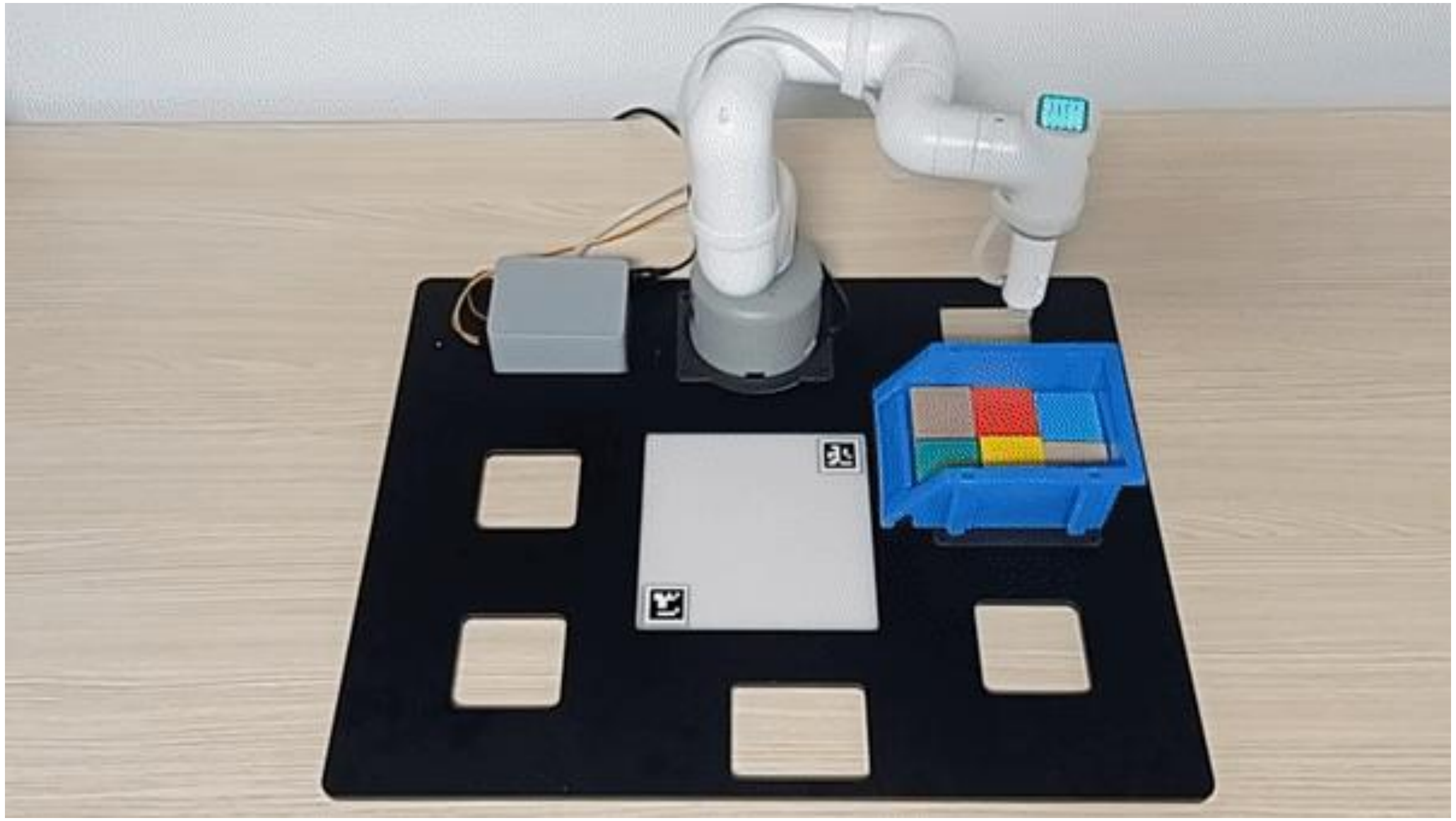
Foundational Models for Robotics:
Removing the Engineer from the Loop



Rogerio Bonatti

April 27, 2023

Speaker Bio: Rogerio Bonatti

Senior Researcher

PhD in Robotics,
School of Computer Science

Research Internships,
FAIR and MSR

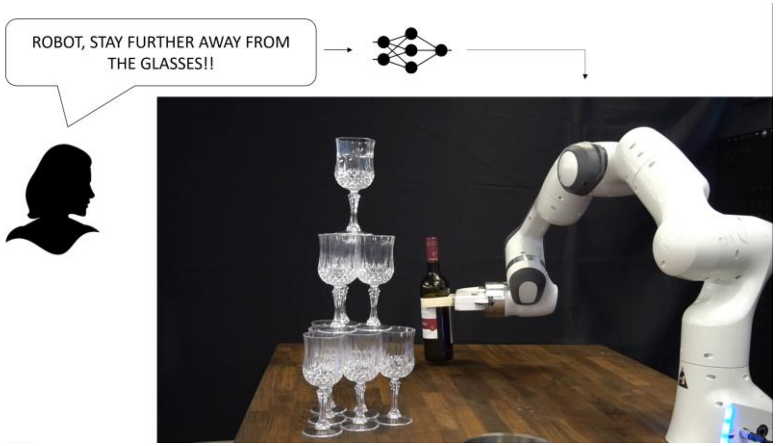
Microsoft

Carnegie Mellon University

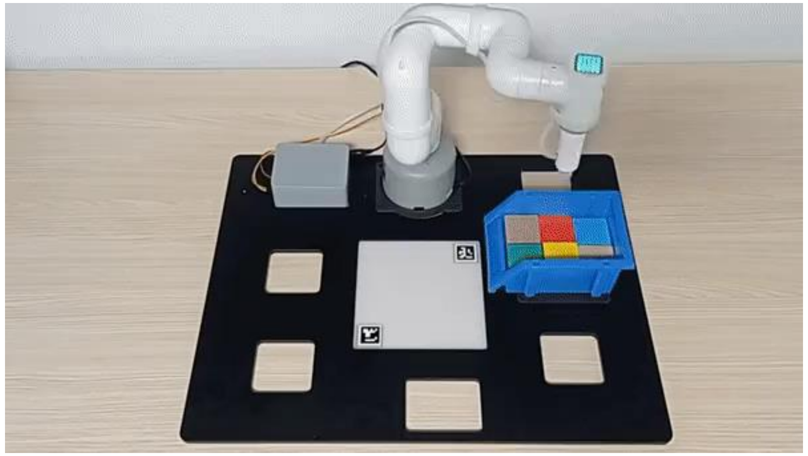
Meta

Research projects:

Multi-modal robot control (vision + language):

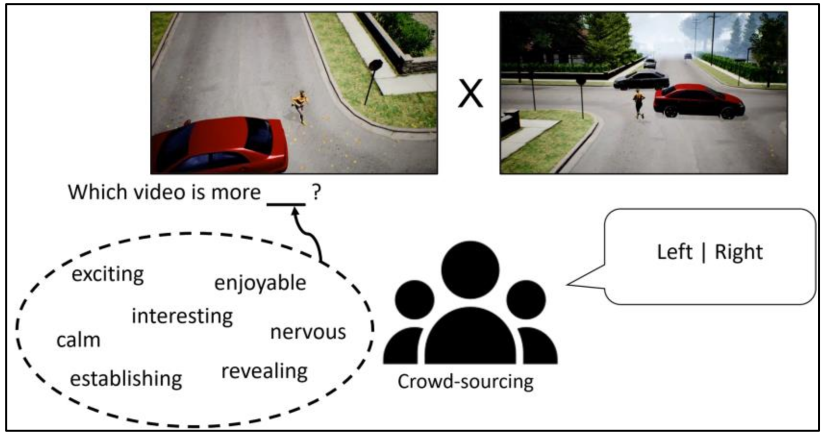


Generative models for decision-making:

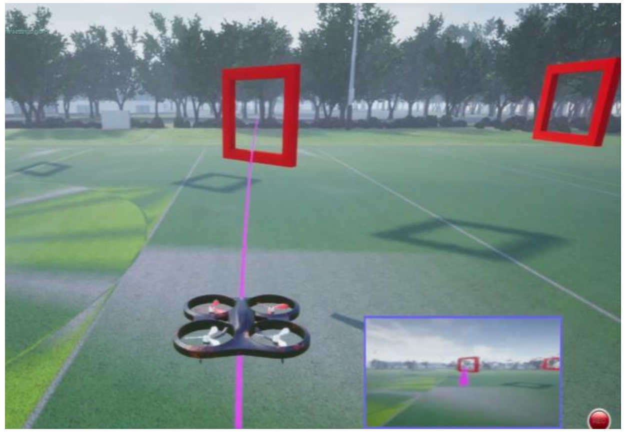


Autonomous drone cinematography:

Learning emotional camera control:

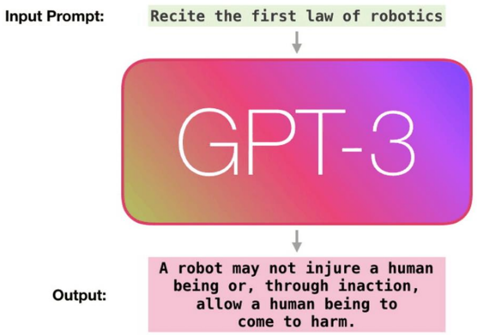


Visuo-motor representation learning:



Large models are revolutionizing generative AI

Text generation
[GPT-X, LLaMA]



Conversation
[ChatGPT]

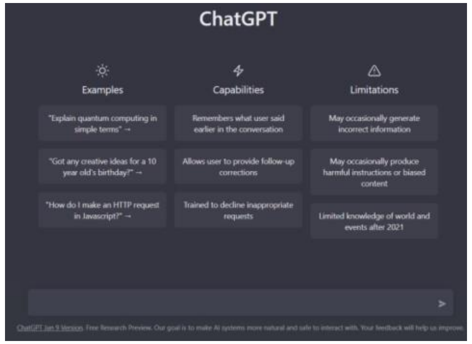
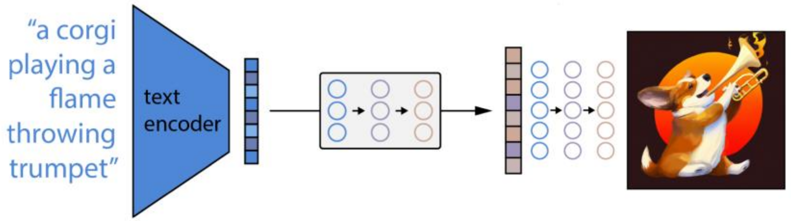


Image generation
[DALL-E, Stable Diffusion]



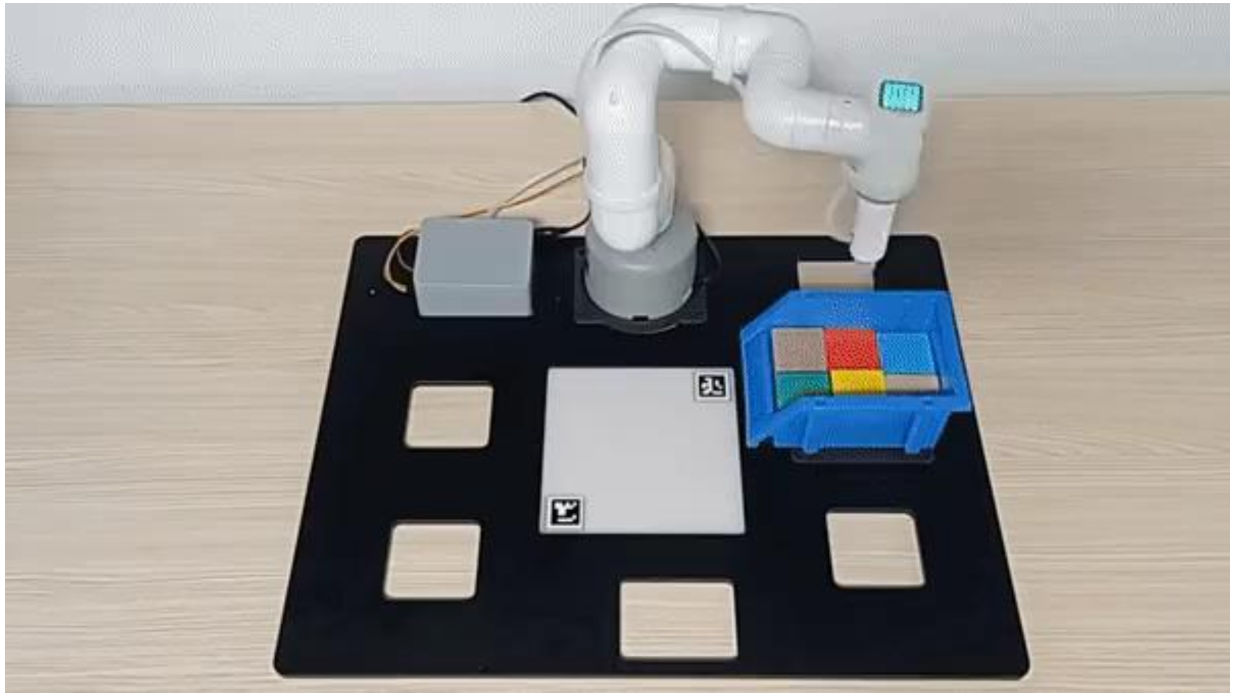
Text-to-video
[NVIDIA Picasso]

Text-to-3D
[Point-e, NVIDIA Picasso]

