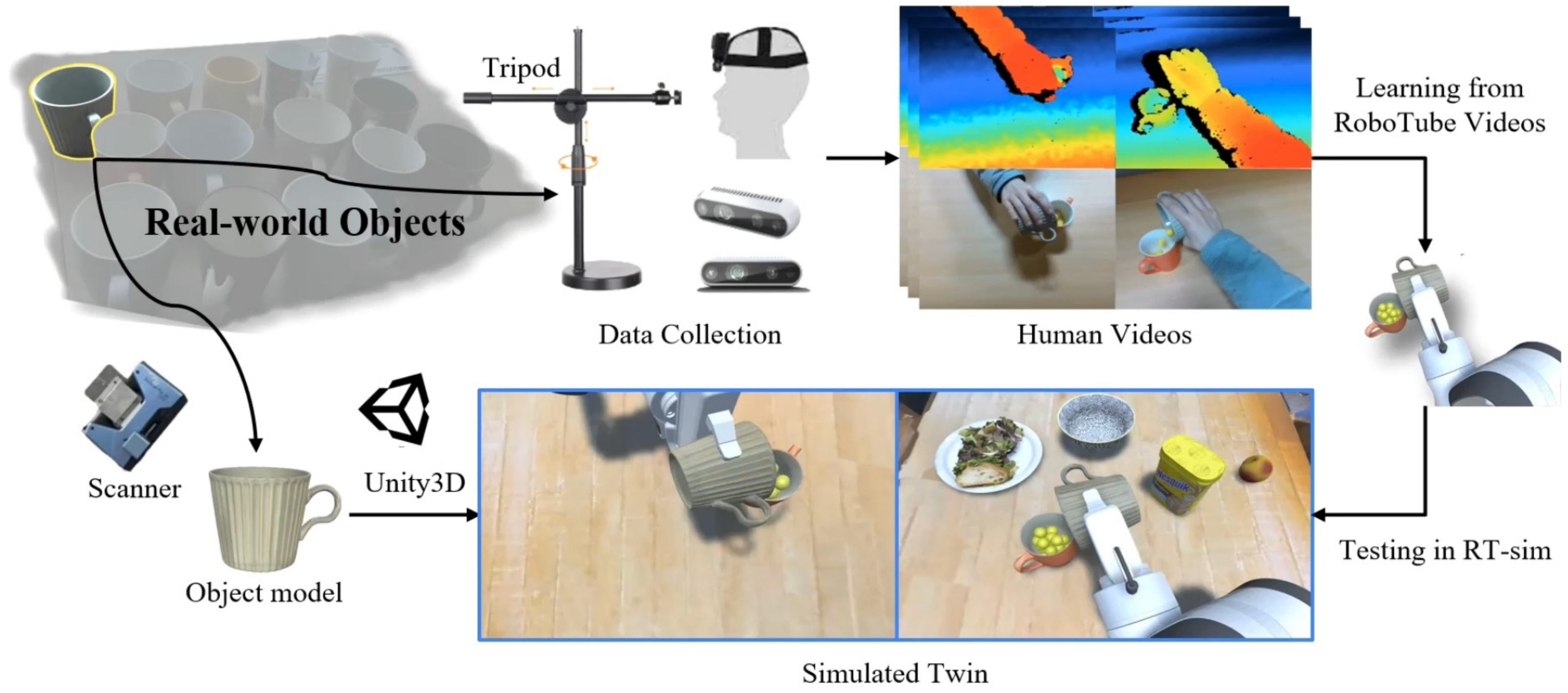
Learning Household Manipulation from Human Videos with Simulated Twin Environments



1. Not all researchers have the robot to evaluate
2. Dataset should avoid missing scenarios

Dataset alone is not enough

Learning Household Manipulation from Human Videos with Simulated Twin Environments

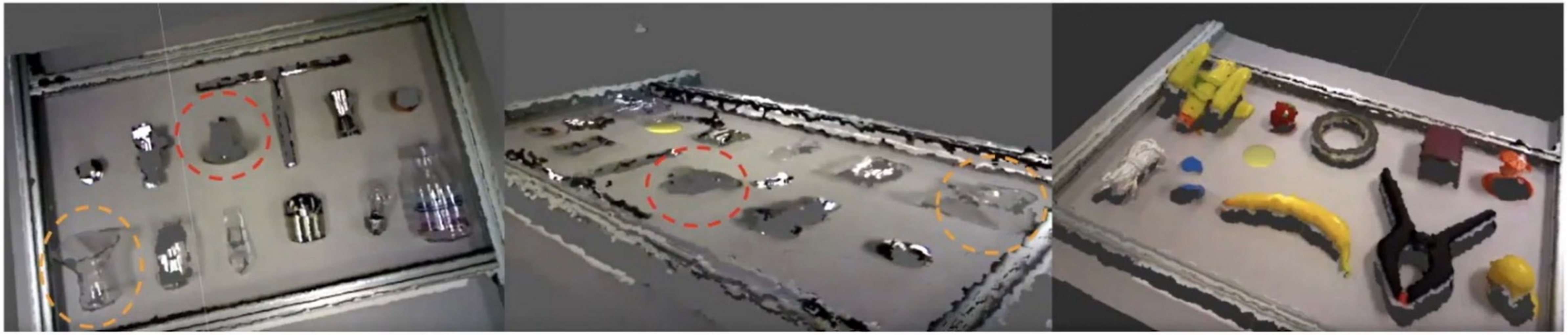


build simulation environments paired with video dataset

1. Pair real–world video with simulation video
2. Large dataset to 5000+ videos and 50 different objects

Multi-modal transfer learning for grasping transparent and specular objects

Depth sensors suffer from transparent objects and specular objects



Overhead view

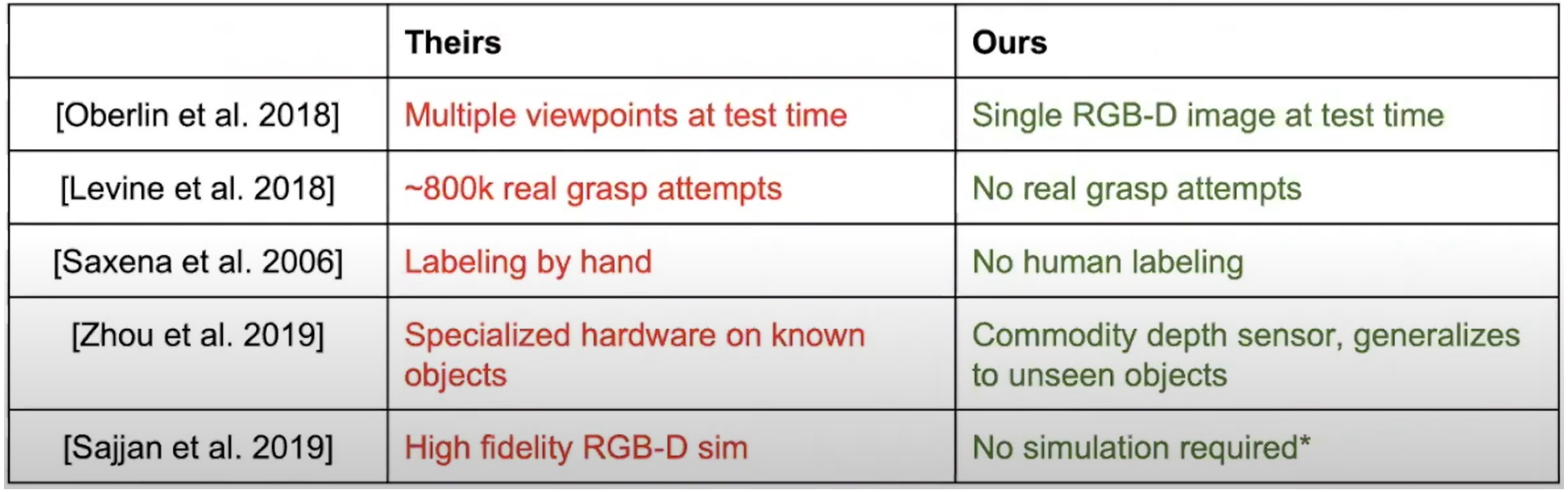
Oblique view
Depth is flat or has holes

Reference image of opaque objects

立志成才报国裕民

Multi-modal transfer learning for grasping transparent and specular objects

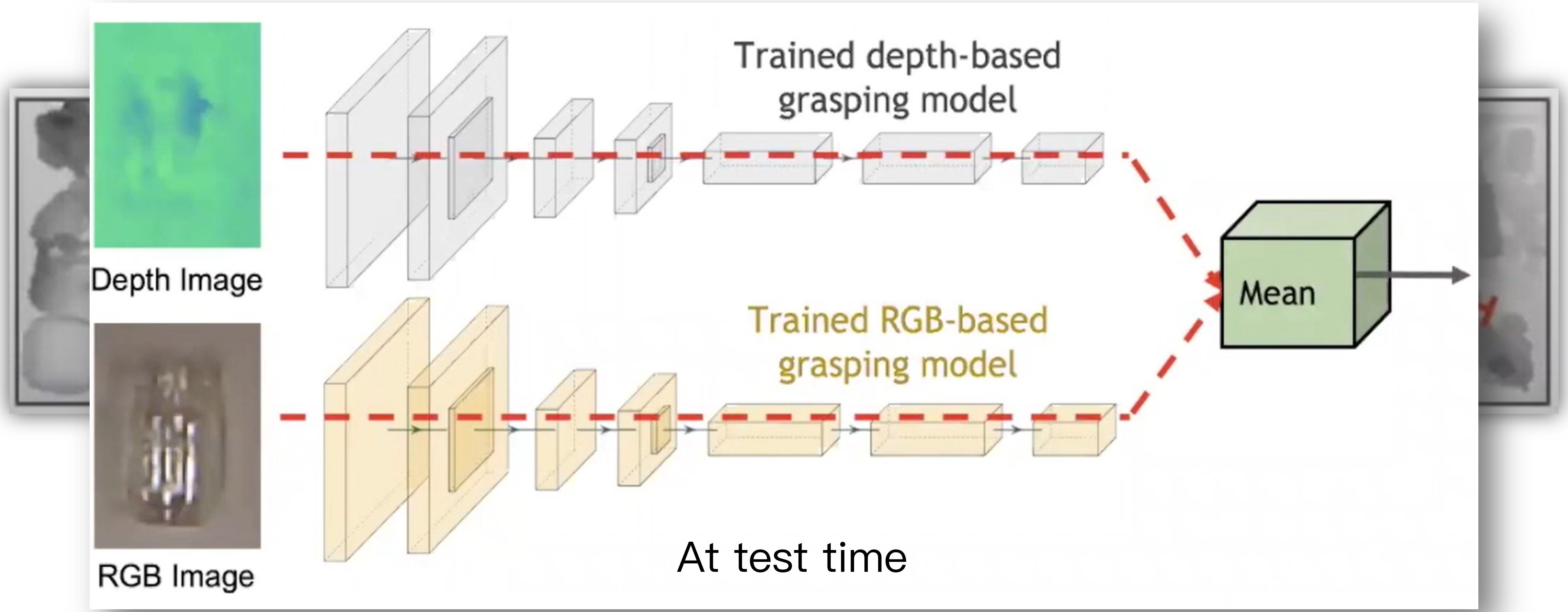
Previous works cost a lot. :P



	Theirs	Ours
[Oberlin et al. 2018]	Multiple viewpoints at test time	Single RGB-D image at test time
[Levine et al. 2018]	~800k real grasp attempts	No real grasp attempts
[Saxena et al. 2006]	Labeling by hand	No human labeling
[Zhou et al. 2019]	Specialized hardware on known objects	Commodity depth sensor, generalizes to unseen objects
[Sajjan et al. 2019]	High fidelity RGB-D sim	No simulation required*

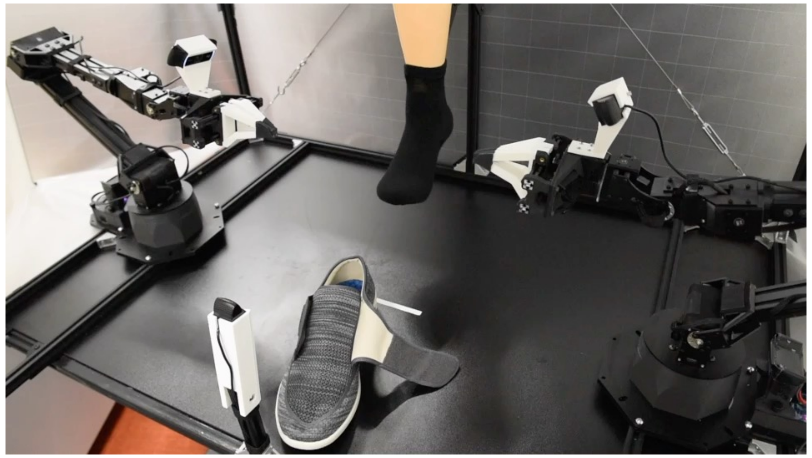
立志成才报国裕民

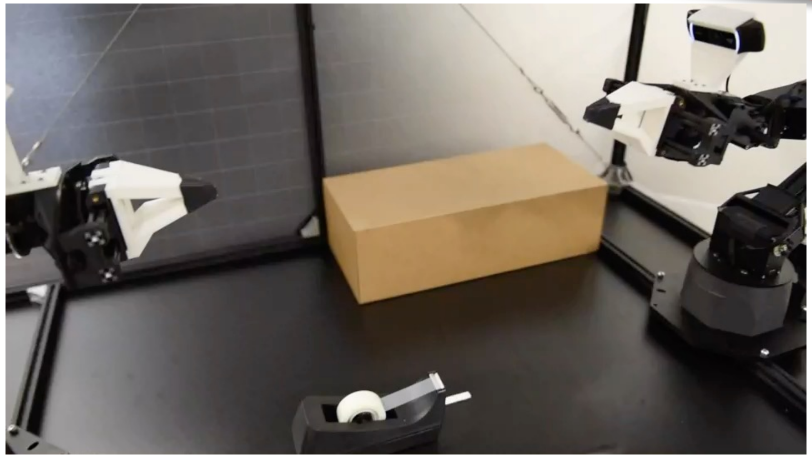
Multi-modal transfer learning for grasping transparent and specular objects

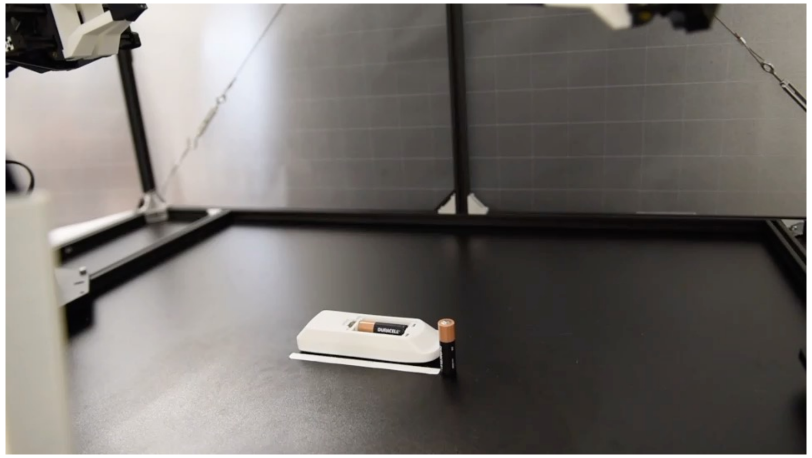

Depth Image
RGB Image
Trained depth-based grasping model
Trained RGB-based grasping model
Mean
At test time

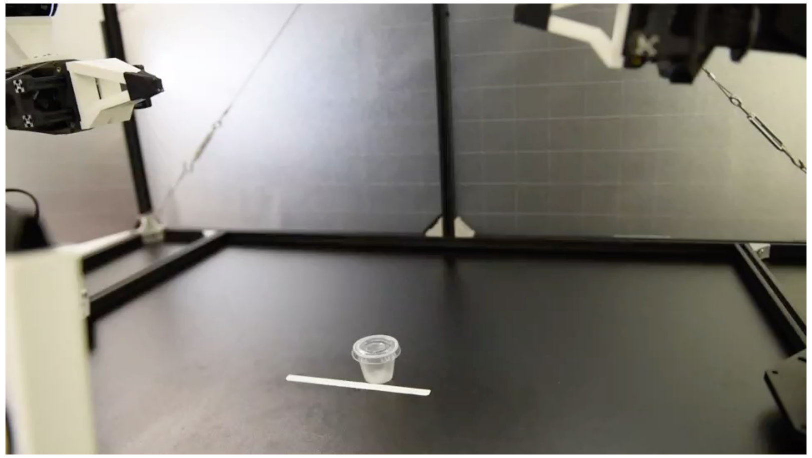
立志成才报国裕民

Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)

