Sistemi Intelligenti Avanzati – 2025/2026
Università degli Studi di Milano

Robot Learning for Motion Planning, Control and Navigation

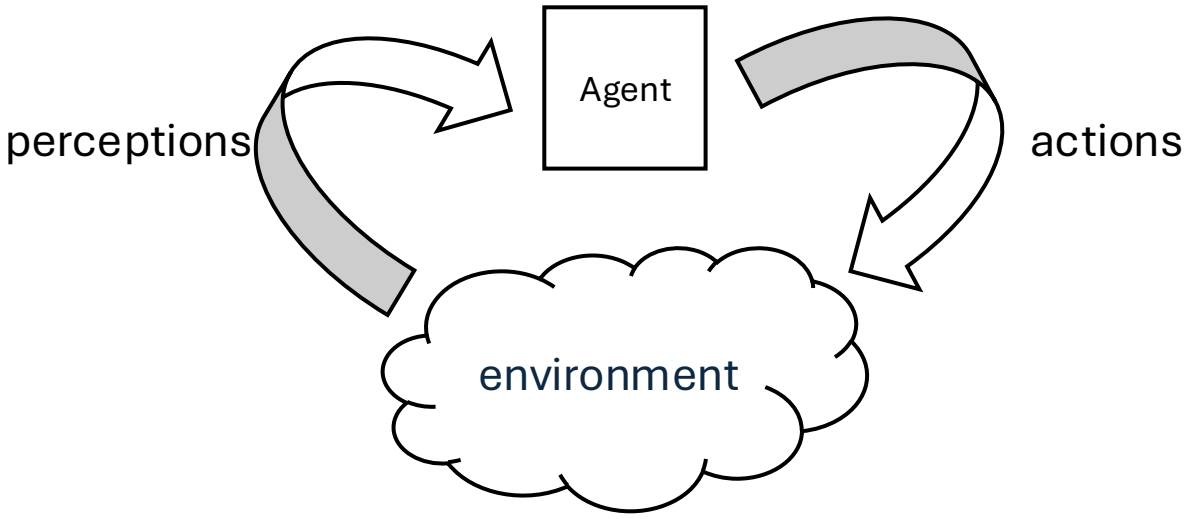
Matteo Luperto
Dipartimento di Informatica
matteo.luperto@unimi.it

Robot development tools

- Heterogeneous set of software/tools/simulators
- Some of them cover the whole robot stack
- Some of them abstract most of the robot stack and focus on a target component
- Some of them cover all robot types
- Some of them focus on specific type of robots

Simulation vs Dataset

- Dataset can be used to train DNN for solving complex task (mostly offline learning)
- Robot learning is more complex as it involves interacting with the environment (online learning)

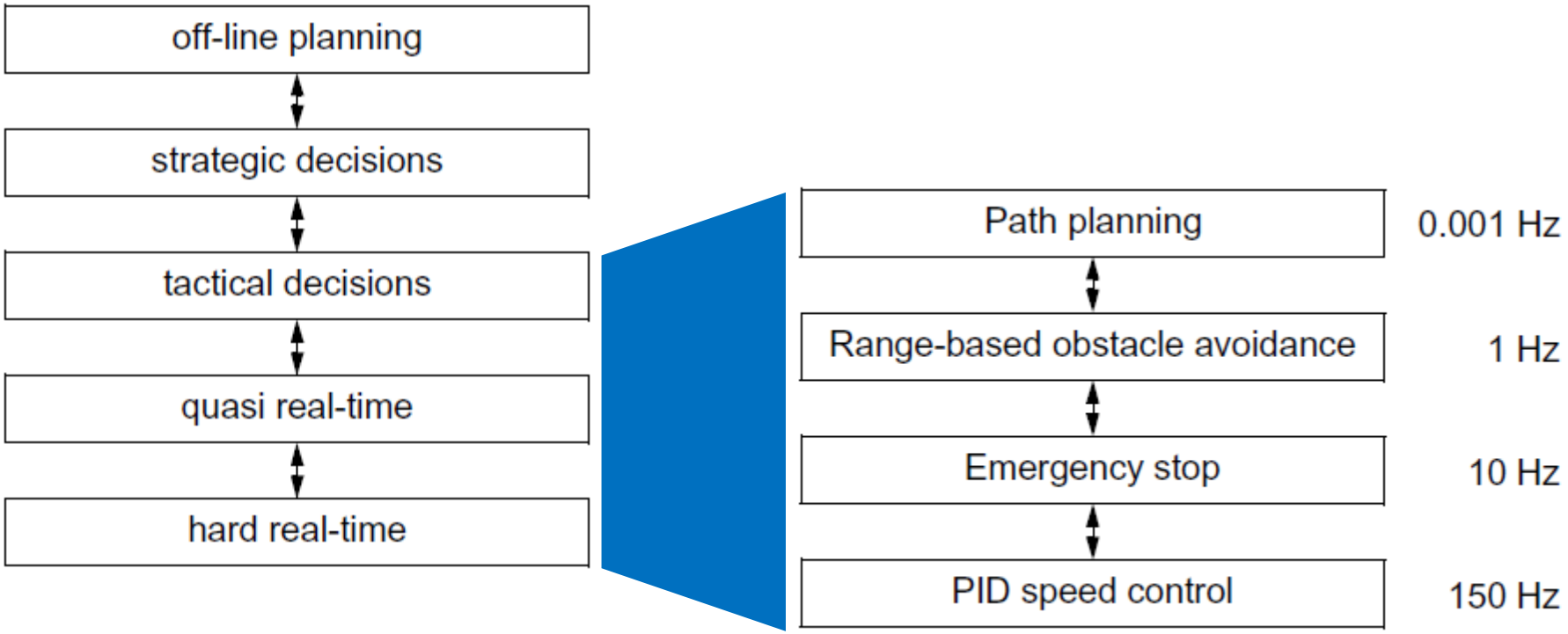


is a set of software libraries and tools that help you build robot applications.

From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

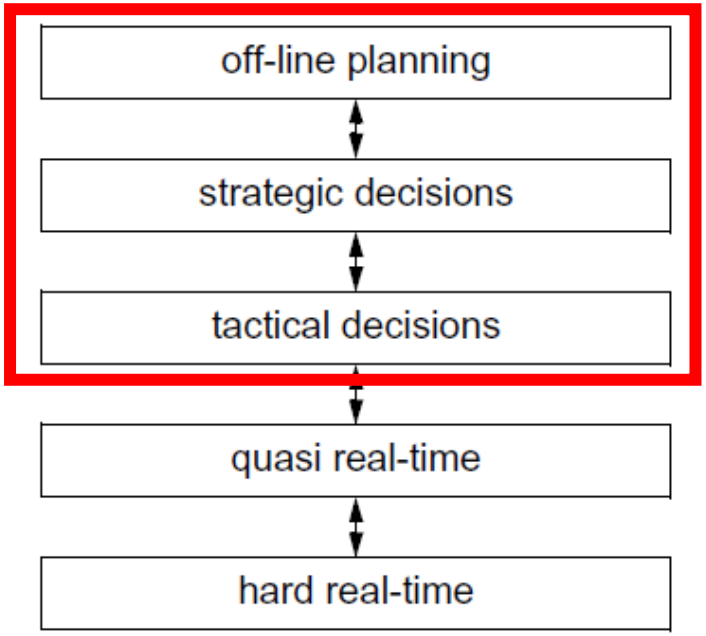
Widely used multi-tool toolbox, it is used for teaching, research, commercial use; it suits all type of robots, sensors, ...

Robot Architecture contd.



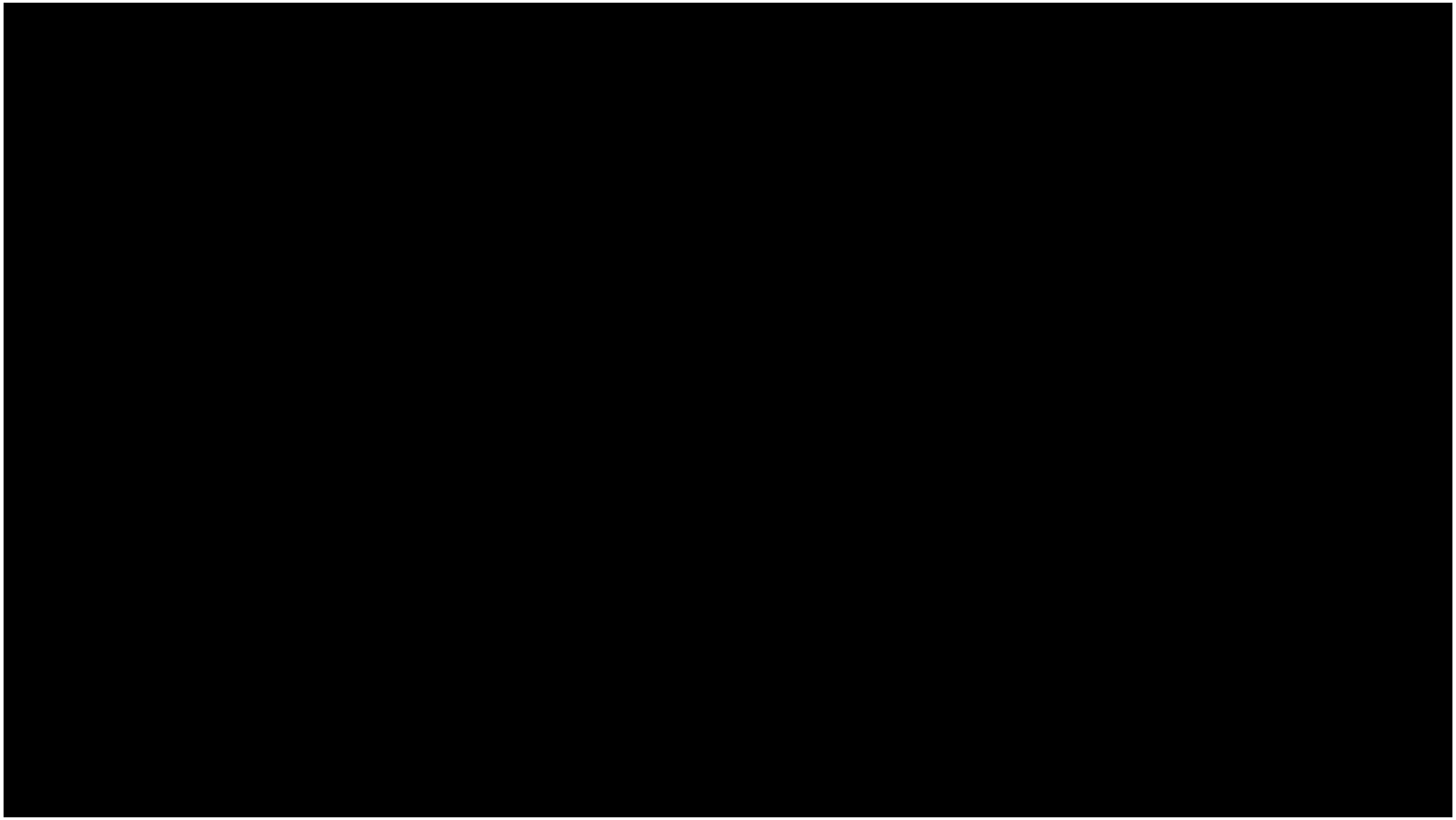
Robots as Embodied AIs

An intelligent agent that interacts with the environment through a physical body within that environment.



Line of research that focuses on the AI part of the robot, while abstracting/simplifying the «robot» part

Habitat



Habitat

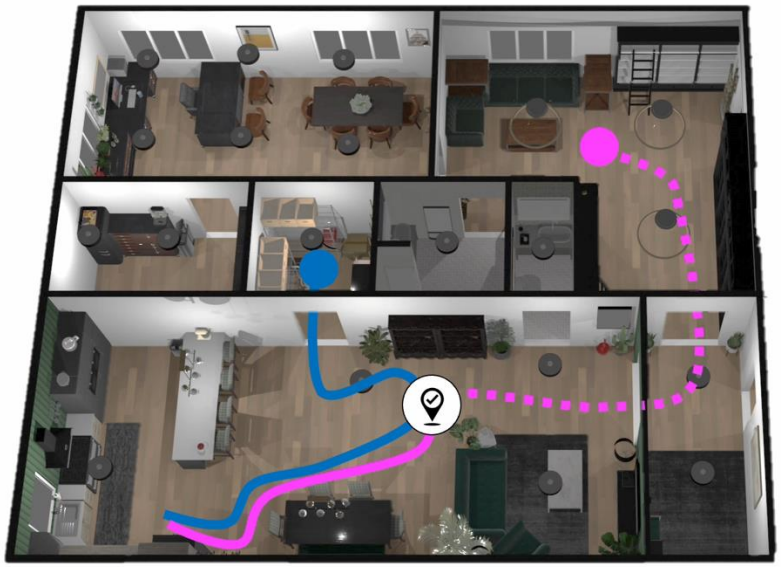
Developed and maintained by Meta/Facebook is designed for developing Embodied AI abstracting from the robot hardware, in photorealistic environments

PROs

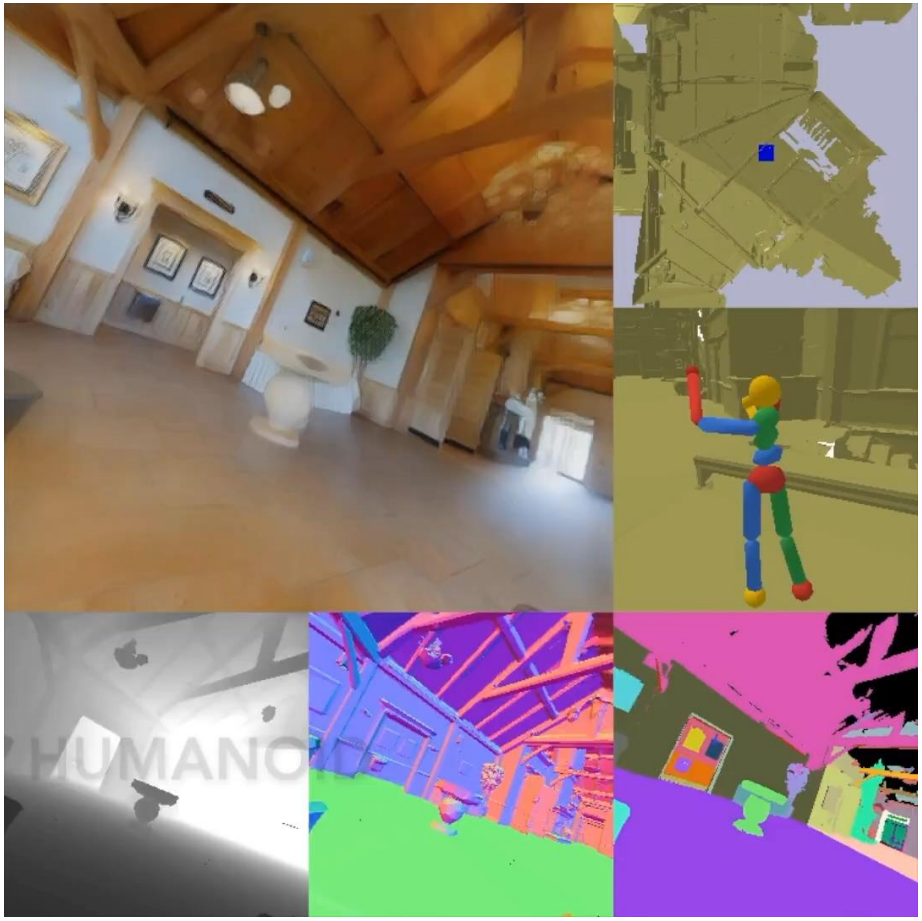
- High level abstraction
- Embeds different environments and tasks

CONs

- Robot perception is not realistic
- Static environments (Habitat 1-2)
- Synthetic ones (Habitat 3)



Gibson



Gibson

Developed by Stanford, offers simulations in 3D photorealistic environments

PROs

- Physics and realism better than Habitat
- Works with ROS + others

CONs

- Static environments, Sim2 real gap
- Limited number of environments

iGibson

Interactive version of Gibson for robot learning

PROs

- Realistic interaction and customization of environments
- ROS integration

CONs

- Environments are synthetic
- Limited number of environments

iGibson

Physical Interaction with Articulated Objects

More than 500 object models

Sourced from open source datasets and cleaned up

Articulated objects can be operated by agents

AI2THOR

AllenAI suite for embodied AI, offers multiple environment and a generative model for creating others (Procedural Thor, or ProcThor)

PROs

- Focusing on interaction with the environments
- Some environments have a real counterpart
- Generative environments (ProcThor, Holodeck, …)

CONs

- Sim2Real gap

AI2THOR


AI2THOR
Open Source Interactive Environments for Embodied AI

MuJoCo

MuJoCo is a free and open source physics engine that aims to facilitate research and development in robotics, biomechanics, graphics and animation, and other areas where fast and accurate simulation is needed.

