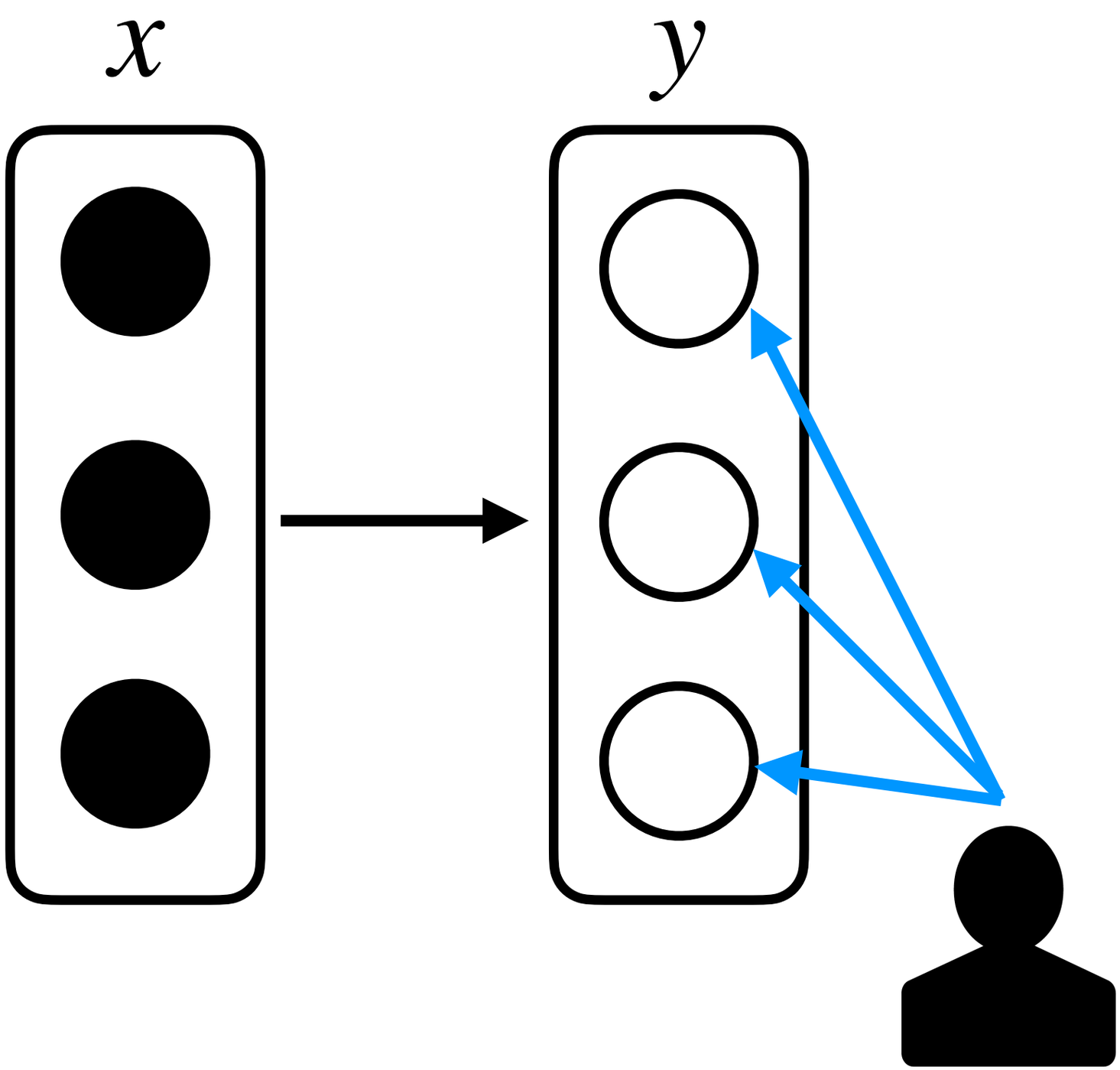
Leveraging Human Input for Self Driving Cars

Gokul Swamy

BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Supervised Learning



*With humans in the loop, learning algorithms should be able to learn **quickly** from data that is both **easy** and **safe** for people to provide.*

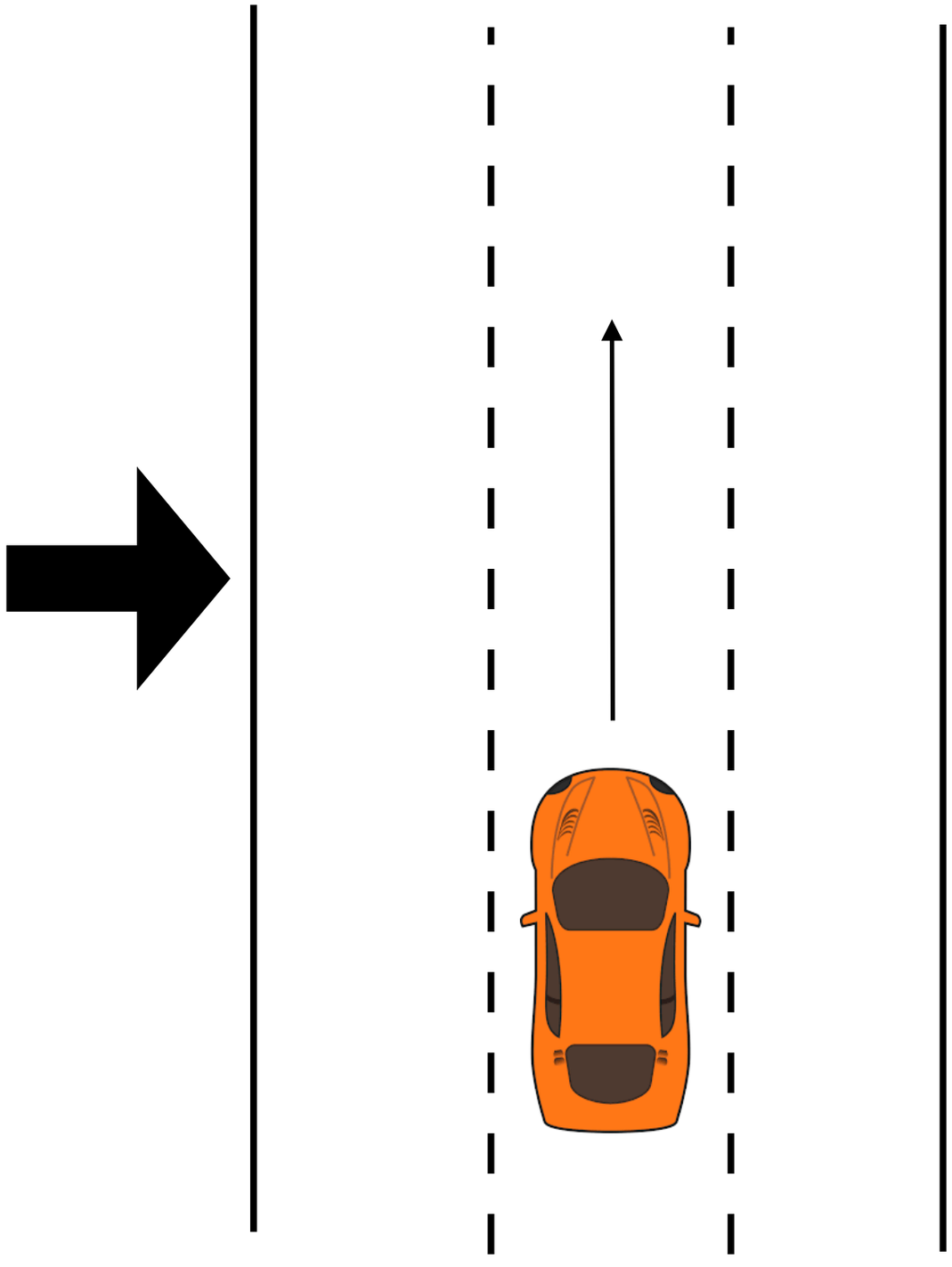
*We can accomplish these goals by taking advantage of the **structure** present in specific problems.*