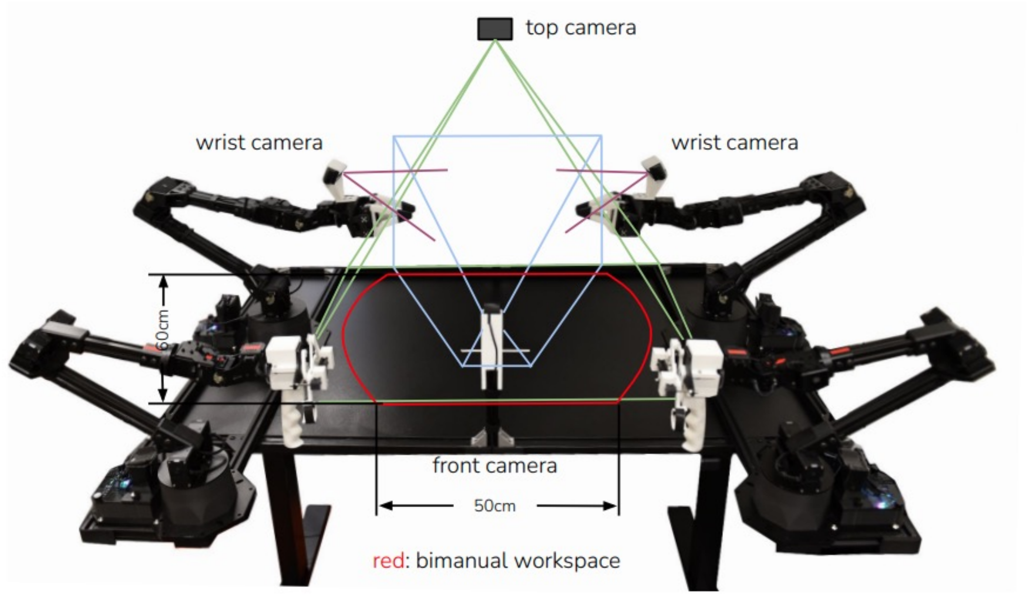


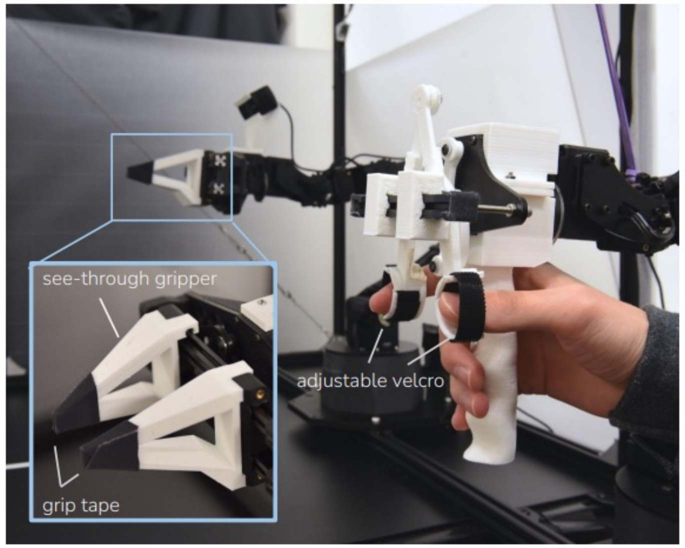






Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)



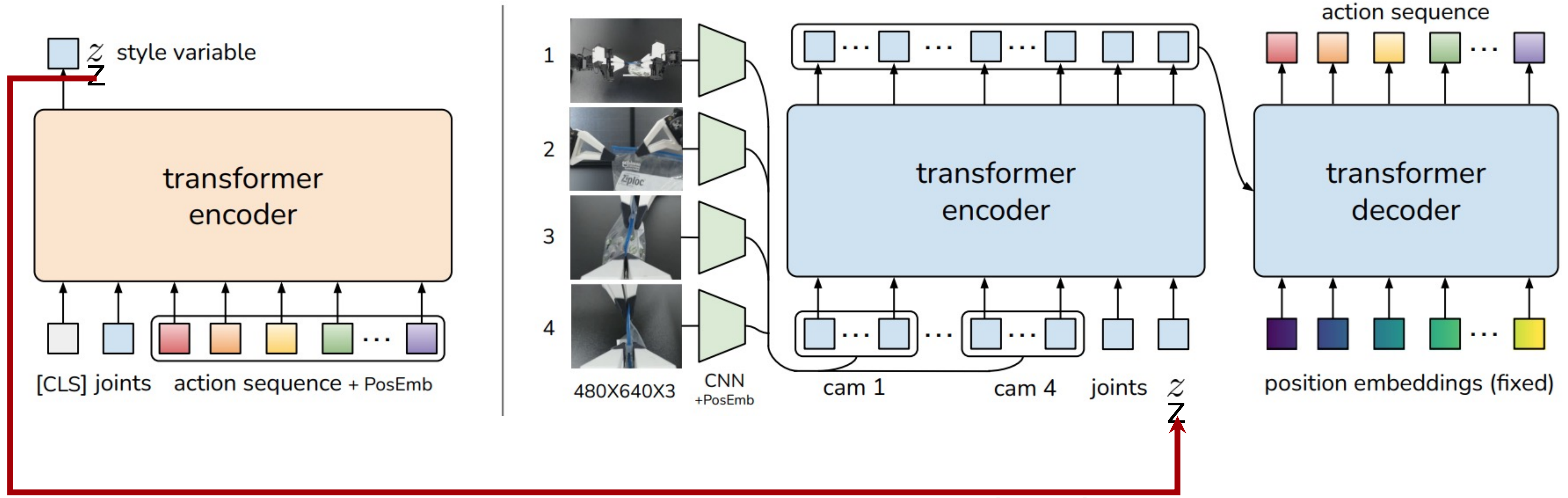


ViperX 6dof Arm (follower)

#Dofs	6+gripper
Reach	750mm
Span	1500mm
Repeatability	1mm
Accuracy	5-8mm
Working Payload	750g

立志成才报国裕民

Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)



Action Chunking with Transformers (ACT)
train a conditional VAE (C–VAE)
left: the encoder part, where z is the style variable
right: the decoder part, where z is connected during training but set to the mean of the prior (i.e. 0) during testing.

立志成才报国裕民

Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)

Action Chunking

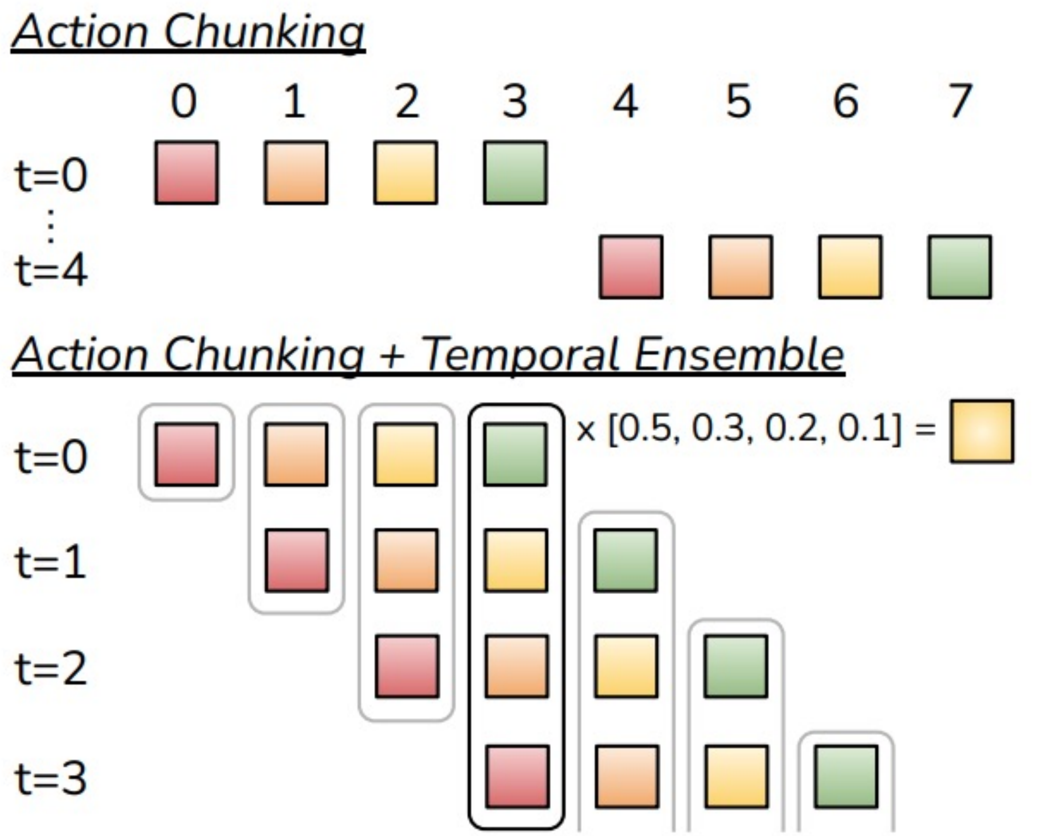
Inspired by action chunking, a neuroscience concept where individual actions are grouped together and executed as one unit, making them more efficient to store and execute.

Temporal Ensemble

weighted sum of action from different action sequence predictions

$w_i = exp(-m \times i)$ where w_0 is for the oldest action and m is the hyperparameter.

The smaller m is, the faster incorporation.



立志成才 报国裕民

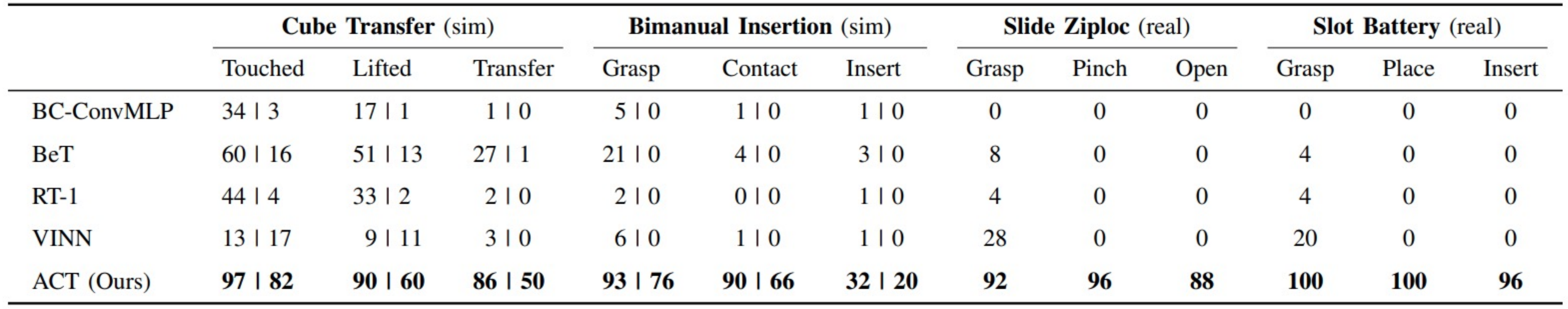
Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT / ALOHA)

Data collection

- 8 tasks (6 real-world, 2 simulated)
- 50 demonstrations for each task
- 50Hz, 8~14s on each task

Training setting

- 80M parameters model
- 5 hours on one 11G RTX 2080 Ti GPU
- 0.01 inference time on the machine

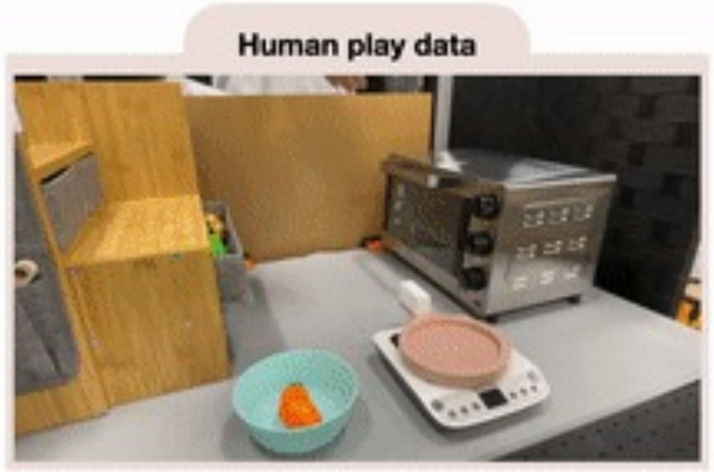


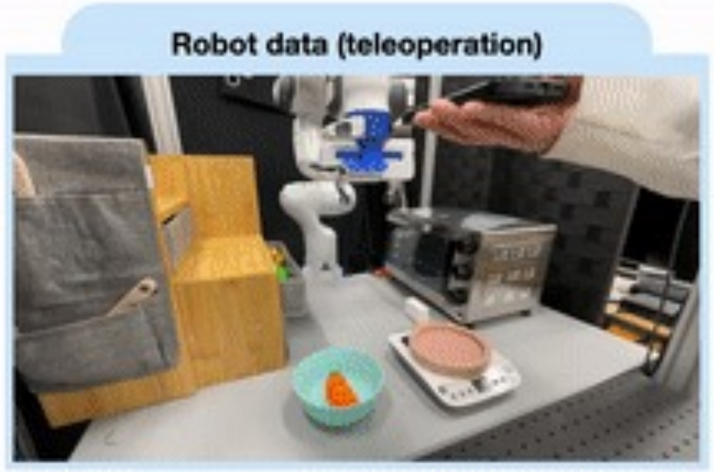
	Cube Transfer (sim)			**Bimanual Insertion** (sim)			**Slide Ziploc** (real)			**Slot Battery** (real)		
	Touched	Lifted	Transfer	Grasp	Contact	Insert	Grasp	Pinch	Open	Grasp	Place	Insert
BC-ConvMLP	34 | 3	17 | 1	1 | 0	5 | 0	1 | 0	1 | 0	0	0	0	0	0	0
BeT	60 | 16	51 | 13	27 | 1	21 | 0	4 | 0	3 | 0	8	0	0	4	0	0
RT-1	44 | 4	33 | 2	2 | 0	2 | 0	0 | 0	1 | 0	4	0	0	4	0	0
VINN	13 | 17	9 | 11	3 | 0	6 | 0	1 | 0	1 | 0	28	0	0	20	0	0
ACT (Ours)	**97 | 82**	**90 | 60**	**86 | 50**	**93 | 76**	**90 | 66**	**32 | 20**	**92**	**96**	**88**	**100**	**100**	**96**

TABLE I: Success rate (%) for 2 simulated and 2 real-world tasks, comparing our method with 4 baselines. For the two simulated tasks, we report [training with scripted data | training with human data], with 3 seeds and 50 policy evaluations each. For the real-world tasks, we report training with human data, with 1 seed and 25 evaluations. Overall, ACT significantly outperforms previous methods.