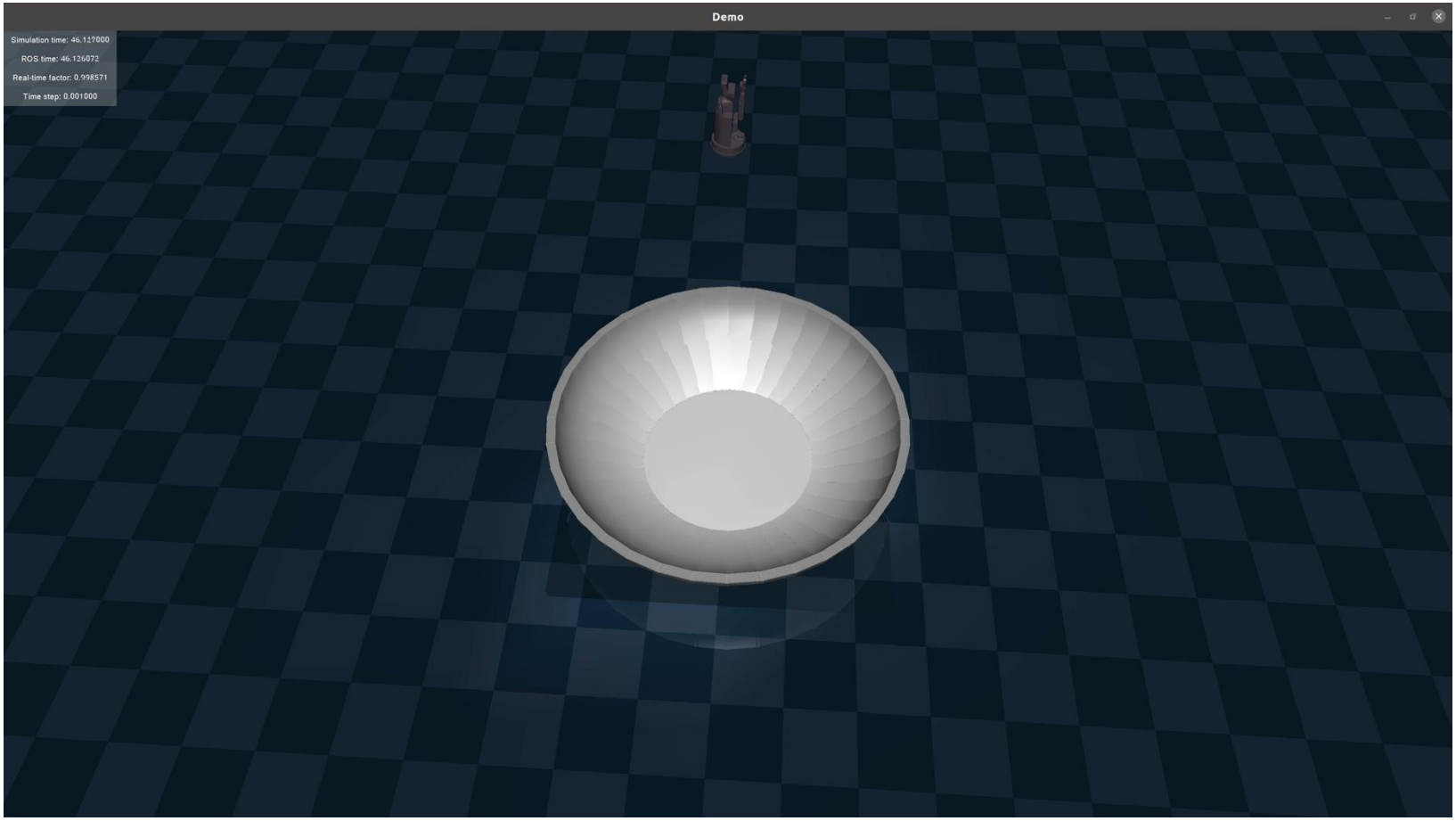
PROs

- Realistic physic simulation
- Low Sim2Real Gap with most robotic platforms
- Low computational capabilities

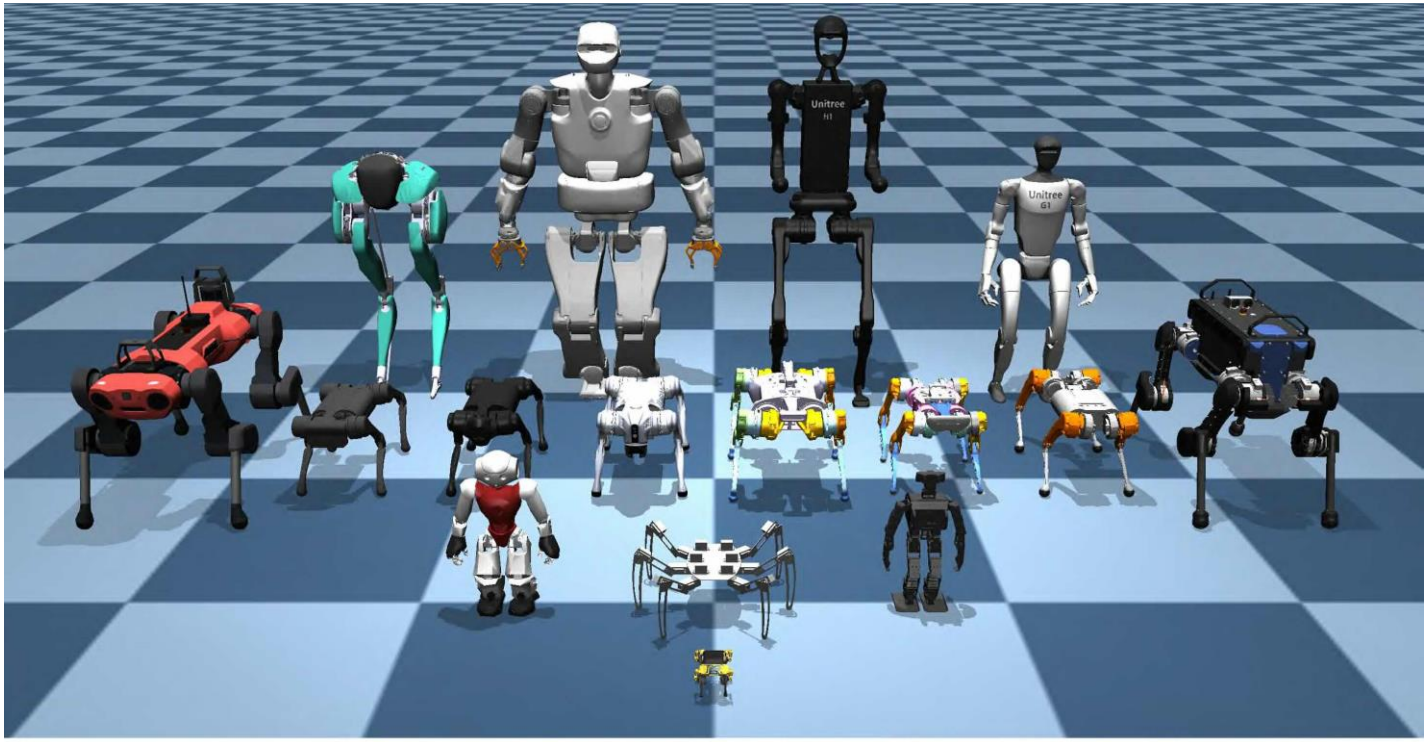
CONs

- Focus on the robot – no environment
- No GPU support (now partially solved)

MuJoCo


Demo
Simulation time: 46.127000
ROS time: 46.126072
Real-time factor: 0.998571
Time step: 0.001000

MuJoCo



NVIDIA Isaac™

NVIDIA Isaac™ is an AI robot development platform consisting of NVIDIA-accelerated libraries, application frameworks, and AI models that accelerate the development of AI robots such as autonomous mobile robots (AMRs), arms and manipulators, and humanoids.

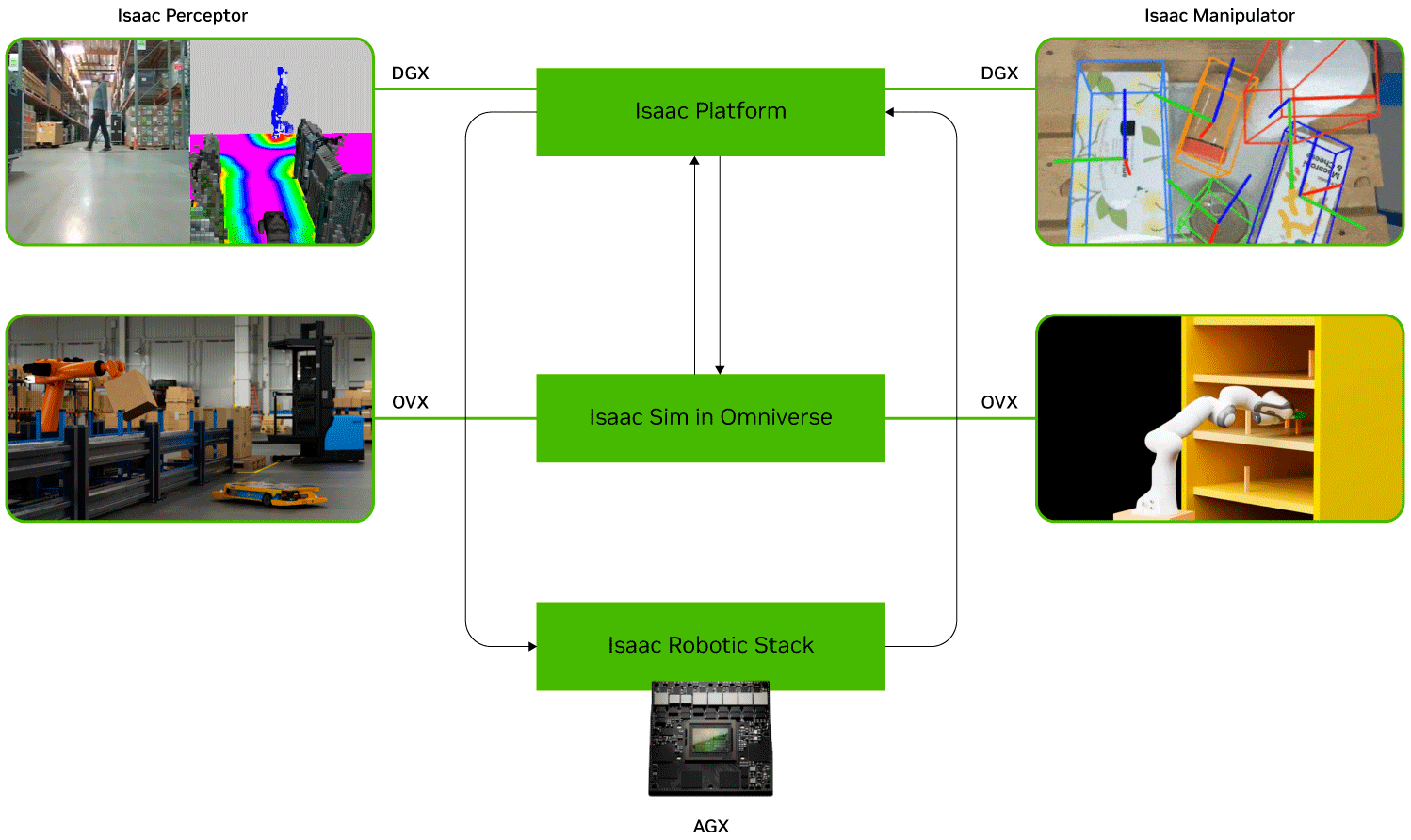
PROs

- Photorealistic simulation
- GPU parallalization
- Integrated with ROS
- Multi-task (manipulation, locomotion, navigation, task planning)

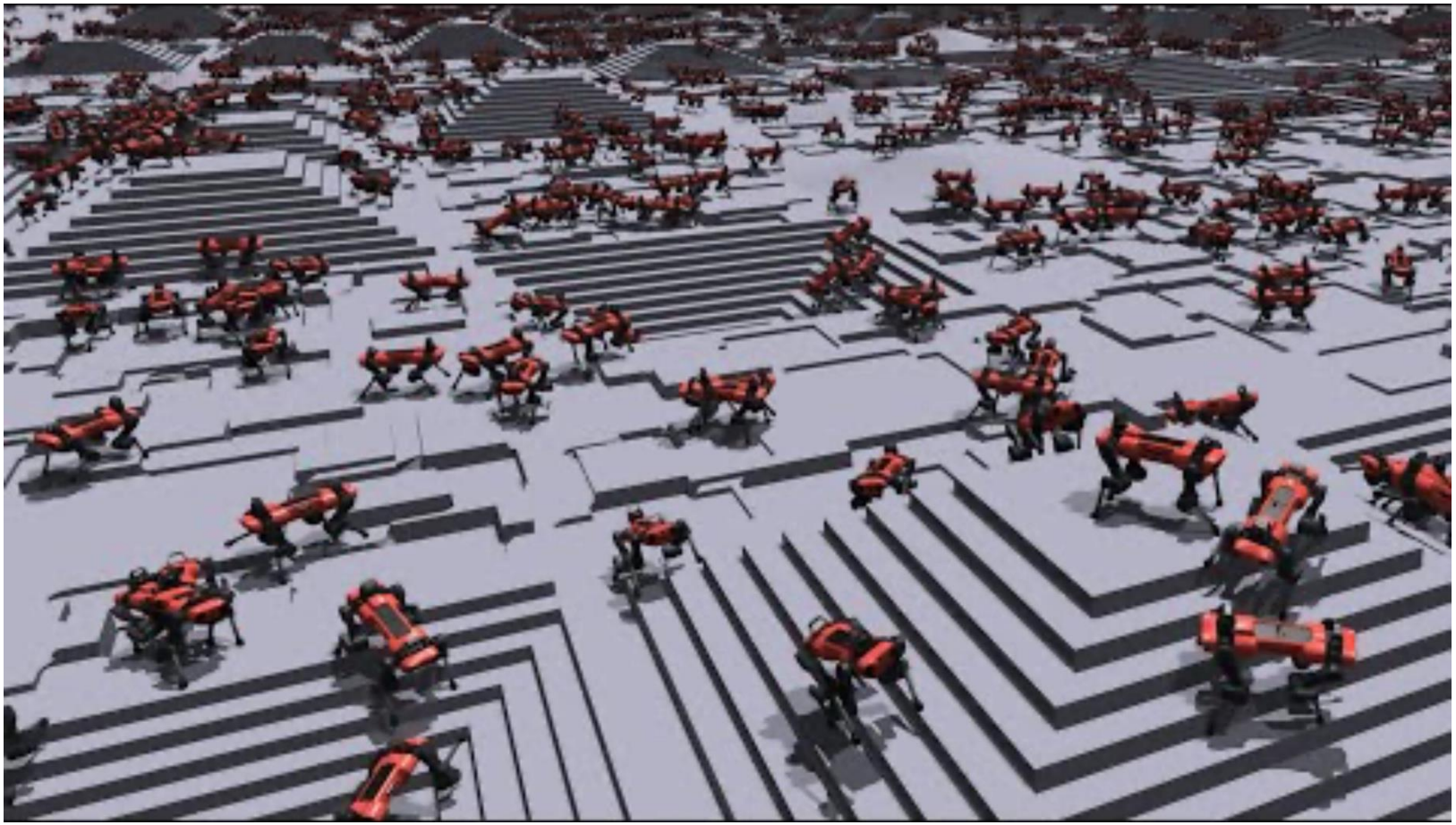
CONs

- Customization
- Hi-end computational requirements

NVIDIA Isaac™



NVIDIA Isaac™



Learning to Walk in Minutes Using Massively Parallel Deep RL, Rudin et al., CORL 2021

CARLA is an example of a framework/simulator designed for one task, autonomous driving

PROs

Scalability via a server multi-client architecture: multiple clients in the same or in different nodes can control different actors.