Weakly-Supervised 3D Intersection Structure Estimation

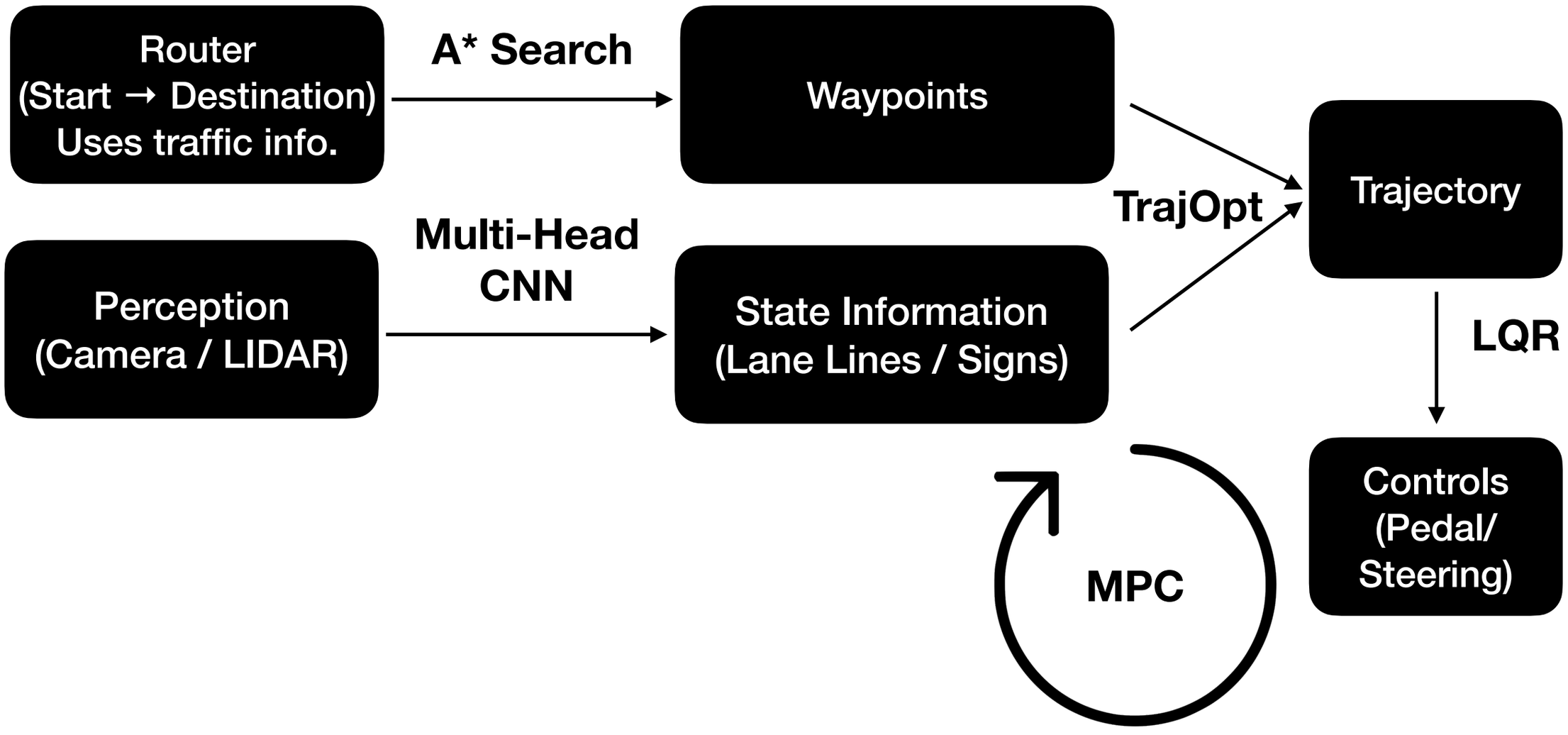
(slides hidden from public version)