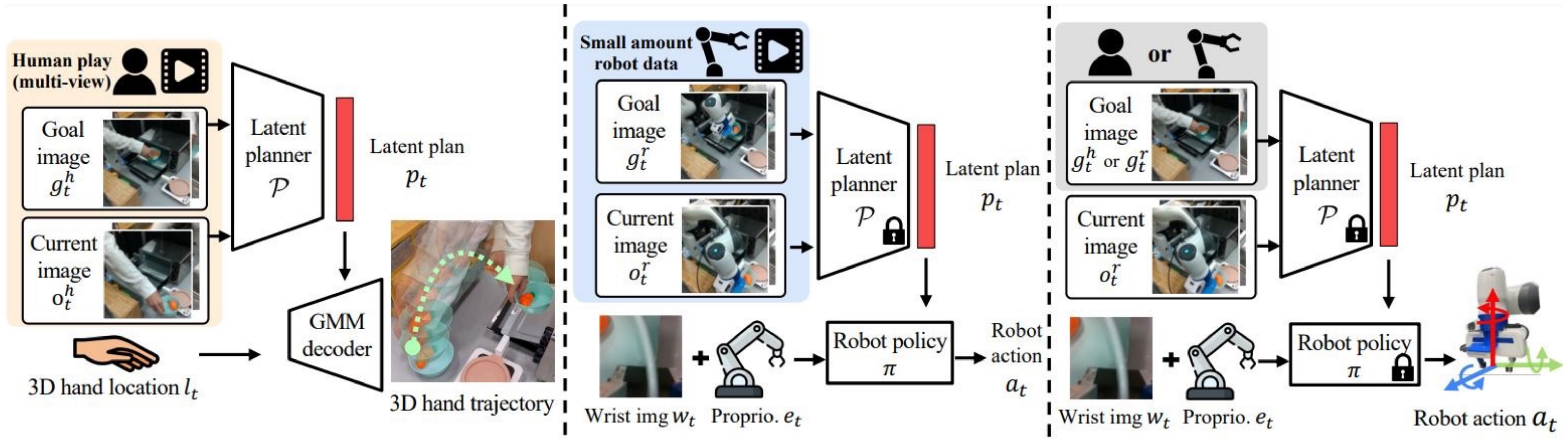
立志成才报国裕民

MimicPlay: Long-Horizon Imitation Learning by Watching Human Play





MimicPlay: Long-Horizon Imitation Learning by Watching Human Play



(a) Training stage 1 - Learning latent plans **(b) Training stage 2 - Plan-guided imitation learning** **(c) Testing stage**

MimicPlay: Long-Horizon Imitation Learning by Watching Human Play

How to bridge the gap between
human appearance
and robot appearance?

human video $o^h \in \mathcal{V}^h$

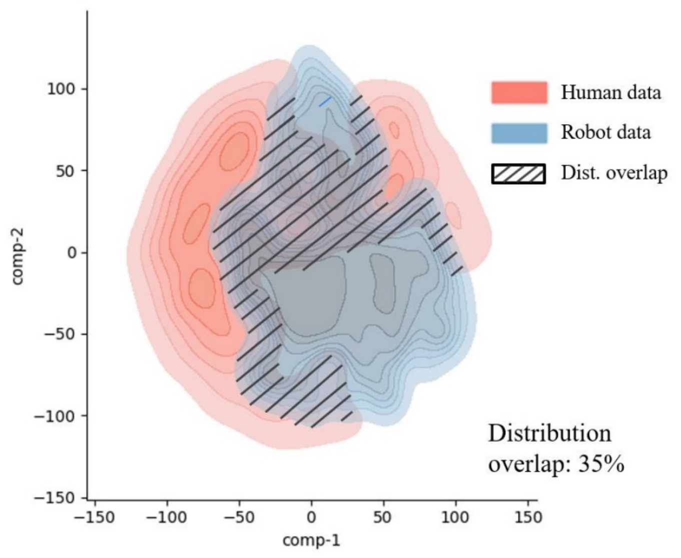
robot video $o^r \in \mathcal{V}^r$

calculate feature embedding **across entire dataset**
$Q^h = E(\mathcal{V}^h), Q^r = E(\mathcal{V}^r)$

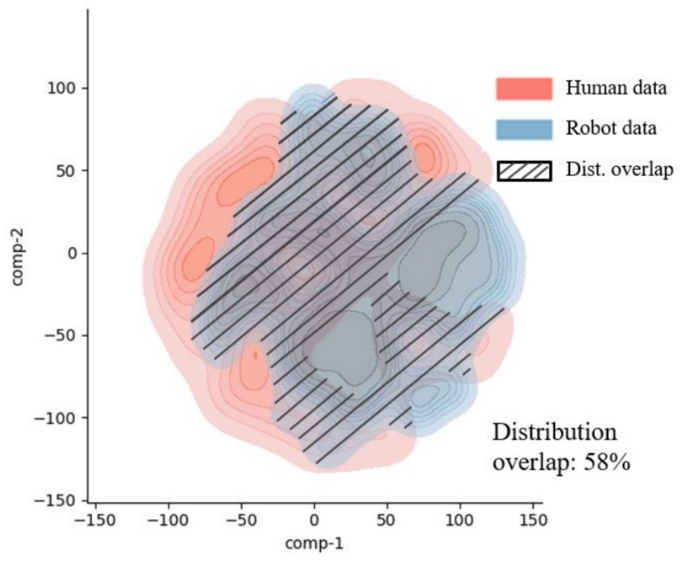
minimize the KL divergence across entire dataset
$\mathcal{L} = D_{KL}(Q^r || Q^h)$

MimicPlay: Long-Horizon Imitation Learning by Watching Human Play

A simple KL regularization seems to mitigate the domain gap



(a) Distribution overlap of Ours (w/o KL)



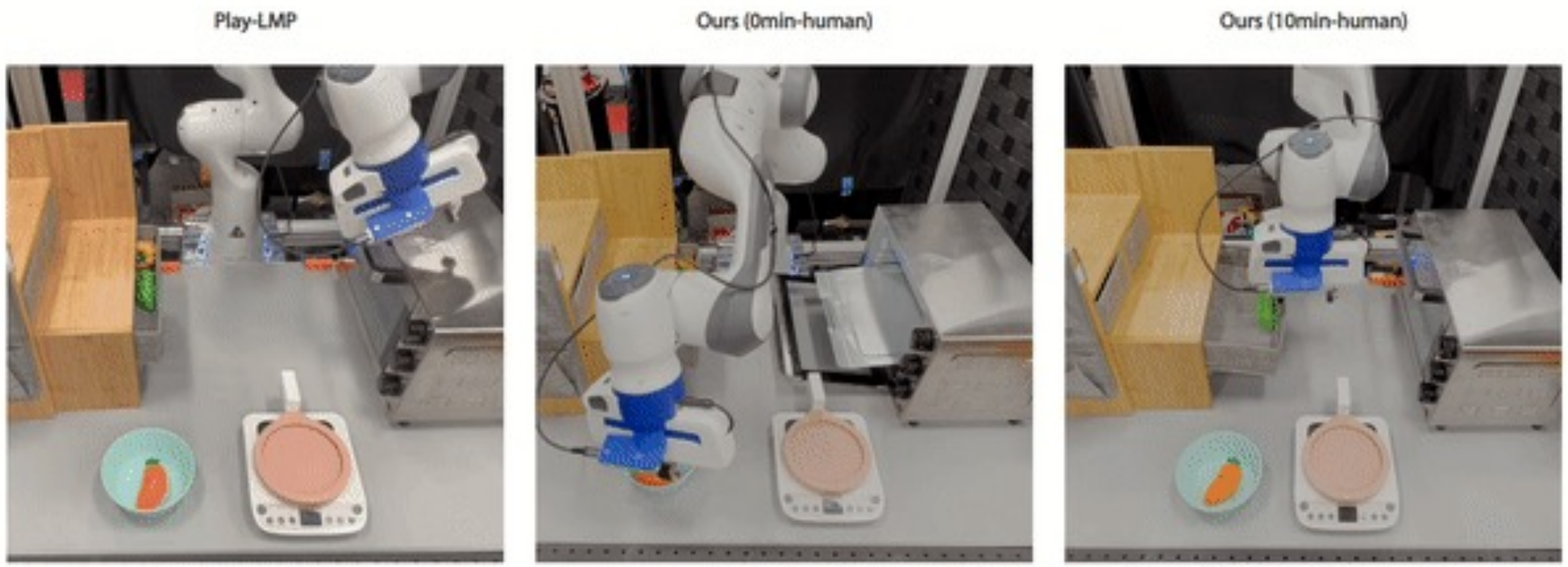
(b) Distribution overlap of Ours

KL Regularization refers to using the **Kullback-Leibler (KL) divergence** as a regularization term in an optimization problem, typically in the context of machine learning. It encourages one probability distribution (often a learned model) to stay close to another reference distribution, serving as a constraint or guidance during training.

MimicPlay: Long-Horizon Imitation Learning by Watching Human Play

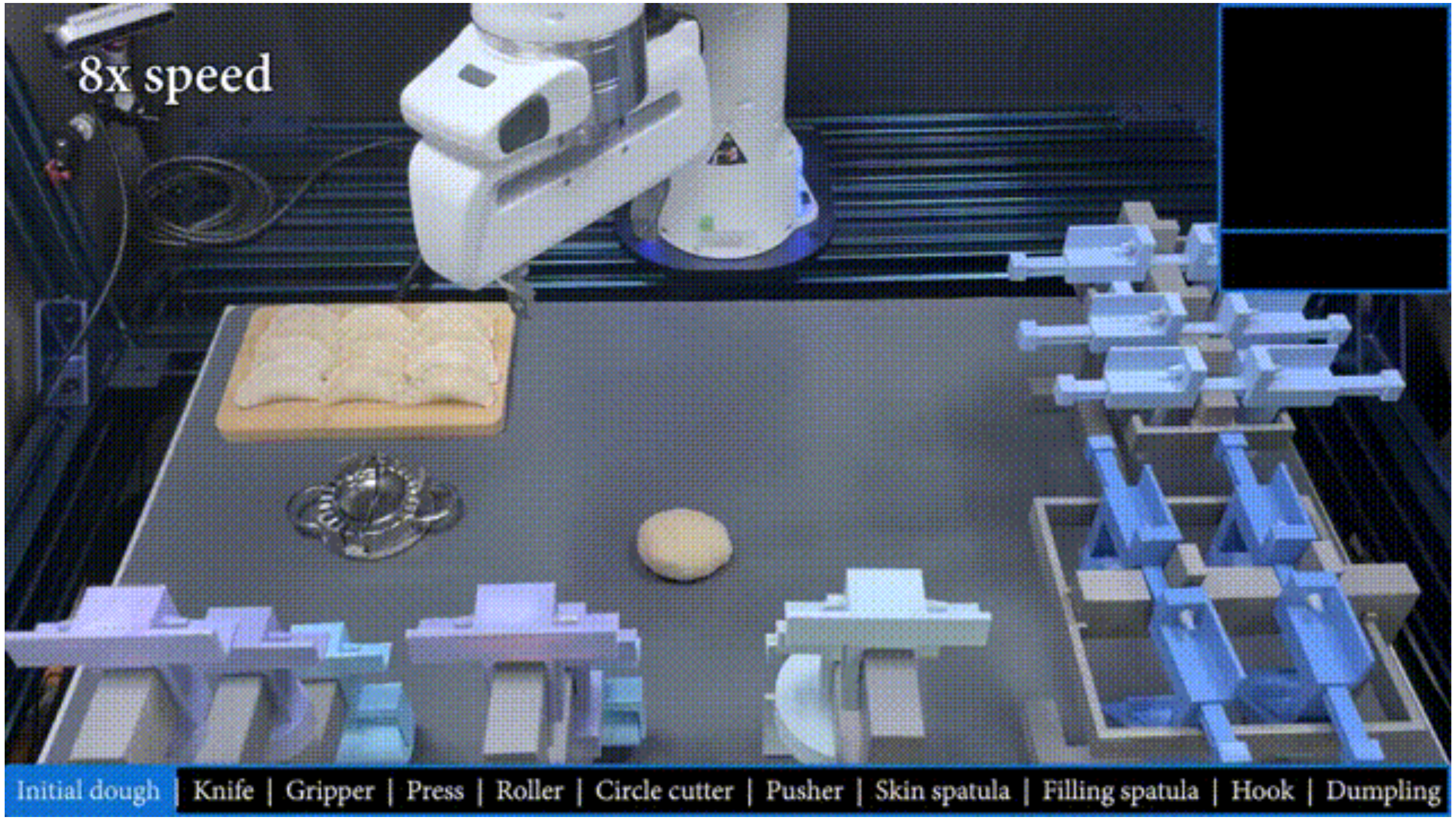
"Do what I just did": Prompting robot manipulation with human motion
AMERICA'S MOMENT
X4
Step 1: Recording human motion

MimicPlay: Long-Horizon Imitation Learning by Watching Human Play



Success rate (%)	Play-LMP	Ours (0min-human)	Ours (10min-human)
20 demos	7%	23%	47% ↑
40 demos	13%	40%	70% ↑

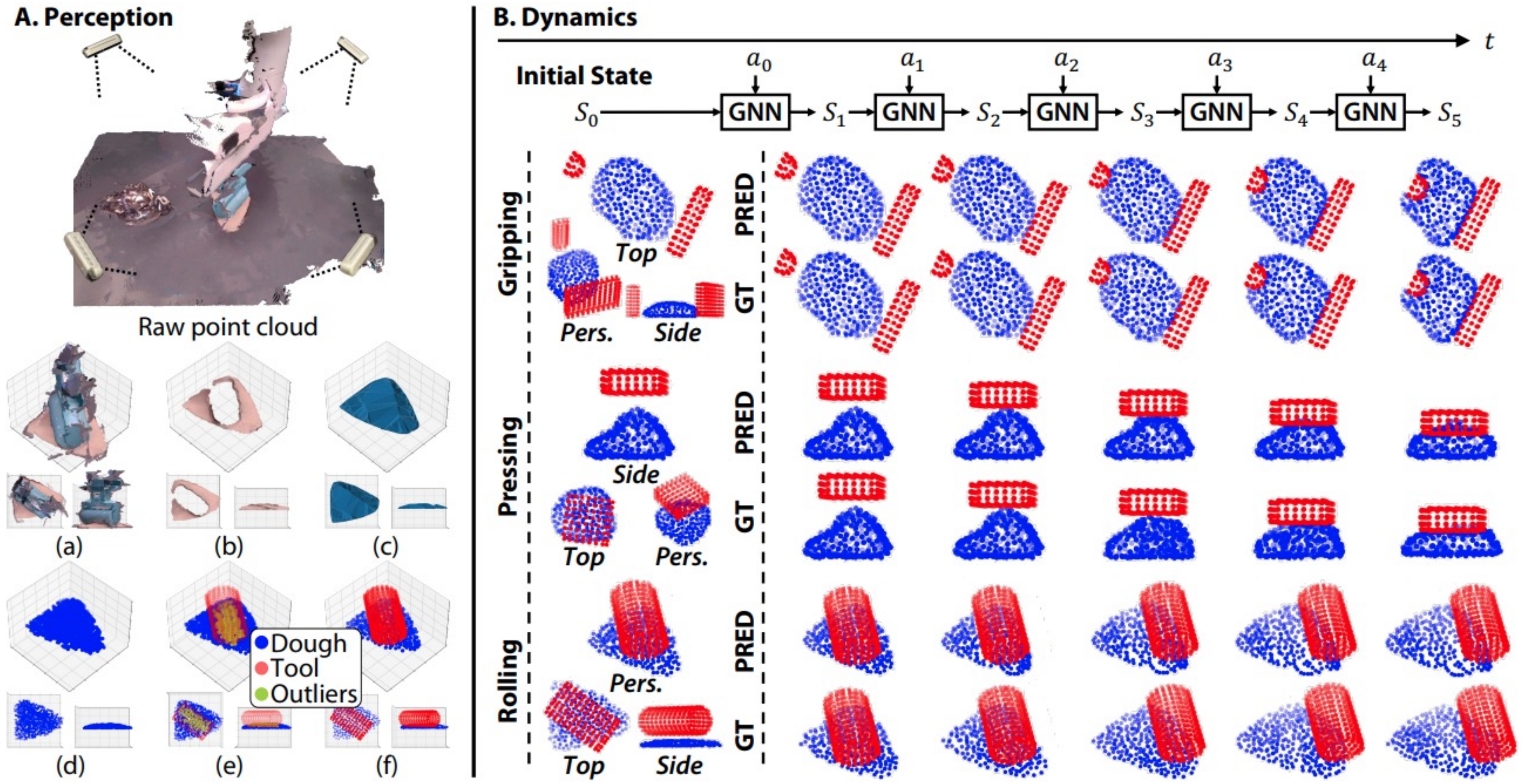
RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools


8x speed
Initial dough | Knife | Gripper | Press | Roller | Circle cutter | Pusher | Skin spatula | Filling spatula | Hook | Dumpling

RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools

Why difficult?

1. deformable representation
2. fine grand manipulation
3. predict dynamics
4. supervision
5. perception (pointcloud?)

