The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

April 22, 2026 CSC 59866

Advanced Topics: Robotic, AR/VR/XR and Autonomous Transportation Agents

Logistics and Motivation

Recall Lecture 21: We discussed smart grid orchestration and decentralized multi-agent resource allocation.

The Final Step: We are moving our agents off the server and out of the digital twin. How do agents interact with the messy, continuous, unpredictable *physical world*?

Today's Agenda

1. **Robotic Agents:** Imitation Learning, VLA Models (RT-2), and Scaling (Open X-Embodiment, $\pi_{0.7}$).
2. **Autonomous Transportation:** World Models (GAIA-1) and Causal Reasoning (Alpamayo-R1).
3. **AR/VR/XR Agents:** Proactive Mixed Reality (Project Astra, YETI).

Robotic Agents

Imitation Learning and Affordable Hardware w/ ALOHA

We can't train a robotic Agent in the real world with pure RL (unless we want our chef agent to smash all the plates through trial and error)

By collecting data from humans, we can train our robot to map pixel movements to joint velocities so that we can learn directly from experts!

ALOHA and Mobile ALOHA were created with relatively cheap (< $32k) telescopic rigs for robots that could do household chores.

With only 50 demonstrations, an Action Chunking with Transformers (ACT) neural network could perform cooking tasks (e.g. flipping shrimp).

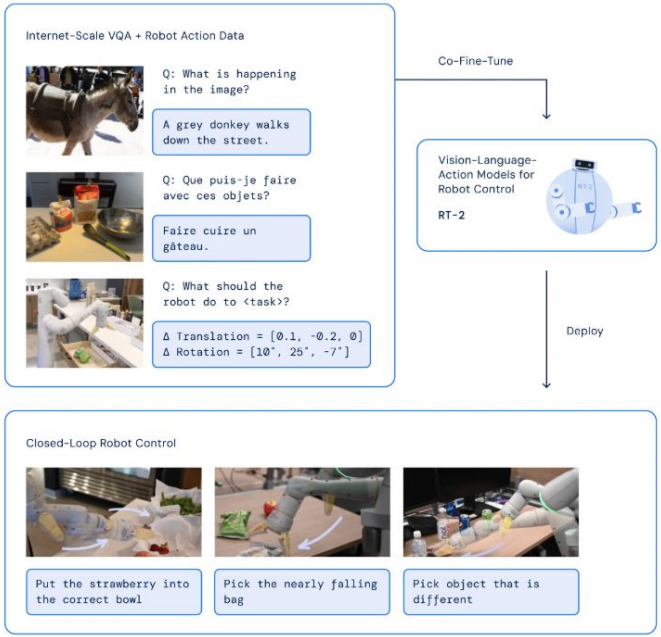
Vision-Language-Action (VLA) Models

ALOHA agents are "narrow", they only know how to wipe a spill if you trained them to wipe a spill.

RT-2 (Google, 2023): What if we treat actions as another "language"?

Researchers tokenized continuous joint commands and mixed them into the training data of a massive web-scale Vision-Language Model.

You can tell RT-2, "Move the extinct animal to the red star." It looks at the table, identifies a toy dinosaur, and moves it, even though it was *never trained on dinosaurs*.



Scaling Embodiment: Open-X & $\pi_{0.7}$

The Open X-Embodiment Dataset: Data from 22 different robot platforms.

The Problem: A robot arm with 6 joints cannot share a neural network output with a robot dog with 12 joints.

The Solution ($\pi_{0.7}$): A steerable generalist foundation model for robotics (Physical Intelligence). It outputs a flow-matched continuous latent representation that is dynamically decoded by the specific robot's hardware.

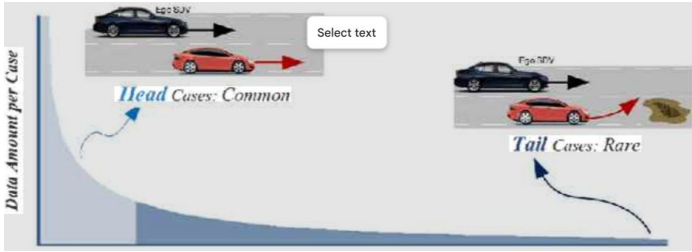
Emergent Steerability: Because of its scale, $\pi_{0.7}$ exhibits emergent capabilities where human users can dynamically steer its real-time physical actions without requiring task-specific retraining.