Flexible API: CARLA exposes a powerful API that allows users to control all aspects related to the simulation, including traffic generation, pedestrian behaviors, weathers, sensors, and much more.

Autonomous Driving sensor suite: users can configure diverse sensor suites including LIDARs, multiple cameras, depth sensors and GPS among others.

Fast simulation for planning and control: this mode disables rendering to offer a fast execution of traffic simulation and road behaviors for which graphics are not required.

Maps generation: users can easily create their own maps following the ASAM OpenDRIVE standard via tools like RoadRunner.

Traffic scenarios simulation: our engine ScenarioRunner allows users to define and execute different traffic situations based on modular behaviors.

ROS and ISAAC integration and ISAAC

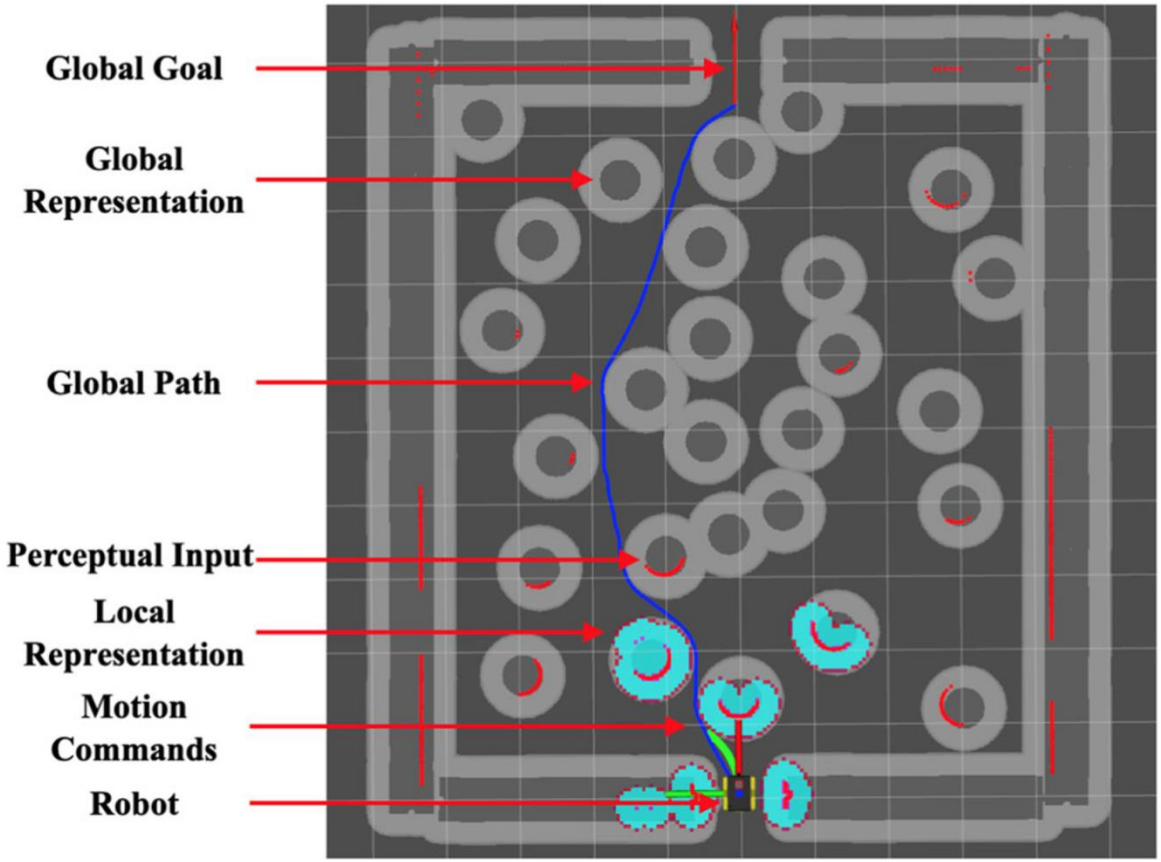
Autonomous Driving baselines: we provide Autonomous Driving baselines as runnable agents in CARLA, including an AutoWare agent and a Conditional Imitation Learning agent.

CONs

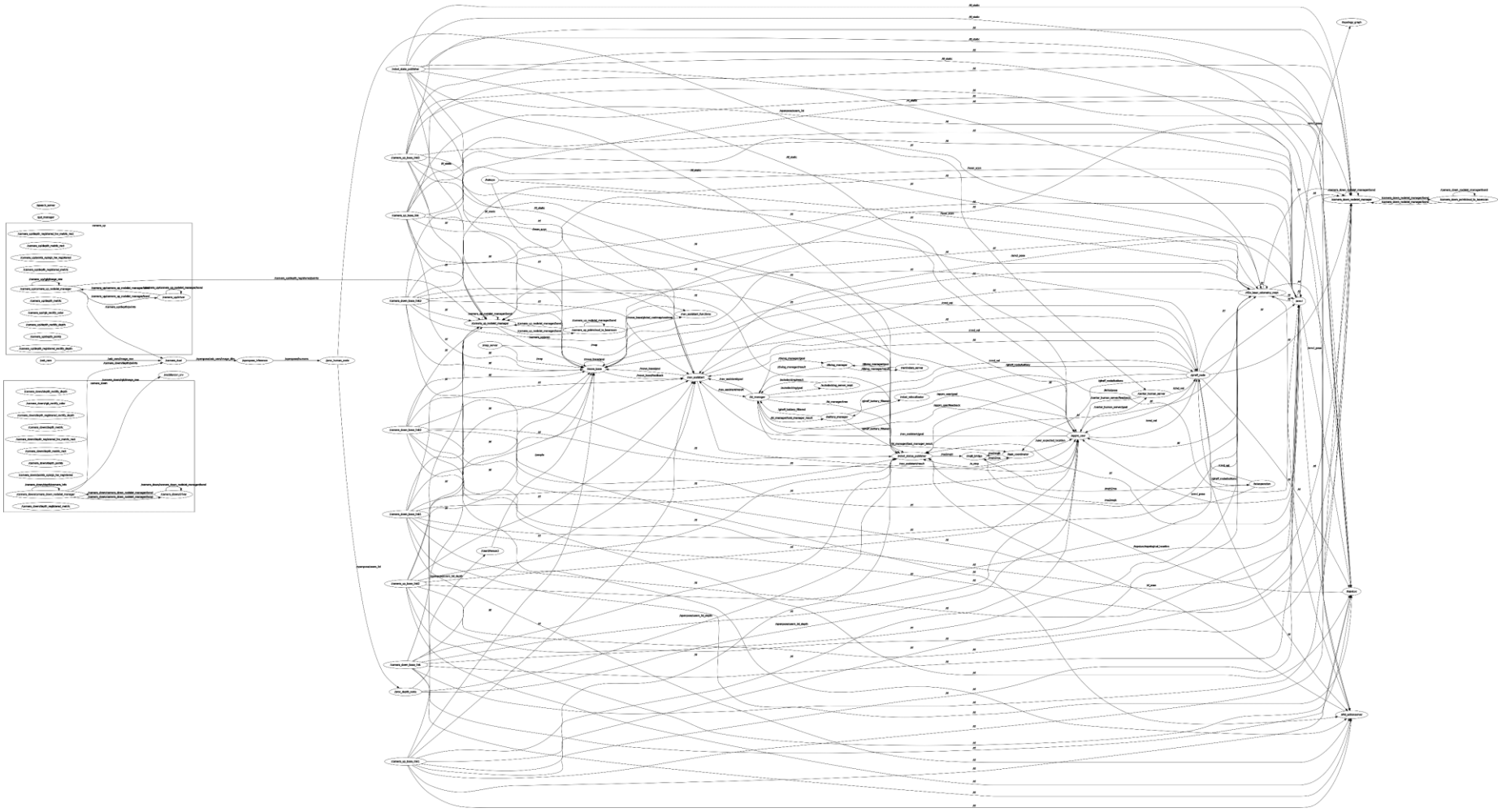
- Sim2Real gap is still an issue

CARLA

Classical Navigation System



Classical Navigation System



Classical Navigation System

PROS

- Stable
- Robust
- Integrated
- Low Sim2Real Gap

CONS

- Suboptimal performance
- Parametrization
- !Learning from experience

Learned Navigation System

CONS

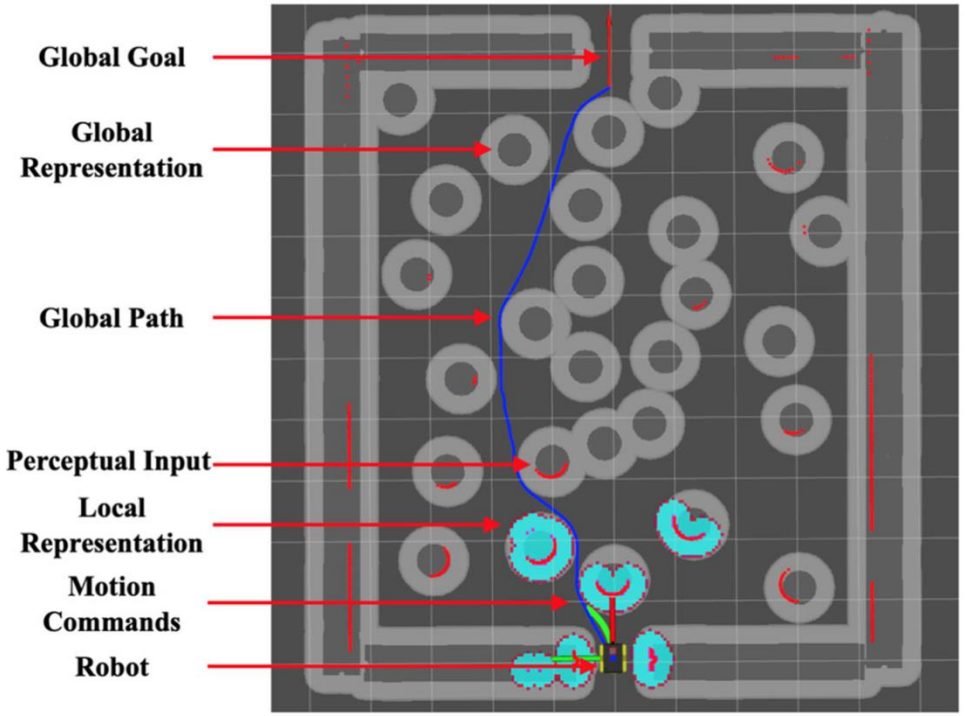
- !Stable
- !Robust
- !Integrated
- High Sim2Real Gap

PROS

- Better performance?
- Auto-Parametrization
- Learning from experience

~~Classical~~ Learning-Based Navigation System

1, Learning replaces the entire navigation stack (including end-to-end)
2. Learning navigation subsystem
3. Learning navigation component



~~Classical~~ Learning-Based Navigation System

What types of robots are more suited for learning?

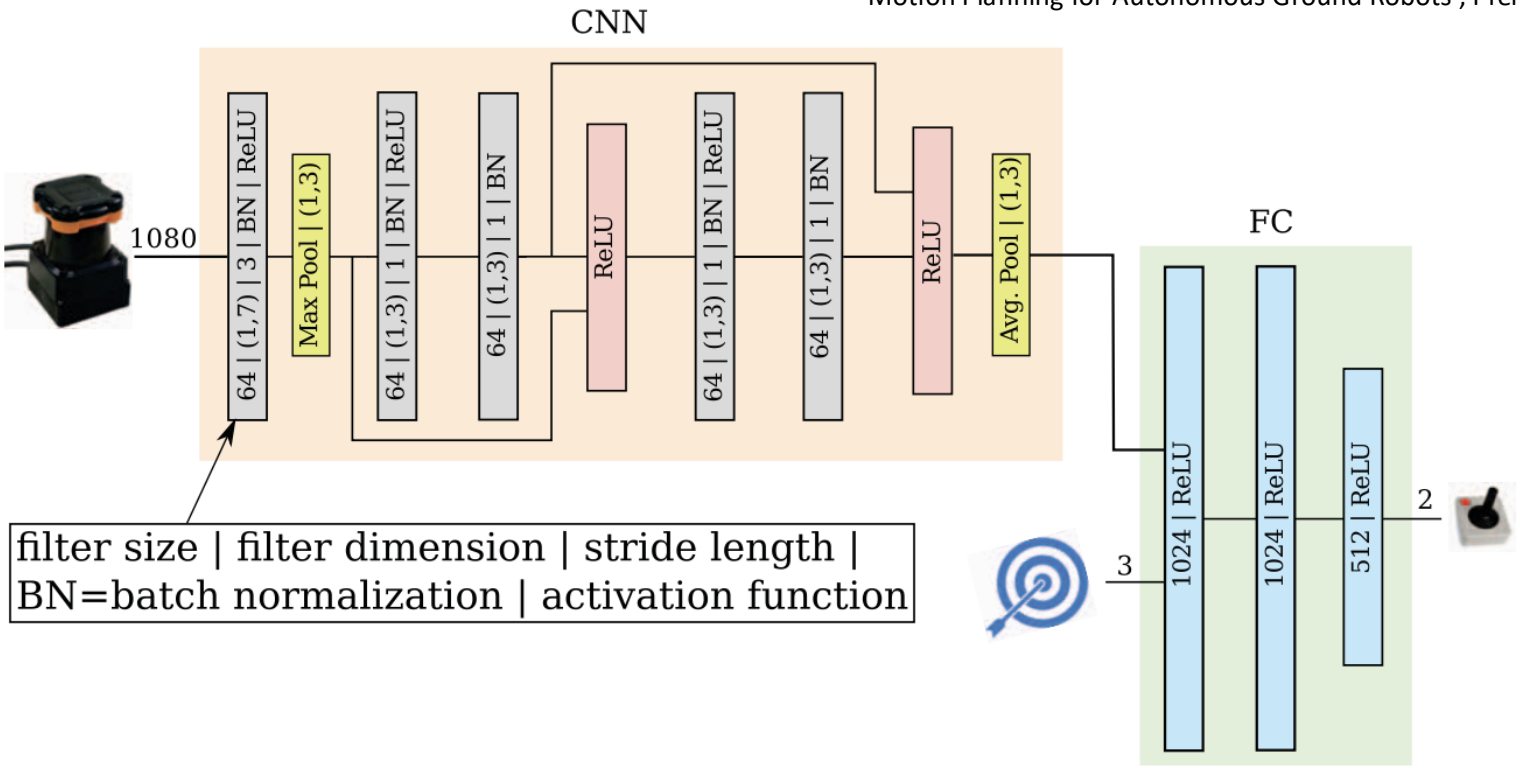
Learning the full stack

Fixed goal (or “point-goal”) navigation

- Geometric navigation
  model navigation as in the classic navigation stack
  PRO: simpler
  CONS: no semantic, similar limitations
- Visual-based navigation
  more “human-like”
  PRO: semantic and vision
  CONS: complexity, generalization