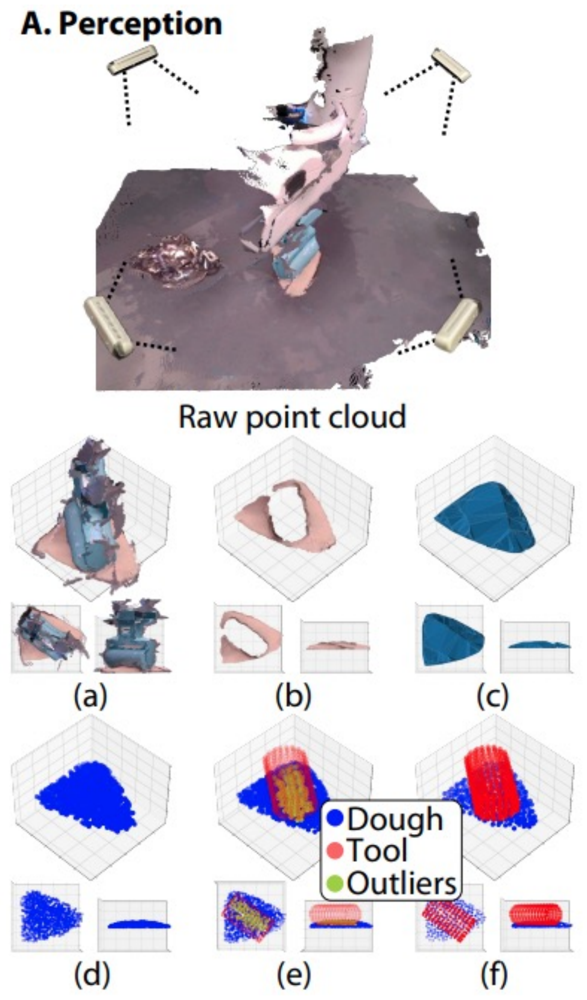


立志成才报国裕民

RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools

Perception

- pointcloud merged from 4 RGBD cameras
- for dough, sample 300 points uniformly on the surface
    - (Poisson disk sampling)
- for tools, uniformly sample the contact surface



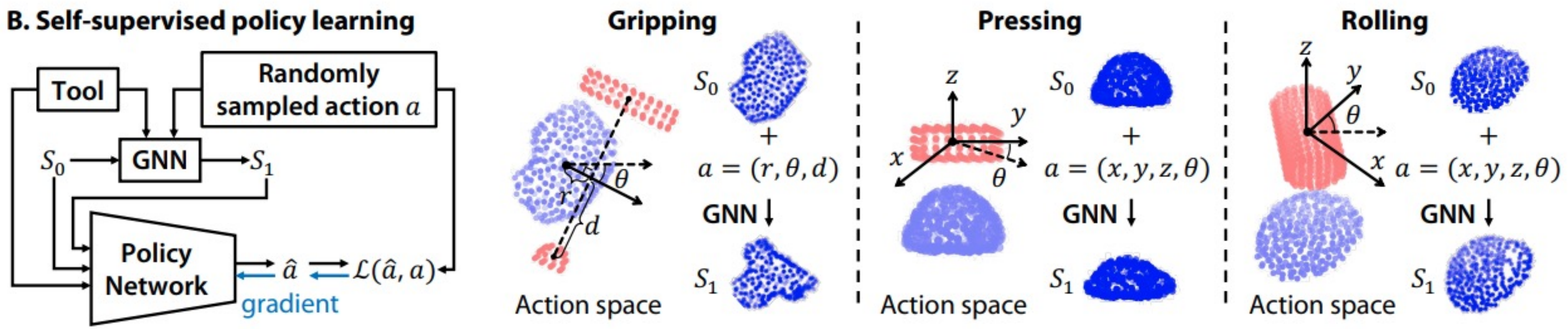
RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools

Dynamics model

- a set of parameterized actions
- given S_t, a_t and tool class, predict S_{t+1}
- given S_t, S_{t+1} predict a_t
- PointNet++, MLP, SoftMax

Tool Selection Model

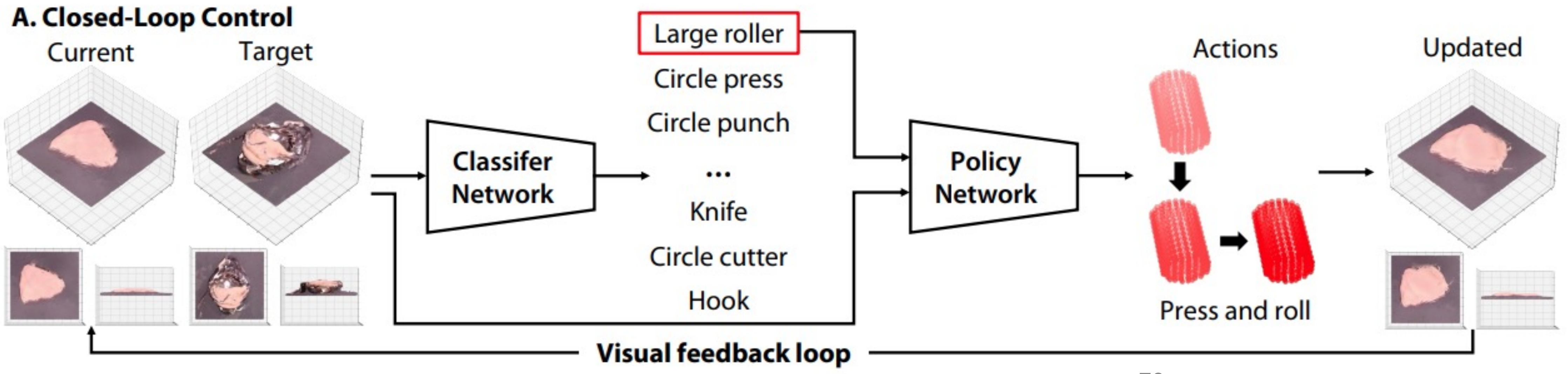
- Supervised Learning
- Labeled by human demonstrated process



RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools

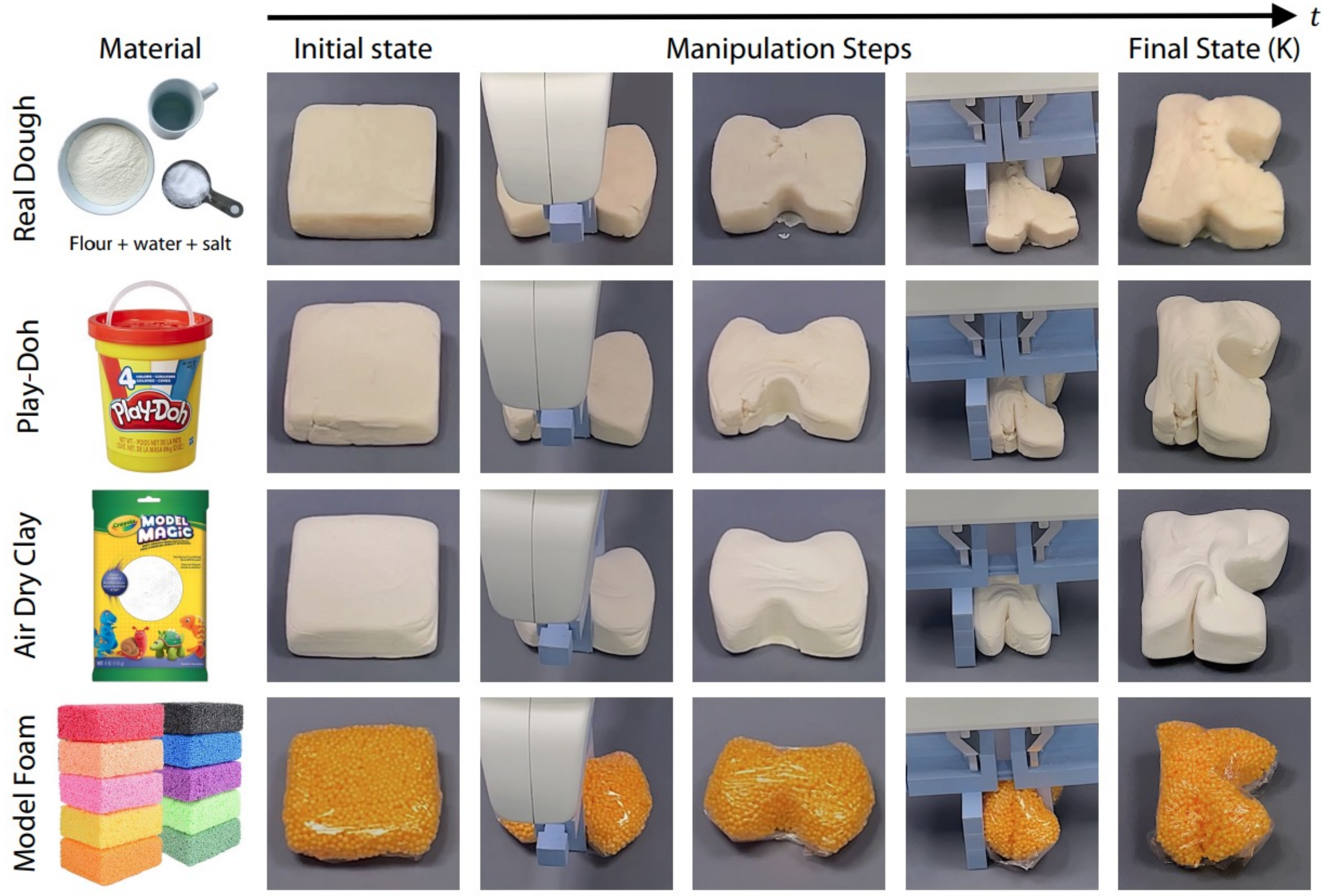
Close-loop Control policy

- given S_t, S_{Target} predict a_t
- select top 3 tool selection, predict 3 possible outcome,
- execute the action with the closest result to the target.

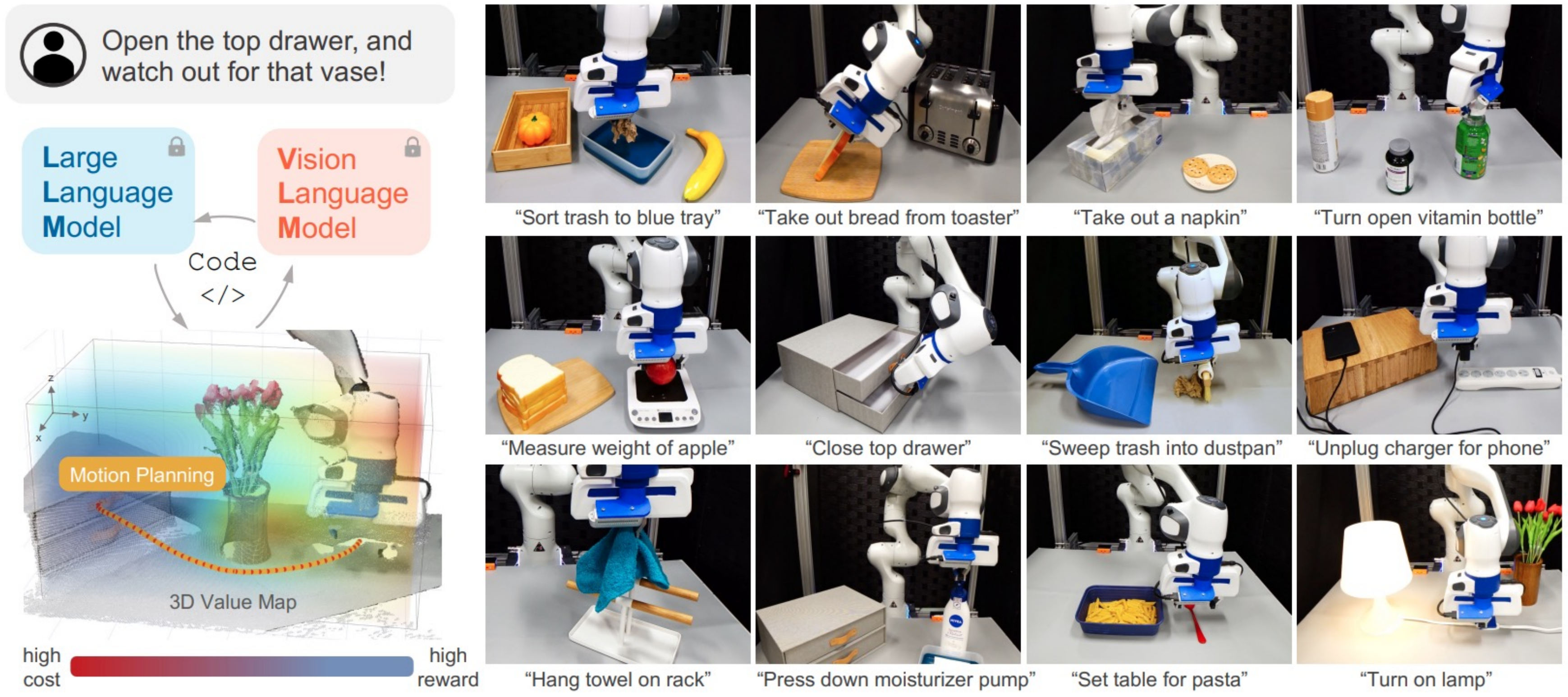


立志成才 报国裕民

RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools



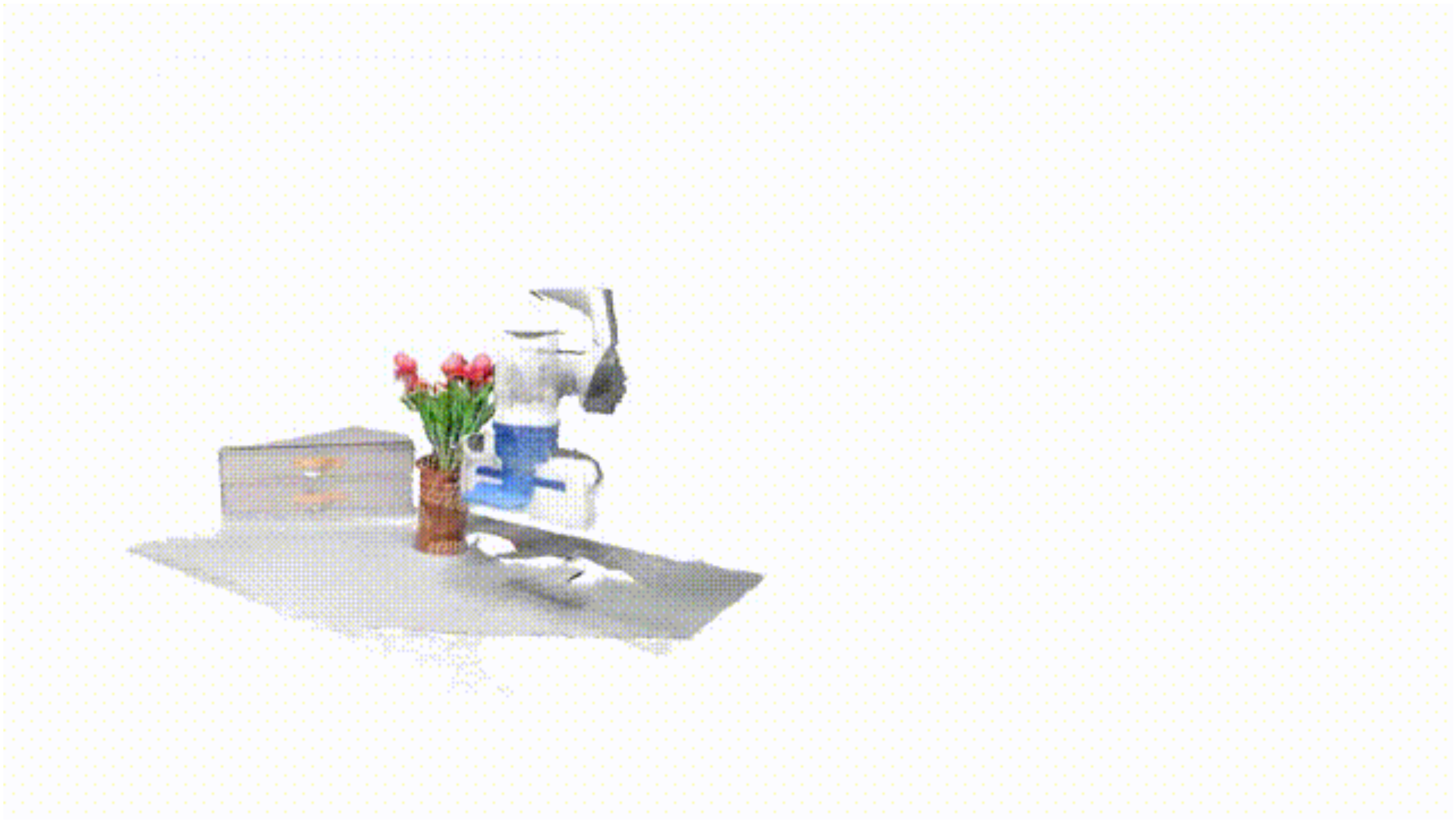
VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models