Modeling Interaction for Self-Driving Cars

Up Till Today: Driving in Isolation



Up Till Today: Driving in Isolation

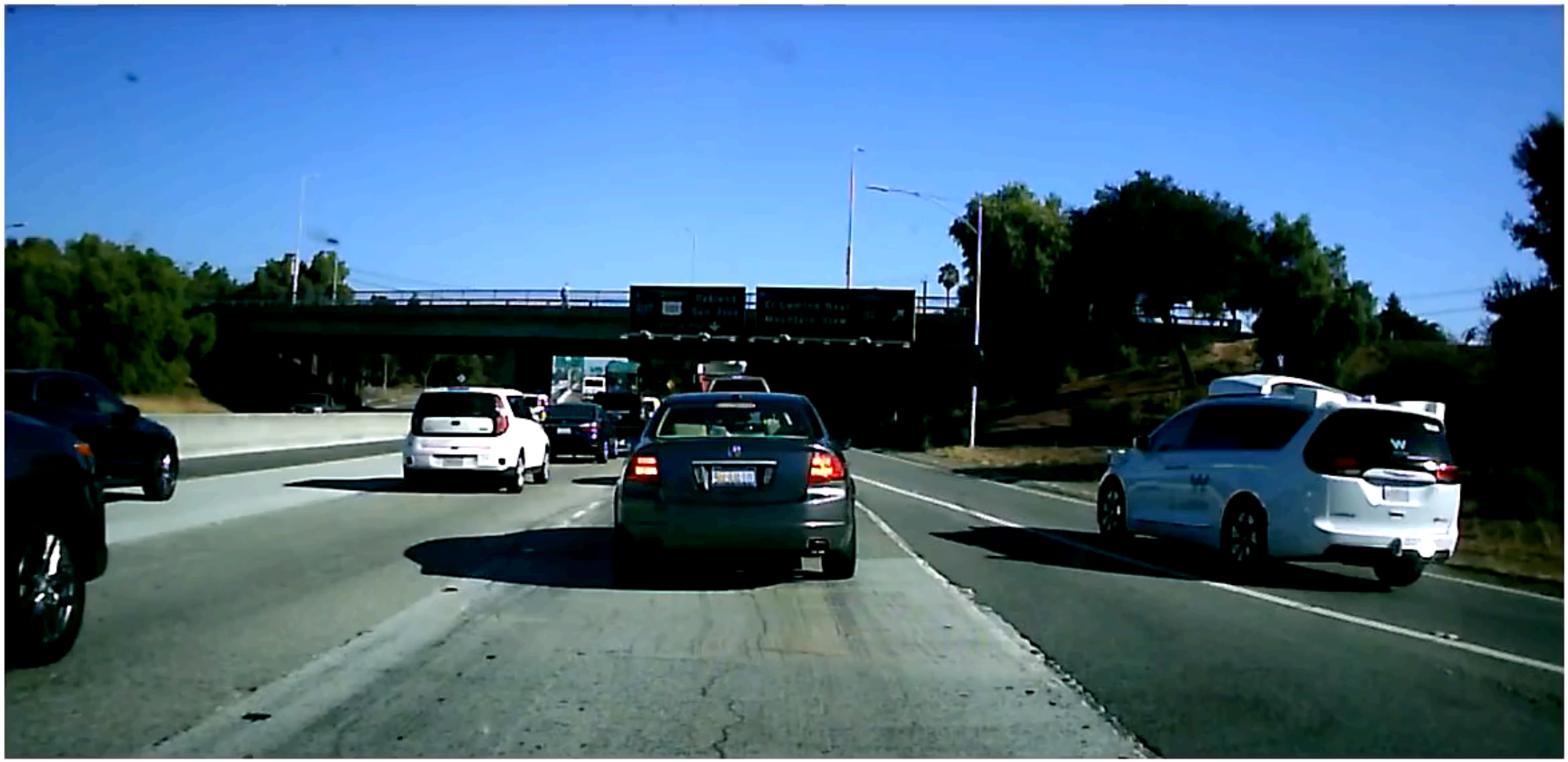


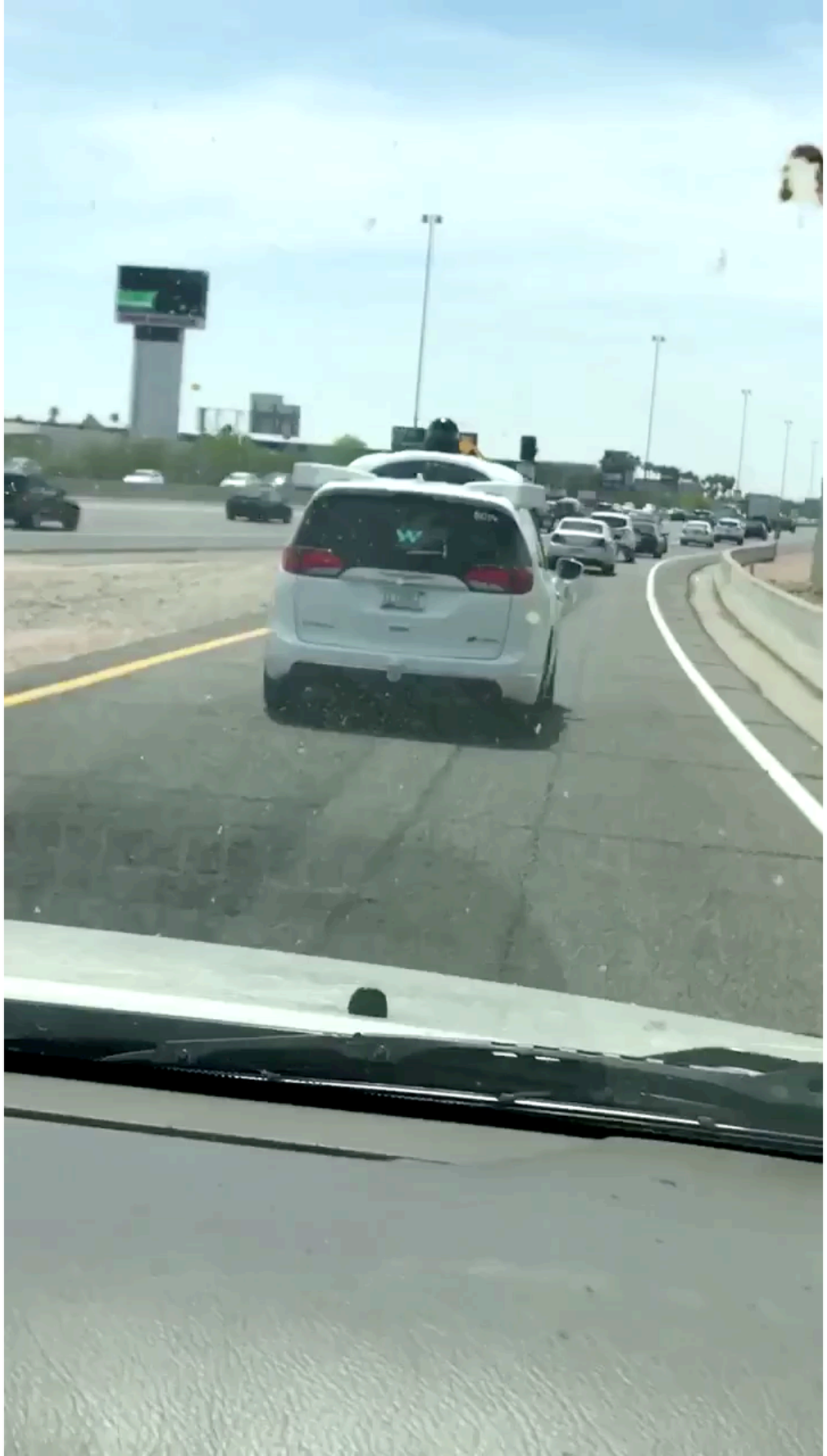
(will talk about some cool perception work if I have time)

The Good

The Bad

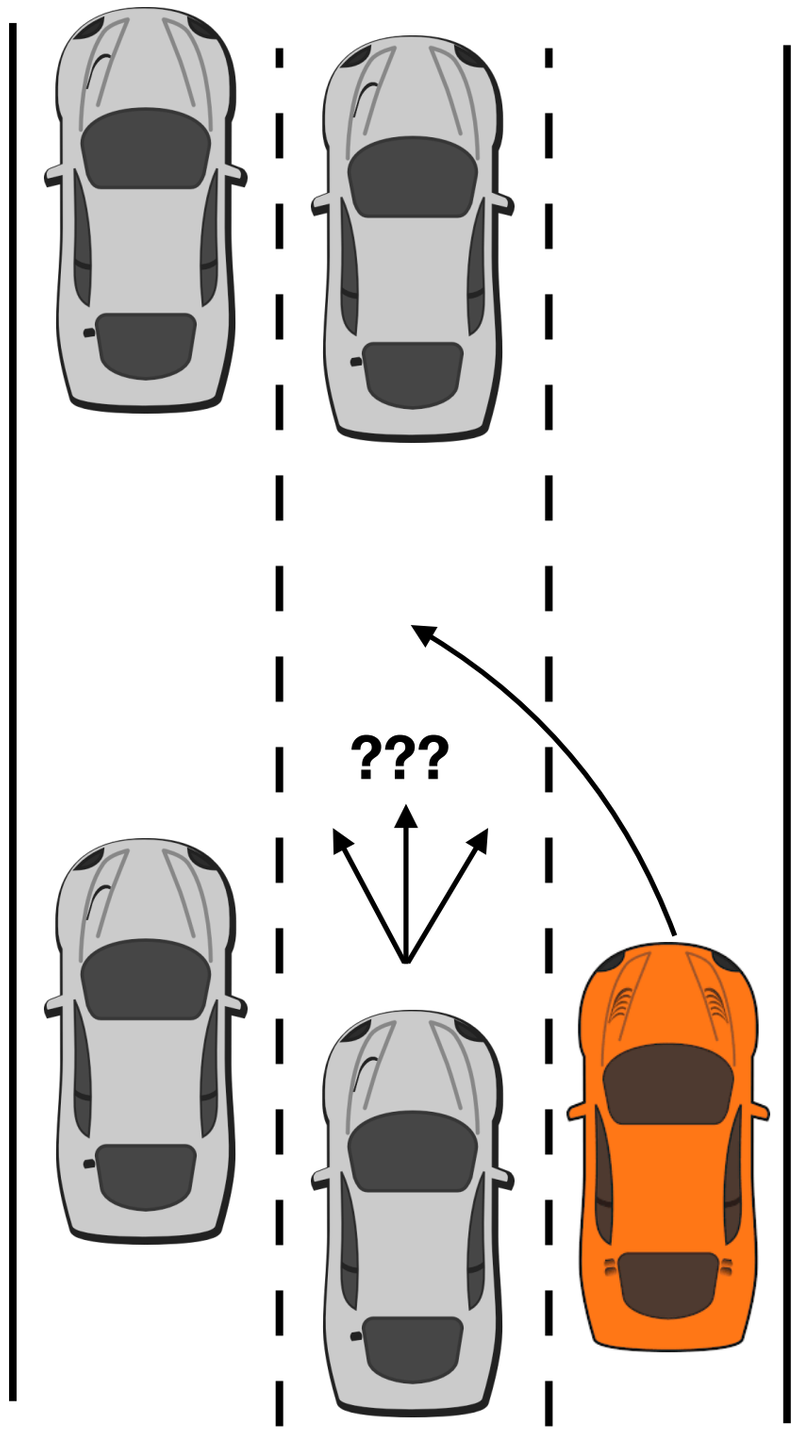
How Does a Waymo Merge into Traffic?