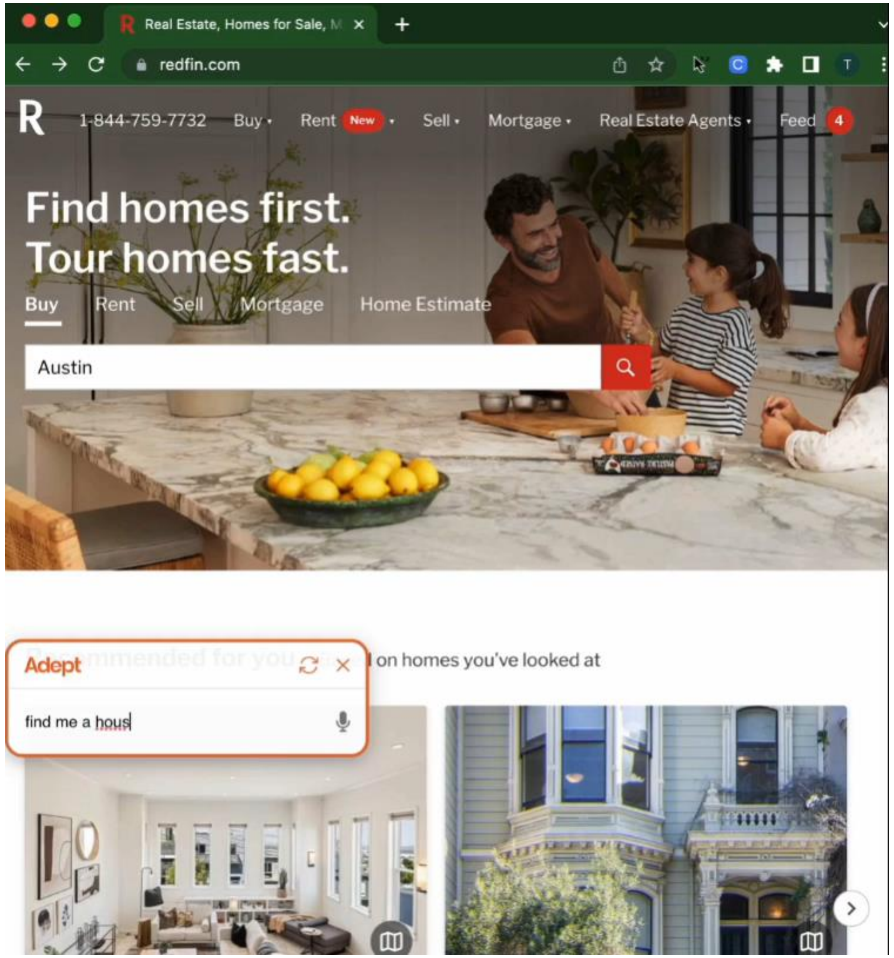
Image editing
[InstructPix2Pix, Prompt-to-Prompt]

But generative AI is just starting to revolutionize decision-making:

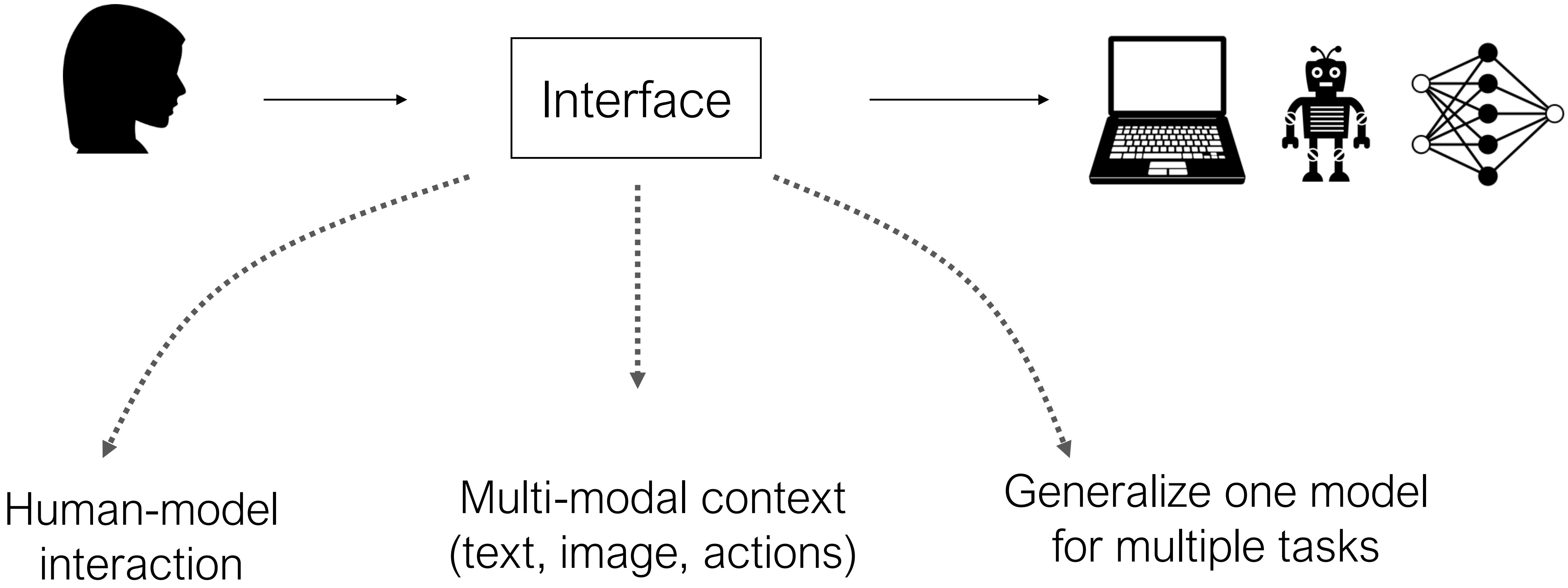
Human-robot interfaces:
[ChatGPT for Robotics]



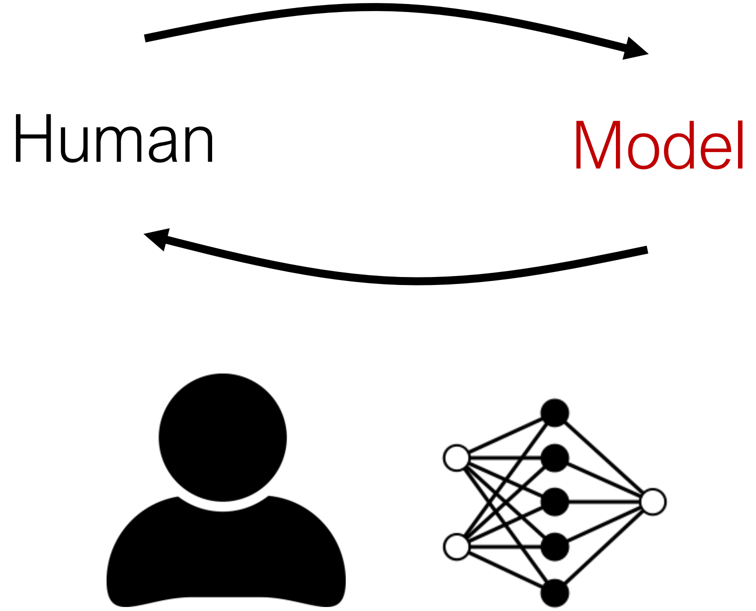
Human-computer interfaces
[video from Adept AI]



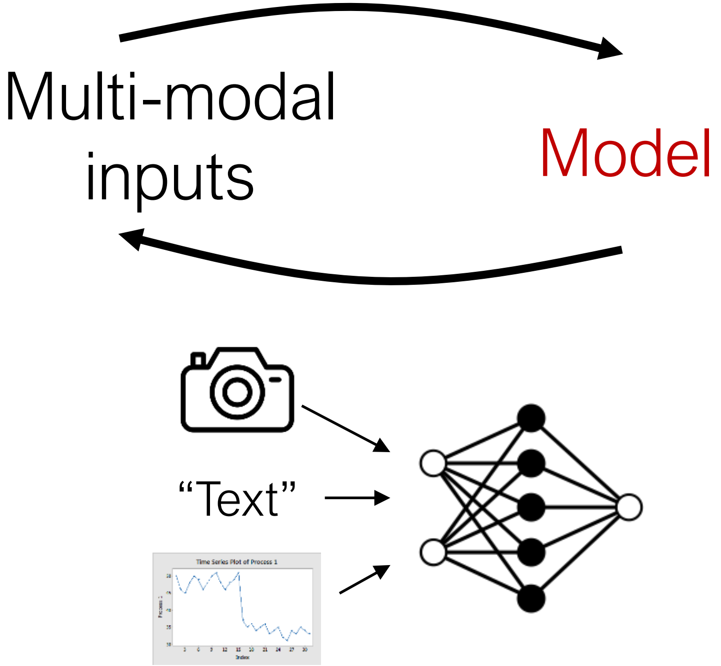
Goal: use large models to empower any user with the capabilities of generative AI



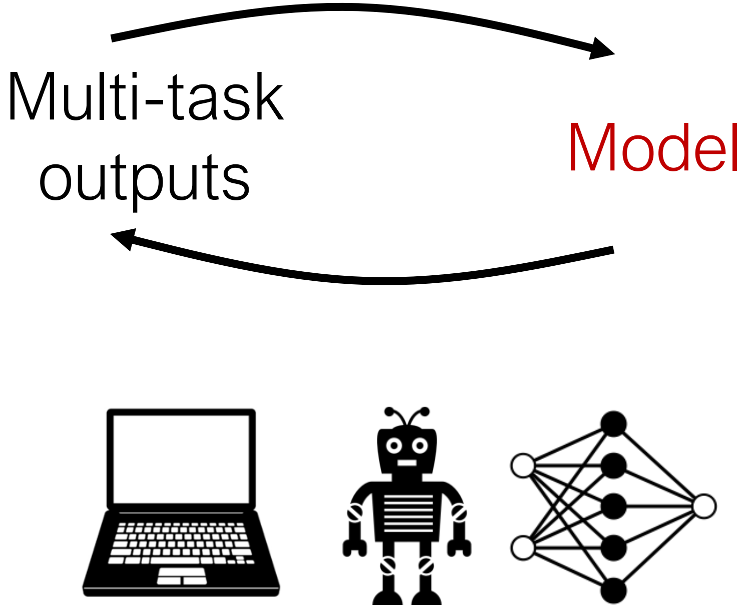
Goal: use large models to empower any user with the capabilities of generative AI



Close the loop between the human user and the model



Fusing multi-modal information in I/O

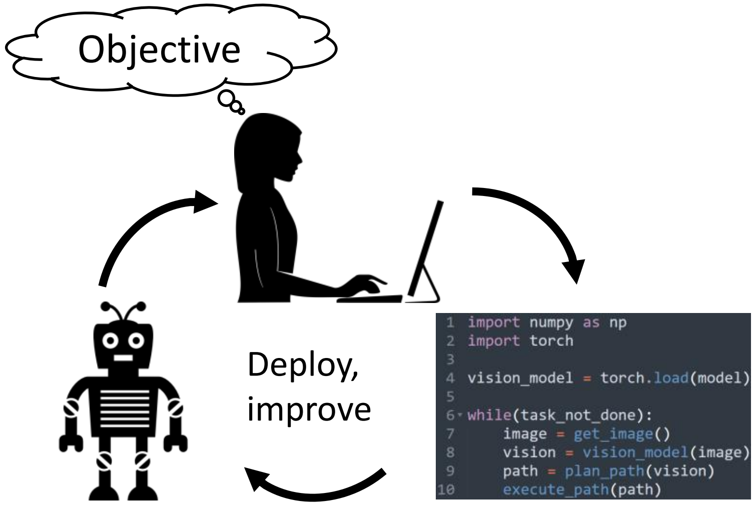


Generalizing a single model to multiple tasks

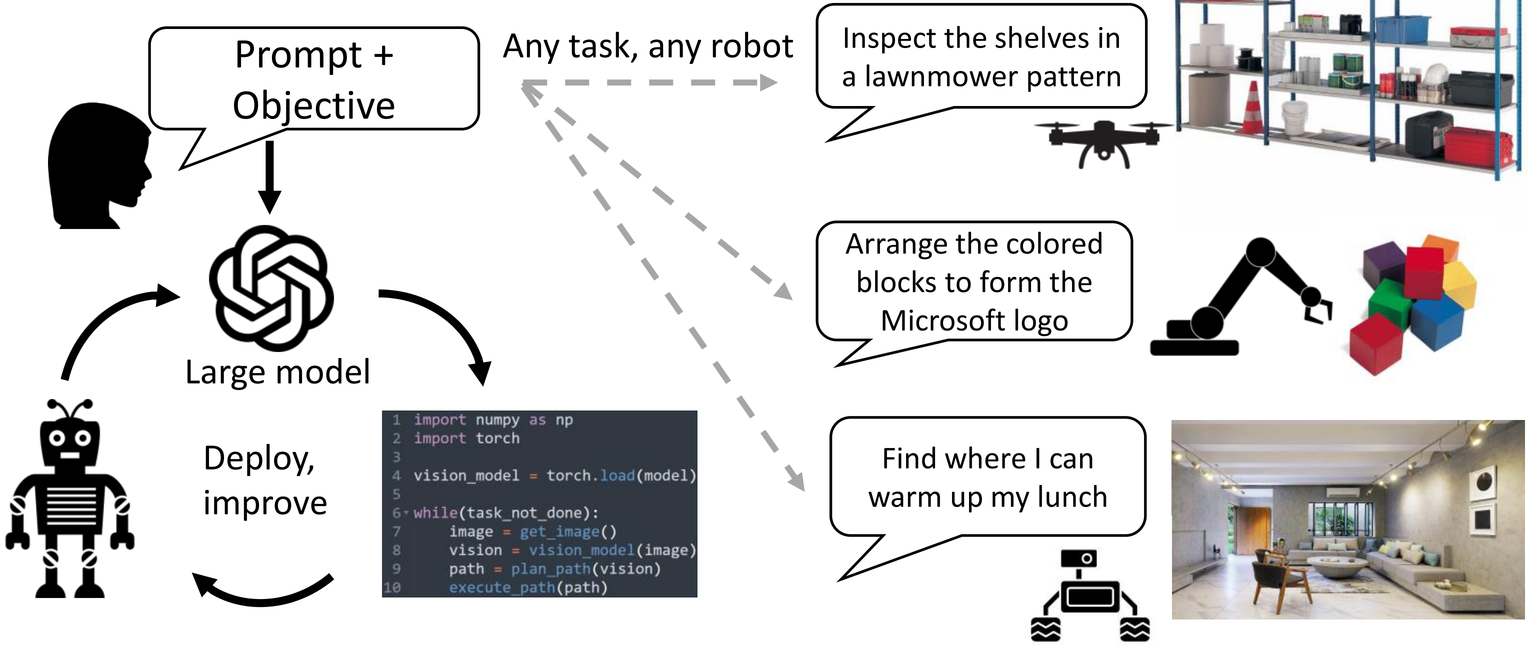
What is the future of human-computer interfaces?

Remove the engineer from the loop. Bring in the user.