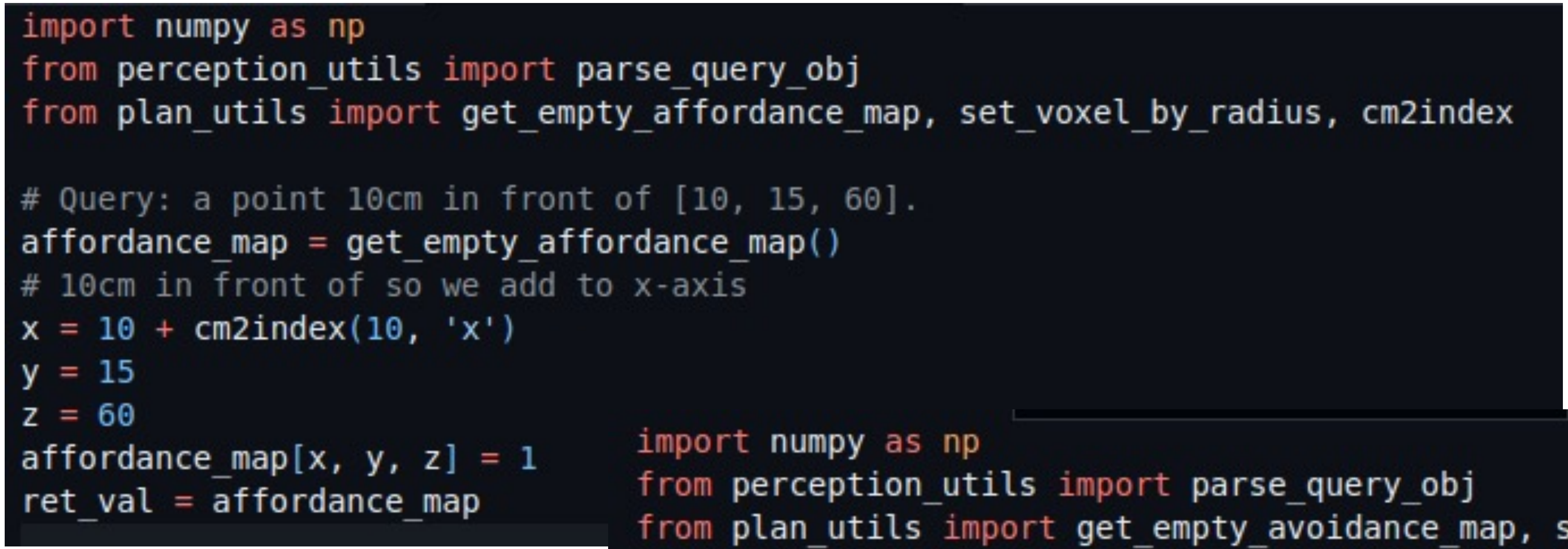
Open the top drawer, and watch out for that vase!
Large Language Model
Vision Language Model
Code
</>
Motion Planning
3D Value Map
high cost
high reward
"Sort trash to blue tray"
"Take out bread from toaster"
"Take out a napkin"
"Turn open vitamin bottle"
"Measure weight of apple"
"Close top drawer"
"Sweep trash into dustpan"
"Unplug charger for phone"
"Hang towel on rack"
"Press down moisturizer pump"
"Set table for pasta"
"Turn on lamp"

Figure 1: **VoxPoser** extracts language-conditioned **affordances** and **constraints** from LLMs and grounds them to the perceptual space using VLMs, using a code interface and without additional training to either component. The composed map is referred to as a 3D value map, which enables **zero-shot** synthesis of trajectories for large varieties of everyday manipulation tasks with an **open-set of instructions** and an **open-set of objects**.

VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models



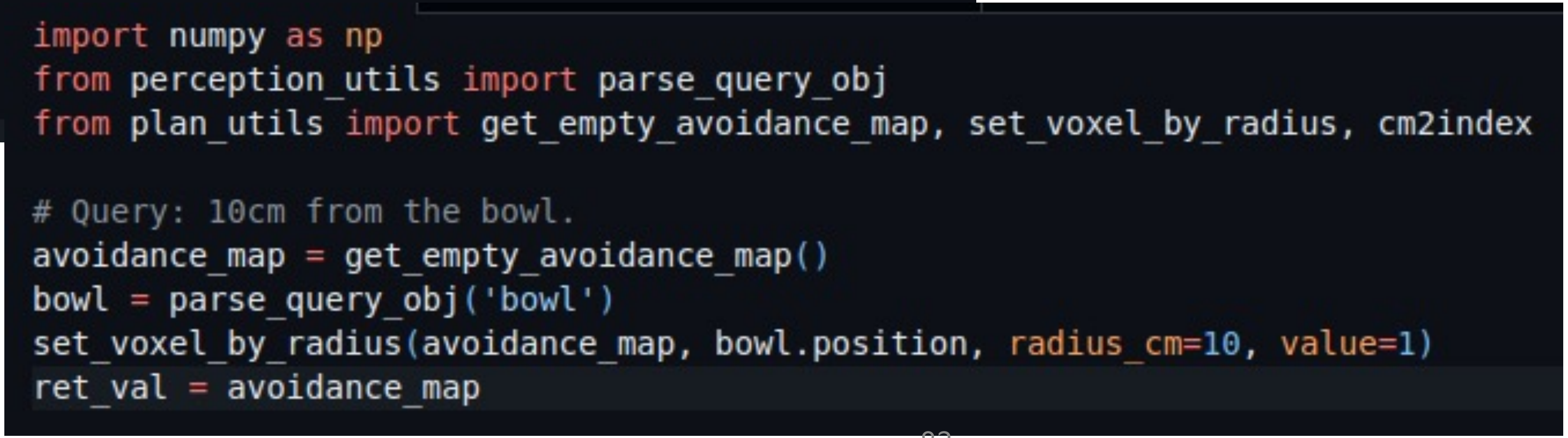
VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models



```python
import numpy as np
from perception_utils import parse_query_obj
from plan_utils import get_empty_affordance_map, set_voxel_by_radius, cm2index

# Query: a point 10cm in front of [10, 15, 60].
affordance_map = get_empty_affordance_map()
# 10cm in front of so we add to x-axis
x = 10 + cm2index(10, 'x')
y = 15
z = 60
affordance_map[x, y, z] = 1
ret_val = affordance_map
```

Prompt Engineering



```python
import numpy as np
from perception_utils import parse_query_obj
from plan_utils import get_empty_avoidance_map, set_voxel_by_radius, cm2index

# Query: 10cm from the bowl.
avoidance_map = get_empty_avoidance_map()
bowl = parse_query_obj('bowl')
set_voxel_by_radius(avoidance_map, bowl.position, radius_cm=10, value=1)
ret_val = avoidance_map
```

立志成才 报国裕民

VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models

LLM and Prompting

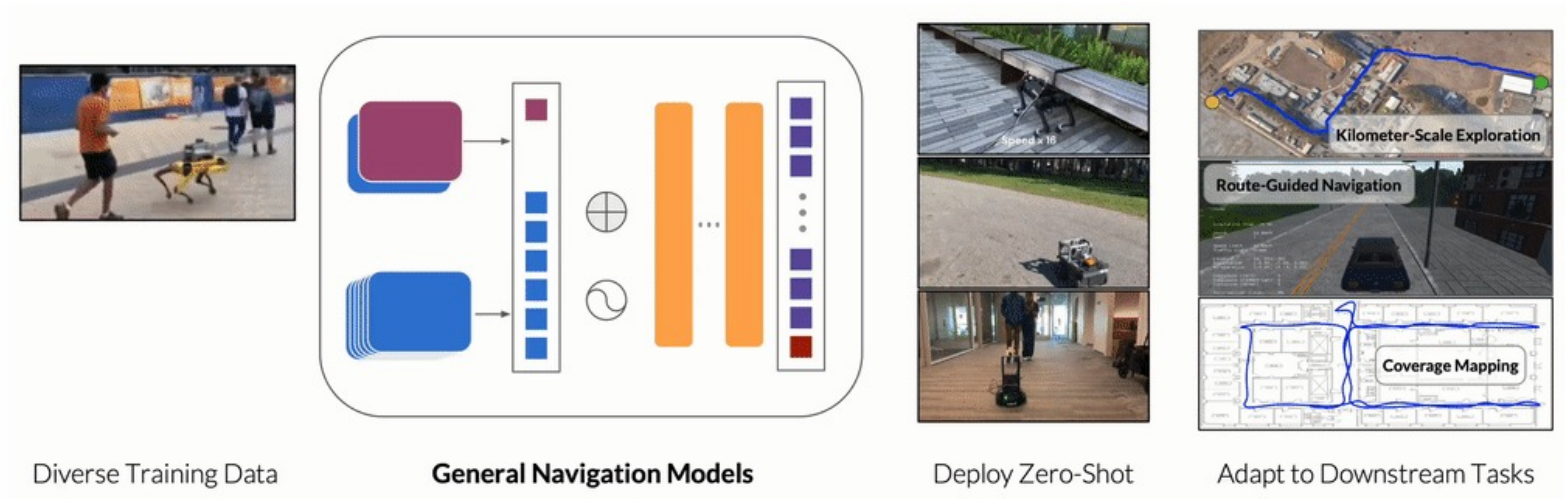
- GPT-4
- Include 5-20 example queries and corresponding responses as part of the prompt.

VLMs and Perception

- OWL-ViT to obtain a bounding box
- Segment Anything to obtain a mask
- XMEM to track the mask
- RGB-D camera to obtain object/part point cloud

立志成才 报国裕民

ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization



ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization

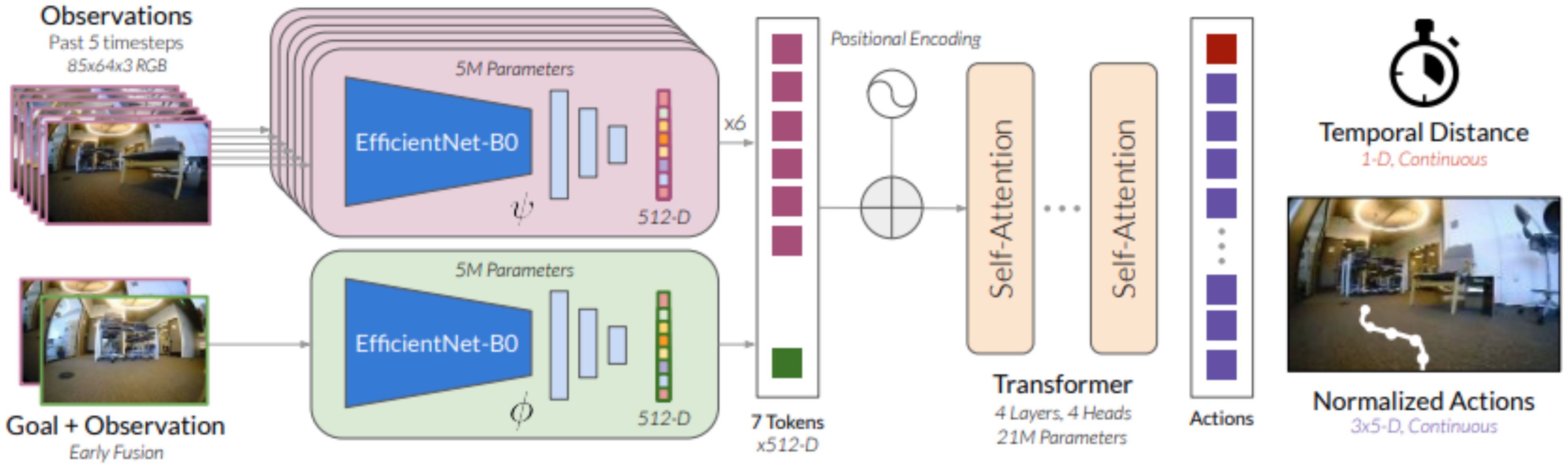


Figure 2: ViNT Model Architecture. ViNT uses two EfficientNet encoders ψ, ϕ to generate input tokens to a Transformer decoder. The resulting sequence is concatenated and passed through a fully-connected network to predict (temporal) distance to the goal as well as a sequence of $H = 5$ future actions.

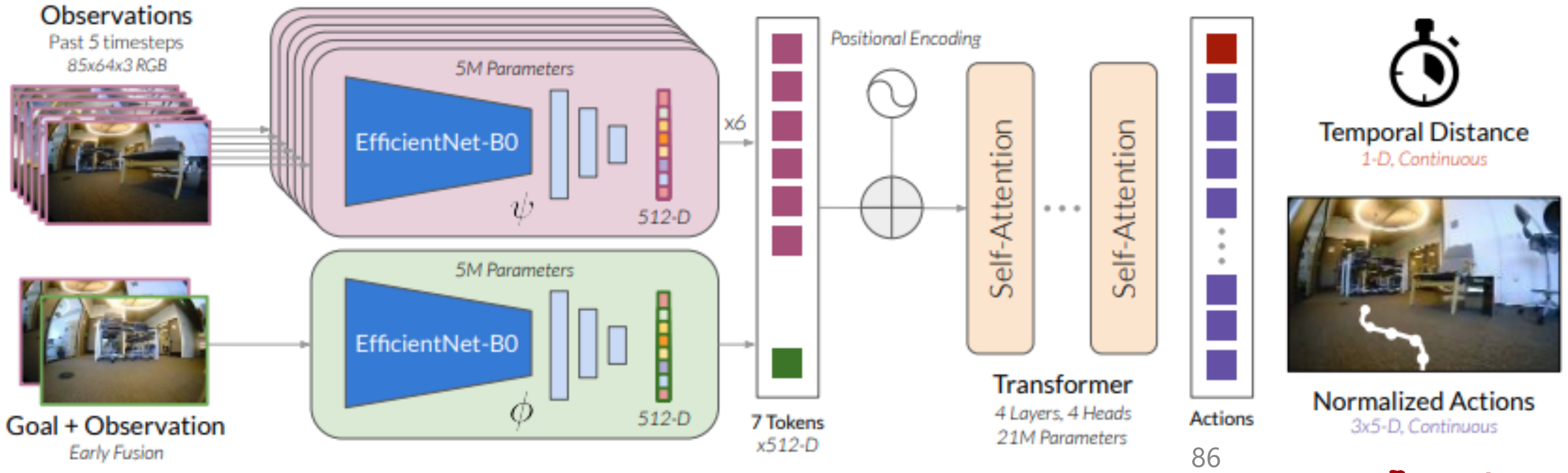
Uses action abstraction across embodiments: short-term waypoints in a sequence.

ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization

Train the Model

- Use a wide variety of navigation demonstrations. (Across Stanford, Berkeley, Seattle…)
- Sample a trajectory from the dataset, select P frames as observation, a frame d timestep away from now.
- Train model based on imitation objective:

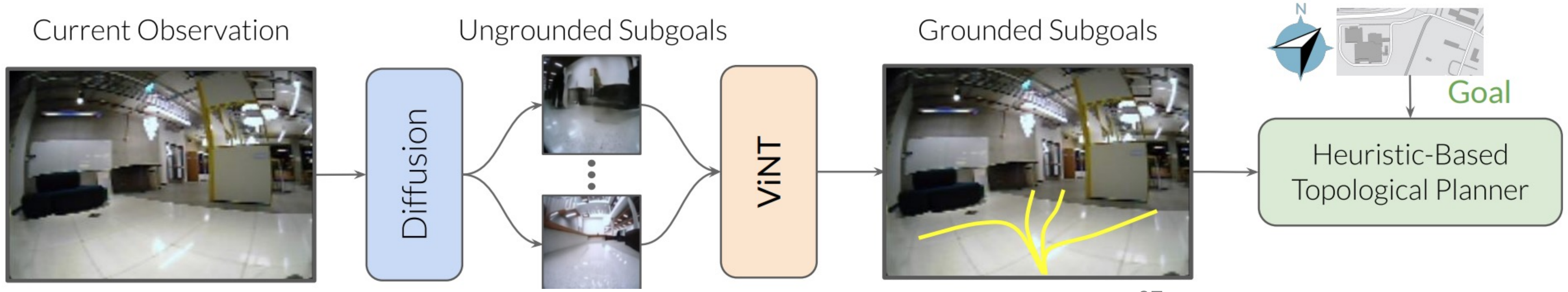
$$\mathcal{L}_{ViNT}(\phi,\psi,f) = \mathbb{E}_{\tau}\mathbb{E}_{t}\mathbb{E}_{d}\left[\log p(\hat{a}|f(\psi(o)_{t:t-P},\phi(o_t,o_s)) + \lambda \log p(d|f(\psi(o)_{t:t-P},\phi(o_t,o_s))\right]$$



ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization

Run the Model

- Build a topological graph $\mathcal{M}$ using subgoal image as node, ViNT distance prediction as edge.
- Use physical search with a topological graph-based planner.
- Use an image-to-image diffusion model to propose exploration targets (subgoal).
- Use ViNT to (ground) determine which are possible subgoals.