Learning the full stack

From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots , Pfeiffer et al., ICRA 17

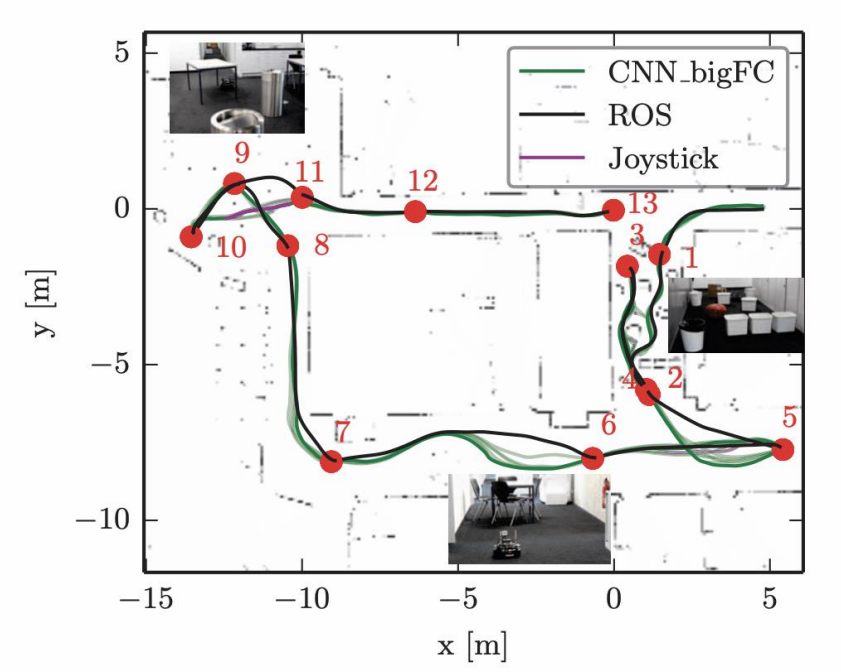


Early works: learning ROS nav stack with a CNN with Imitation Learning

Learning the full stack

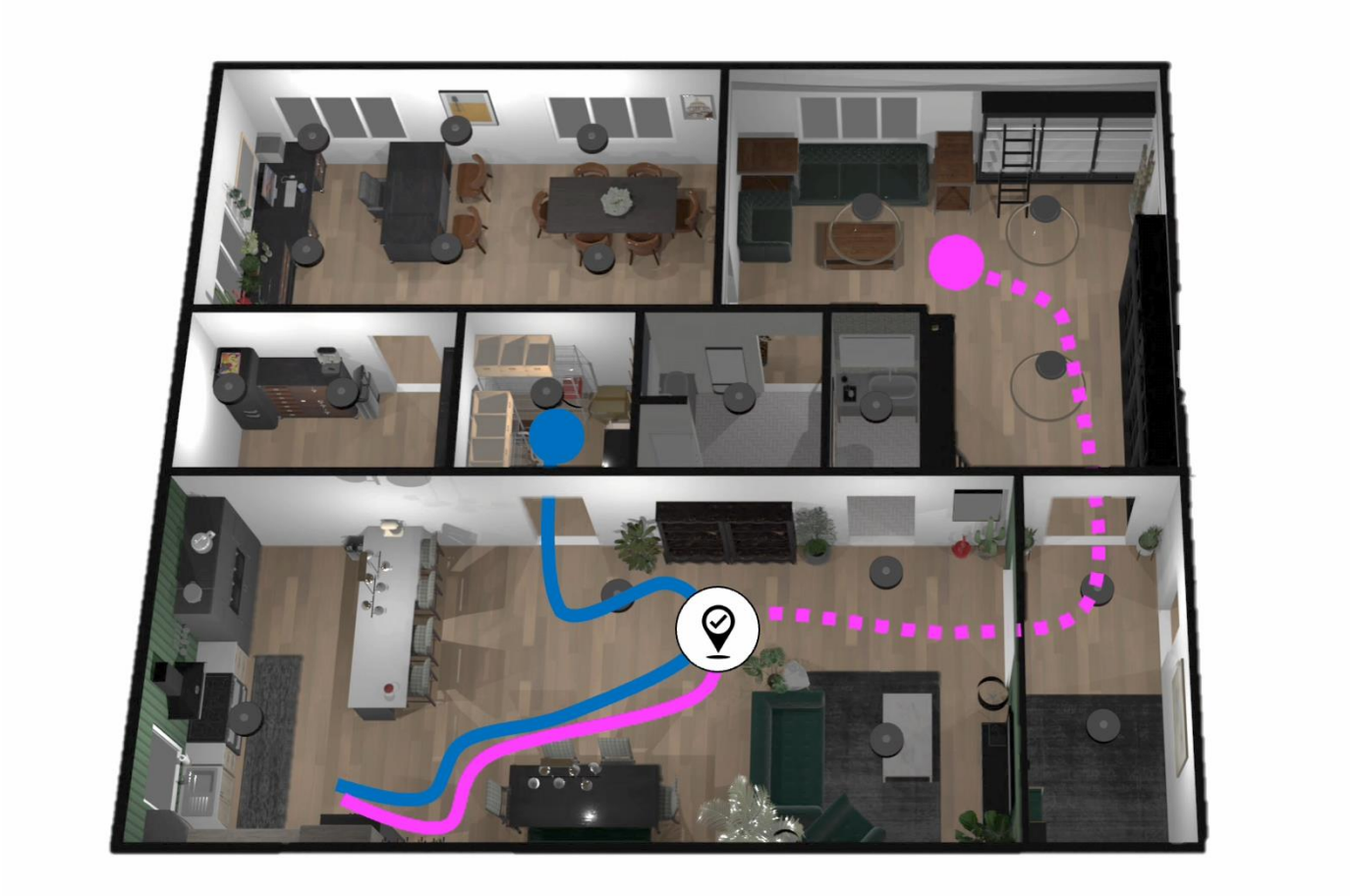
Target	1	2	3	4	5	6	7	8	9	10	11	12	13
CNN_smallFC	1	0	1	0	4	0	0	0	2	4	6	0	0
CNN_bigFC	0	0	1	1	0	1	0	0	0	0	4	1	0

From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots , Pfeiffer et al., ICRA 17

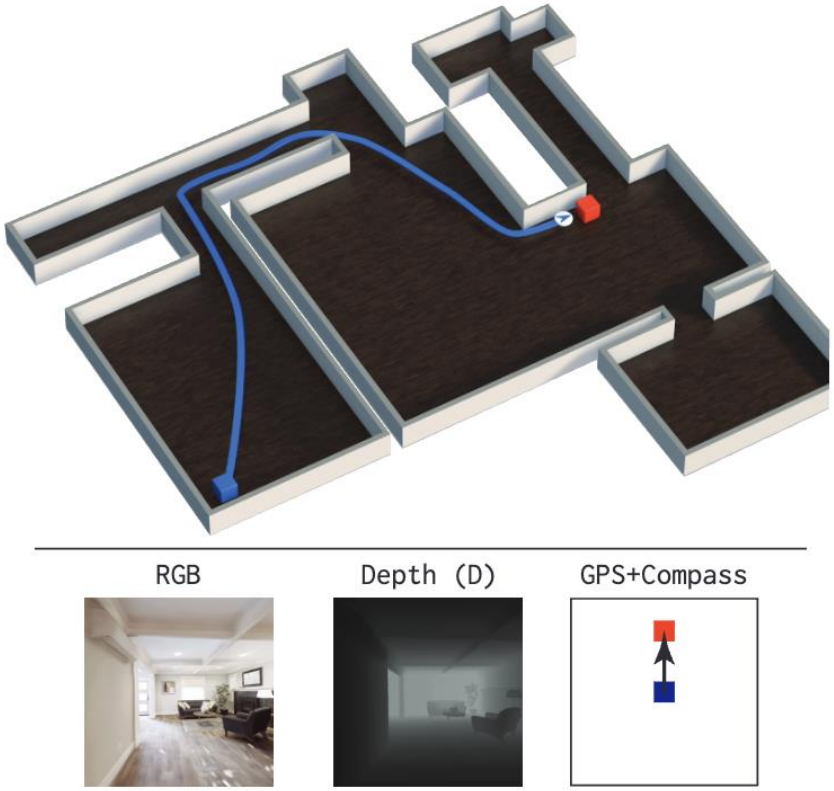


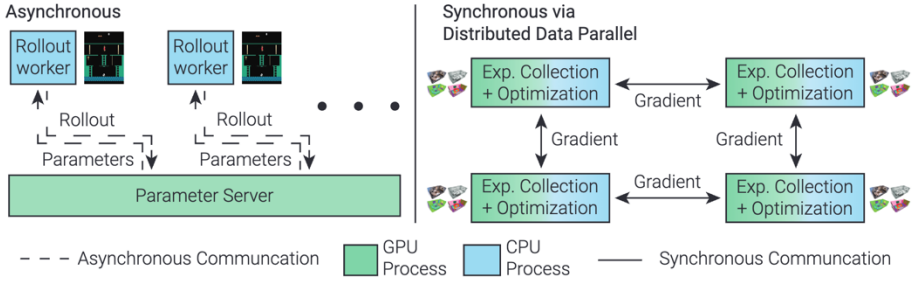
Early works: learning ROS nav stack with a CNN with Imitation Learning

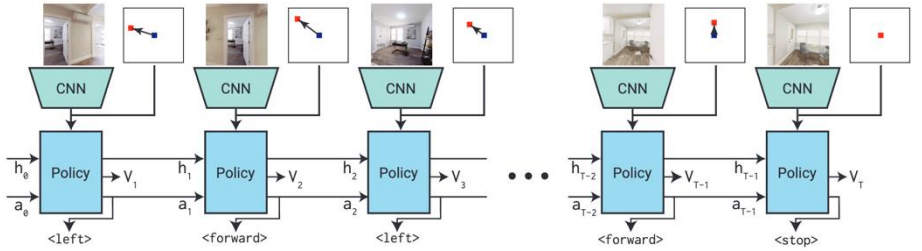
Later works: embodied AIs



DD-PPO Learning Near-Perfect Point Goal Navigators from 2.5 Billion Frames



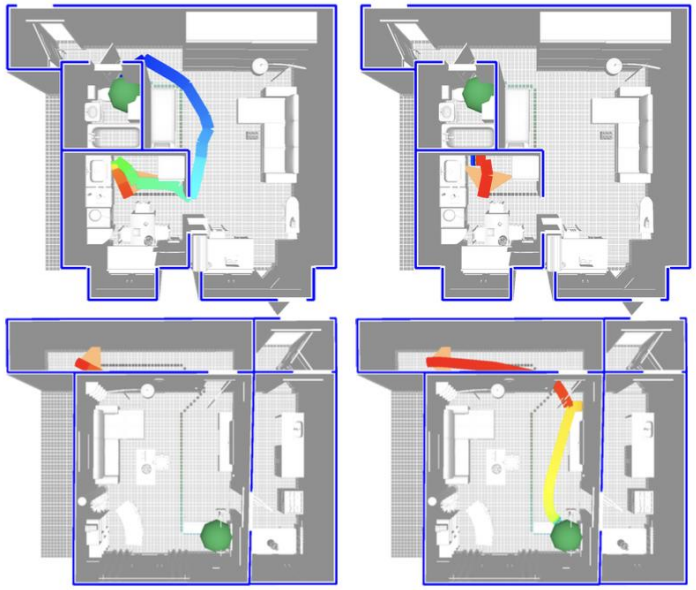




Decentralized – Distributed Proximal Policy Optimization, Wijmans et al, ICLR 2020 (Meta AI)

SIM2Real Gap

Benchmarking Classic and Learned Navigation in Complex 3D Environments, Mishkin, Gosovitskiy, Koltun, INTEL Lab, Arxiv 2019

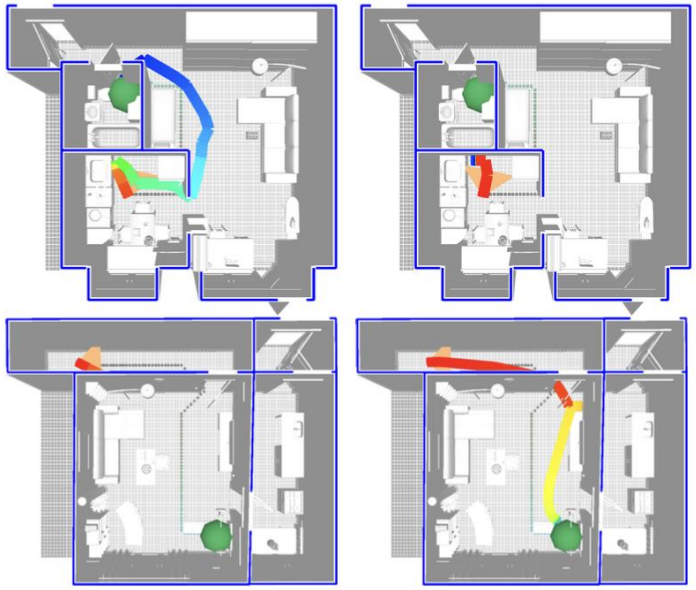


Classic pipeline Learned agent

Learned outperform Classic, but in simulations

SIM2Real Gap

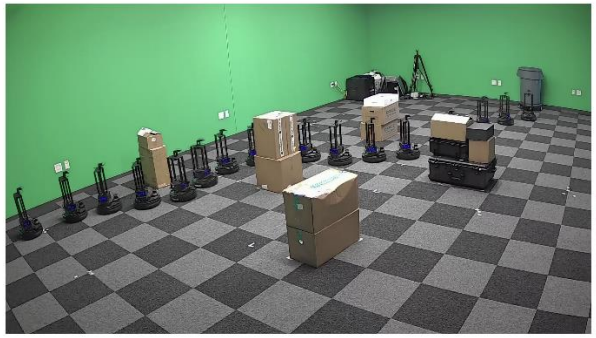
Benchmarking Classic and Learned Navigation in Complex 3D Environments, Mishkin, Gosovitskiy, Koltun, INTEL Lab, Arxiv 2019



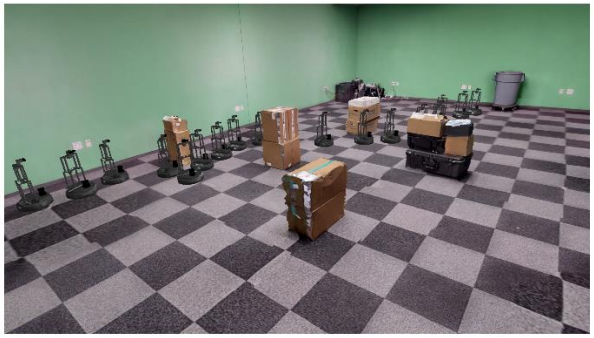
Classic pipeline Learned agent

Learned navigation stack outperform classic navigatio stack, but in simulations

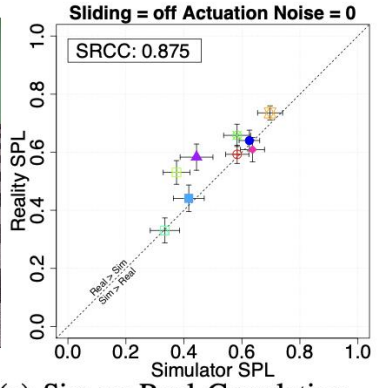
Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?