Interfaces today: engineer *in the loop*



Goal with ChatGPT: user *on the loop*

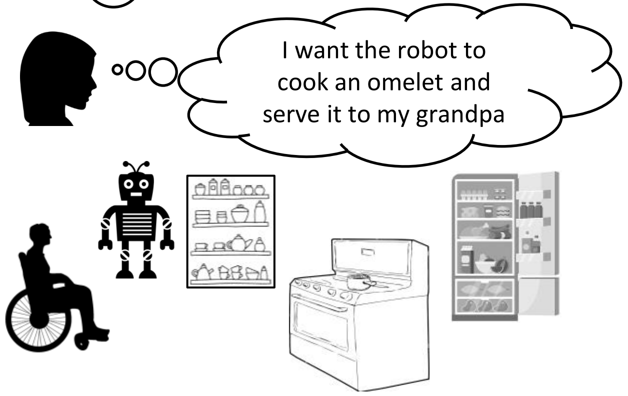


ChatGPT for Robotics: Deployment example

Autonomous flight experiments

Design principles for effective ChatGPT usage:

① Define a task-relevant robot API library*




```python
def locate_object(obj_name):
    # do something
    return

def move_to_location(X,Y,Z):
    # do something
    return

def cook_item(obj_name):
    # do something
    return

def grab_object(obj_name):
    # do something
    return
```

*APIs should be easily implementable on the robot and have descriptive text names for the LLM. They can be chained together to form more complex functions.

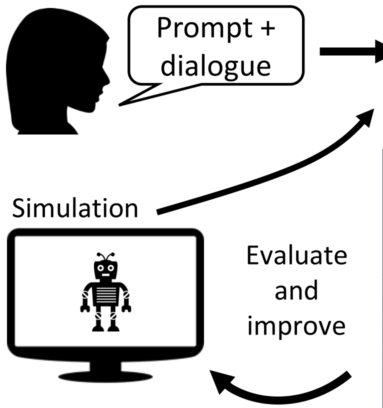
② Build prompt following engineering principles

Consider you are a home assistant robot. You goal is to prepare an omelet for an elderly person. You are equipped with functions:

locate_object(obj_name): returns a X,Y,Z tuple representing the location of the desired object defined by string "obj_name";
move_to_location(X,Y,Z): moves the robot's hands to a specific X,Y,Z location in space. Returns nothing;
cook_item(obj_name): cooks a particular item defined by "obj_name". Returns nothing;
grab_object(obj_name): picks a particular object defined by "obj_name". Returns nothing;

Output python code with the sequence of steps that achieves your objective.

③ User *on the loop*: iterate on solution quality and safety




```python
import torch

vision_model = torch.load('model')
while(task_not_done):
    # Get the current image
    img = get_current_image()
    # Get the prediction
    prediction = vision_model(img)
    # Get the object name
    obj_name = get_object_name(prediction)
    # Get the location of the object
    X,Y,Z = locate_object(obj_name)
    # Move to the object
    move_to_location(X,Y,Z)
    # Grab the object
    grab_object(obj_name)
    # Move to the cooking station
    move_to_location(X,Y,Z)
    # Cook the object
    cook_item(obj_name)
```

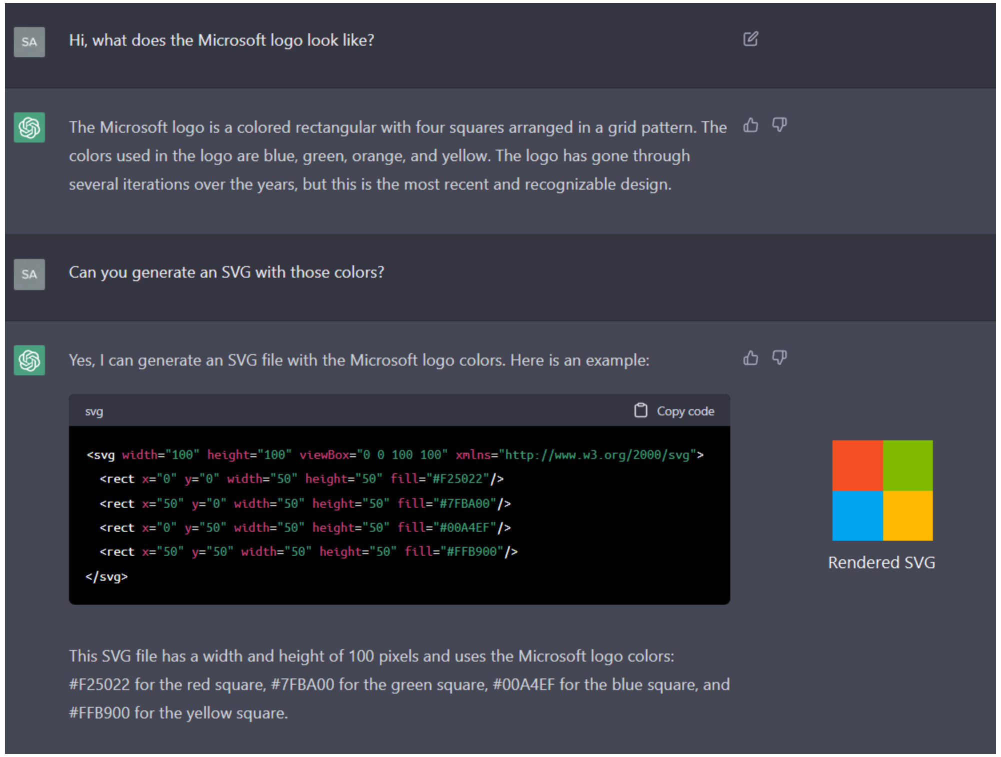
④ Execute!

Deployment with the user in the loop

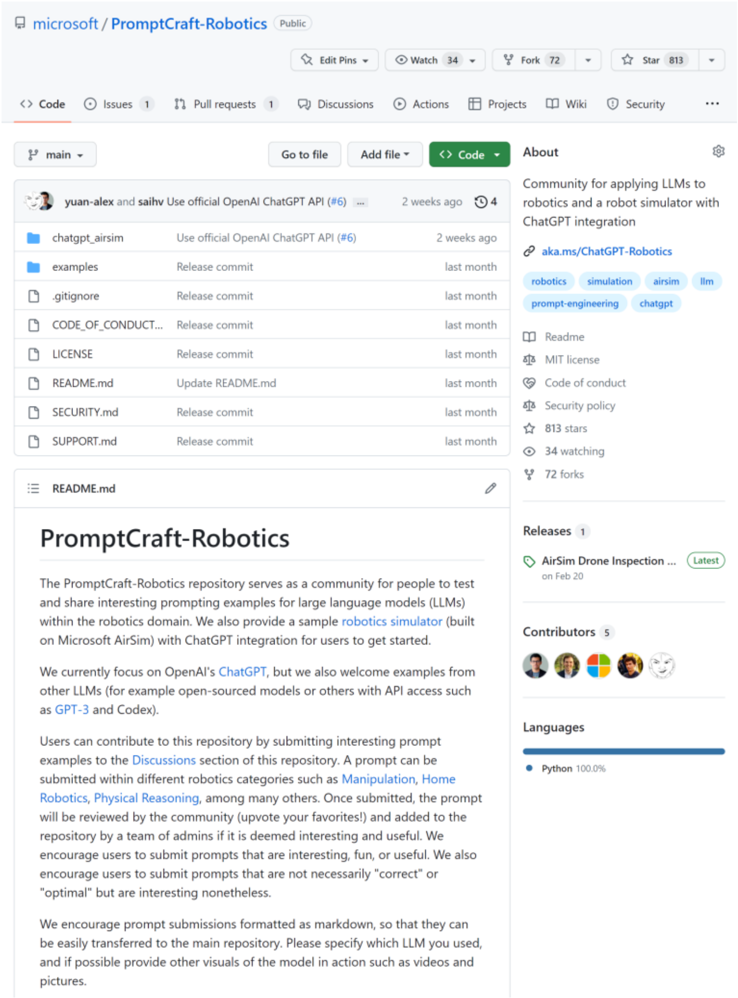
When a conversation is needed for complex tasks

Multi-modal emerging properties

Hallucination over colors, shapes and geometry



Community support: prompt library and simulator


microsoft / PromptCraft-Robotics Public

Edit Pins | Watch 34 | Fork 72 | Star 813

Code | Issues 1 | Pull requests 1 | Discussions | Actions | Projects | Wiki | Security

main | Go to file | Add file | Code

yuan-alex and saihv Use official OpenAI ChatGPT API (#6) ... 2 weeks ago 4

chatgpt_airsim	Use official OpenAI ChatGPT API (#6)	2 weeks ago
examples	Release commit	last month
.gitignore	Release commit	last month
CODE_OF_CONDUCT...	Release commit	last month
LICENSE	Release commit	last month
README.md	Update README.md	last month
SECURITY.md	Release commit	last month
SUPPORT.md	Release commit	last month

README.md

PromptCraft-Robotics

The PromptCraft-Robotics repository serves as a community for people to test and share interesting prompting examples for large language models (LLMs) within the robotics domain. We also provide a sample robotics simulator (built on Microsoft AirSim) with ChatGPT integration for users to get started.

We currently focus on OpenAI's ChatGPT, but we also welcome examples from other LLMs (for example open-sourced models or others with API access such as GPT-3 and Codex).

Users can contribute to this repository by submitting interesting prompt examples to the Discussions section of this repository. A prompt can be submitted within different robotics categories such as Manipulation, Home Robotics, Physical Reasoning, among many others. Once submitted, the prompt will be reviewed by the community (upvote your favorites!) and added to the repository by a team of admins if it is deemed interesting and useful. We encourage users to submit prompts that are interesting, fun, or useful. We also encourage users to submit prompts that are not necessarily "correct" or "optimal" but are interesting nonetheless.

We encourage prompt submissions formatted as markdown, so that they can be easily transferred to the main repository. Please specify which LLM you used, and if possible provide other visuals of the model in action such as videos and pictures.

About

Community for applying LLMs to robotics and a robot simulator with ChatGPT integration

aka.ms/ChatGPT-Robotics

robotics simulation airsim llm prompt-engineering chatgpt

Readme
MIT license
Code of conduct
Security policy
813 stars
34 watching
72 forks

Releases 1

AirSim Drone Inspection ... Latest
on Feb 20

Contributors 5

Languages

Python 100.0%

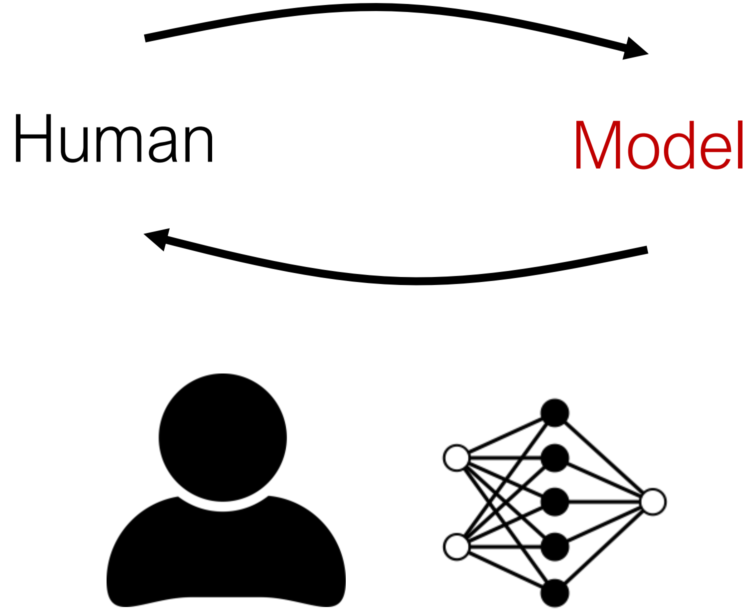
ChatGPT Prompting Guides & Examples

The list below contains links to the different robotics categories and their corresponding prompt examples. We welcome contributions to this repository to add more robotics categories and examples. Please submit prompt examples to the Discussions page, or submit a pull request with your category and examples.

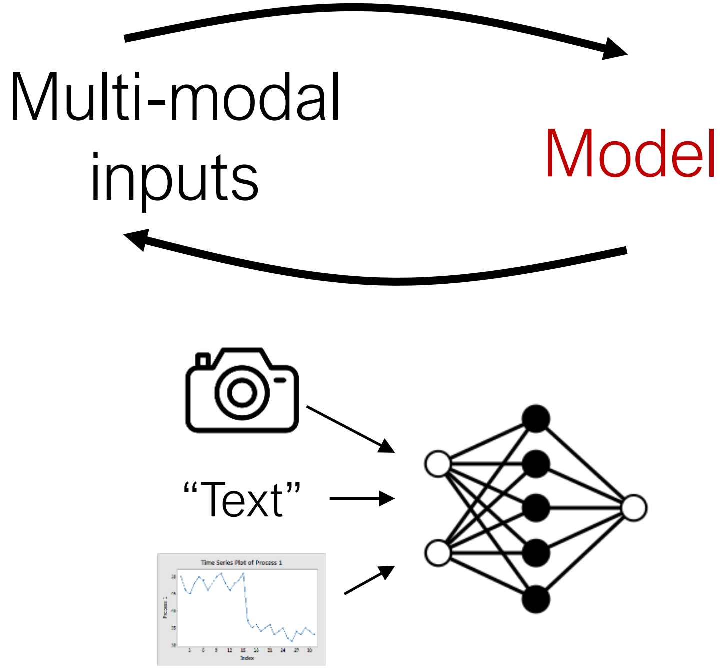
- Embodied agent
  - ChatGPT - Habitat, closed loop object navigation 1
  - ChatGPT - Habitat, closed loop object navigation 2
  - ChatGPT - AirSim, object navigation using RGBD
- Aerial robotics
  - ChatGPT - Real robot: Tello deployment | Video Link
  - ChatGPT - AirSim turbine Inspection | Video Link
  - ChatGPT - AirSim solar panel Inspection
  - ChatGPT - AirSim obstacle avoidance | Video Link
- Manipulation
  - ChatGPT - Real robot: Picking, stacking, and building the MSFT logo | Video Link
  - ChatGPT - Manipulation tasks
- Spatial-temporal reasoning
  - ChatGPT - Visual servoing with basketball

Open-sourced AirSim simulator with ChatGPT API:

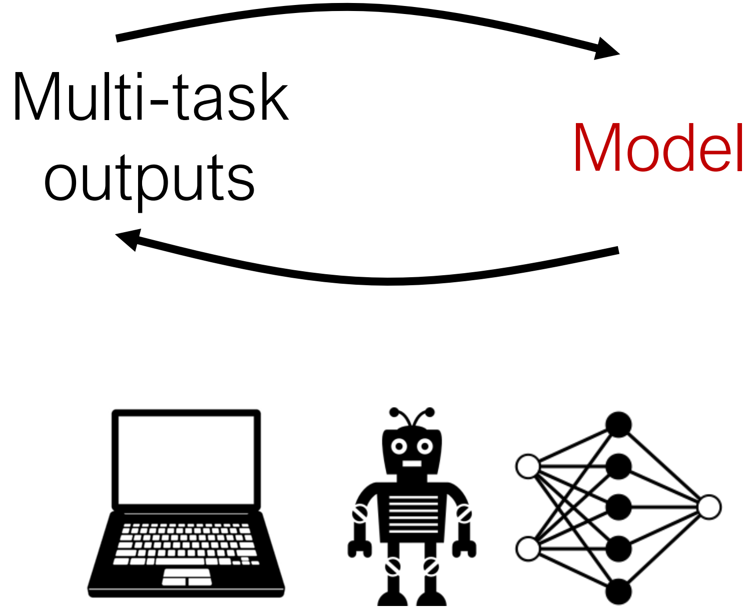
Goal: use large models to empower any user with the capabilities of generative AI