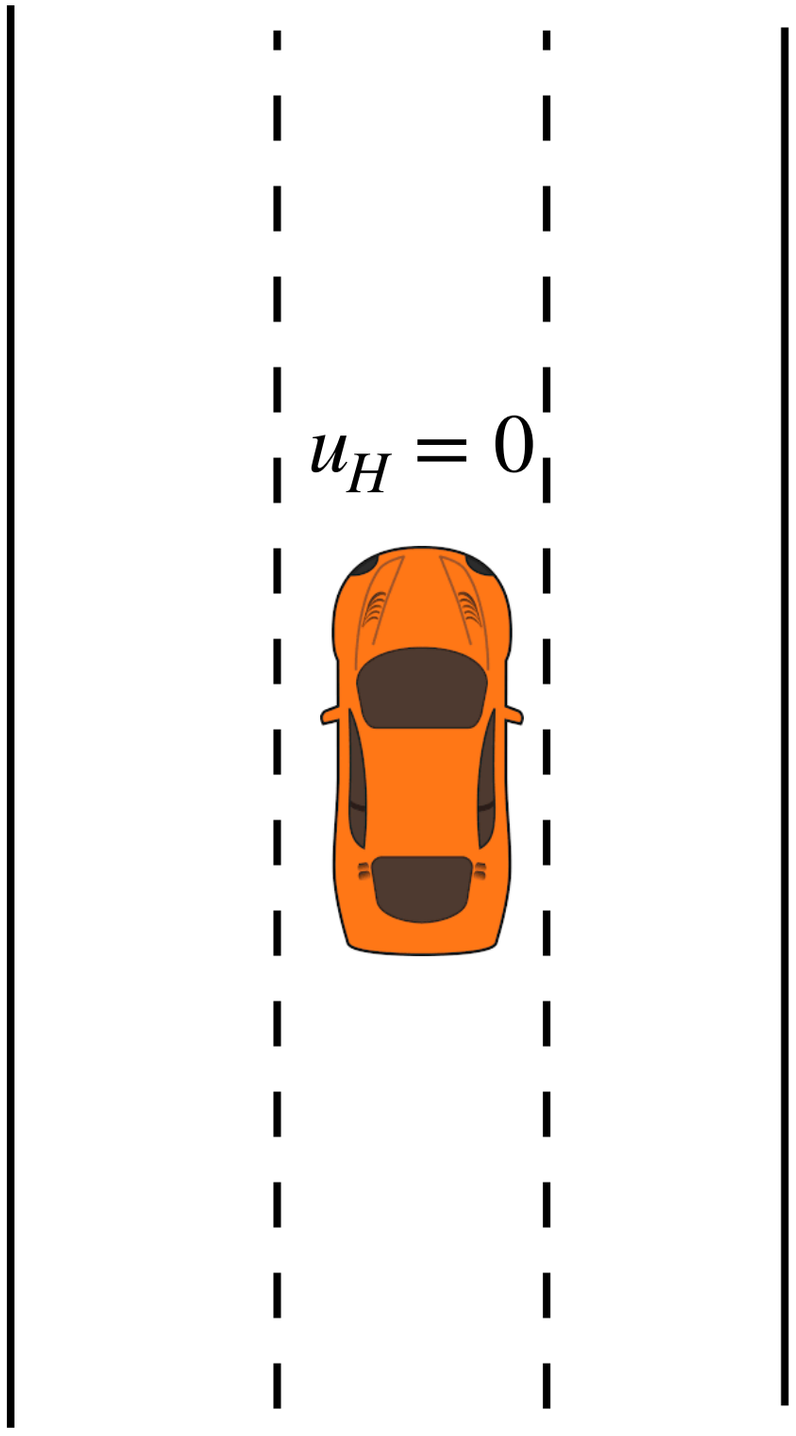
The Ugly

What was the problem here?



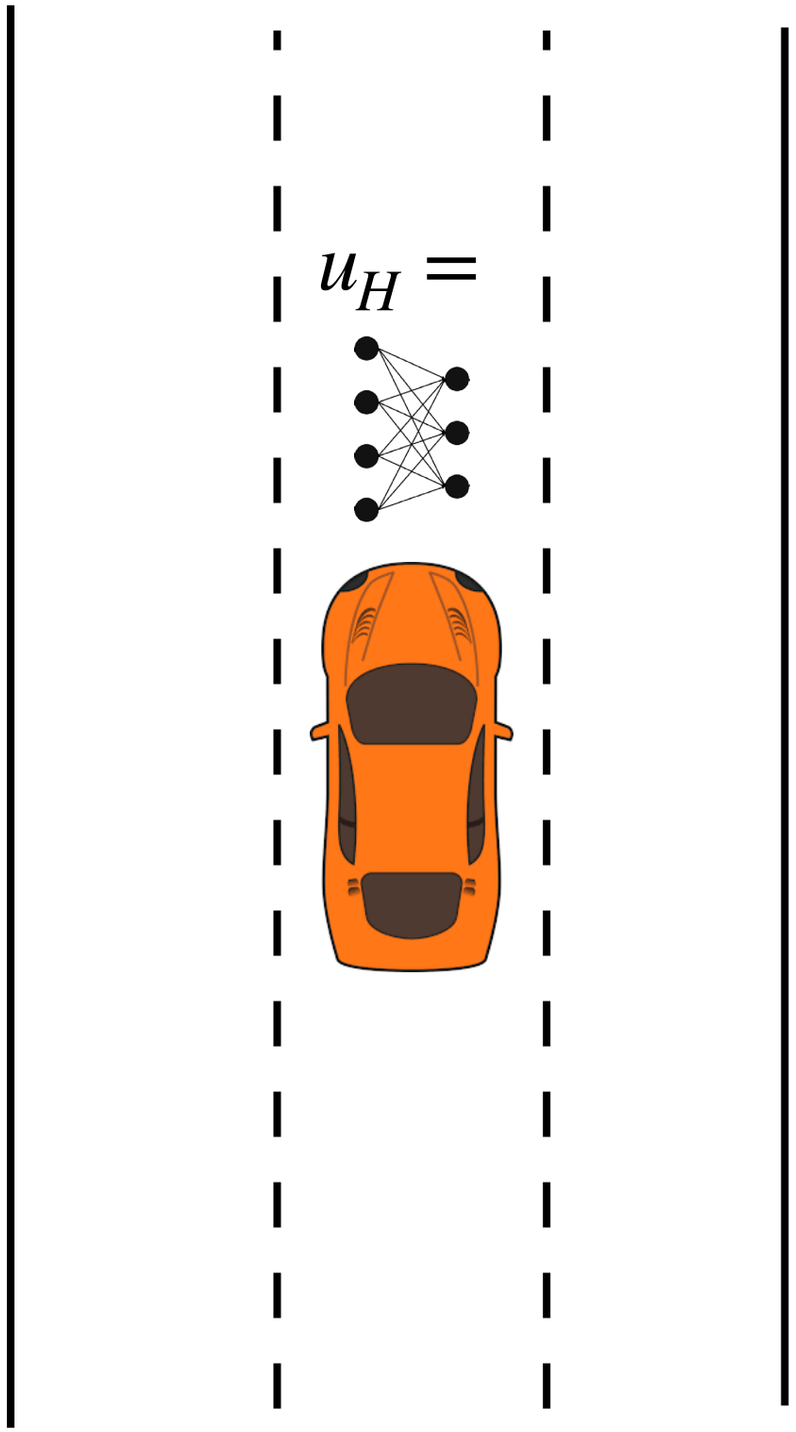
Behavior Prediction

- Approach V0: Constant Velocity
  - Will work decently well on highways
  - Or for things like drones