ViNT: A Large-Scale, Multi-Task Visual Navigation Backbone with Cross-Robot Generalization

For Downstream Tasks

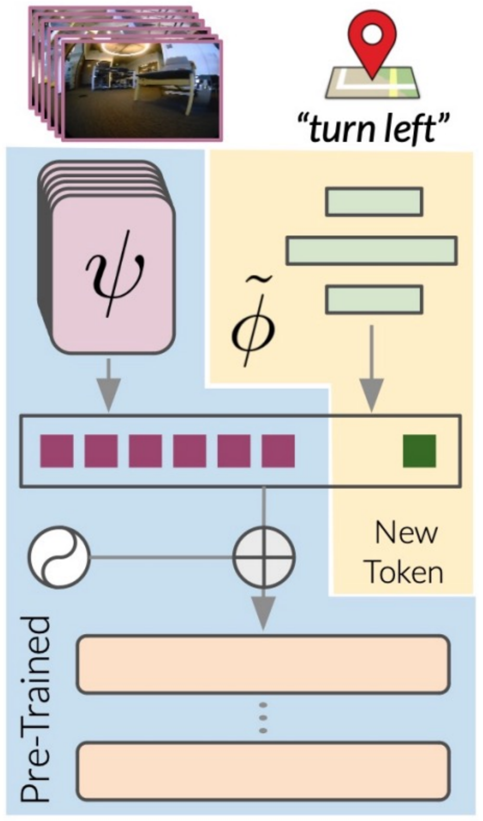


Figure 4: Adapting ViNT to different goals using a new tunable goal token.

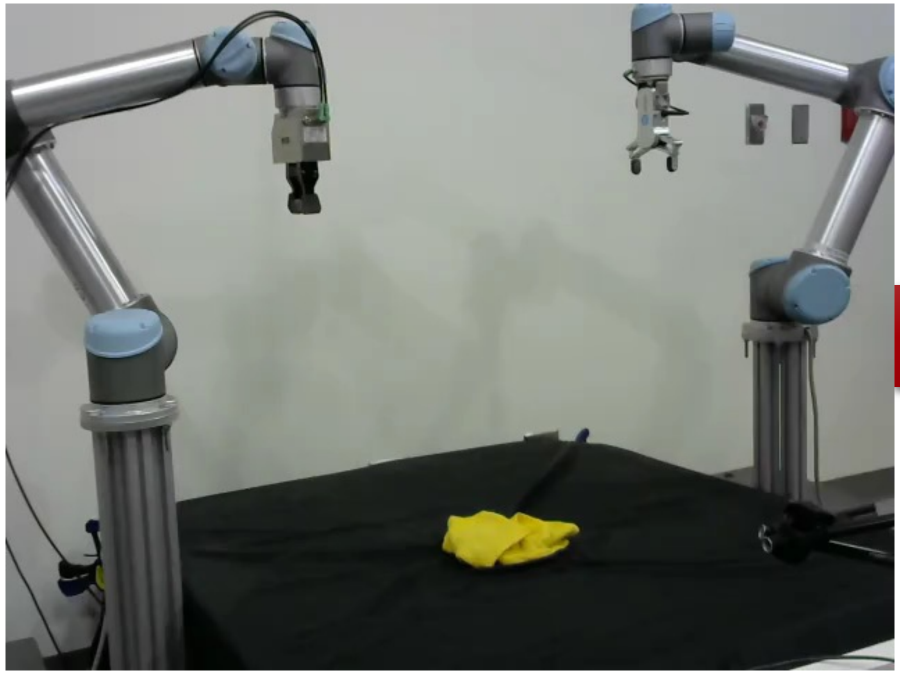
- Add a new branch of tokenizer to the network
- But incredibly small amount of data (1 hour compared to 80 hours)

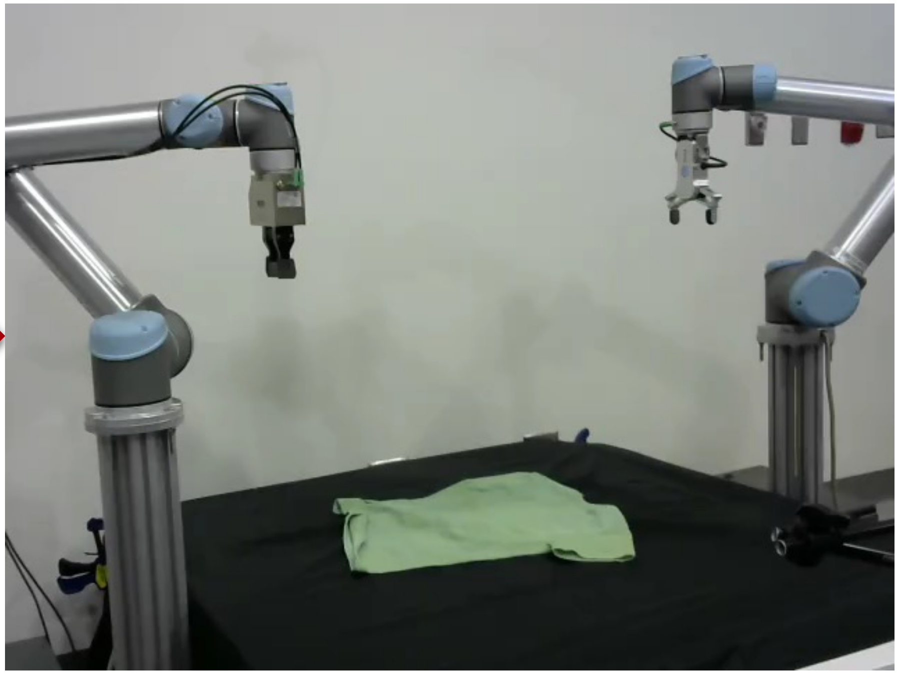
立志成才 报国裕民

Athletic Intelligence

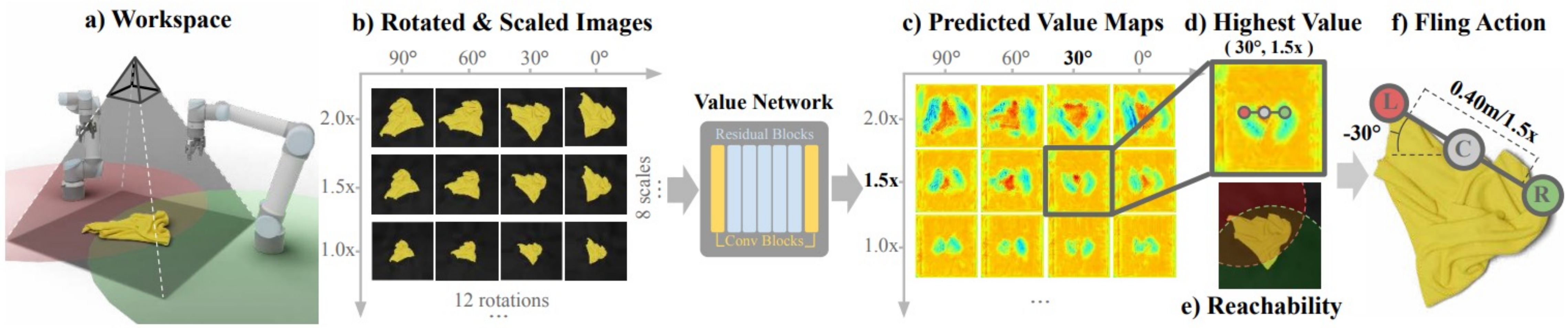
- FlingBot: The Unreasonable Effectiveness of Dynamic Manipulations for Cloth Unfolding
- TossingBot: Learning to Throw Arbitrary Objects with Residual Physics
- Dynamic Handover: Throw and Catch with Bimanual Hands
- Legged Locomotion in Challenging Terrains using Egocentric Vision
- Robot Parkour Learning
- Learn Humanoid Locomotion with Transformers

FlingBot: The Unreasonable Effectiveness of Dynamic Manipulations for Cloth Unfolding





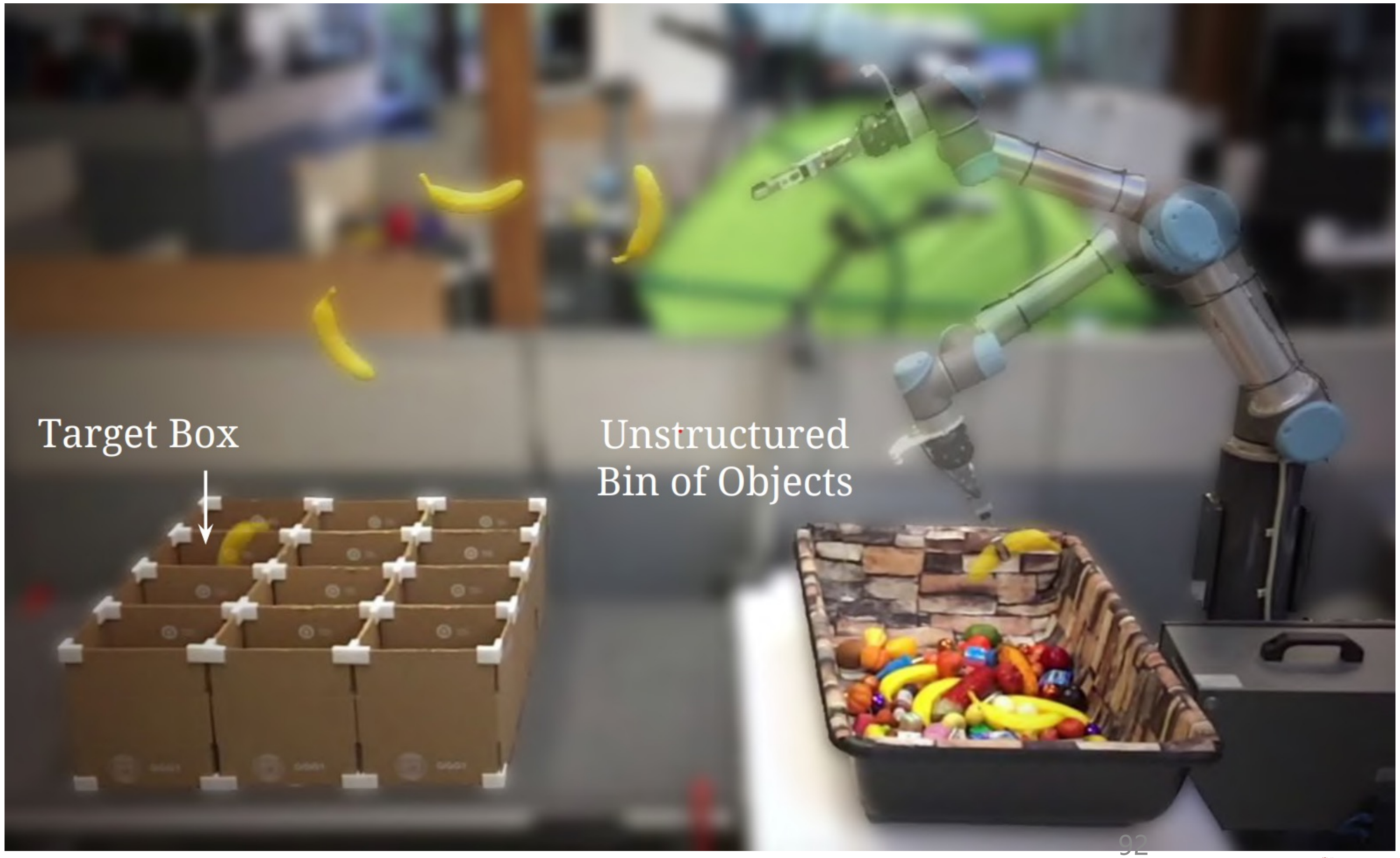
FlingBot: The Unreasonable Effectiveness of Dynamic Manipulations for Cloth Unfolding



Defines a grid of discrete action space; predict the position of the highest value.

Train with a predefined fling action.

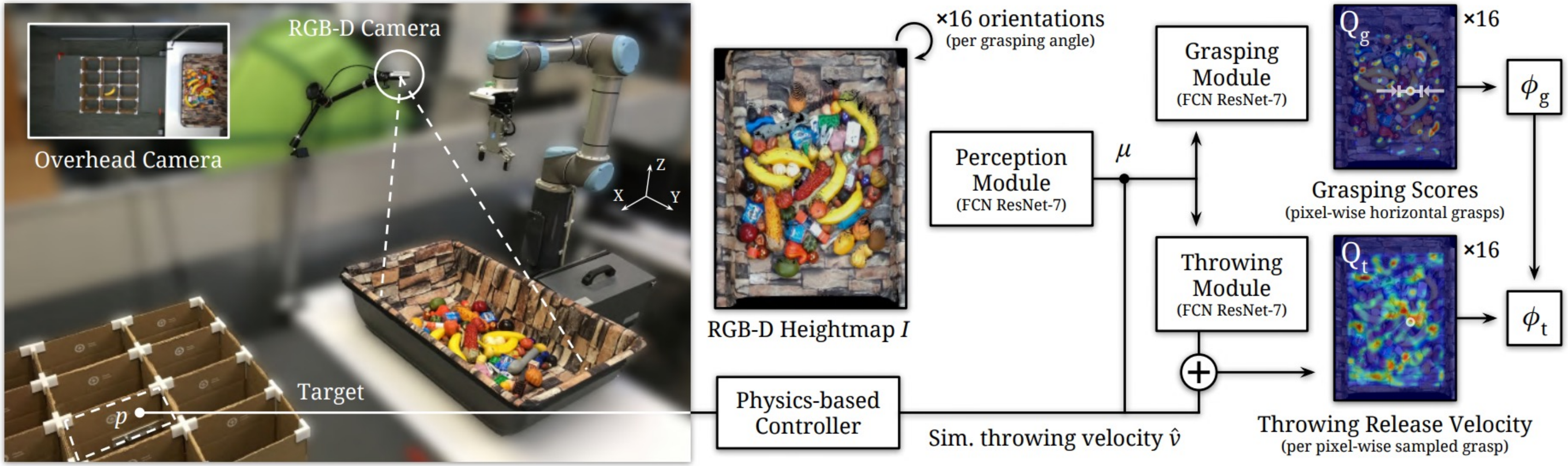
TossingBot: Learning to Throw Arbitrary Objects with Residual Physics



TossingBot: Learning to Throw Arbitrary Objects with Residual Physics

Tossing arbitrary objects is non-trivial even for human

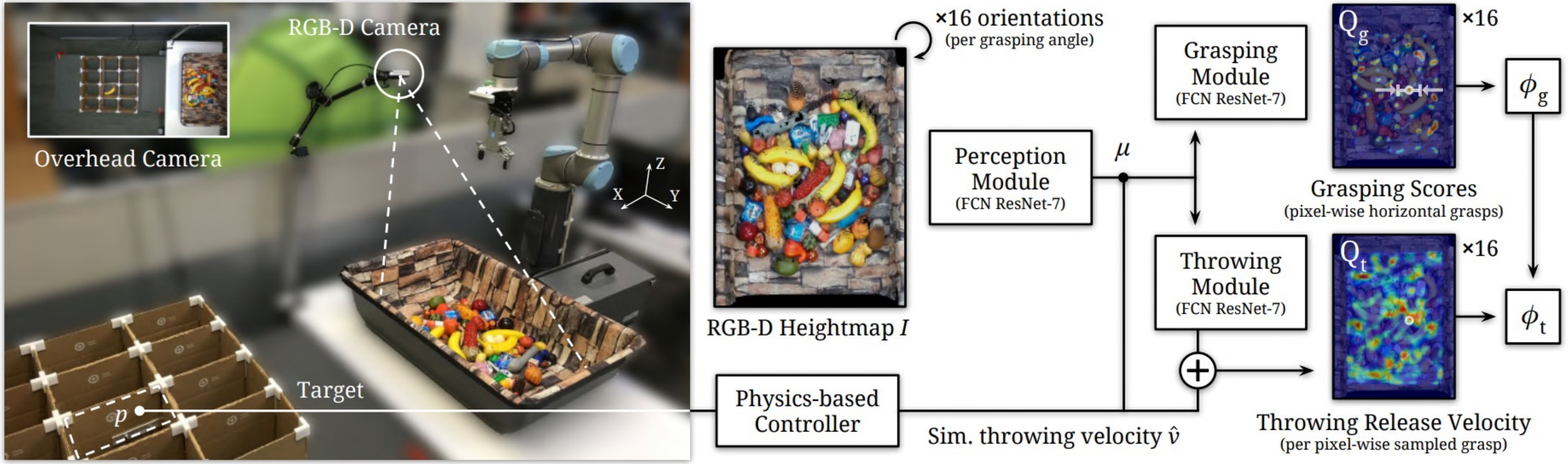
TossingBot: Learning to Throw Arbitrary Objects with Residual Physics