Close the loop between the human user and the model



Fusing multi-modal information in I/O

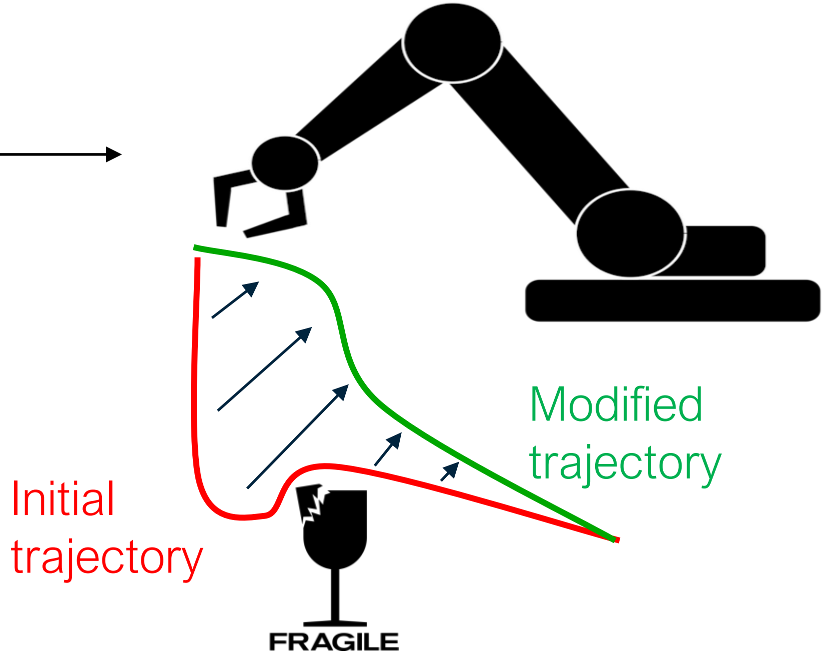


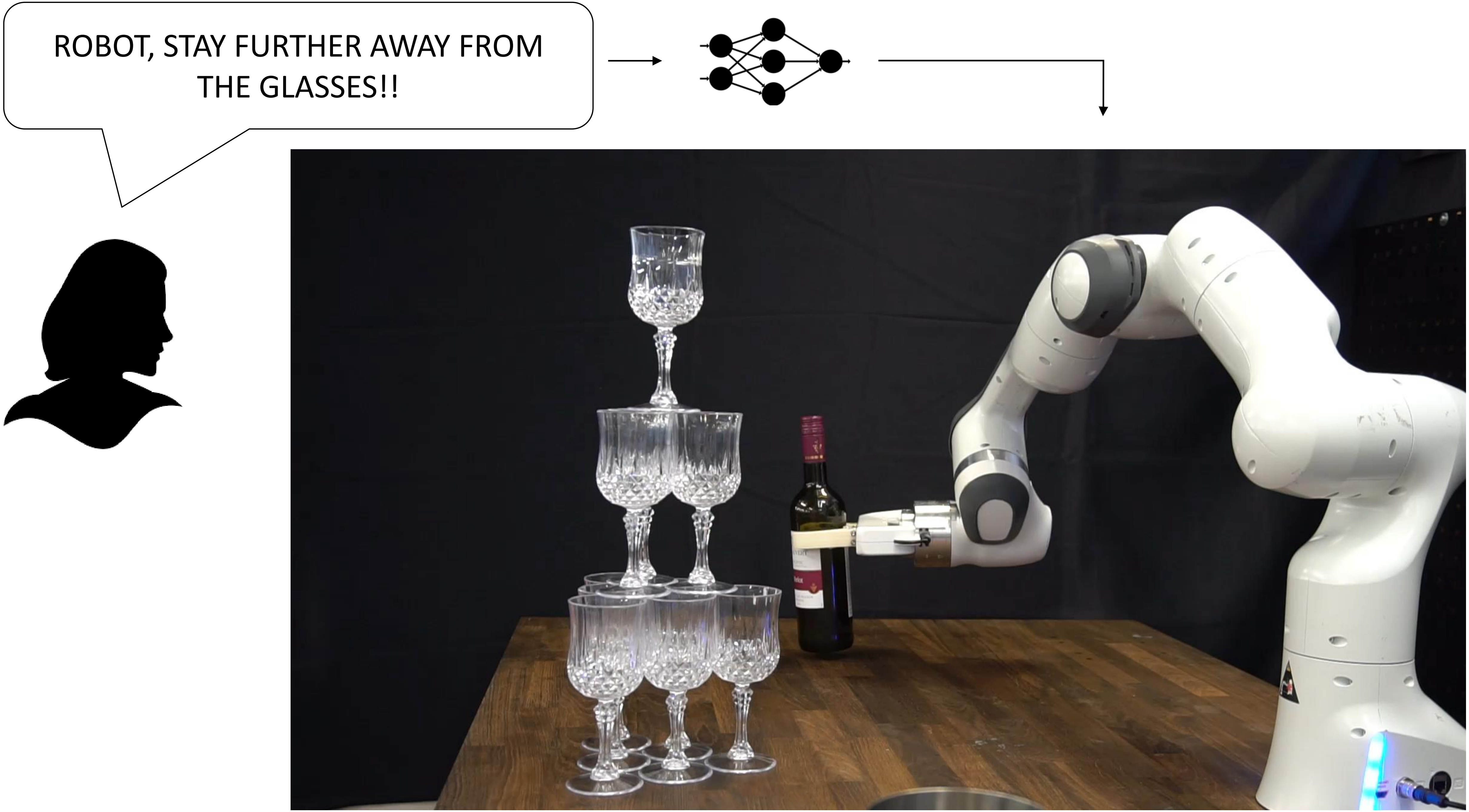
Generalizing a single model to multiple tasks

Goal: reshape robot trajectories based on user language inputs

Interface

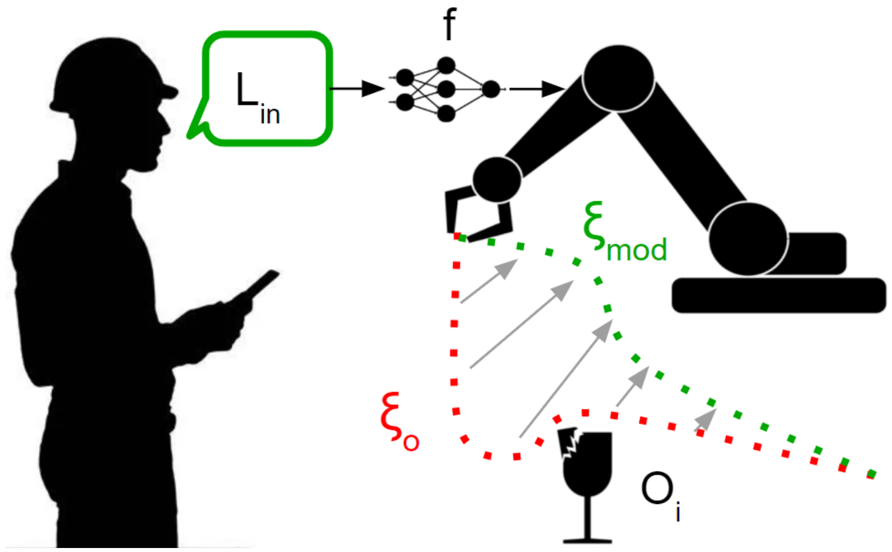
- Natural Language
- Mouse-keyboard
- Programming
- Drawing
- Kinesthetic teaching
- Touch-screens



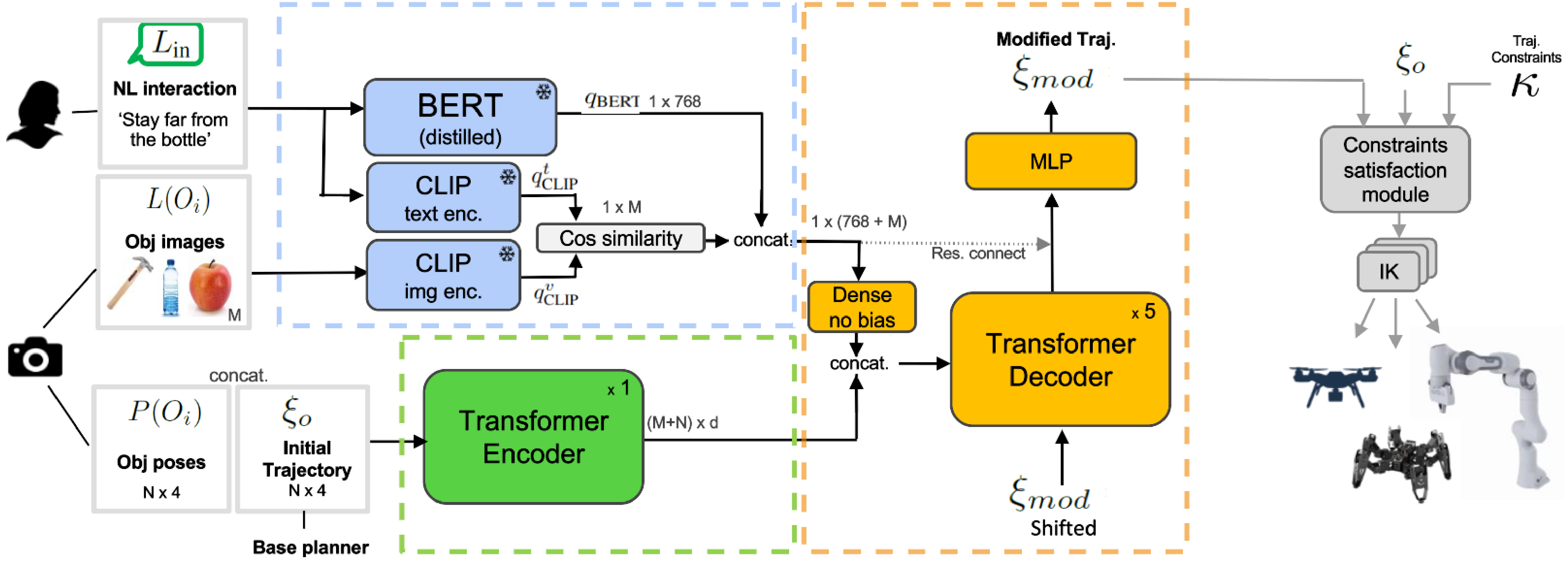

ROBOT, STAY FURTHER AWAY FROM THE GLASSES!!

Mathematical definitions

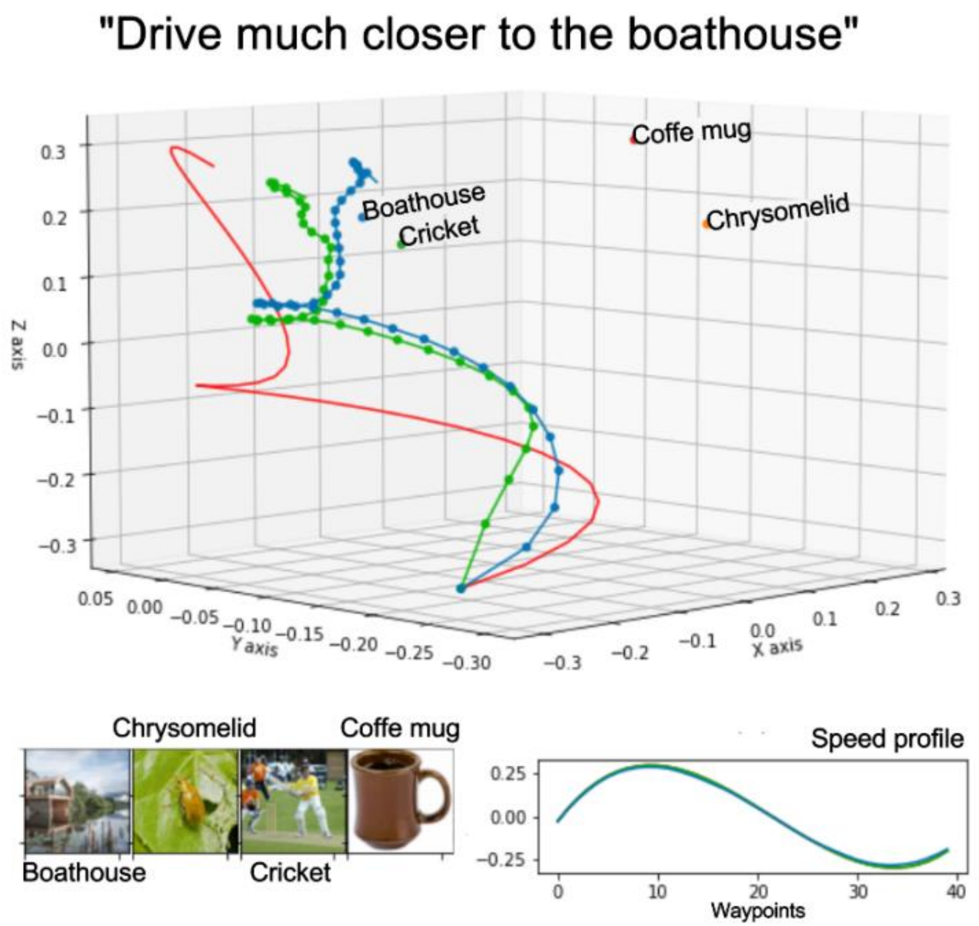
- Original trajectory: $\xi_o : [-1, 1] \rightarrow \mathbb{R}^4 \quad \{(x_1, y_1, z_1, v_1), ..., (x_N, y_N, z_N, v_N))\}$
- Objects in the scene: $\mathcal{O} = \{O_1, ..., O_M\}$
  - Position $P(O_i) \in \mathbb{R}^3$
  - Image label: $I(O_i)$ 
- Language input: L_{in}
- Learning objective: $\xi_{mod} = f(\xi_o, L_{\text{in}}, \mathcal{O})$