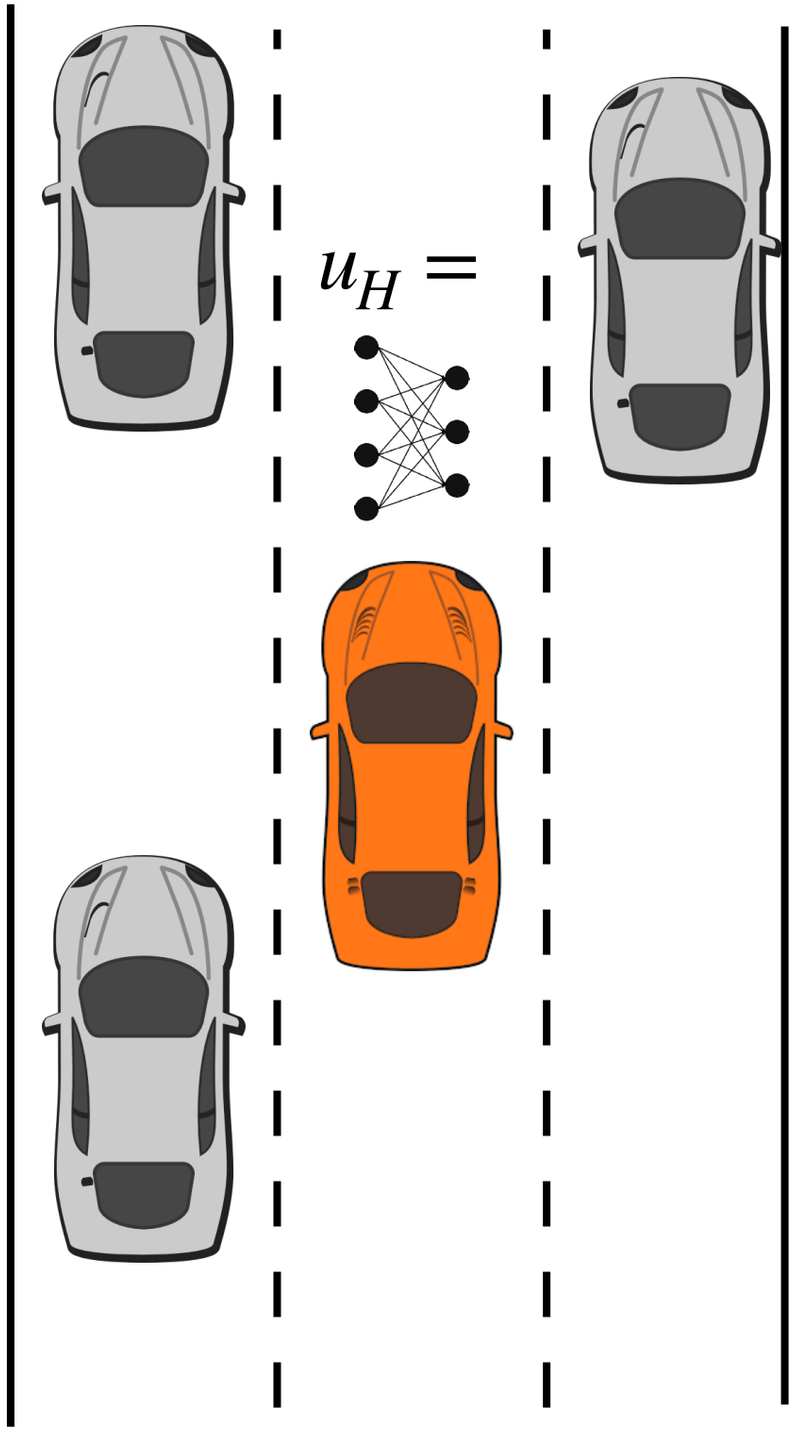
Behavior Prediction

- Approach V1: Isolated Prediction
  - Fit network to map from state to action of car.
  - **Q**: Why will this not work well?
  - **A**: Need to incorporate information about other cars.



Behavior Prediction

- Approach V2: Full Prediction
  - Fit network to map from state of all cars to action of a particular car
- **Q**: Why might this not work well?
- **A**: Your actions influence the actions of other cars



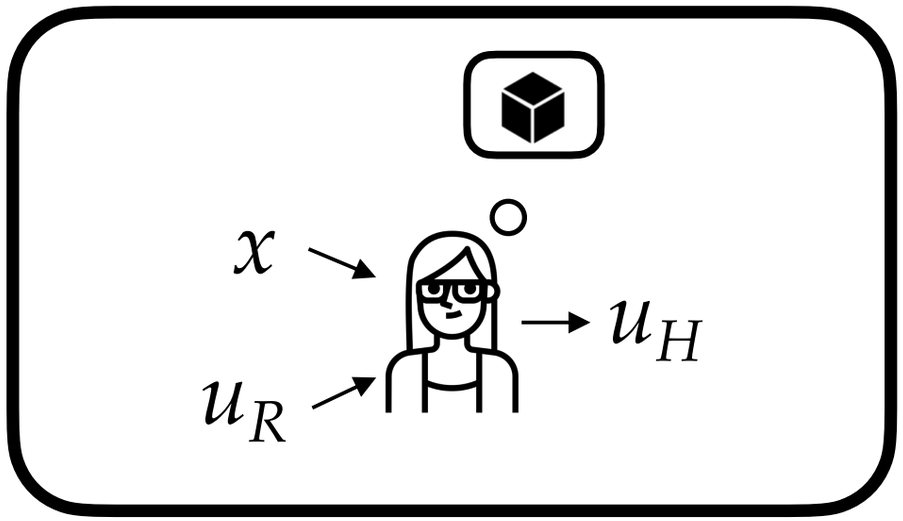
Behavior Prediction

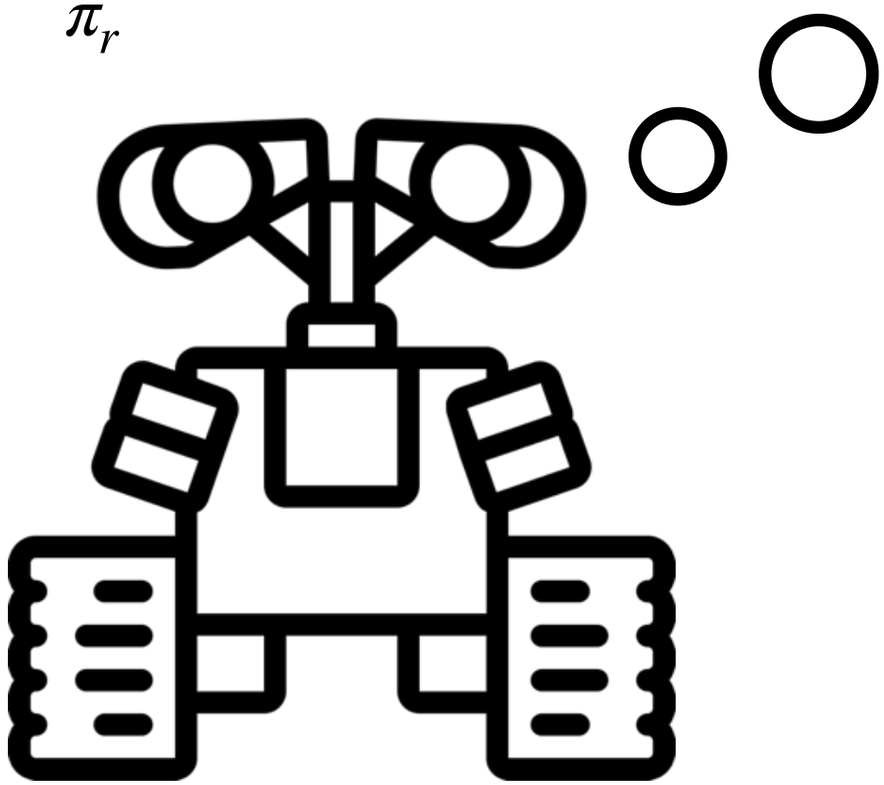
- **Problem**: other drivers ***react*** to the positions and actions of your car
  - Predictions → Your Actions → Other Actions → Pred.
  - So if you update your policy, you change input distribution to network
- **Solution**: learn a function that models the person's response to a robot's action
  - Question: what kind of function to learn?