System Overview

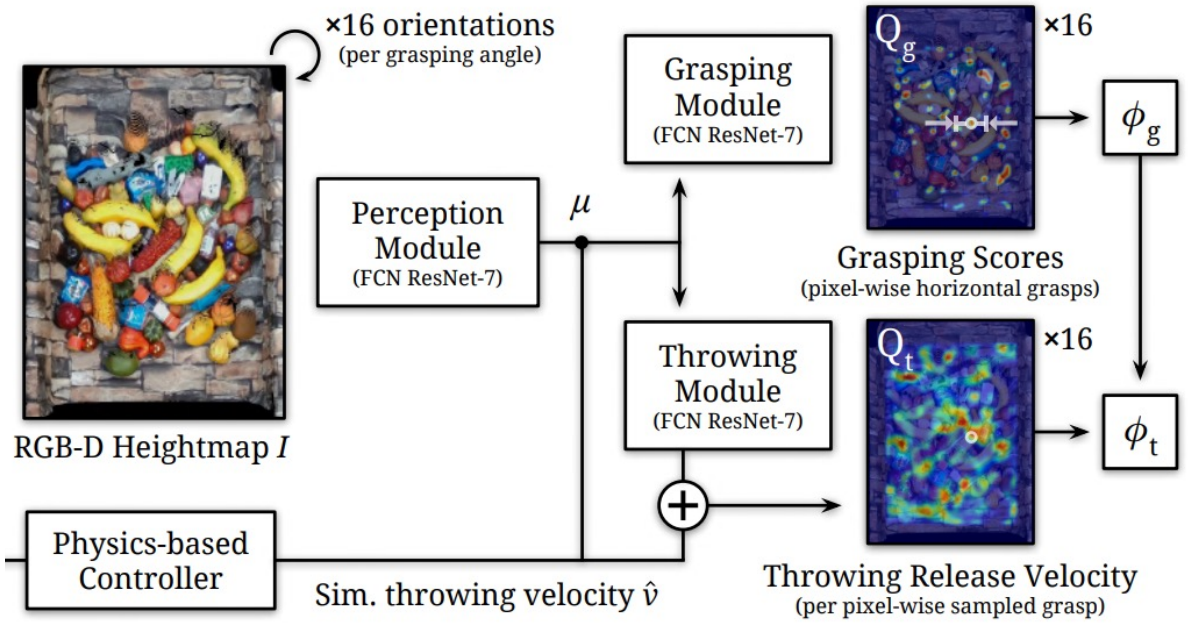
- RGB–D camera + target location –> throwing release velocity $\hat{v}$ and throw release point

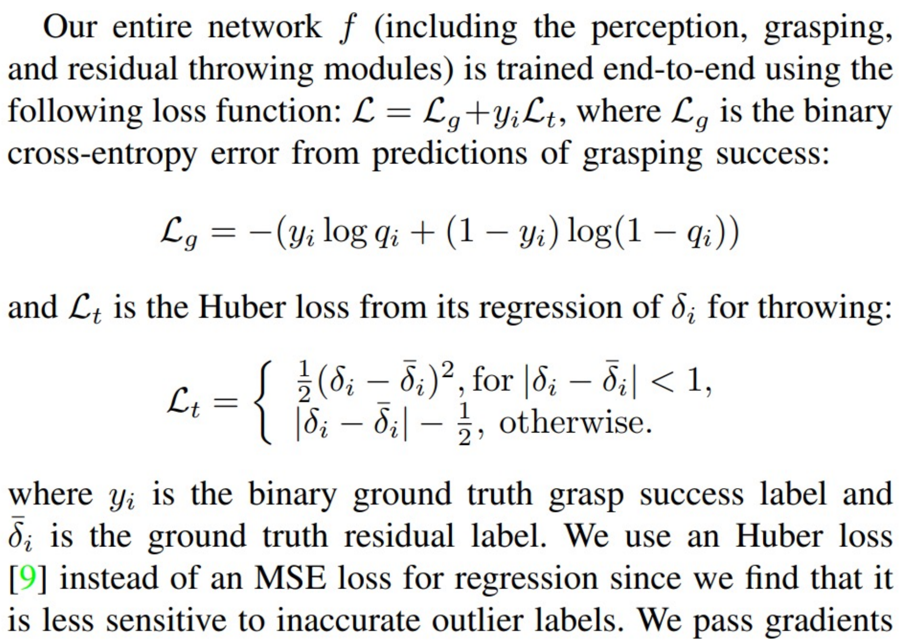
TossingBot: Learning to Throw Arbitrary Objects with Residual Physics



The work outputs throwing command $\phi_t = (r, v), r = (r_x, r_y, r_z), v = (v_x, v_y, v_z)$

TossingBot: Learning to Throw Arbitrary Objects with Residual Physics





Our entire network f (including the perception, grasping, and residual throwing modules) is trained end-to-end using the following loss function: $\mathcal{L} = \mathcal{L}_g + y_i \mathcal{L}_t$, where $\mathcal{L}_g$ is the binary cross-entropy error from predictions of grasping success:

$$\mathcal{L}_g = -(y_i \log q_i + (1 - y_i)\log(1 - q_i))$$

and $\mathcal{L}_t$ is the Huber loss from its regression of δ_i for throwing:

$$\mathcal{L}_t = \begin{cases} \frac{1}{2}(\delta_i - \bar{\delta}_i)^2, \text{for } |\delta_i - \bar{\delta}_i| < 1, \\ |\delta_i - \bar{\delta}_i| - \frac{1}{2}, \text{ otherwise.} \end{cases}$$

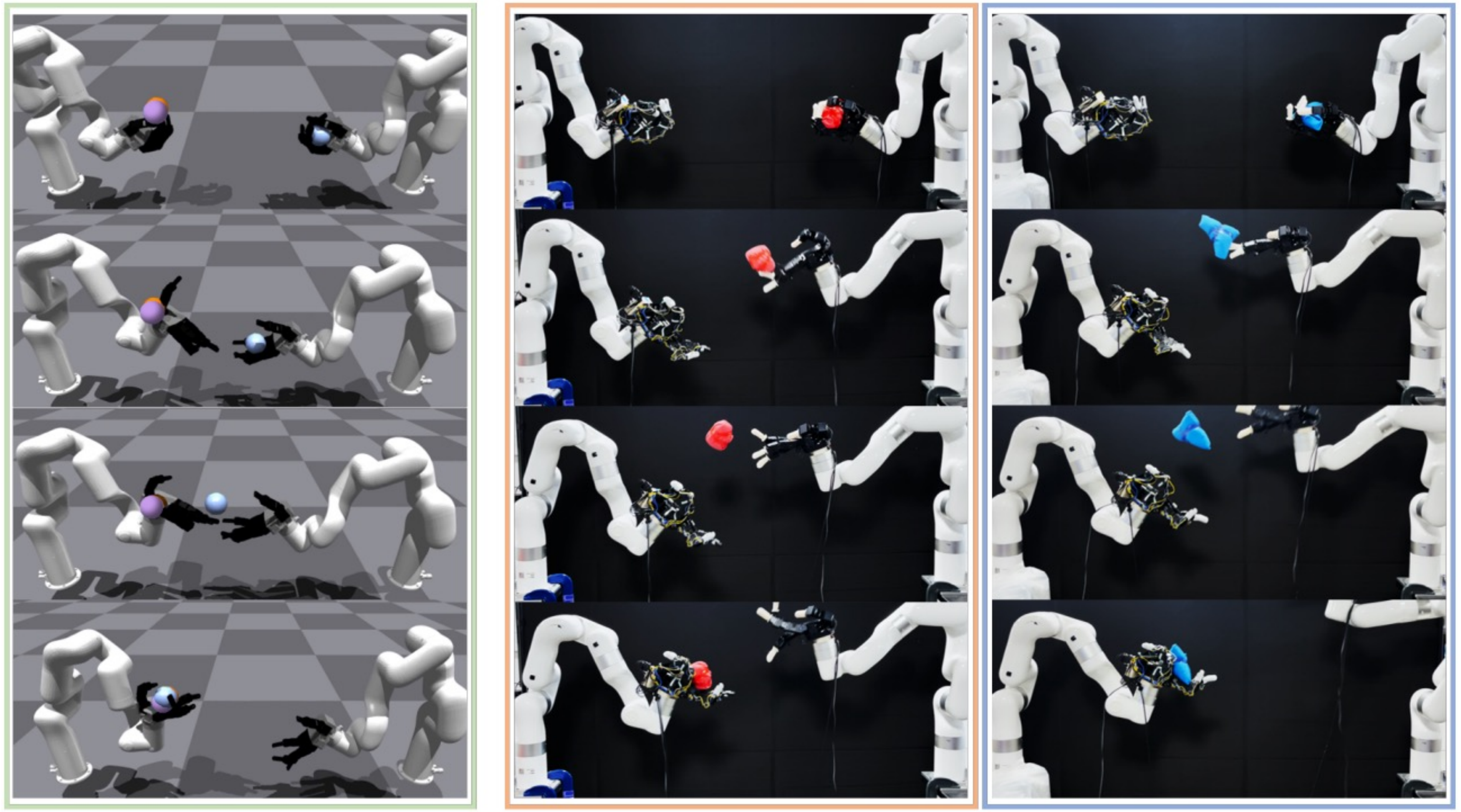
where y_i is the binary ground truth grasp success label and $\bar{\delta}_i$ is the ground truth residual label. We use an Huber loss [9] instead of an MSE loss for regression since we find that it is less sensitive to inaccurate outlier labels. We pass gradients

- binary cross-entropy loss for success grasping prediction
- Huber loss for velocity residual prediction

Using actual landing location $\overline{p}$ and executed release velocity v to supervise residual prediction

TossingBot: Learning to Throw Arbitrary Objects with Residual Physics

Dynamic Handover: Throw and Catch with Bimanual Hands

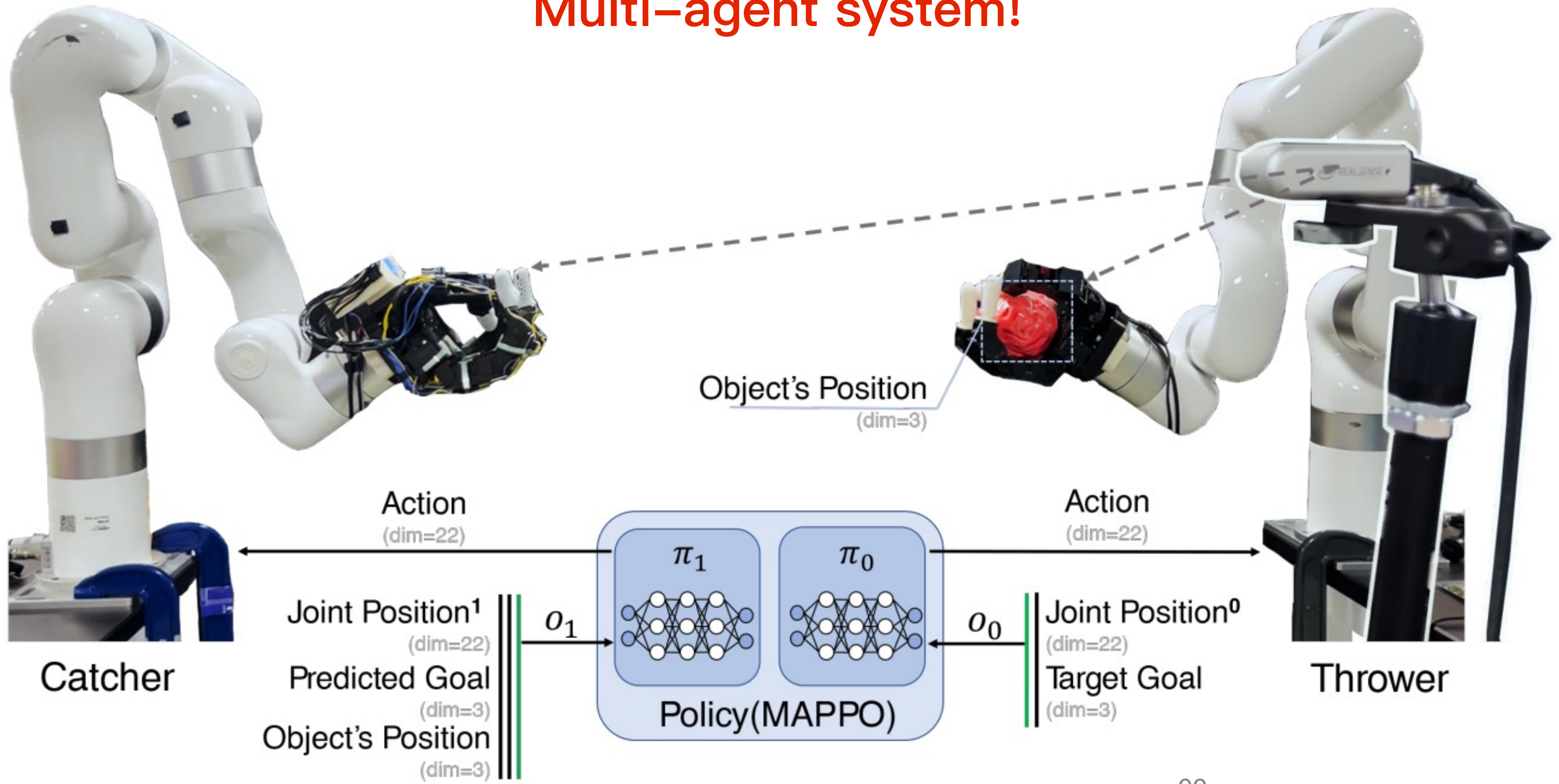


Training in Simulation

Test in Real Robot System

Dynamic Handover: Throw and Catch with Bimanual Hands

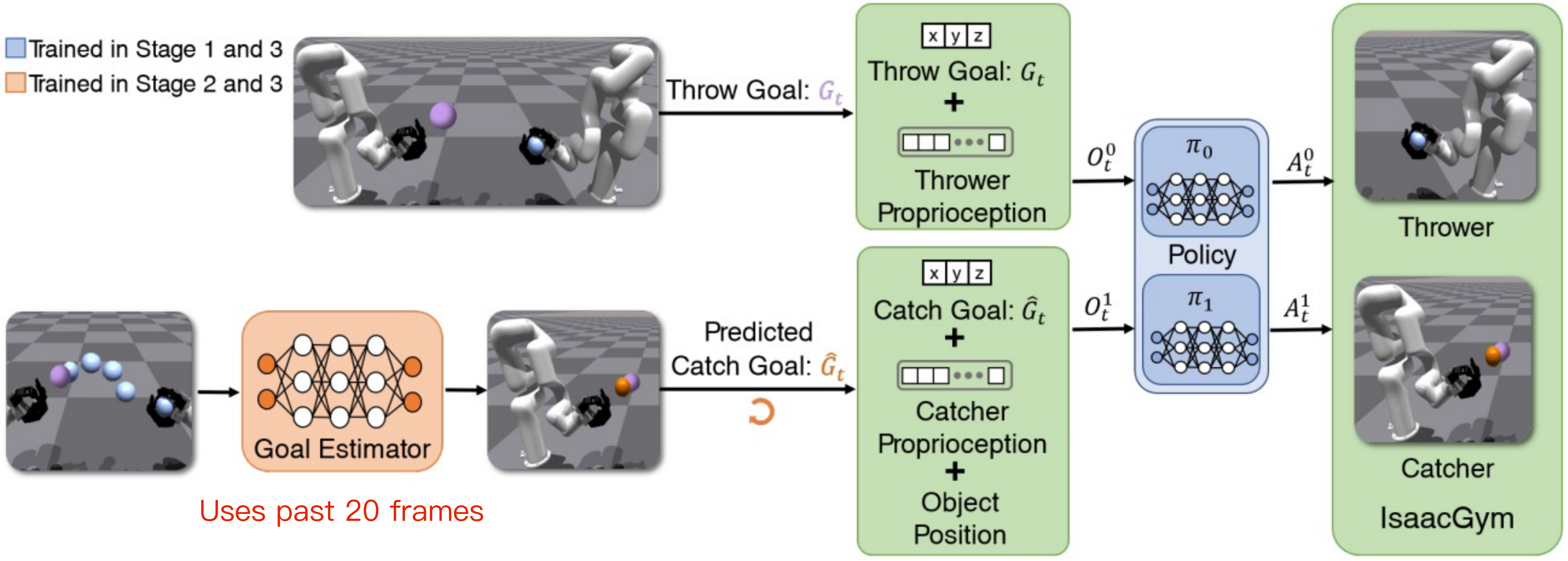
Multi-agent system!