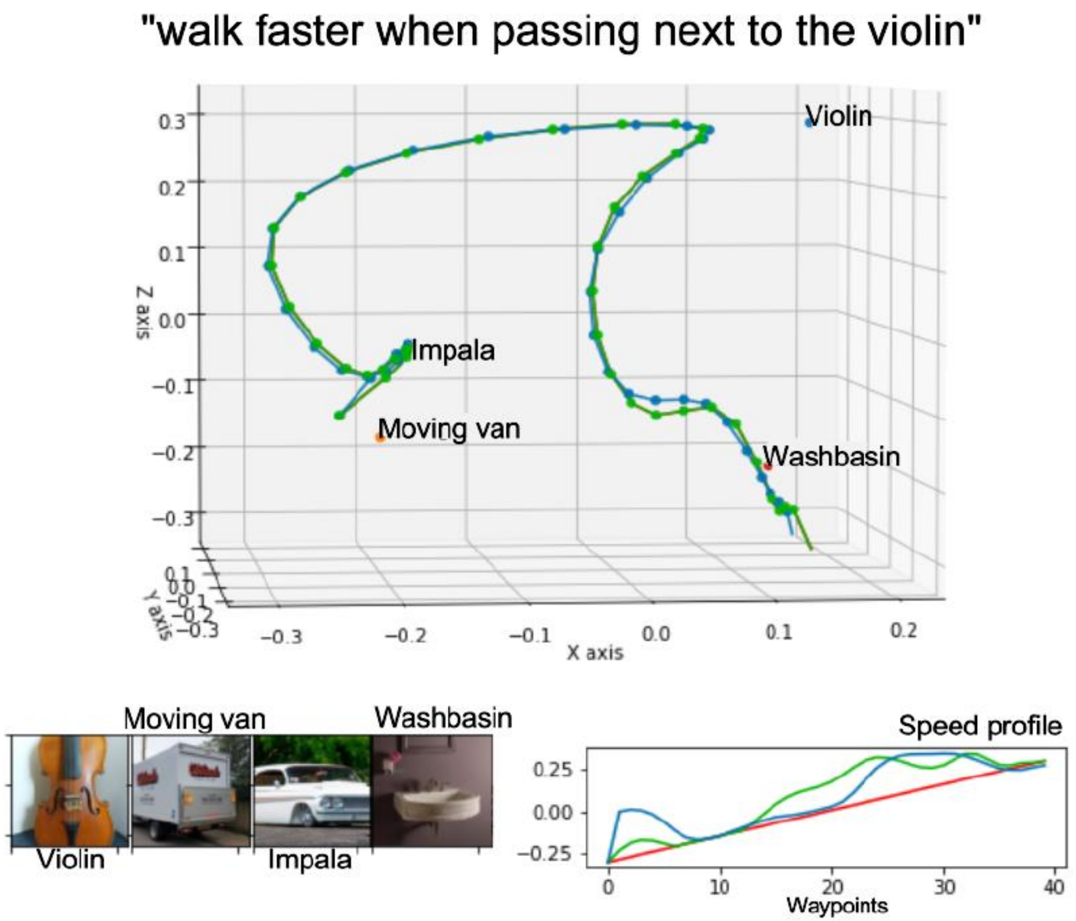


System overview



It is expensive to collect real-world trajectory correction examples. We use procedural data generation.





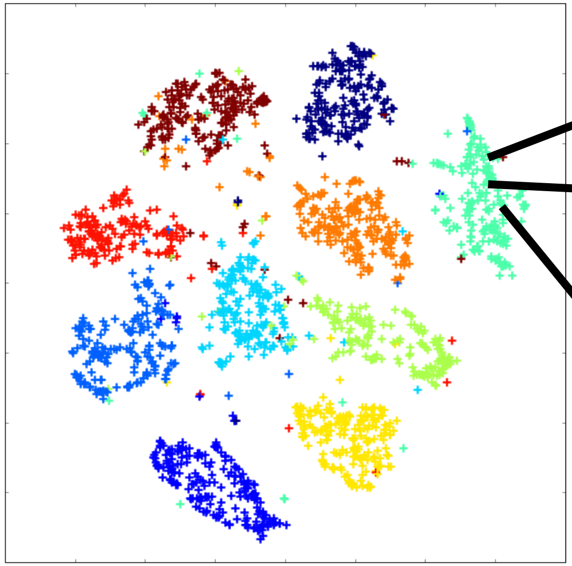
Key idea: use frozen LLM embeddings to compensate for low synthetic dataset variations

WIKIPEDIA
The Free Encyclopedia

Large language models

- GPT-3
- BERT
- CLIP

Latent sentence embeddings:



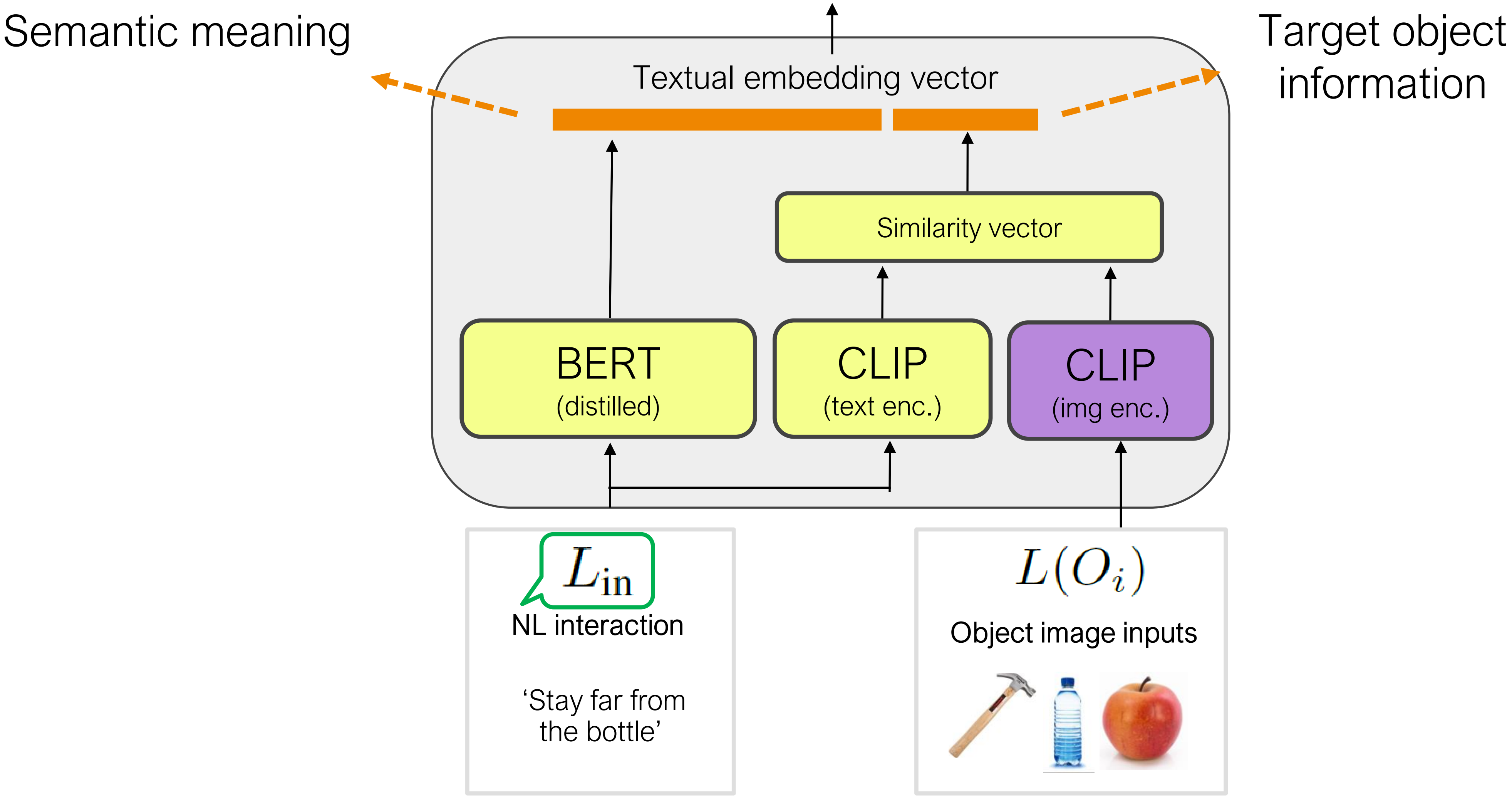
Training sentences
"Stay far away from x"
"Keep some distance from x"

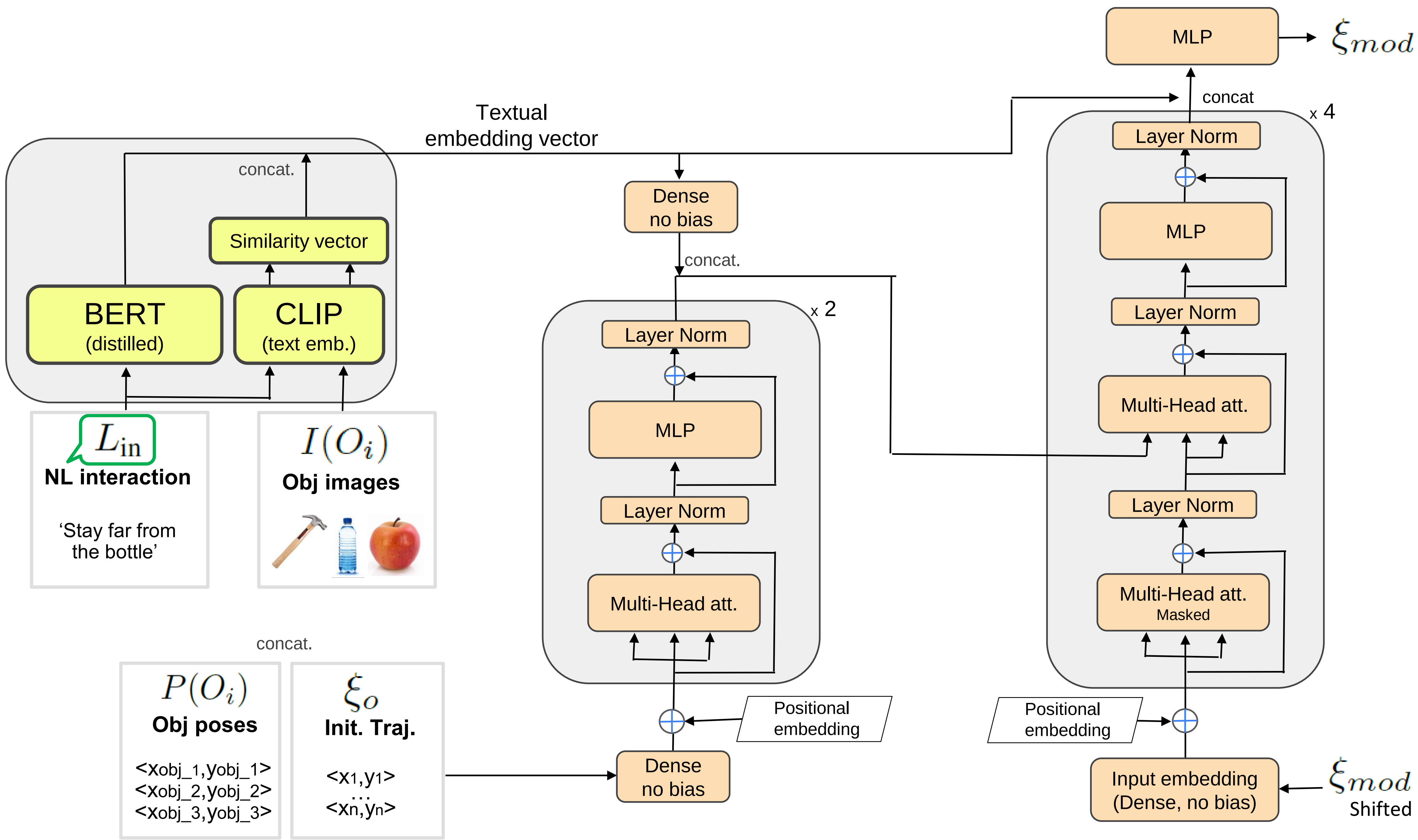
Unseen vocabulary
"Avoid being close to x"
"Move away from x"

Advantages:

- Richer representations
- Less training data required
- More robust to vocabulary variations

Semantic sentence embedding

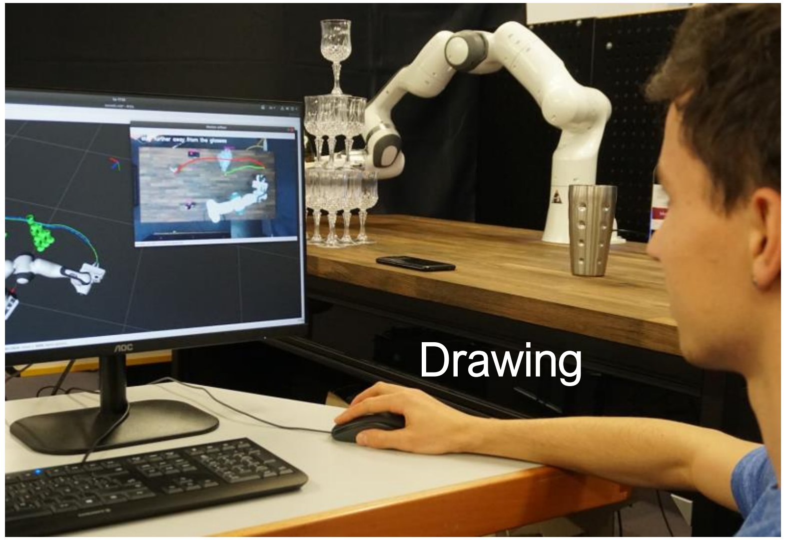




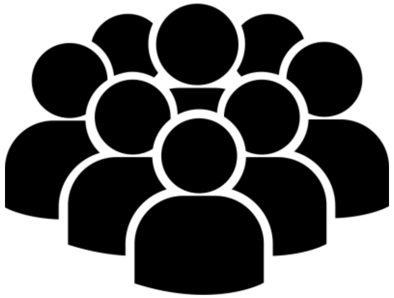
MLP
ξ_{mod}
concat
x 4
Layer Norm
MLP
Layer Norm
Multi-Head att.
Layer Norm
Multi-Head att.
Masked
Positional embedding
Input embedding (Dense, no bias)
ξ_{mod}
Shifted
Textual embedding vector
concat.
Similarity vector
BERT (distilled)
CLIP (text emb.)
L_{in}
NL interaction
'Stay far from the bottle'
$I(O_i)$
Obj images
Dense no bias
concat.
x 2
Layer Norm
MLP
Layer Norm
Multi-Head att.
Positional embedding
Dense no bias
concat.
$P(O_i)$
Obj poses
<xobj_1,yobj_1>
<xobj_2,yobj_2>
<xobj_3,yobj_3>
ξ_o
Init. Traj.
<x1,y1>
...
<xn,yn>