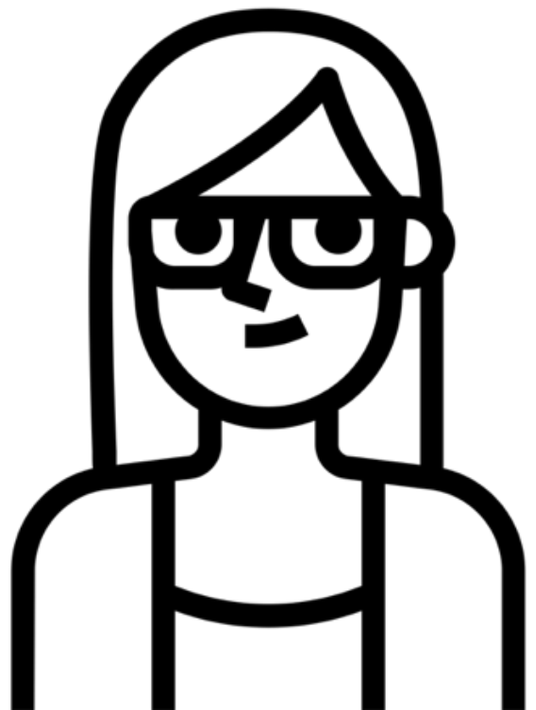
Joint work w/

Black-Box Models

$$\max_{\pi_r} E[U_r(\pi_r, \pi_h)]$$

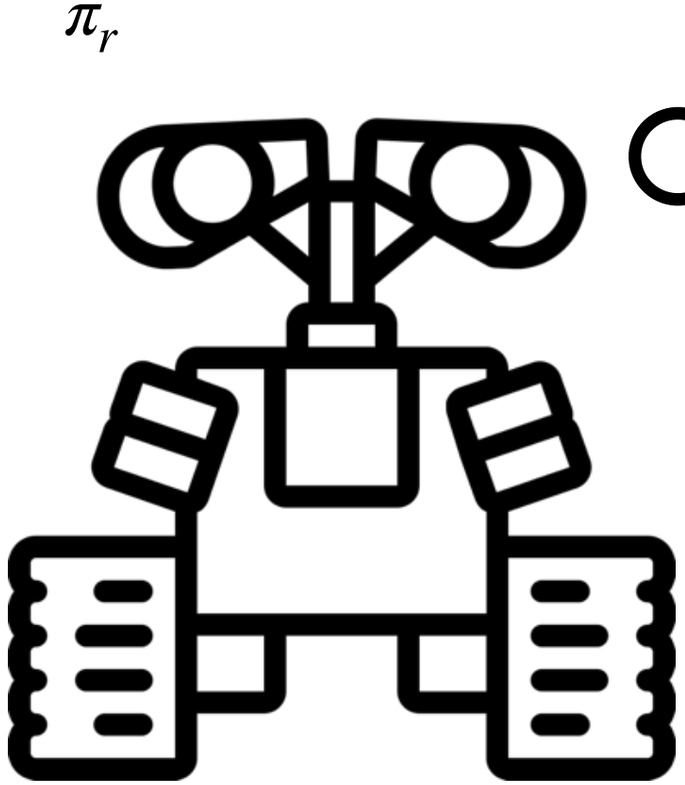


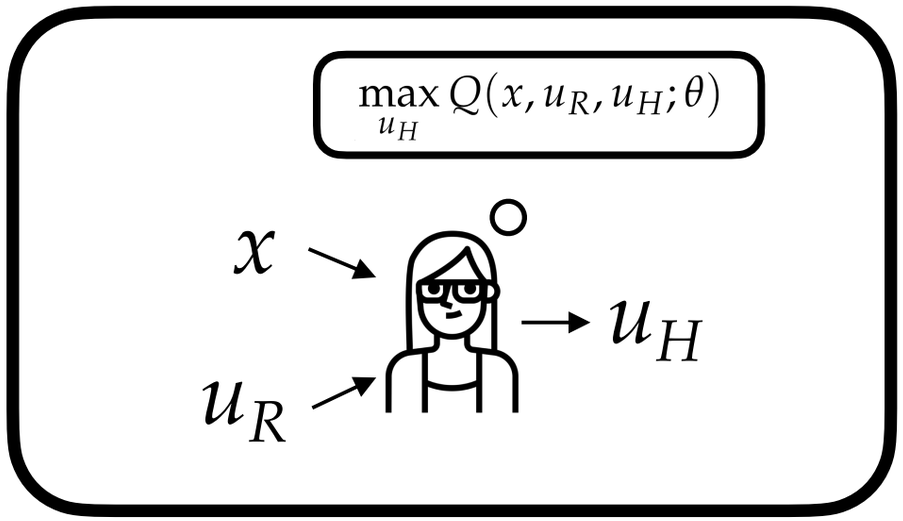


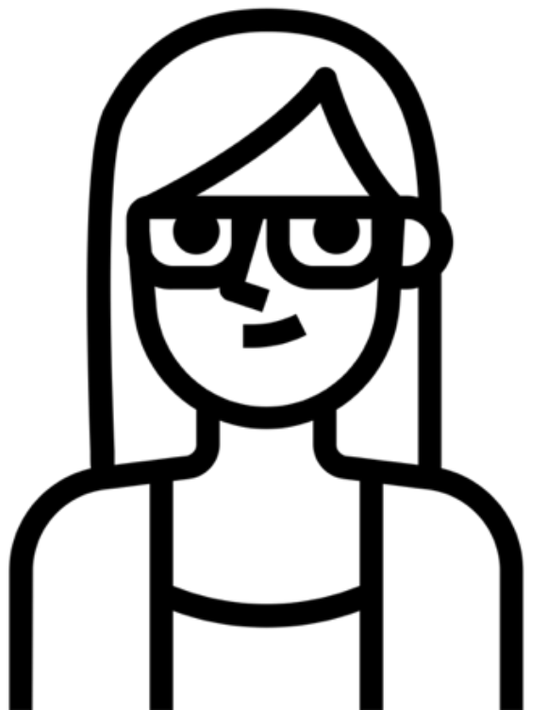


Theory of Mind

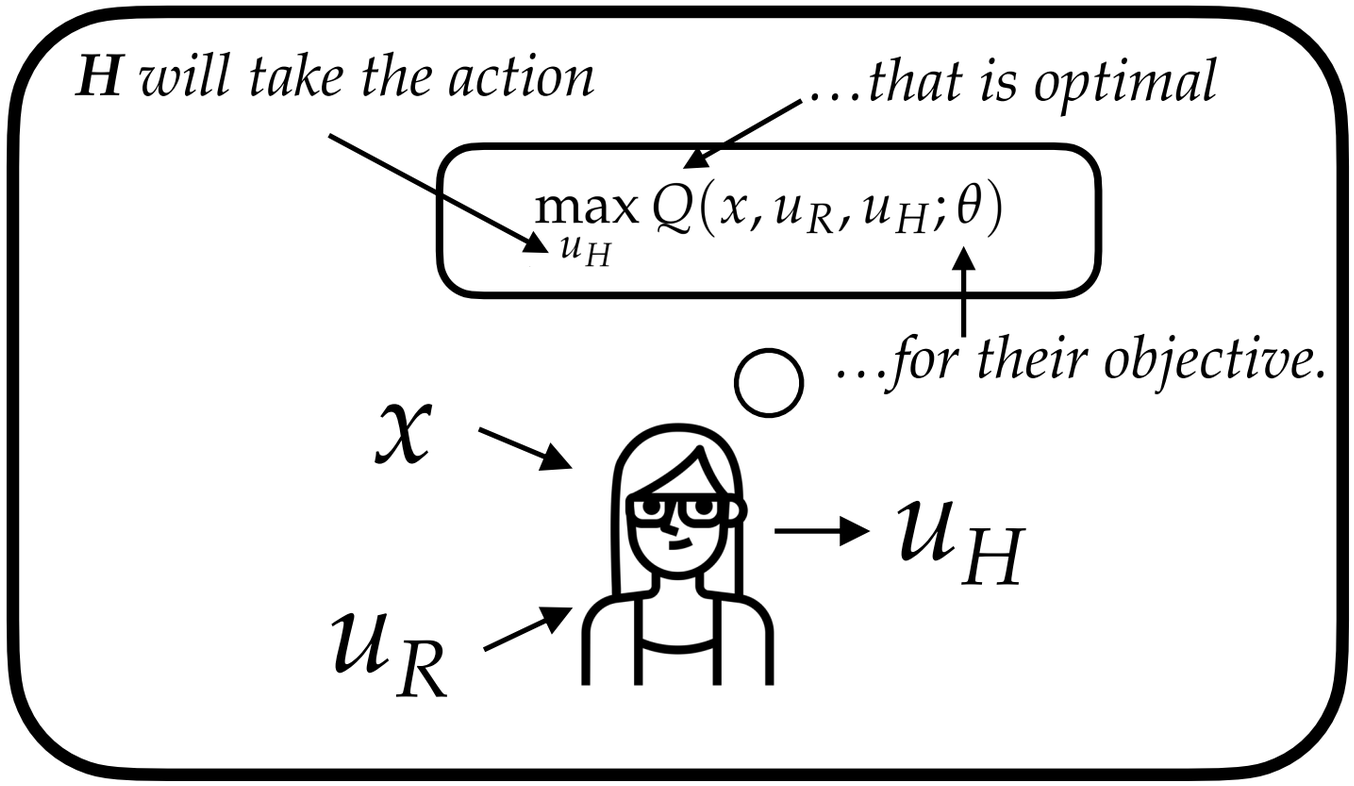
$$\max_{\pi_r} E[U_r(\pi_r, \pi_h)]$$

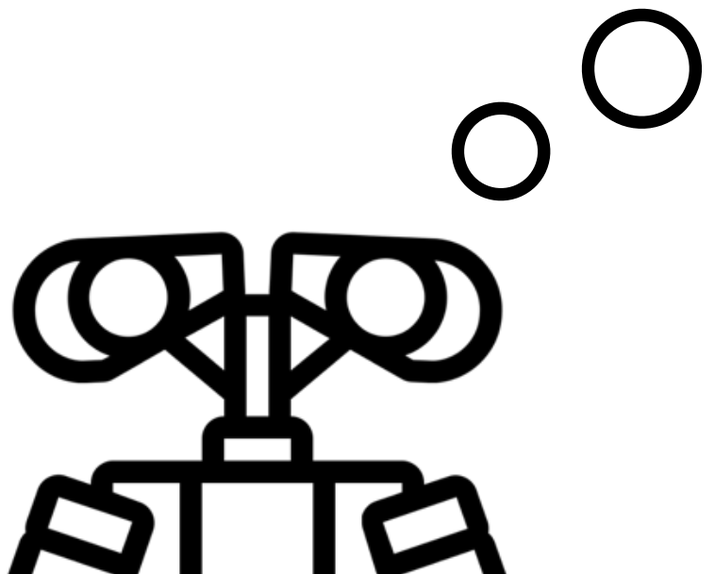






Theory of Mind





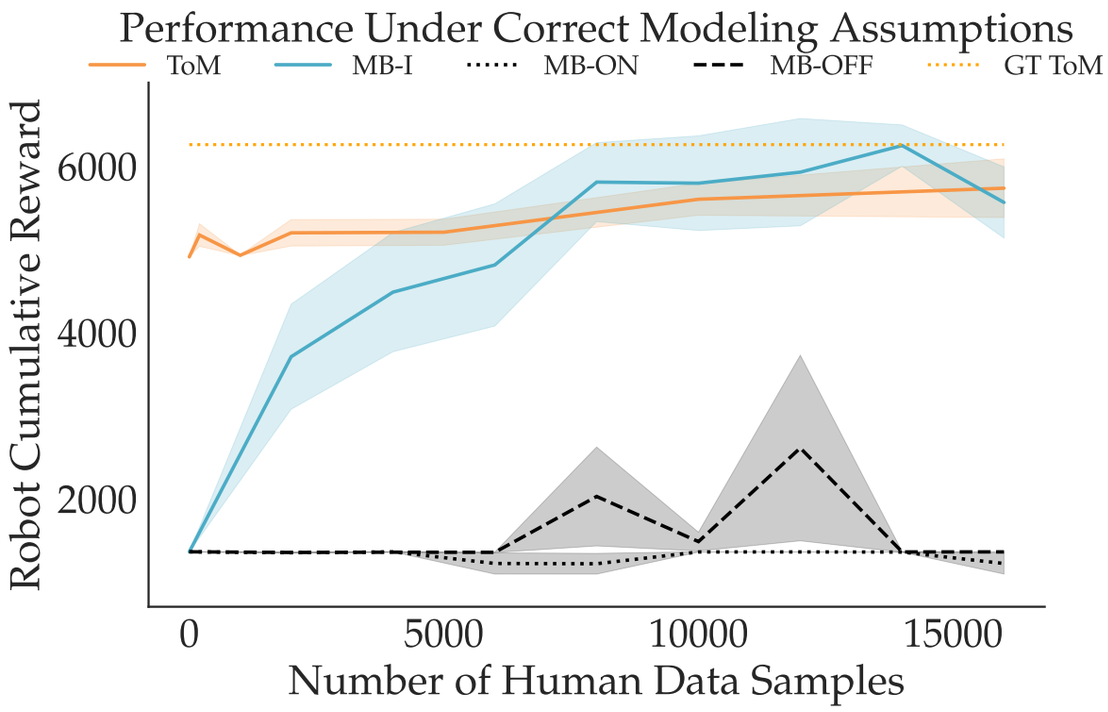