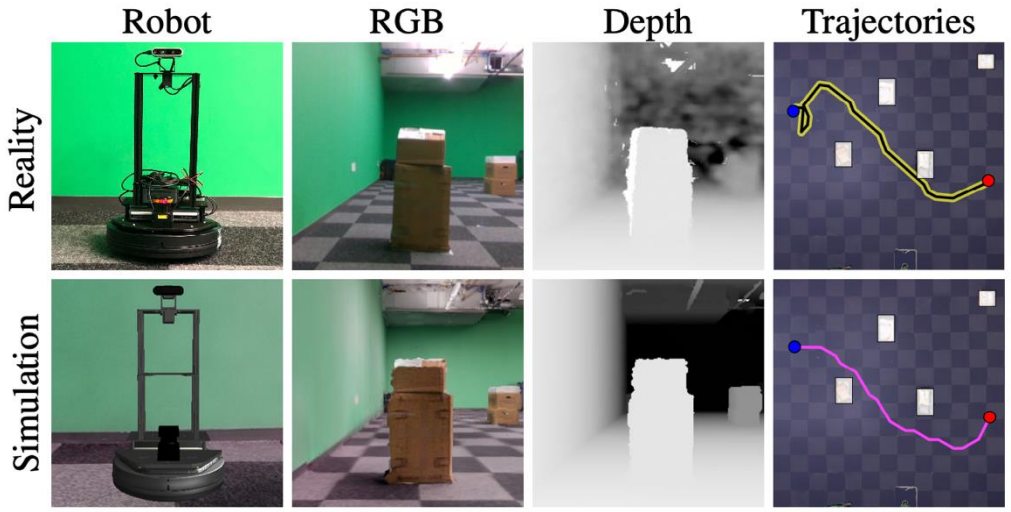
(a) Reality.



(b) Simulation.



(c) Sim-vs-Real Correlation.



Kadian, Wijmans et al. (Meta AI), IEEE RAL, 2020

The end!

Embodied AI trained in simulation can solve every tasks.

No map is required, only vision-based end-to-end DRL navigation.

No Sim2Real gap.

Navigation in robotics is solved, we can all move on.

Sci-fi robot will be among us soon!

Not so much

… a large concentration of learning-based work is able to solve the classical navigation problem, very few approaches actually *improve* upon classical techniques: currently, for navigation as a robotics task, learning methods have been used mainly to replicate what is already achievable by classical approaches; and for learning, navigation is mostly a good testbed to showcase and to improve learning algorithms.

We note that 43 out of the 74 surveyed papers are end-to-end approaches, and 15 of those lack the ability to navigate to user-defined goals. This inability to navigate to defined goals is not common in the classical navigation literature.

Not so much

While the majority of papers surveyed sought to solve the classical navigation problem, very few actually demonstrate improved performance over classical approaches. 46 of the 74 papers surveyed dealt with the classical problem, and only eight of those showed improved performance over classical solutions. The other 38 only achieved navigation in relatively simple environments (e.g., sparse obstacles, uniform topology, same training and deployment environment) and mostly did not compare with classical approaches.

One explanation is that the research community initially focused on answering the question of whether or not navigation was even possible with learning approaches, and focused on the unique benefits of such approaches in that extensive human engineering (e.g., filtering, designing, modeling, tuning, etc.) is not required or that alternate interfaces and constraints can be leveraged. Very few proposed learning approaches have been compared to classical solutions, which leads us to advocate for such performance comparison to be done in future work, along with comparisons along the other dimensions, such as training versus engineering effort and neural architecture search versus parameter tuning

Wrapping up

- End-to-end point-goal object-goal or image-goal navigation methods have achieved near-perfect results in visually realistic simulations
- Their applicability for real-word navigation remains unclear, has they have mostly been validated in limited real or lab setting.
- Transfer of visual navigation methods to real world is debatable.

Learning navigation subsystems

Integrating learning-based methods in modular architecture (e.g., local planning, semantic exploration) is promising.

The most interesting results are with robots where the focus in navigation is still on kinematics, as quadrotor or quadrupeds.

Few works integrate learning with wheeled platforms.

Mobile manipulation

- Manipulation is a task where learning gives great benefits (difficult kinematics, difficult to be modelled explicitly, low uncertainty, low semantic)
- Usually when we talk about manipulation, the robot is table-top
- New trend: Mobile Manipulation MoMa
- Integration of MoMa of learning and classical modules is one of the current challenges.
- MoMa is extremely task-dependent. Generalization is difficult as it involves a complex world model and semantic and spatial understanding.

Next steps in learned navigation

- The current best practice is to use machine learning at the subsystem or component level.
- «Special» type of navigation are good problem for testing learned-based navigation systems (e.g., social navigation, rough terrain, …) as those settings are complex for classical methods.
- Two critical aspects are *safety* and *explainability* – currently not addressed by learned methods. (Classical methods have both)
- *Parameter tuning* is a problem of classical framework but literature in learning-based navigation did little to explicitly acknowledge or characterize the very real costs of hyperparameter search and training overhead. Manual hyperparameter search.

Next steps in learned navigation

- Further development of machine learning components that continually improve based on real deployment experience.
- Most traditional navigation systems require manual intervention by human engineers in order to overcome failures or unsatisfactory performance during deployment.
- Learning-based systems, in theory, should be able to automatically process data generated by such events and improve without human involvement.
- However, current learning-based systems still separate the training and deployment phases, even for online RL approaches (due to, e.g., onboard computational constraints).
- This separation means that such systems are not continually learning from deployment experience.

Next steps in learned navigation

- Real-world learning is difficult. Especially when perception and vision is involved.
- Improving stability and sample efficiency in RL algorithms is thus important.
- Integration of different learned modules or of different learned task is an open problem
- Benchmarking Real-World success is also an open problem; can we trust sim results? Can we trust LLMs stability and robust performance?

Sistemi Intelligenti Avanzati – 2025/2026
Università degli Studi di Milano

Robot Learning using VLA and VLMs

Matteo Luperto
Dipartimento di Informatica
matteo.luperto@unimi.it

Avoiding the ~~Sim2~~Real gap with real world data

- Real-world learning is difficult. Especially when perception and vision is involved.
- Can LLMs provide a world model for a robot?
- What if we can use only real-world data for training?

Few research groups can do that, the most relevant one are those of Chelsea Finn (Stanford), Sergey Levine (Berkley) and their startup Physical Intelligence (π).

For locomotion and control, Marco Hutter (ETH) is doing the same for legged robotics.

VINT VIKING BADGER + others

- Kahn, G., Abbeel, P., & Levine, S. (2021). Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, *6*(2), 1312-1319. https://sites.google.com/view/badgr
- Shah, D., & Levine, S. (2022). Viking: Vision-based kilometer-scale navigation with geographic hints. *CORL 2022* *https://sites.google.com/view/viking-release*
- Hirose, Noriaki, et al. "ExAug: Robot-conditioned navigation policies via geometric experience augmentation." *ICRA 2023* *https://sites.google.com/view/exaug-nav*
- Shah, D., Sridhar, A., Dashora, N., Stachowicz, K., Black, K., Hirose, N., & Levine, S. (2023). ViNT: A foundation model for visual navigation. *CORL 2023* *https://general-navigation-models.github.io/vint/index.html*

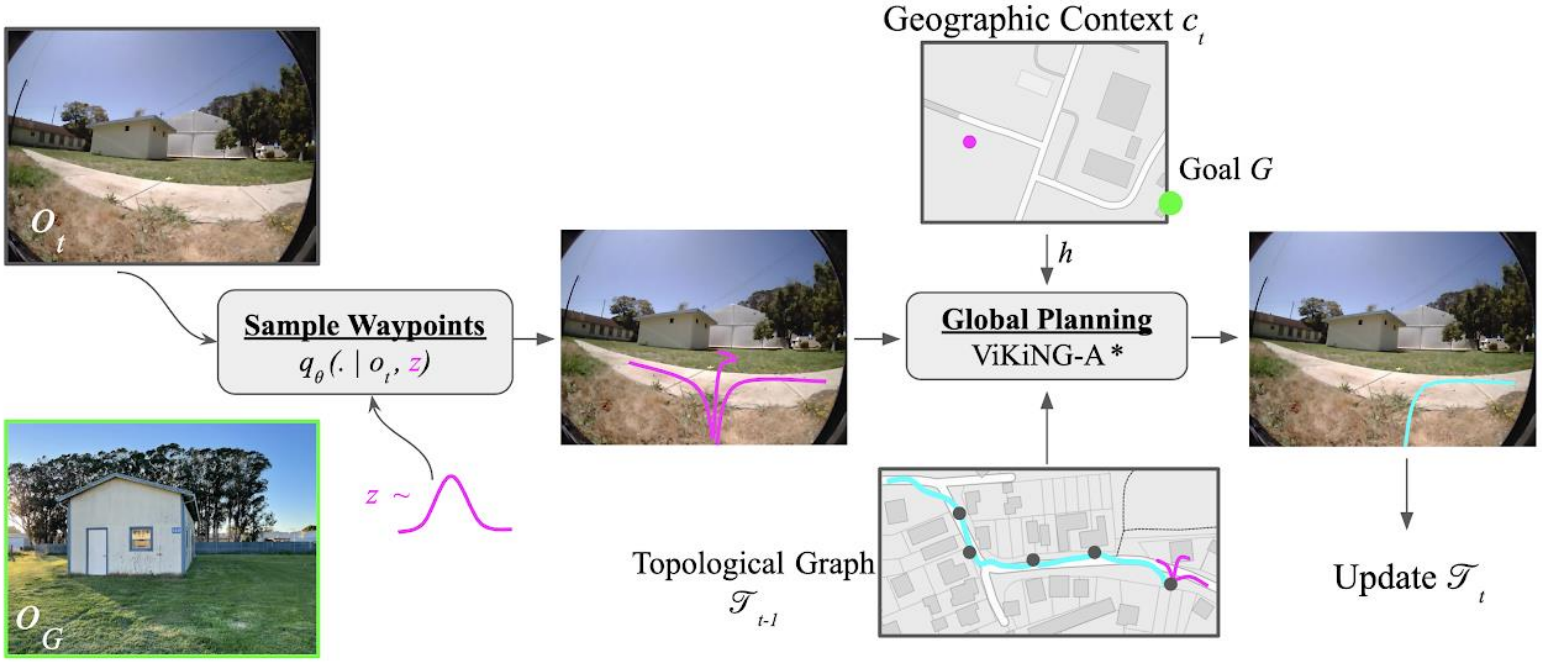
BADGR


BADGR: An Autonomous Self-Supervised Learning-Based Navigation System
Gregory Kahn, Pieter Abbeel, Sergey Levine
BAIR

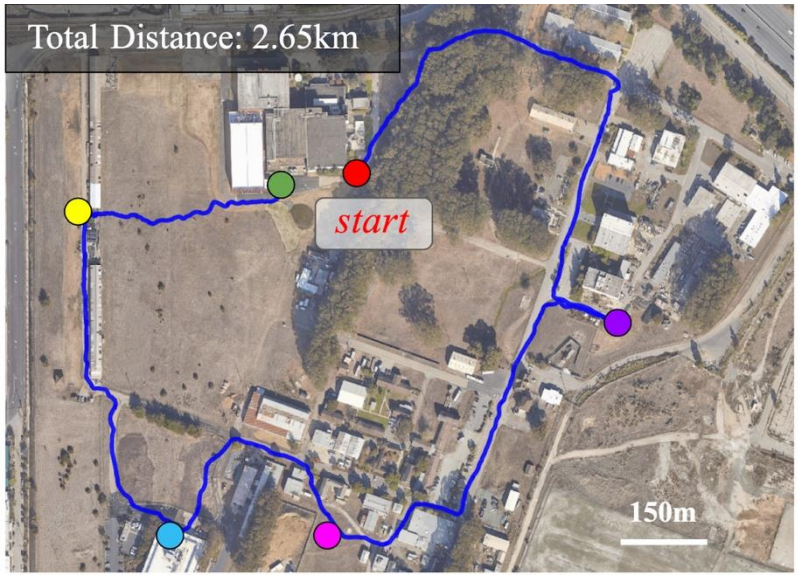
BADGR

- Learning only with a real robot that can experience catastrophic failures
- The robot learned to go through grass with experience (geometric method avoid grass)
- The robot learned to not go through trees and houses with experiences (geometric method avoid houses)

Fig. 4: While collecting data, the robot will periodically require a manual intervention to reset from catastrophic failures, though recovery is usually automatic.


Geographic Context c_t
Goal G
o_t
h
Sample Waypoints
$q_\theta(. \mid o_t, z)$
Global Planning
ViKiNG-A *
$z \sim$
o_G
Topological Graph
$\mathcal{T}_{t-1}$
Update $\mathcal{T}_t$

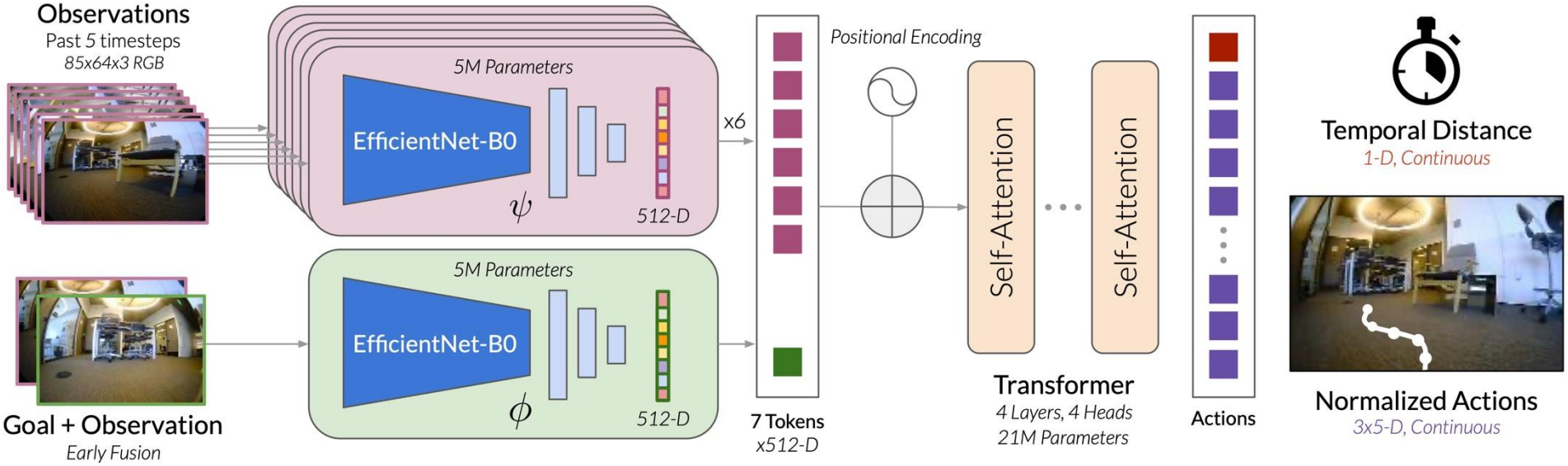
ViKiNG


Total Distance: 2.65km
start
150m

ViNT

https://general-navigation-models.github.io/vint/ 2023

ViNT uses a Transformer-based architecture to encode (current and past) visual observations and goals using an EfficientNet CNN, and predicts temporal distance and normalized actions in an embodiment-agnostic manner.