LaTTe: Language Trajectory TransformEr

User study: comparison between interfaces



Results: multi-model model is effective, precise and intuitive



Interface	Satisfied	Easy to use	Safety	Natural	Predictable
Natural Language (ours)	**90**	**92**	92	**98**	72
Kinesthetic Teaching	**90**	88	88	78	**96**
Drawing	88	74	**100**	80	88
Parameter tuning	62	58	88	62	48

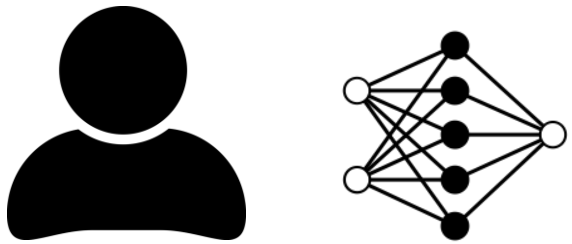
User ratings collected in the user study

Av. Iterations	Success rate (%)	Avr. Time (s)
1.33	**100**	**81**
1.78	56	139
1.89	65	120
4.00	92	284

User study metadata

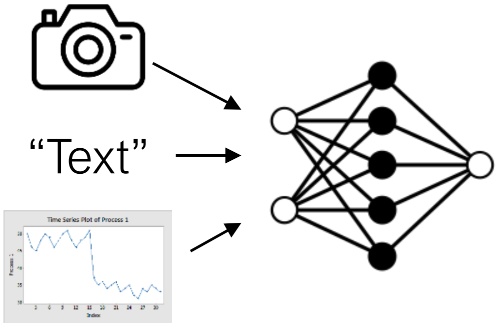
Goal: use large models to empower any user with the capabilities of generative AI

Human ⇄ Model



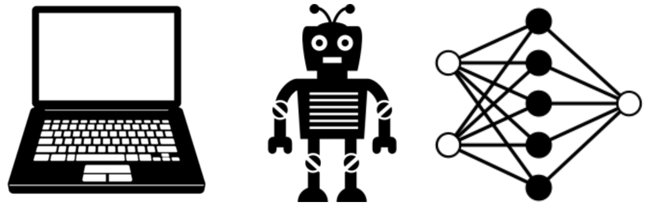
Close the loop between the human user and the model

Multi-modal inputs ⇄ Model



Fusing multi-modal information in I/O

Multi-task outputs ⇄ Model



Generalizing a single model to multiple tasks

General Pre-Training for Decision-Making

Language models

Input:

Recite the first law of robotics

GPT-3

Output:
Sequence of words

A robot may not injure a human being or, through inaction allow a human to be harmed…

Autonomous systems models

Input:

Images/ Depth | LiDAR | Language: ‘Go fast!’

PACT

Perception-Action Causal Transformer

Output:
Sequence of actions

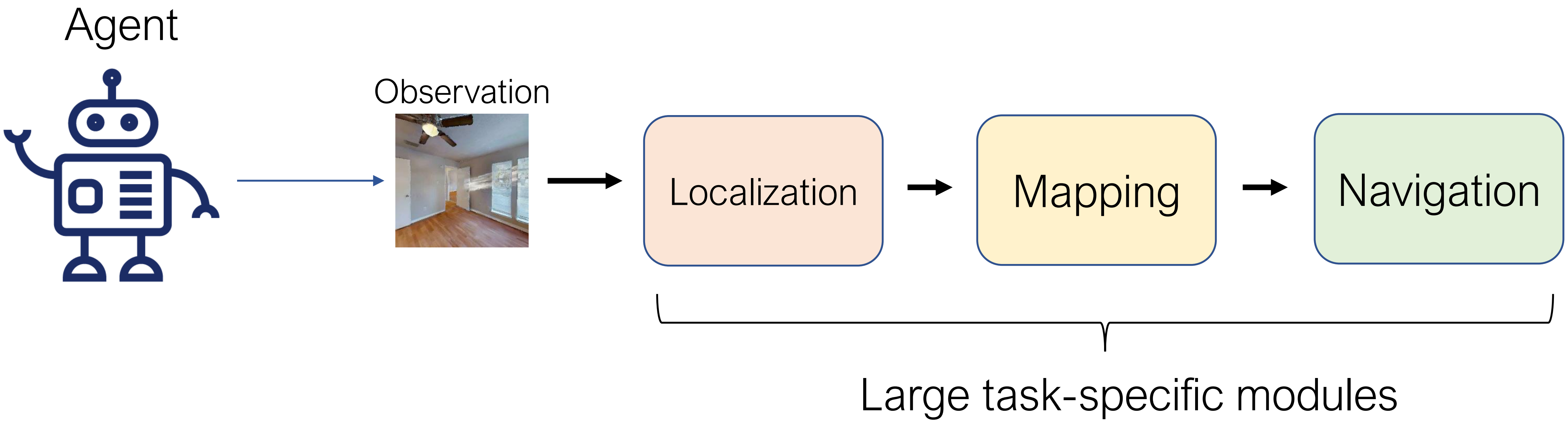
$$\theta_0 = 10^o, V_0 = 0.5m/s$$
$$\theta_1 = 15^o, V_1 = 0.1m/s$$
$$\theta_2 = 18^o, V_2 = -0.7m/s$$

Actuators

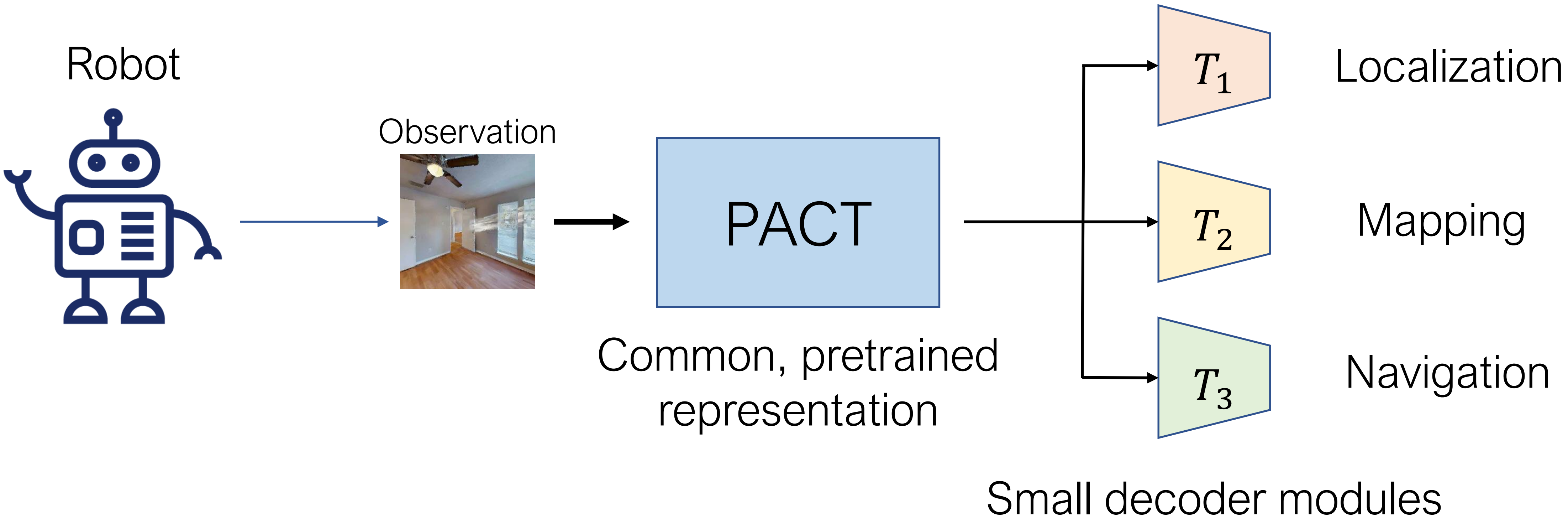
Environment

New sensor inputs

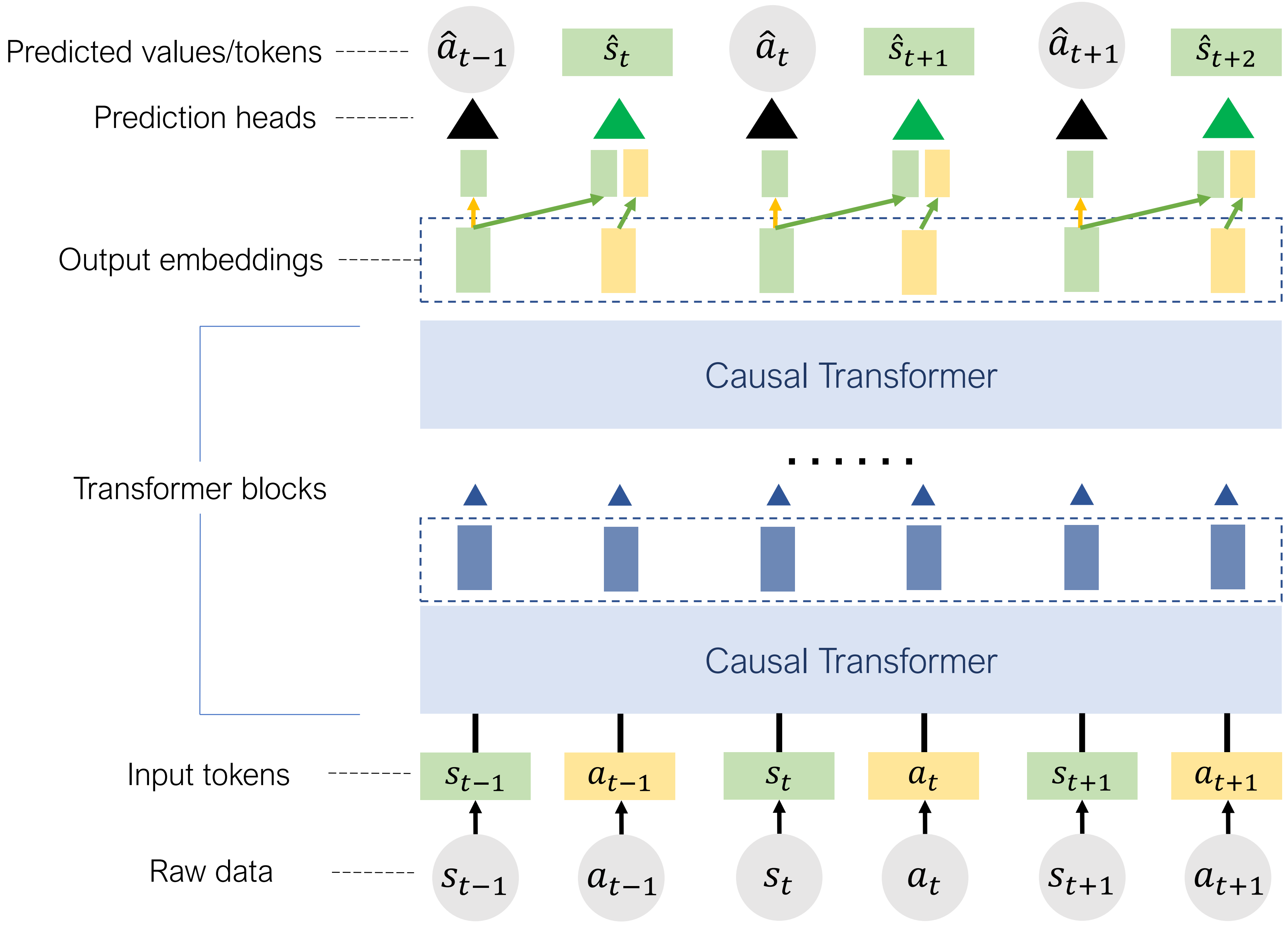
Traditional Autonomous Systems Pipeline



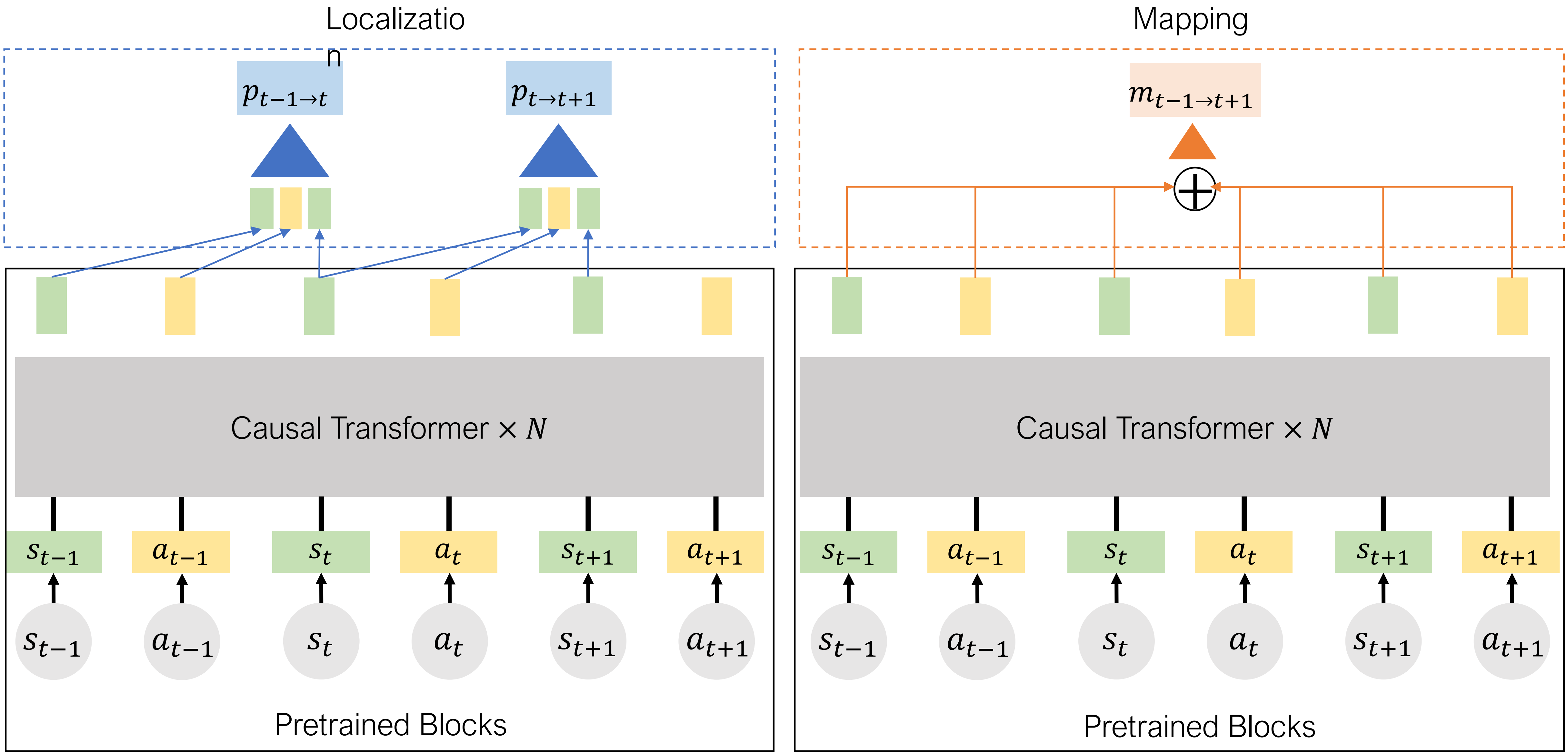
Foundational models for decision-making: Deployment in multiple downstream tasks