*Thus, **R** should take the action that will cause **H** to take the action that jointly maximizes* $U_r(u_r, u_h)$

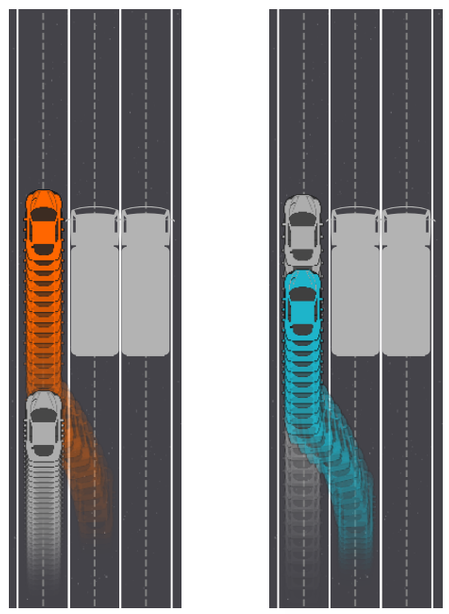
Inferring Utility Functions

- **Approach**: Inverse Optimal Control / Reinforcement Learning
    - RL: Given reward function, find best actions
    - IRL: Given actions, find reward function that would have produced these actions
- High level blueprint: Fix a set of features a person driving could care about. Figure out what weights on these features would produce observed behavior on average.