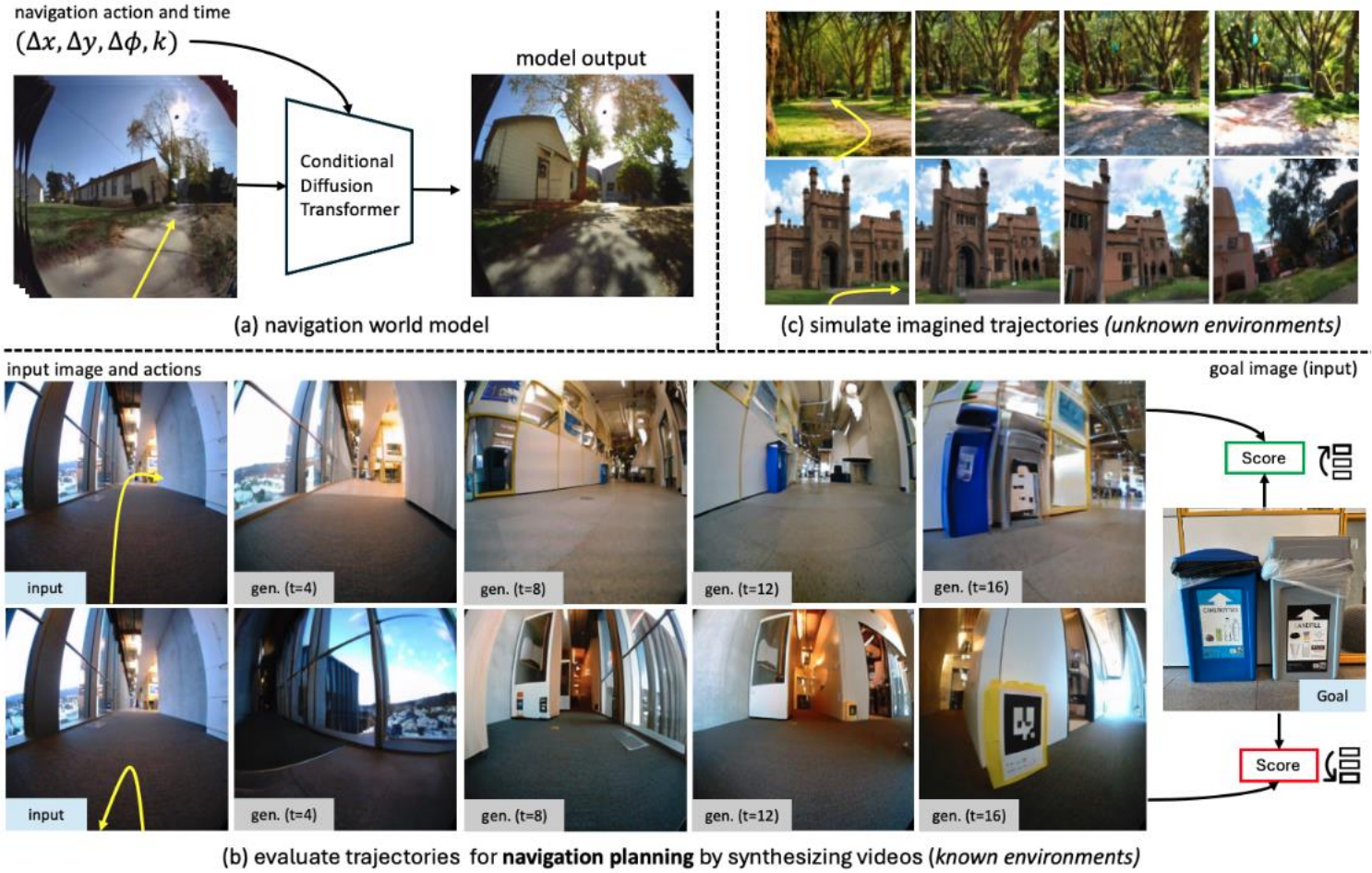
NoMaD

https://general-navigation-models.github.io/nomad/ 2023

NoMaD is a novel architecture for robotic navigation in previously unseen environments that uses a unified diffusion policy to jointly represent exploratory task-agnostic behavior and goal-directed task-specific behavior. NoMaD provides high capacity (both for modeling perception and control) and the ability to represent complex, multimodal distributions.

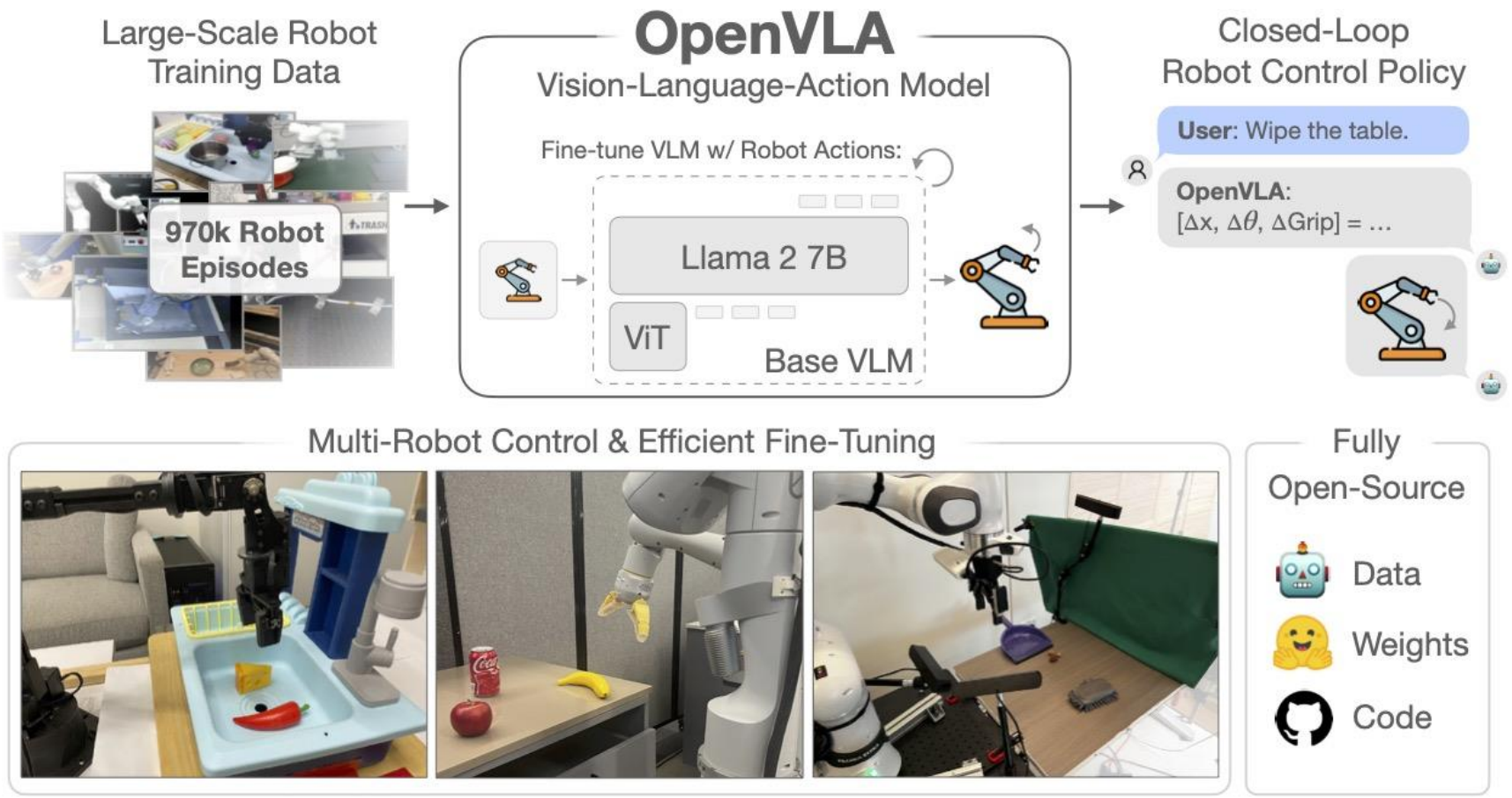
Navigation World Models

https://www.amirbar.net/nwm/ 2024

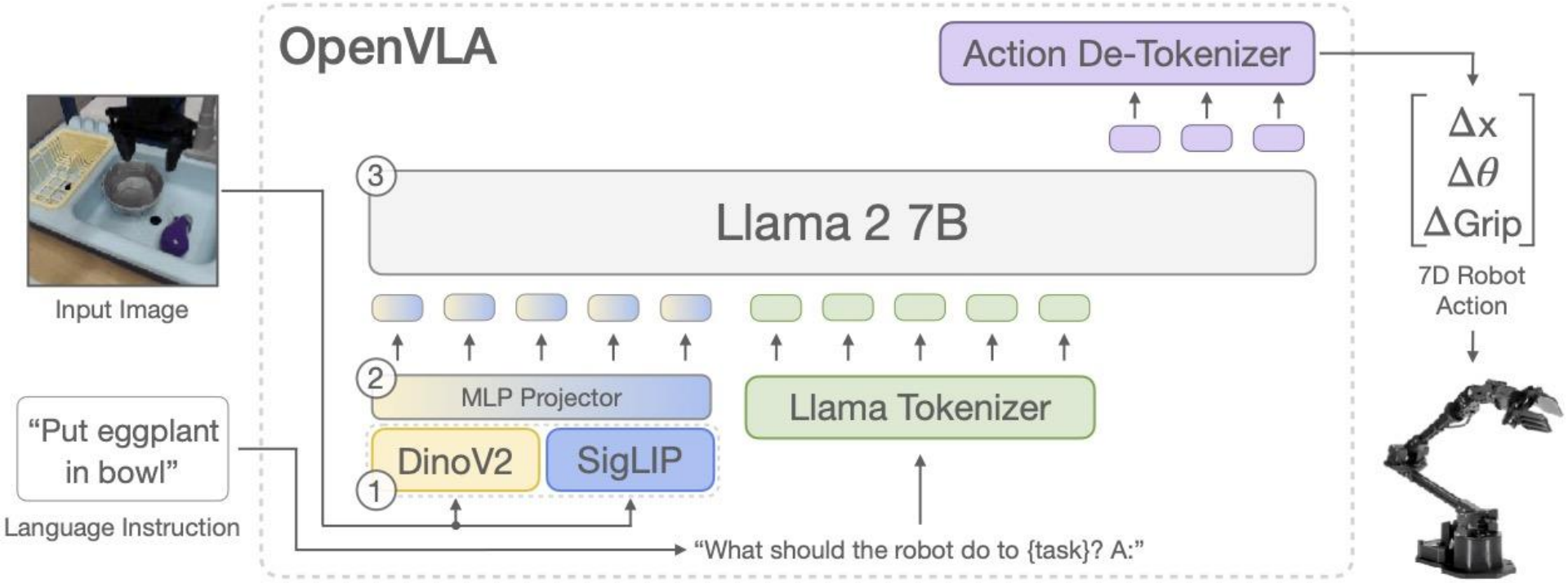


- Conditional Diffusion Transformer (CDiT) as a *world model* to generate the visual input of future trajectories.
- NoMaD as a navigation policy

OpenVLA https://openvla.github.io/ 2024



OpenVLA https://openvla.github.io/ 2024



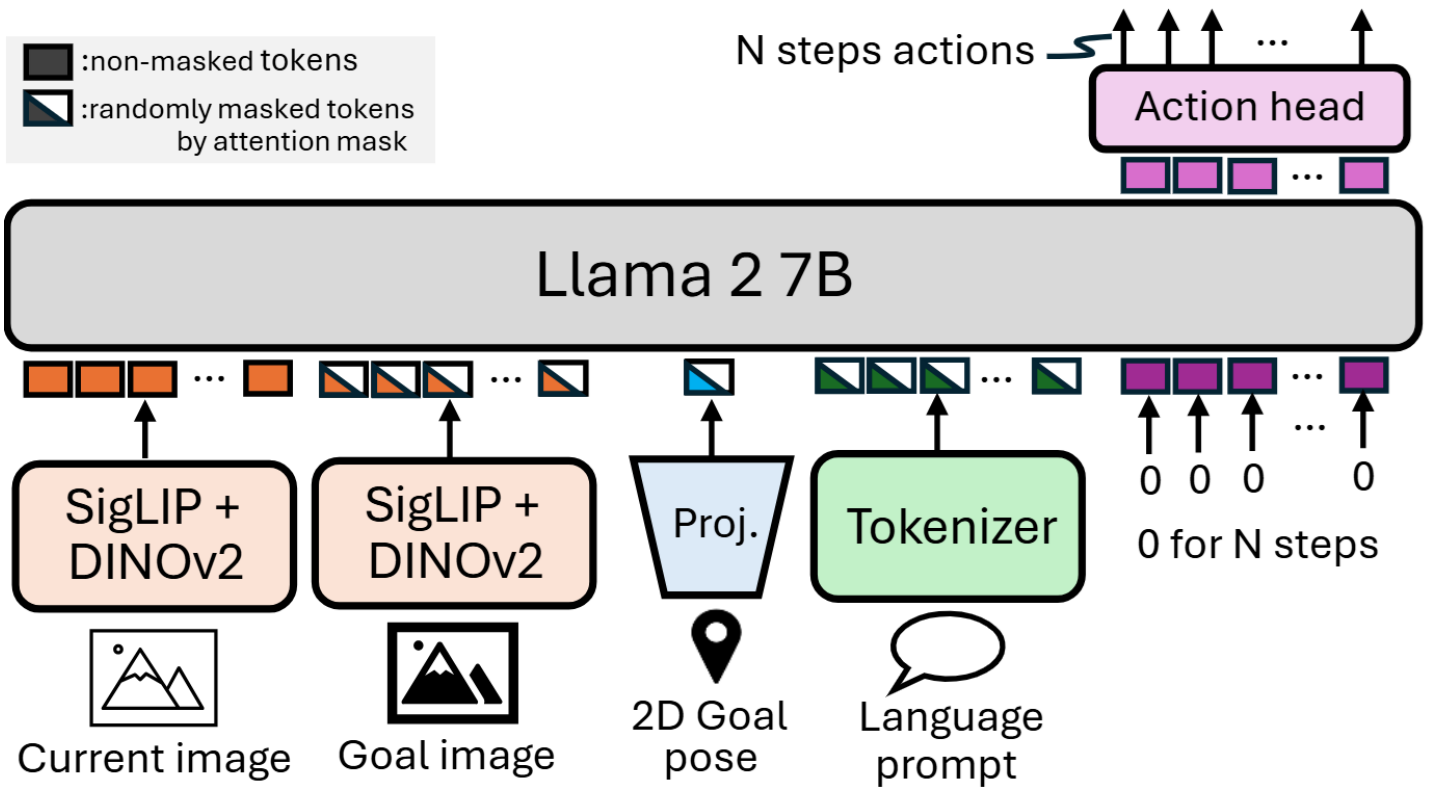
(1) a **fused visual encoder**, consisting of a SigLIP and a DinoV2 backbone, that maps image inputs to a number of "image patch embeddings'", (2) a **projector** that takes the output embeddings of the visual encoder and maps them into the input space of a large language model, and (3) a Llama 2 7B **language model backbone** that predicts tokenized output actions.
These tokens get decoded into continuous output actions that can be directly executed on the robot.

To train OpenVLA, they used 970k robot manipulation trajectories a dataset that spans a wide range of tasks, scenes and robot embodiments. OpenVLA is trained on a cluster of 64 A100 GPUs for 15 days.

Output: discrete token (predicts actions like text) than is then de-tokenized into robot actions. Prediction is step by step, low freq.

OmniVLA

https://omnivla-nav.github.io/



Key: training on a dataset of **9,500** hours across **10** platforms, including human-collected data, covering a wide range of environments.