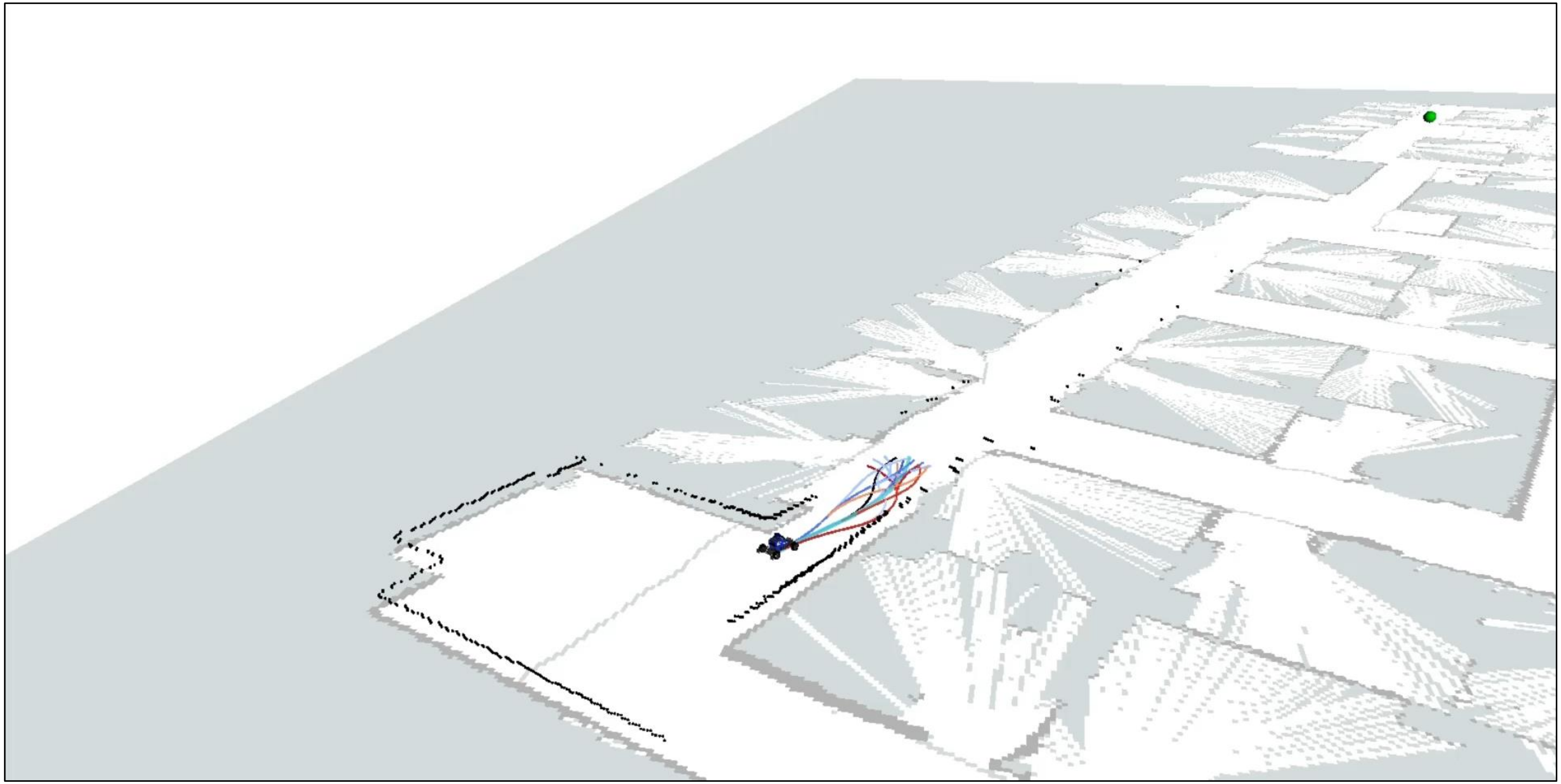
Pre-training network:

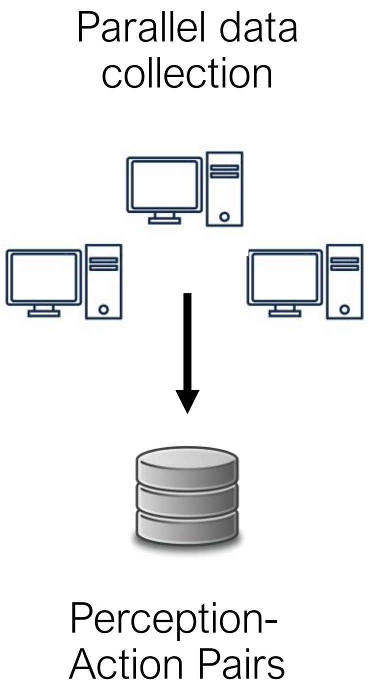


Downstream task networks:

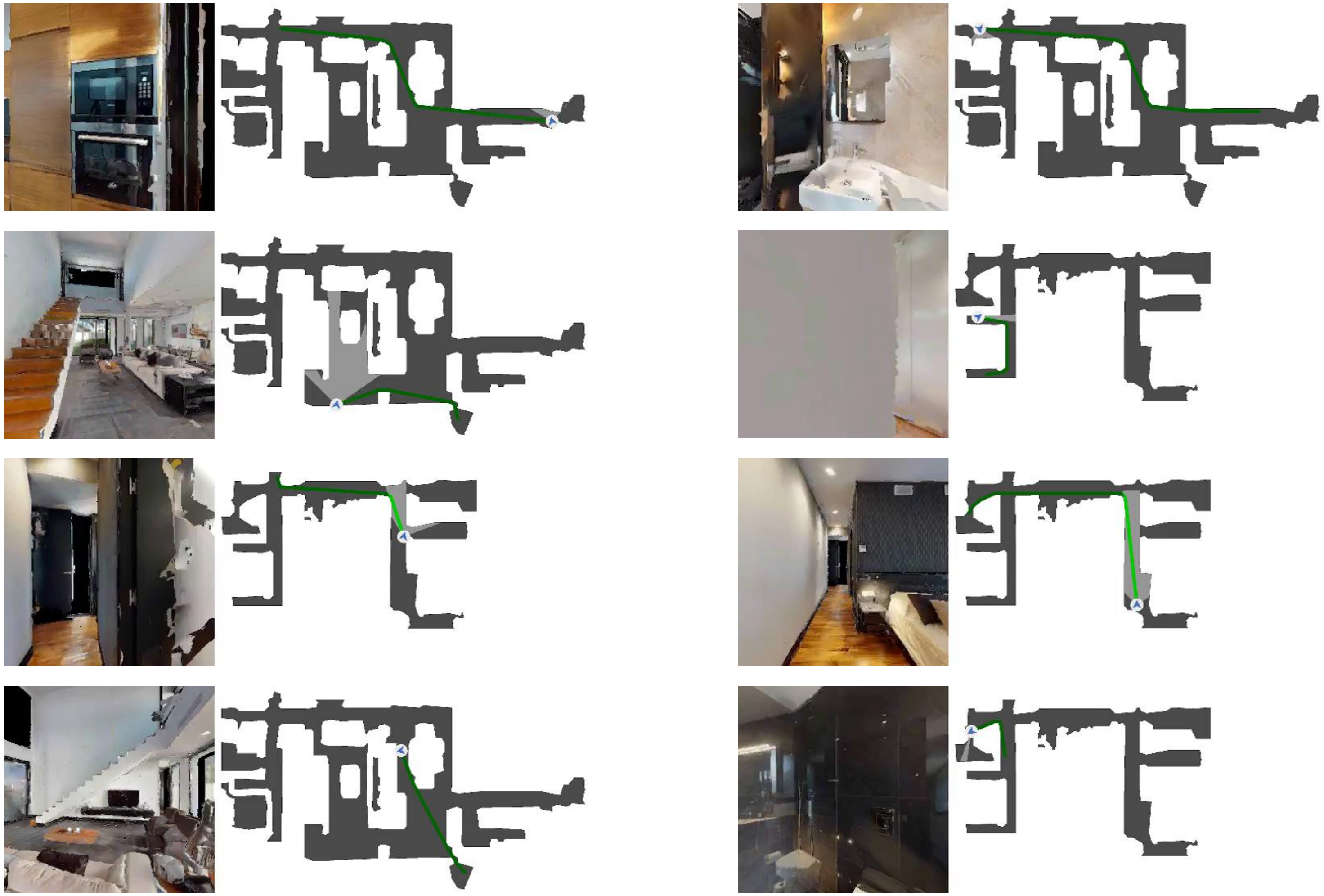


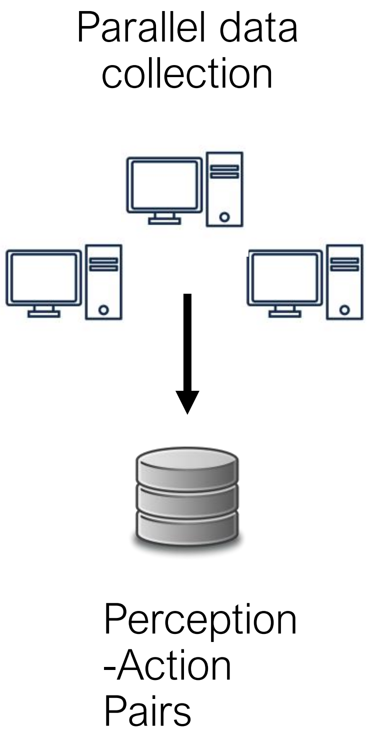
Pre-training data collection for MuSHR: millions of perception-action pairs