立志成才 报国裕民

Dynamic Handover: Throw and Catch with Bimanual Hands

A set of pre–defined goal positions



Uses past 20 frames

Dynamic Handover: Throw and Catch with Bimanual Hands

Stage 1: Multi–Agent Reinforcement Learning

Multi–Agent RL:

- no parameter / info sharing between agents
- same input / output definition

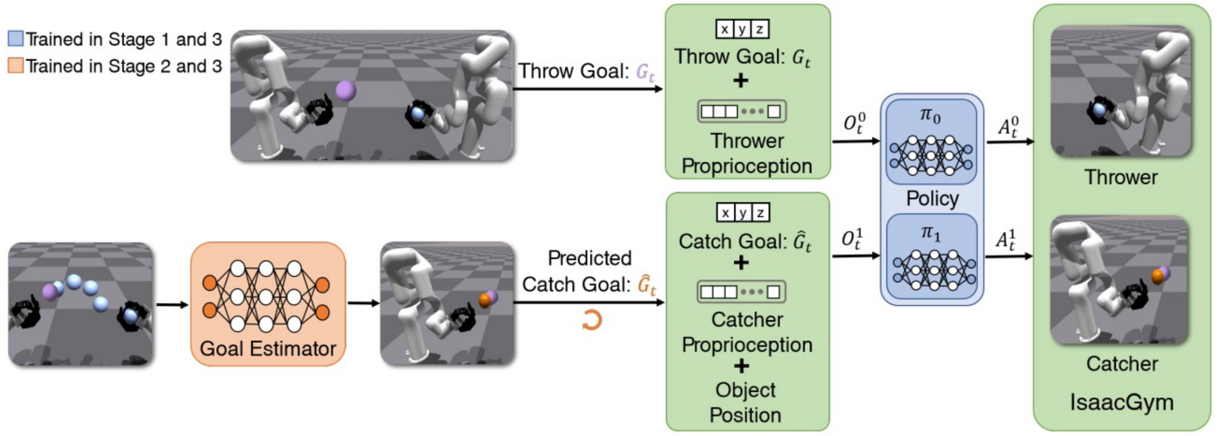
Stage 2: Goal Estimator Learning

Due to sim–to–real issue, thrower in real world **does not** always throw the object in the given goal.

- Using past k frames as input
- Predict the goal position

Stage 3: Jointly training two agent and goal estimator

"This joint training helps reduce compounding errors when integrating the goal estimator with the policy."



Dynamic Handover: Throw and Catch with Bimanual Hands

Settings	Known Obj.	Novel Obj.
w/o Multi-Agent	0.89 ± 0.07	0.24 ± 0.05
w/o Goal Estimation	0.88 ± 0.04	0.22 ± 0.04
w/o Both	0.93 ± 0.07	0.12 ± 0.06
Ours	**0.95 ± 0.07**	**0.37 ± 0.04**

Table 1: **Ablation Study in Simulation:** Success Rate of throwing and catching task on different objects in simulation. We use **11** trained objects and **14** novel objects. The results are averaged on 5 seeds, each seed has 100 trails.

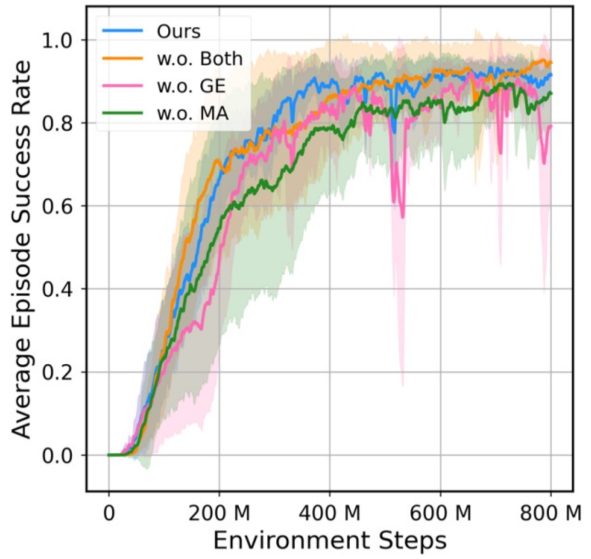


Figure 4: **Training Curves.** The plot shows multi-object training curves of our method and 3 baselines.

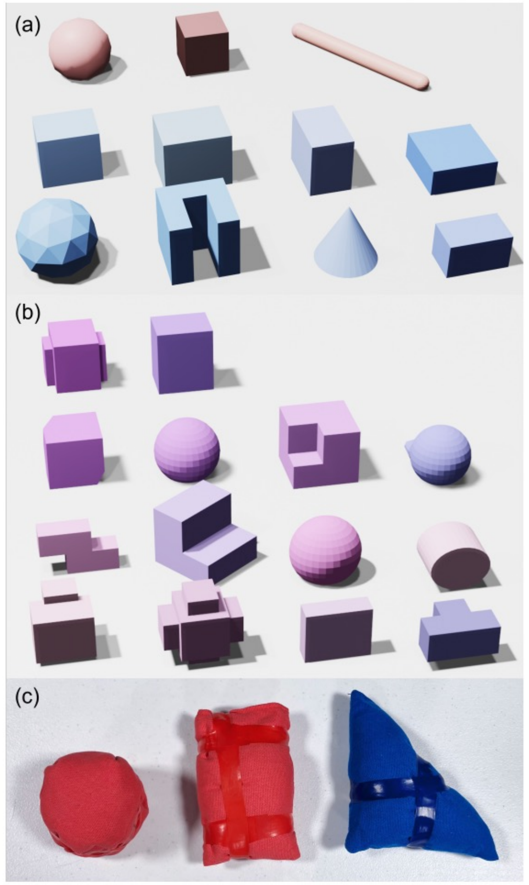
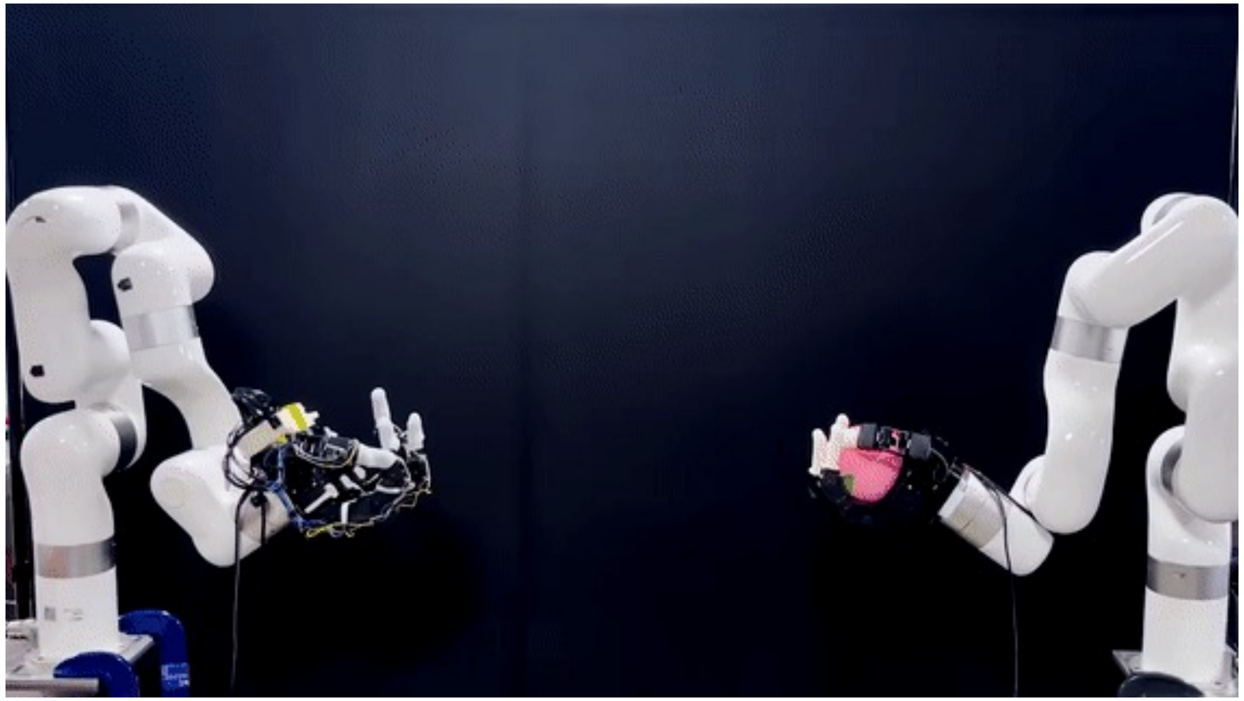
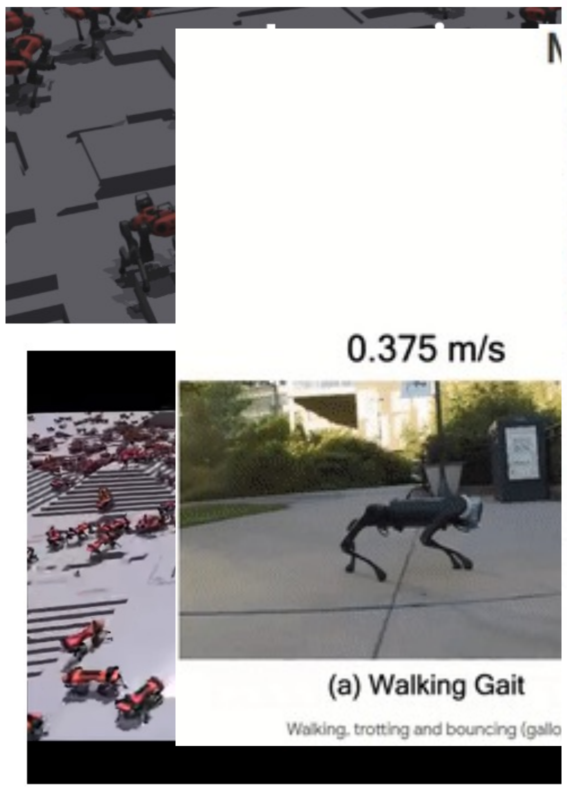
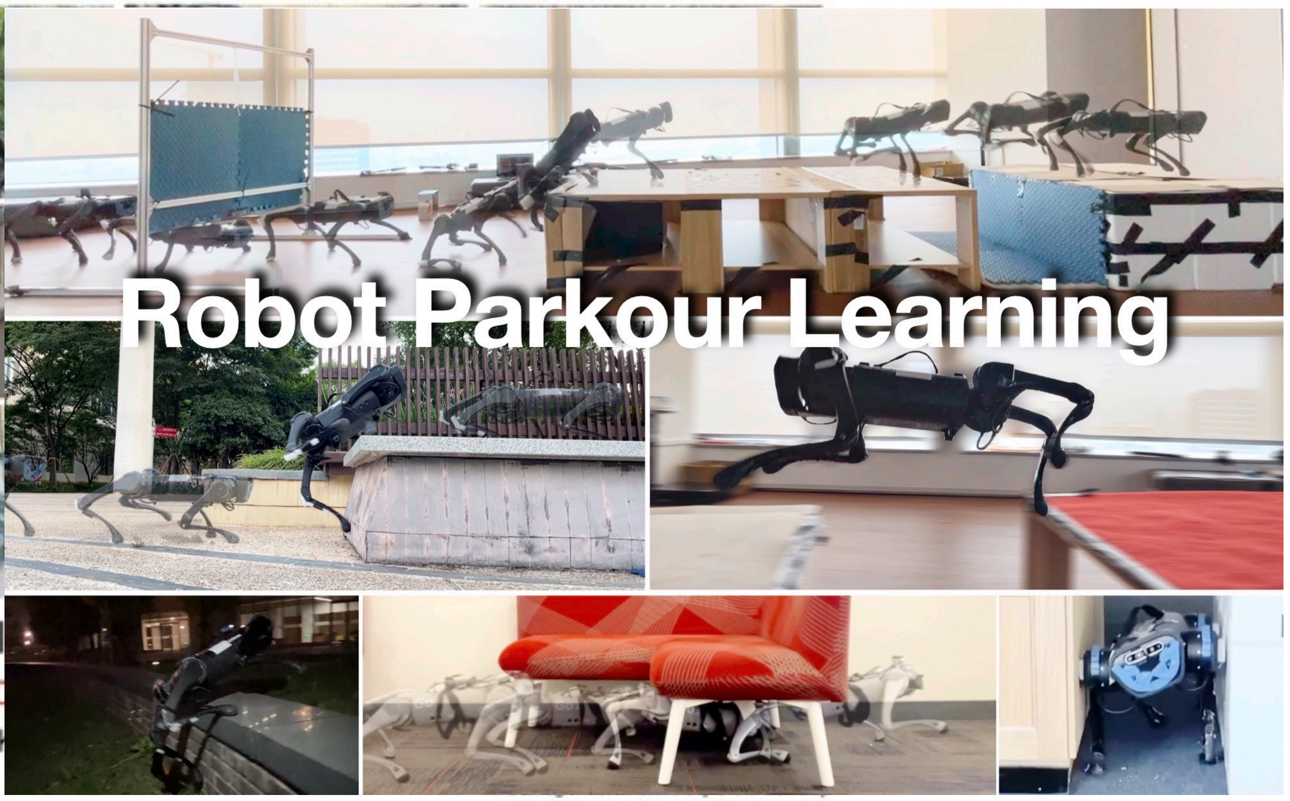


Figure 5: **Objects Sets.** (a) Training objects. (b) Additional objects in evaluation. (c) Real-world objects.



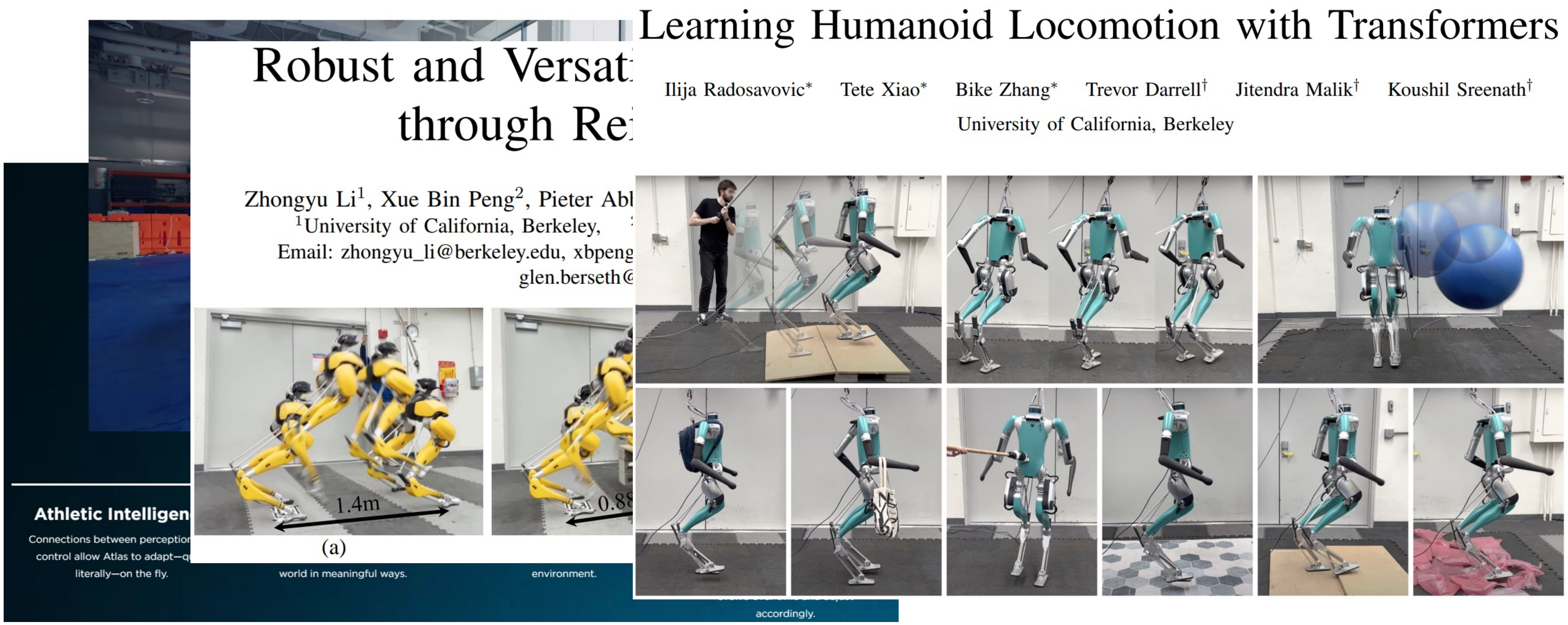
Athletic Intelligence (quadruped robot)





2020: RL is able to work on quadruped locomotion
2021: RL is simple enough to train quadruped robot
2022: quadruped robot can utilize vision to guide the gait
2023: quadruped robot outperforms all other mobile robots

Athletic Intelligence (bipedal robot)



Athletic Intelligen
Connections between perceptio
control allow Atlas to adapt—q
literally—on the fly.

world in meaningful ways.

environment.

accordingly.

Robust and Versat
through Re

Zhongyu Li[1], Xue Bin Peng[2], Pieter Abb
[1]University of California, Berkeley,
Email: zhongyu_li@berkeley.edu, xbpeng
glen.berseth@

(a)

Learning Humanoid Locomotion with Transformers

Ilija Radosavovic* Tete Xiao* Bike Zhang* Trevor Darrell† Jitendra Malik† Koushil Sreenath†
University of California, Berkeley