Autonomous Transportation

The Autonomous Driving Problem

Self-driving cars are essentially massive, heavy multi-agent reinforcement learning problems operating in safety-critical environments.

The "Long Tail" Problem: Neural networks learn the 99% of normal driving easily. The 1% of bizarre edge cases (e.g., driving the wrong way down the highway) causes catastrophic failures.



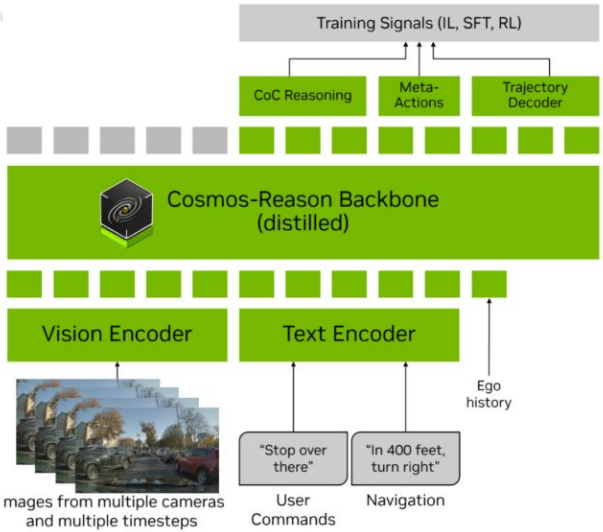
Interpretable VLAs

How do we solve the long tail? We force the model to *reason out loud* before it acts.

Alpamayo-R1 (NVIDIA, 2026): Introduces a Chain of Causation (CoC).

Before outputting steering angles, the model must output causal text explaining *why* it is making the decision.

They use Reinforcement Learning (RL) to explicitly penalize the model if its text reasoning contradicts its physical driving actions!



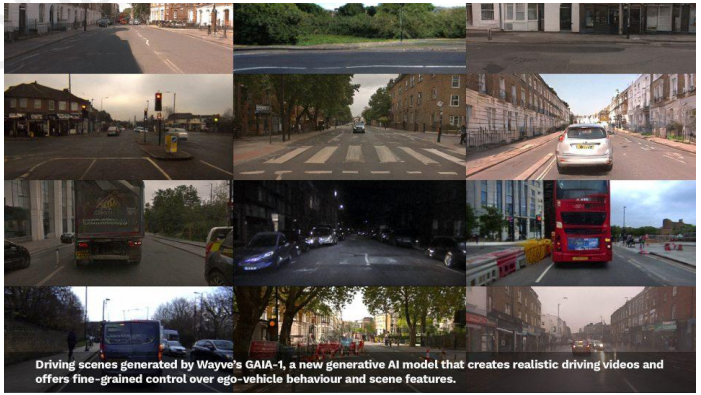
World Models vs. Simulators

To train better cars, we need better simulators. But hand-coding a video game simulator is rigid and lacks real-world diversity.

GAIA-1 (Wayve, 2023): A Generative World Model.

It predicts the *future state of the world* (video tokens) conditioned on the car's current action and text instructions.

It allows us to generate infinite, photo-realistic, completely hallucinated driving data to train our agents on edge cases that never actually happened in reality.



End-to-End Driving with Diffusion (MVLAD-AD)

The Output Bottleneck: LLMs generate output token-by-token (autoregressively). If an agent needs to plan a 5-second trajectory, waiting for 50 tokens to generate one by one is too slow for highway driving.

MVLAD-AD (2026): Replaces the autoregressive token generation with a **Diffusion Model**.

Instead of predicting the next steering angle, the agent starts with pure noise and denoises it into a complete, geometrically sound trajectory all at once, conditioned on the Vision-Language embeddings.

The Result: Faster inference latency and smoother trajectories by avoiding the compounding errors of token-by-token prediction.