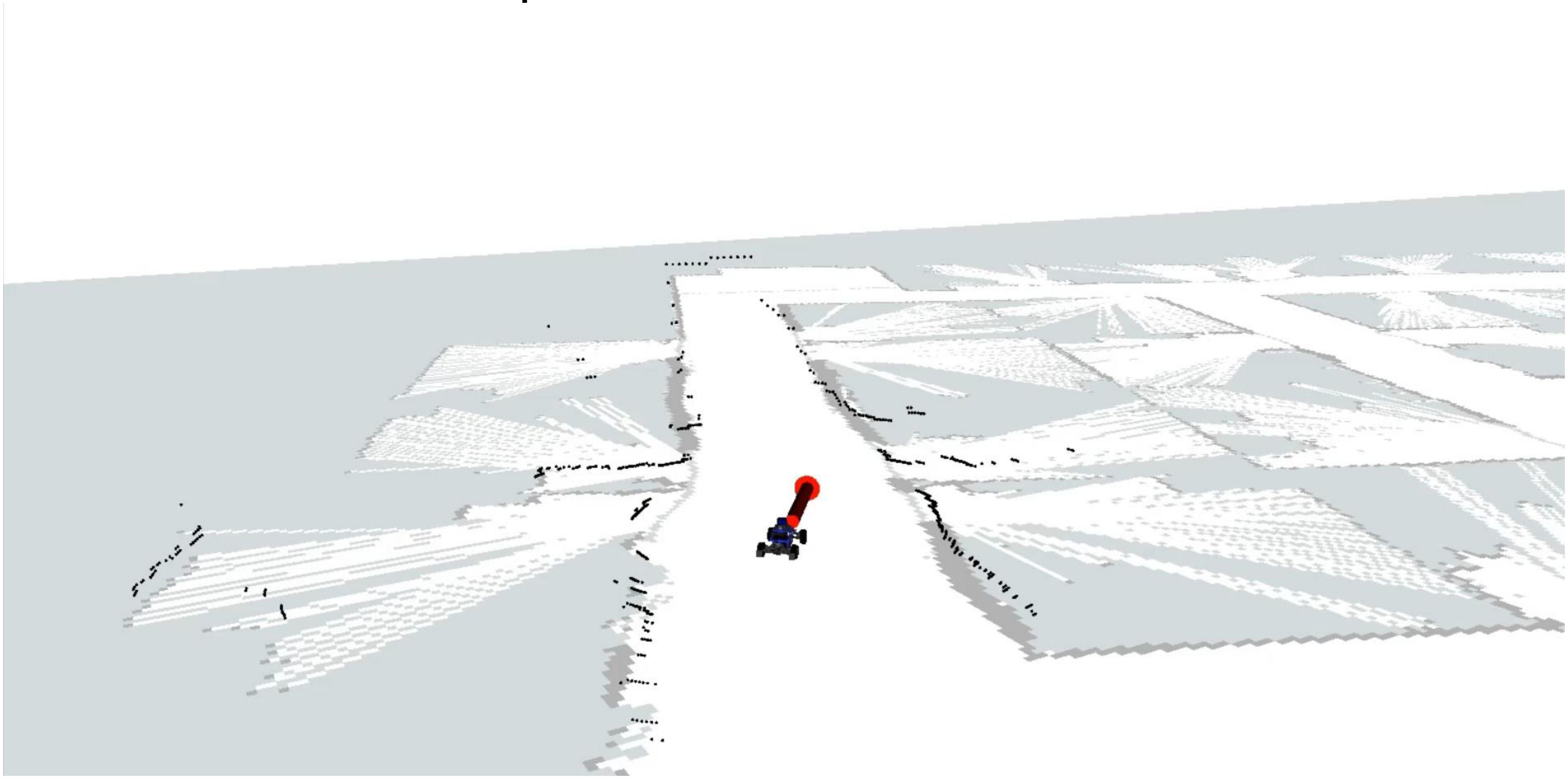


Pre-training data collection for Habitat: millions of perception-action pairs

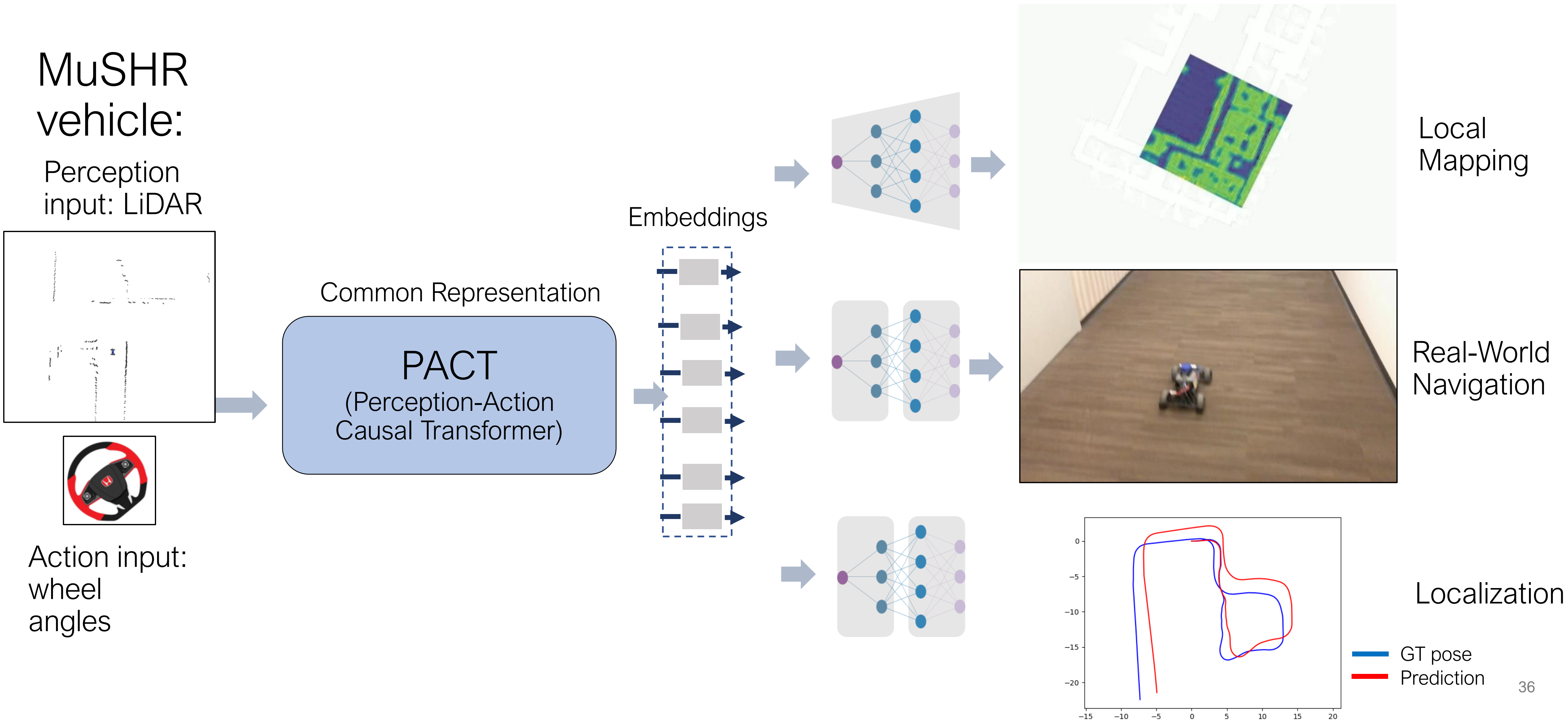




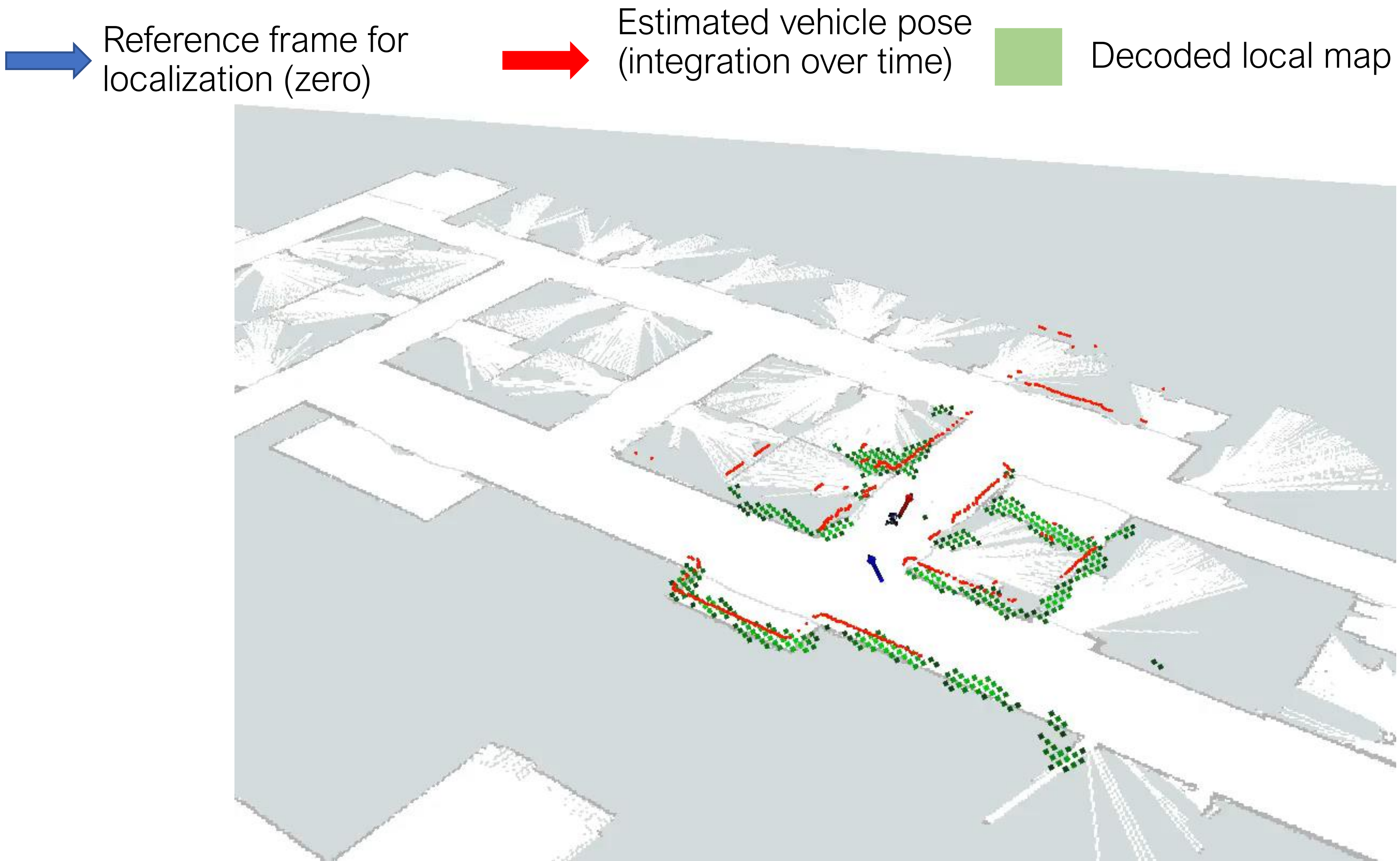
What does the pre-trained model learn?



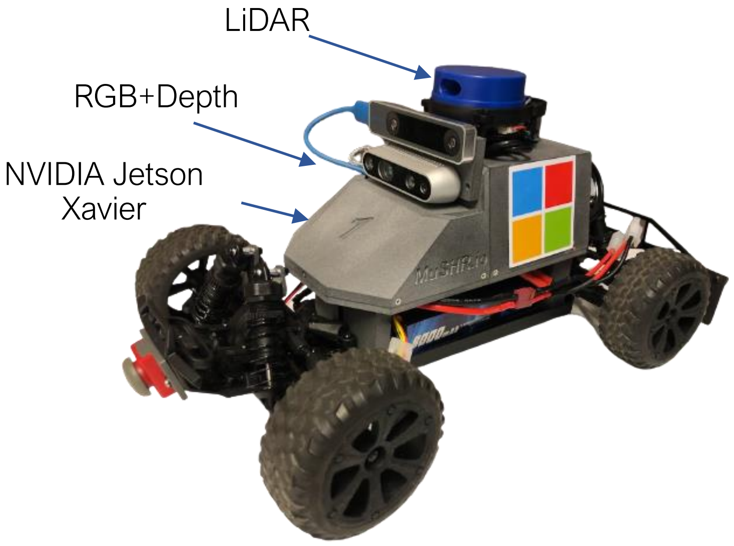
PACT applied towards multiple downstream tasks

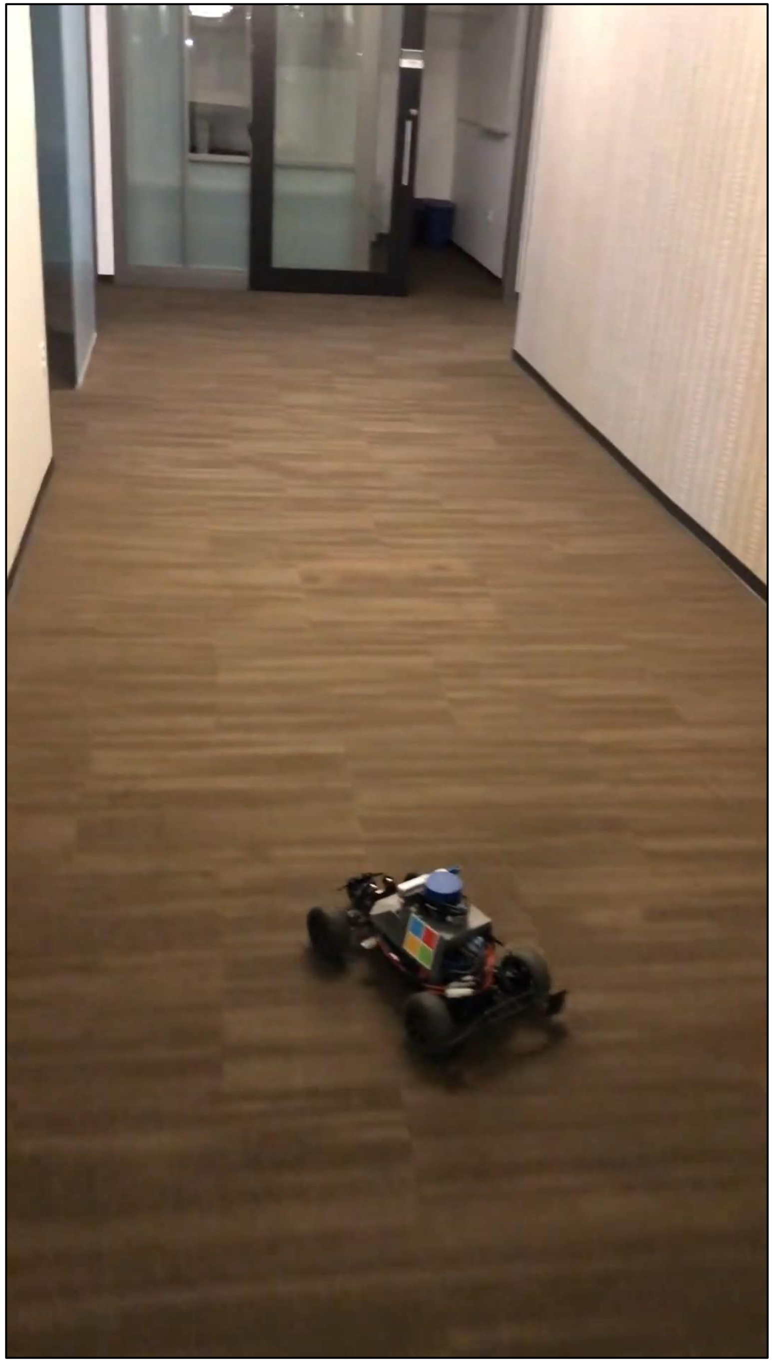


Simultaneous navigation, localization and mapping model deployment in simulation



MuSHR deployment in the wild:

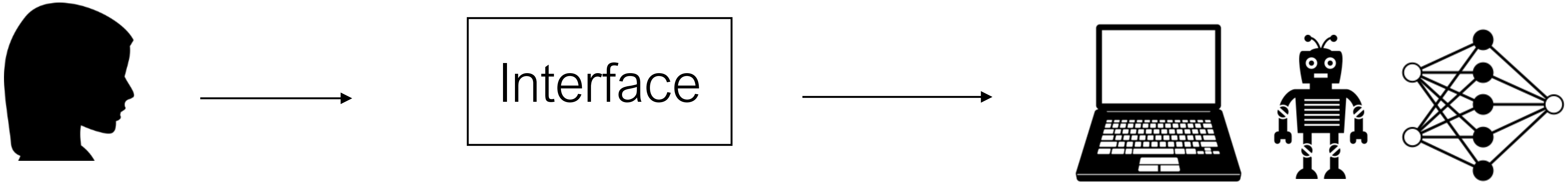


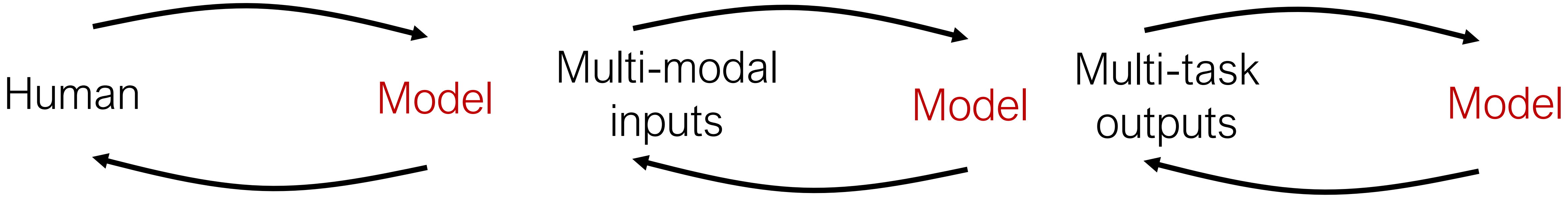


Pre-trained model overcomes sim-to-real gap:

- Model dynamics
- Actuator and processing delays
- Sensor noise and imperfect LiDAR returns (e.g. glass surfaces)

Use large models to empower any user with the capabilities of generative AI





Close the loop between the human user and the model

Fusing multi-modal information in I/O

Generalizing a single model to multiple tasks

Thanks to collaborators:

- Microsoft:
  - Ashish Kapoor
  - Sai Vemprala
  - Shuang Ma
  - Shuhang Chen
  - Felipe Frujeri





- TUM, Germany:
  - Arthur Bucker
  - Luis Figueredo
  - Sami Haddadin



Q&A + discussions