OmniVLA: An Omni-Modal Vision-Language-Action Model for Robot Navigation

Noriaki Hirose[1, 2], Catherine Glossop[1], Dhruv Shah[1, 3], Sergey Levine[1]




1: Berkeley UNIVERSITY OF CALIFORNIA

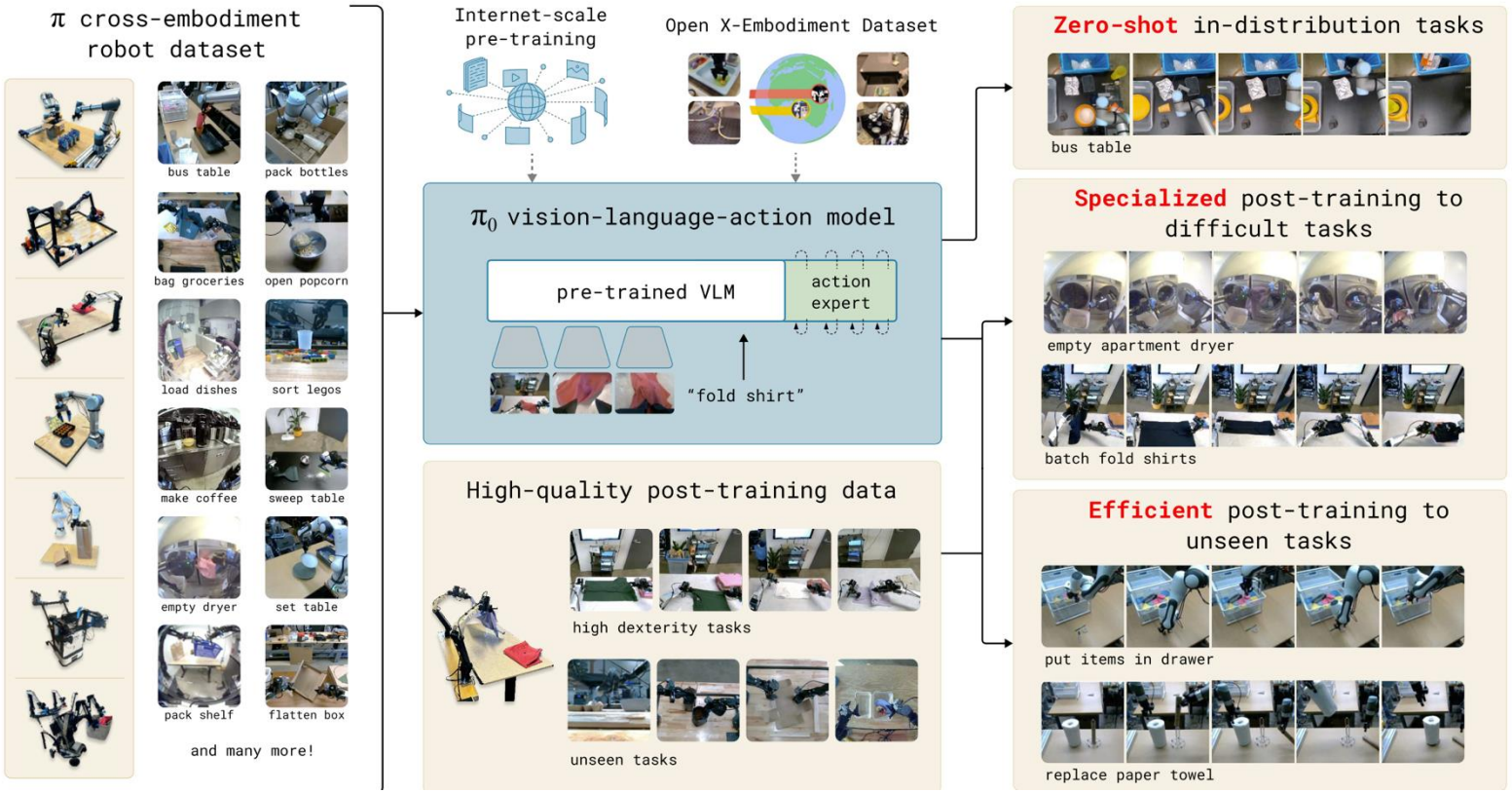
2: TRINA Toyota Motor North America

3: PRINCETON UNIVERSITY

Key: training on a dataset of **9,500** hours across **10** platforms, including human-collected data, covering a wide range of environments. GNM and LeLaN are themselves mixtures of 7 and 5 publicly available datasets, respectively

π_0

https://www.pi.website/blog/pi0 2024



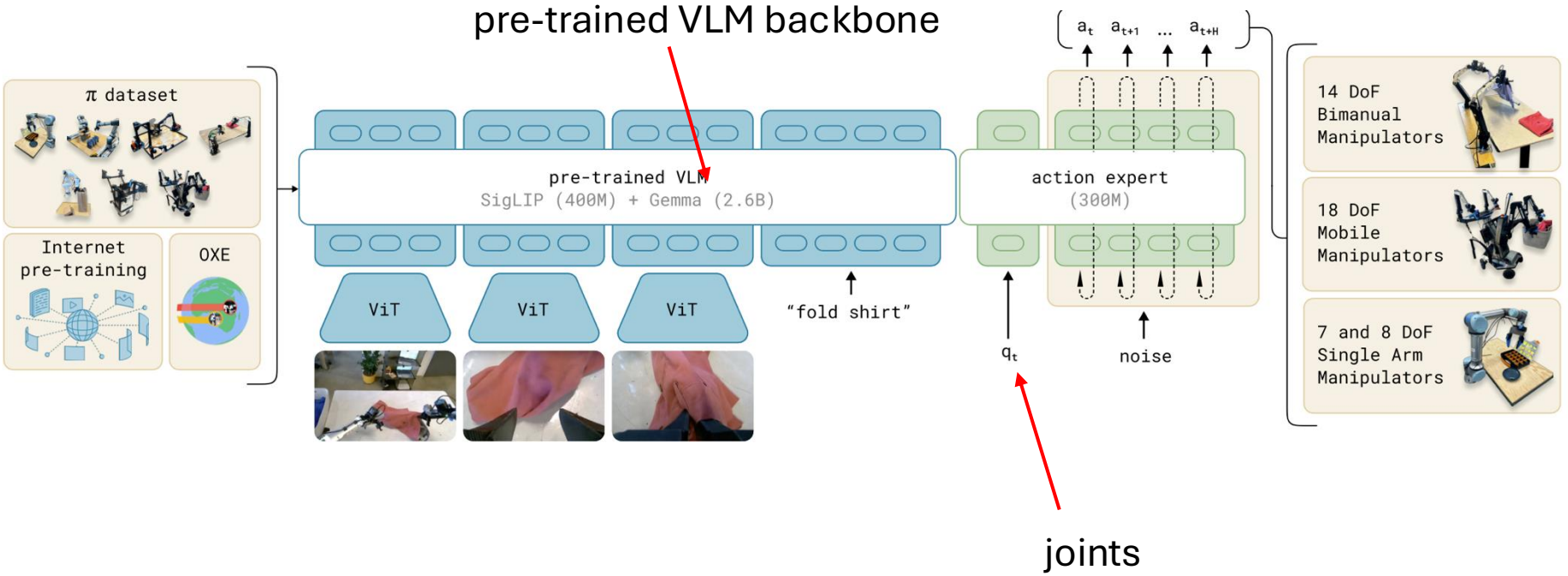
Key here is training; pre-training & post-training separation as in LLMs and VLM

The action expert directly outputs *continuous actions tokens* for the robot using a *flow matching* loss (a type of diffusion).

It predicts the next 50 steps one at a time.

π_0

https://www.pi.website/blog/pi0 2024



Step 1: pre-training the model on over 10k hours of real world data

Goal: learn a base model that exhibits broad capabilities and generalization

Step 2: post-train the action-expert head for a specific task

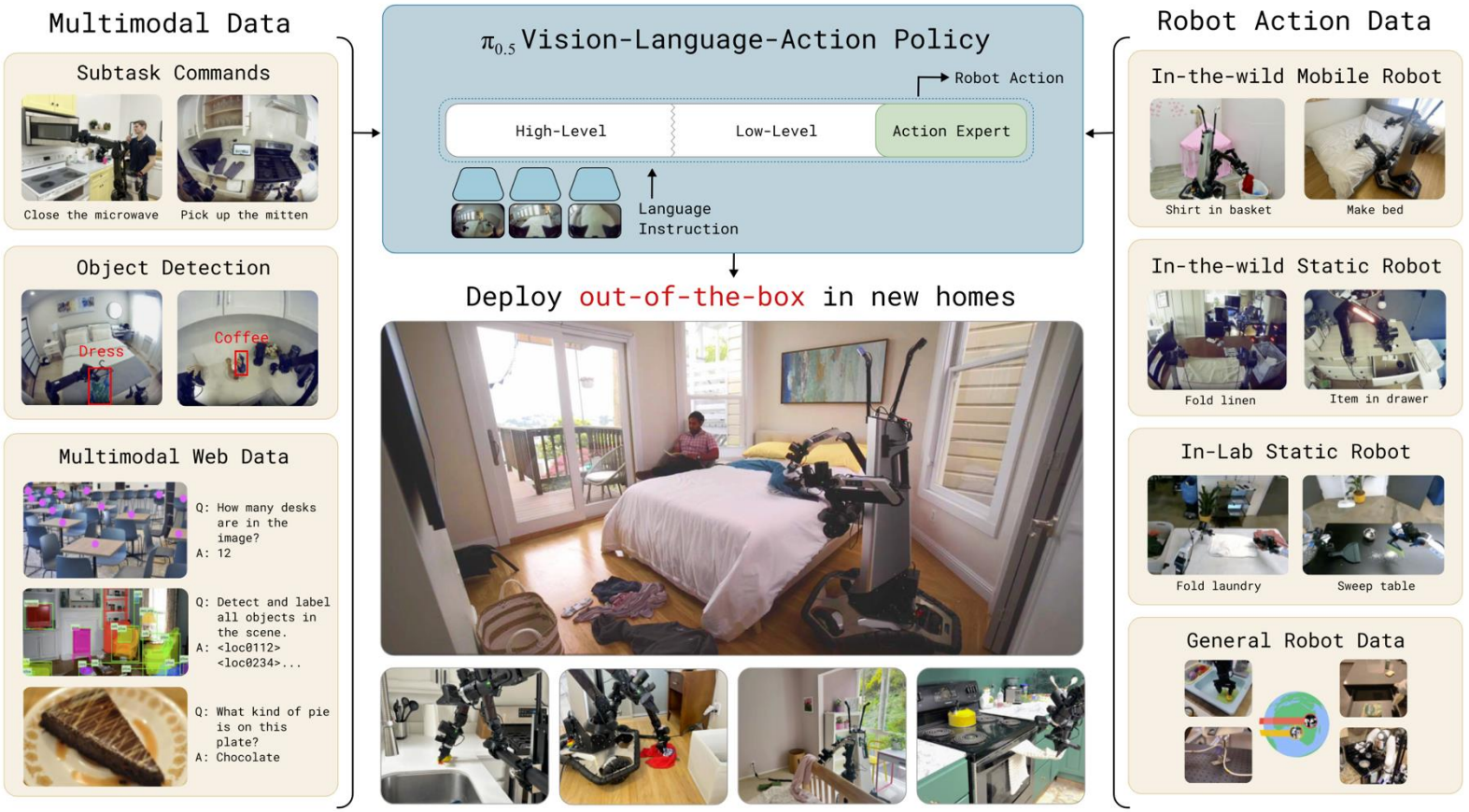
Goal: learn a consistent and fluent strategy for the task

- Easy tasks require 5h of data
- Complex tasks require > 100h of data

Problem: learning continuous action token in Step 1 degrades the VLM backbone

$\pi_{0.5}$

https://www.pi.website/blog/pi05 2025



$\pi_{0.5}$ model transfers knowledge from a heterogeneous range of data sources, including other robots, high-level subtask prediction, verbal instructions, and data from the web, in order to enable broad generalization across environments and objects.

$\pi_{0.5}$ https://www.pi.website/blog/pi05 2025

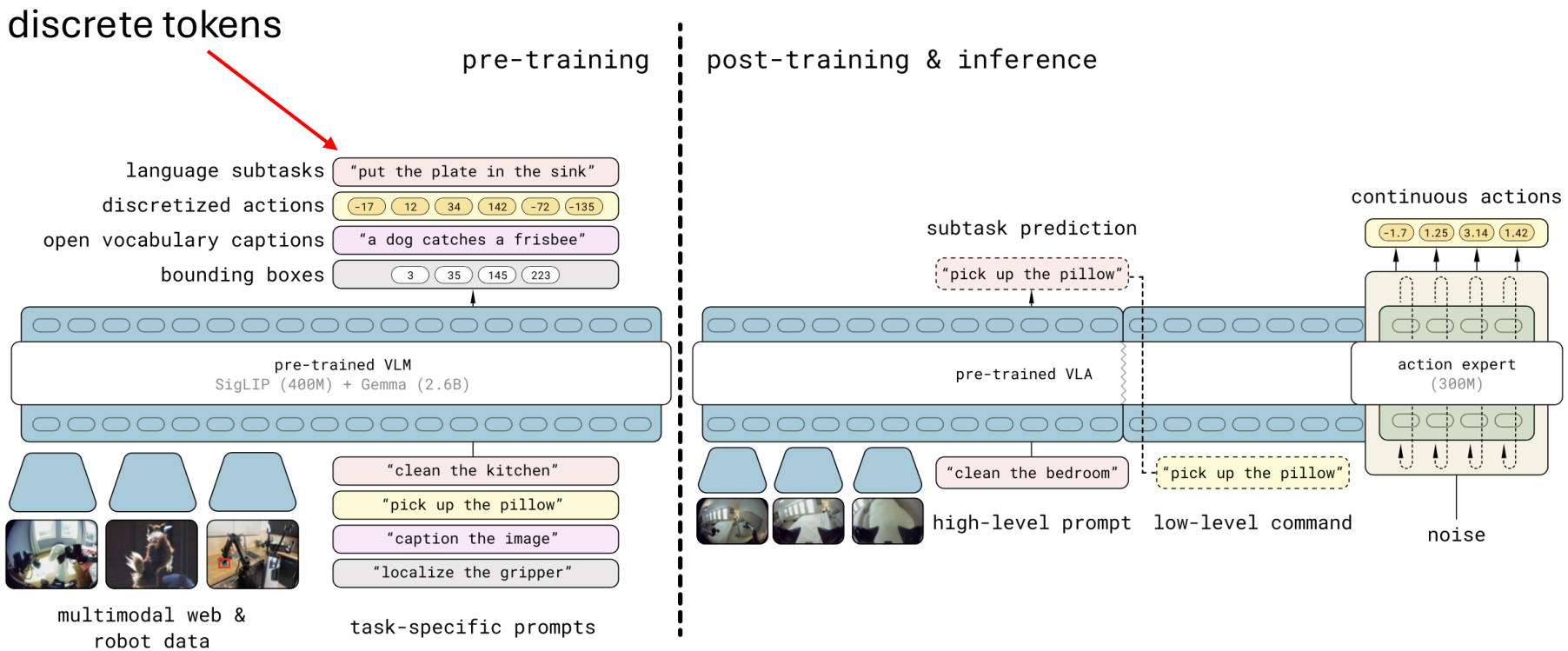
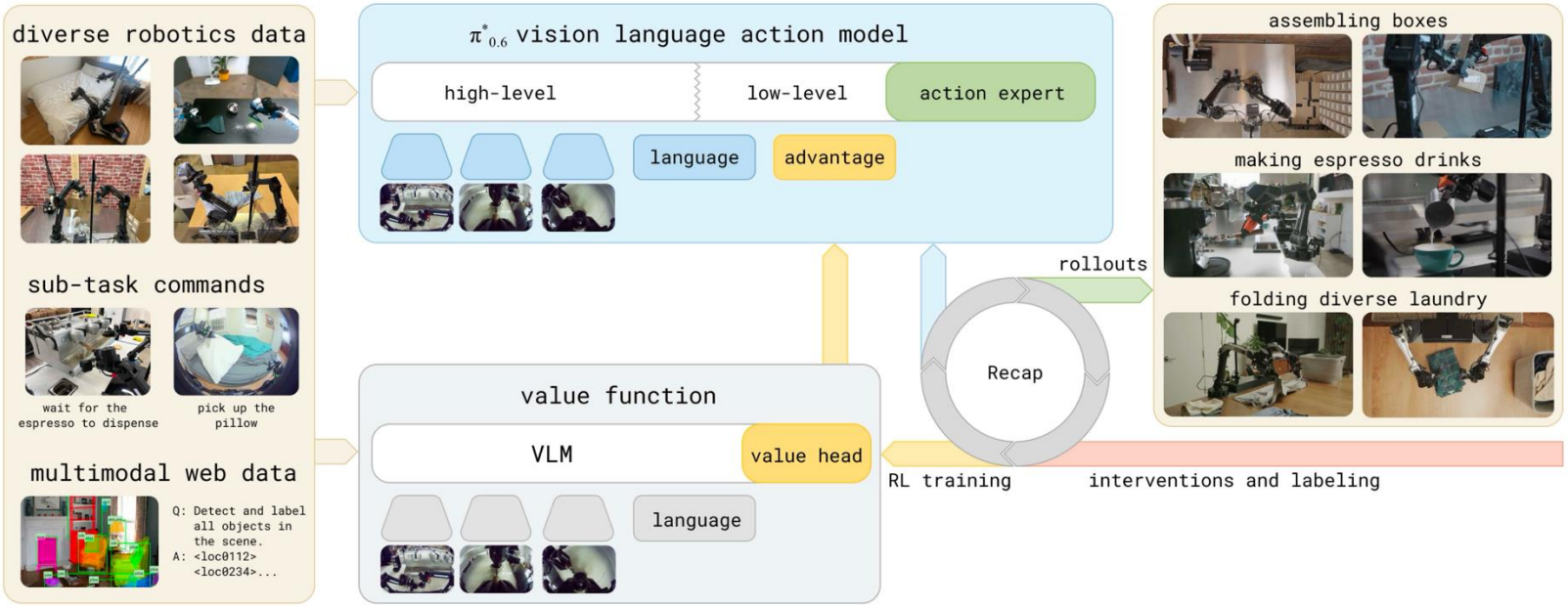