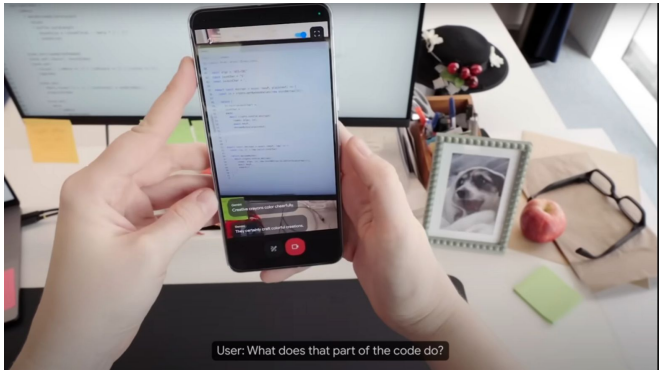
AR/VR/XR Agents

Ubiquitous Agents (Project Astra)

We are moving away from pulling a phone out of our pocket to type a prompt.

Project Astra: Google's framework for real-time, multimodal, ubiquitous agents. The agent "sees" what you see and "hears" what you hear constantly through smart glasses (XR).

The context window is streaming live video, requiring ultra-low latency processing to be genuinely conversational and helpful in the physical world.



Proactive Agents (YETI)

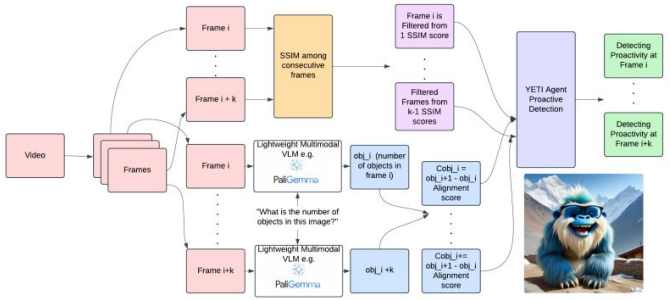
A reactive agent waits for you to say, "Help me fix this." A *proactive* agent intervenes before you make a mistake.

The YETI Framework (2025): How does an AR agent know *when* to interrupt you without being annoying?

It calculates an **Alignment Signal** comparing the user's current trajectory to the optimal trajectory.

The Math: If a user is assembling IKEA furniture and picks up the wrong screw, the Structural Similarity Index (SSIM) between their view and the "correct" view drops.

When the alignment signal falls below a threshold for a duration of *k* seconds, the Agent triggers a proactive AR visual/audio intervention.



The YETI Architecture

How does YETI process real-time video without massive latency?

Step 1: Keyframe Extraction: The AR glasses do not send 60fps video to the cloud. They sample at roughly 1 fps to reduce bandwidth.

Step 2: VLM Goal Alignment: A Vision-Language Model establishes what the "correct" current state should look like based on the user's ultimate goal.

Step 3: Continuous Comparison: The local device constantly calculates the SSIM between the user's current view and the VLM's generated "correct" keyframe.

Tradeoffs in Proactive Agents

What happens if we narrow the window of time (k) and set a high threshold?

We don't want the AI agent to intervene frequently and unnecessarily (e.g. the "Clippy" effect).

Some hyperparameters to tune the proactive agent:

- If we have too small k but a high threshold, we run the risk of creating a micromanager agent.
- If we have a low threshold but a high k, then the opposite problem can happen and you have an AI that just sits by while letting you make (potentially dangerous!) mistakes.

Summary and Points for Senior Projects

Robotics: VLA models (RT-2, $\pi_{0.7}$) combine web-scale reasoning with physical action chunking and steerability.

Transportation: Safety-critical systems require Causal Reasoning (Alpamayo) and Generative World Models (GAIA-1) to handle edge cases.

AR/XR: The interface of the future is egocentric and proactive (YETI, Astra), calculating mathematical thresholds to assist users in real-time.

For your Senior Projects: If your agent operates in a physical or simulated spatial environment, how are you handling the continuous action space? Are you using pure RL, or are you utilizing a VLA to leverage pre-trained knowledge?

Questions?