Fig. 3: **Model overview.** $\pi_{0.5}$ is trained in two stages. First, a pre-training stage combines all of the different data sources to produce an initial VLA with discrete tokens. This stage uses data from diverse robotic platforms, high-level semantic action prediction, and data from the web. Robotic data uses the FAST action tokenizer to represent actions as discrete tokens [64]. Second, a post-training stage specializes the model for low-level and high-level inferences for mobile manipulation, leveraging the most task-relevant data, including verbal instructions from human supervisors. This stage uses flow matching to represent the action distribution, enabling efficient real-time inference and the ability to represent fine-grained continuous action sequences. At inference time, the model first infers a high-level subtask, and then predicts the actions based on this subtask.

$\pi^*_{0.6}$

https://www.pi.website/blog/pistar06 11/2025



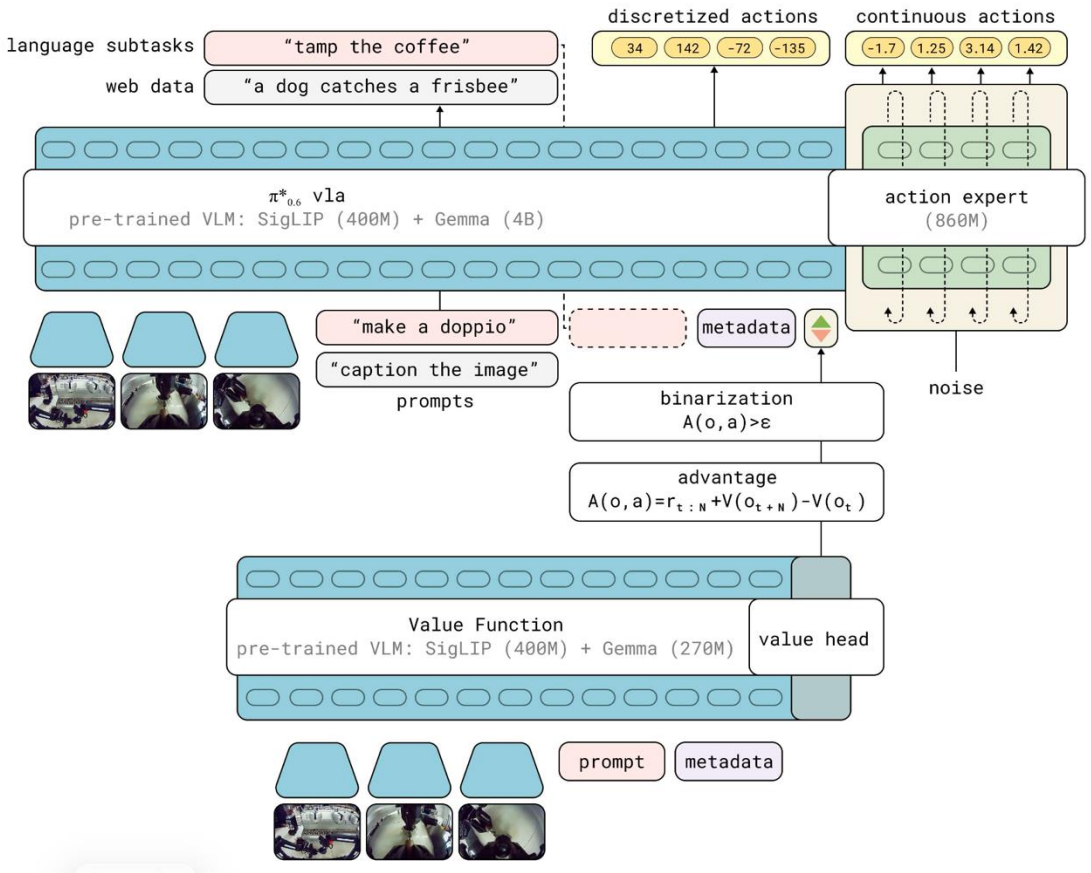
$\pi^*_{0.6}$ adds learning from experience into the framework by introducing an advantage term and expert supervision during the training phase.

Advantage: *conditioning* trick, adds a value specifying if the training data show a good (+1) or bad (-1) move. In deployment, you ask only for type-(+1) moves.

(It also uses a bigger backbone (5B vs 3B) and higher resolution images).

$\pi^*_{0.6}$

https://www.pi.website/blog/pistar06 11/2025



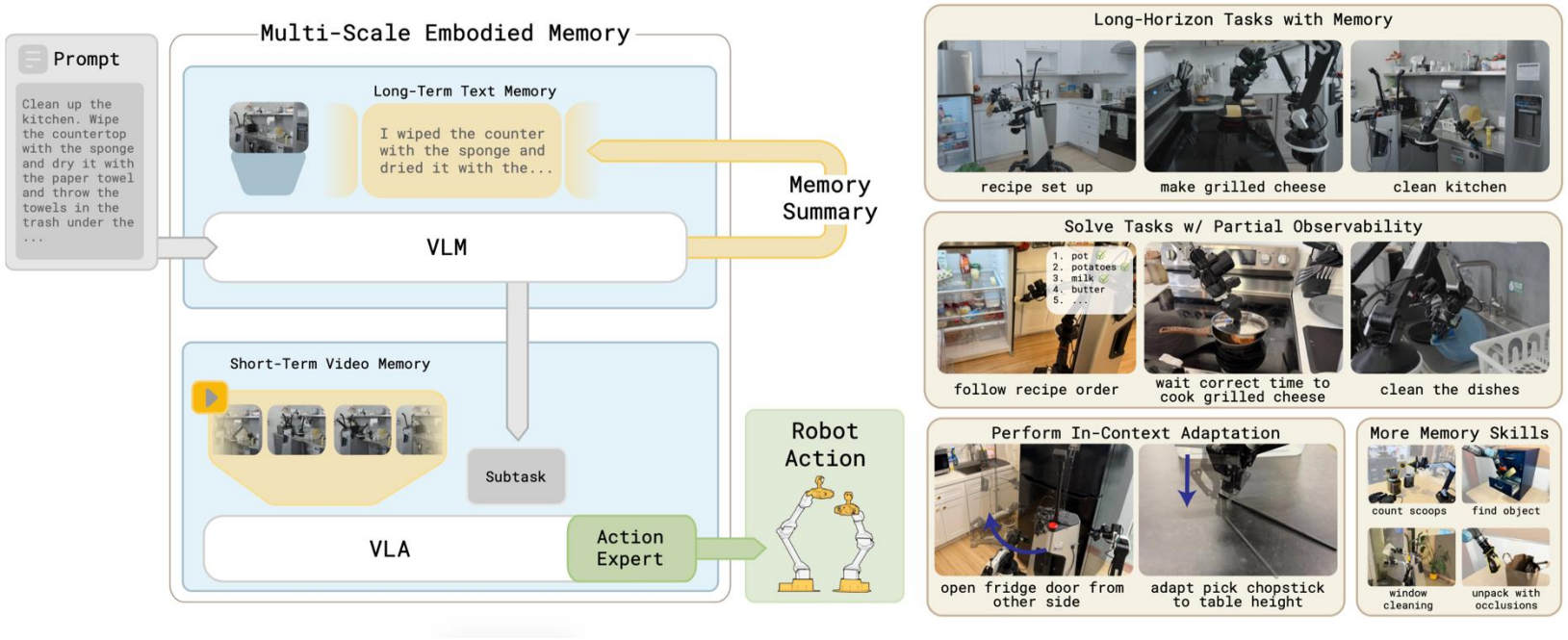
After pretraining, a policy improvement loop starts, for the target task.

- $\pi^*_{0.6}$ is finetuned with expert demonstration (advantage = 1, supervised finetuning)
- Robot acquires more experience autonomously on the task by using the advantage network prediction
- Some of the run are monitored by an expert teloperator who gives corrections

Result: $\pi^*_{0.6}$ can run for **13 hours straight** making espresso

$\pi^*_{0.6}$+MEM

https://www.pi.website/research/memory 3/3/26

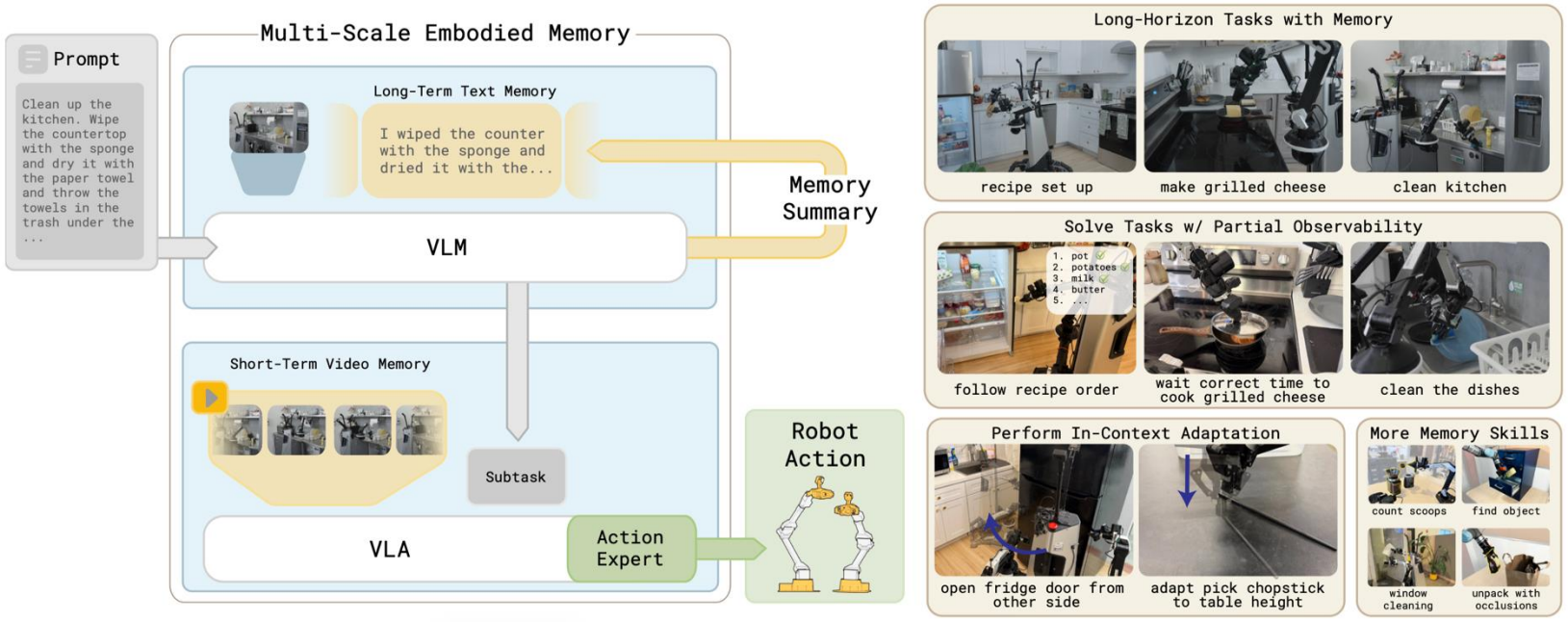


Problem: the VLA has no memory, and a *goldfish* syndrome for performing complex tasks.

1. Clean the table
2. Clean the sink
3. Clean the fridge
4. Clean the table (again)

$\pi^*_{0.6}$+MEM

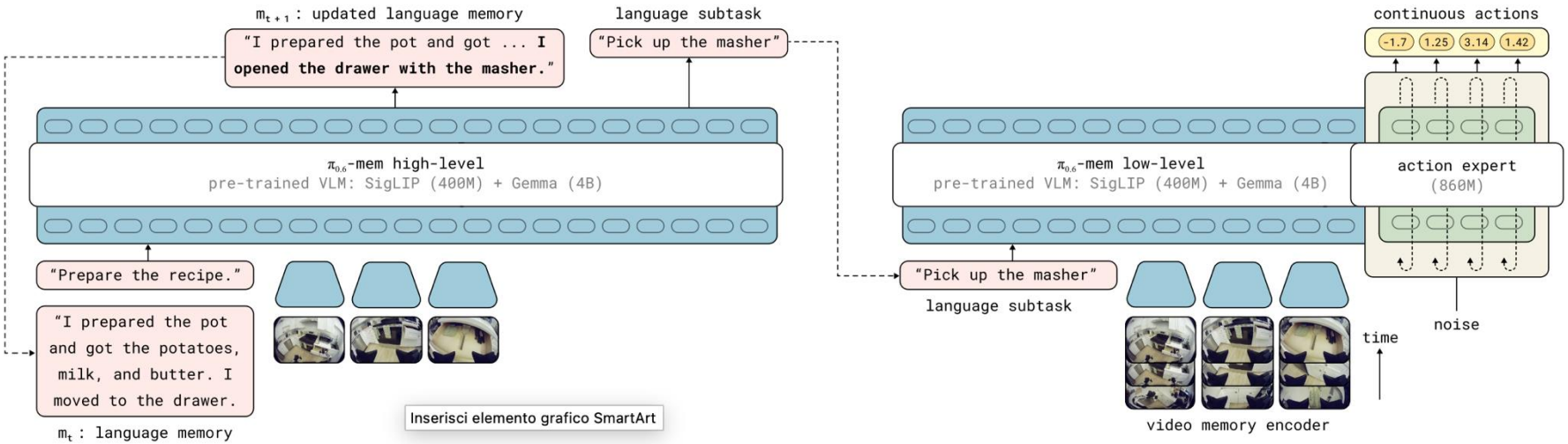
https://www.pi.website/research/memory 3/3/26



Solution: add natural language descriptions of the tasks already performed by the robot along with the other high-level embeddings during the progress of the task.

$\pi^*_{0.6}$+MEM

https://www.pi.website/research/memory 3/3/26



Solution: add natural language descriptions of the tasks already performed by the robot along with the other high-level embeddings during the progress of the task.