Theory of Mind Results

- Can learn model from recorded human-human driving data
  - Much safer than throwing faulty car on road





ToM

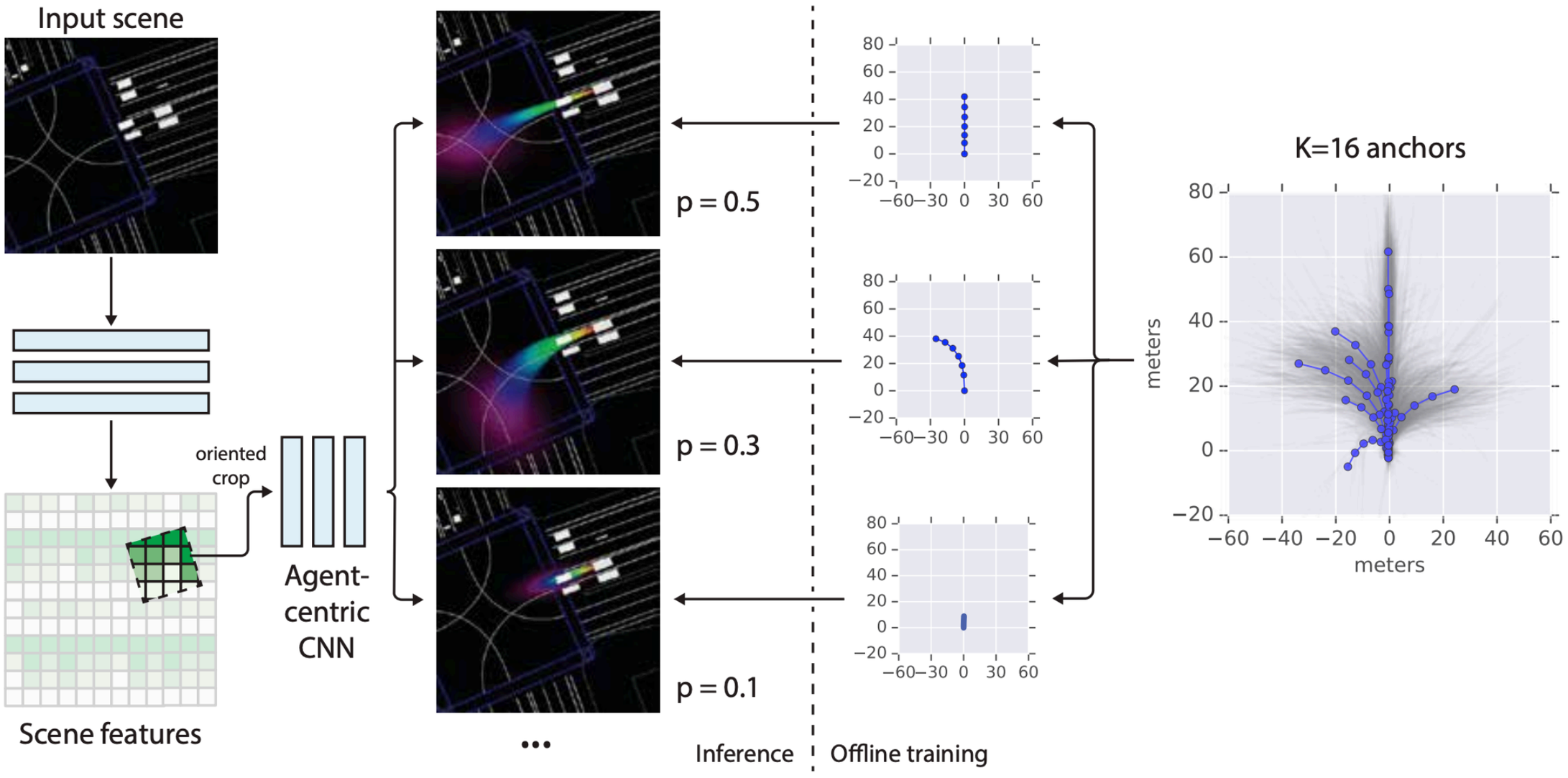
MB-I

Scaling to the Real World

- Unfortunately, running a nested optimization for each other car on the road is computationally infeasible to do online
  - Have only accounted for subset of pairwise interactions

- **Best of both worlds**: At training time, use ToM as data source to train network. At deployment time, use network for fast inference via a single forward pass.

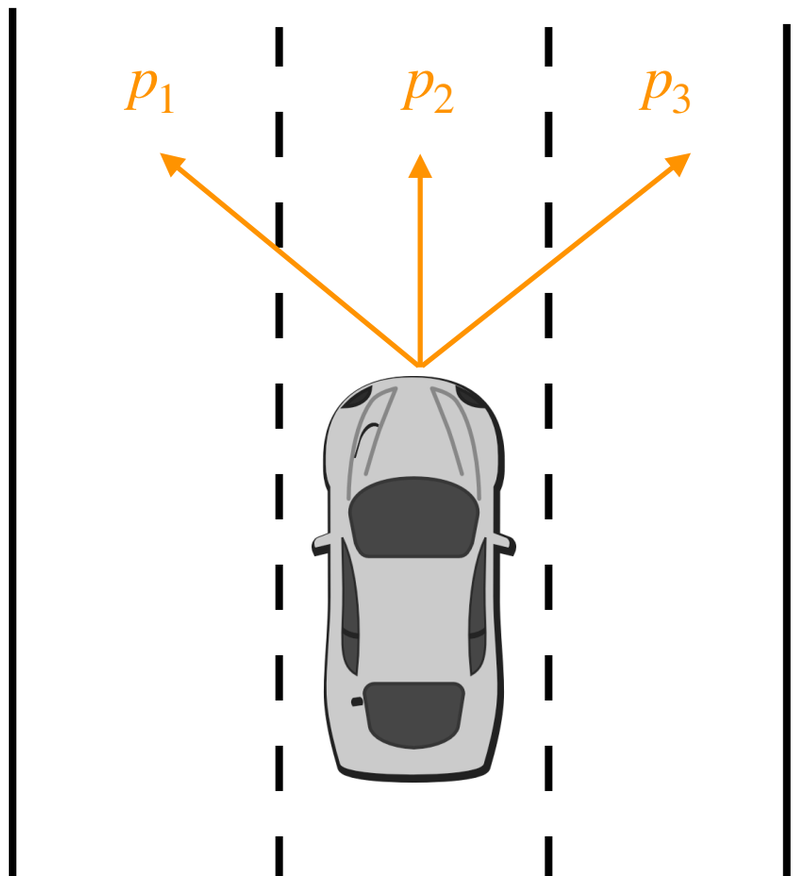
Predictive Network Architecture: Waymo's MultiPath



Leveraging Problem Structure

- **Given**: we have history of state/action pairs of other agents from perception + other state information (lights)

- Roughly, where can a car go in the following situation?

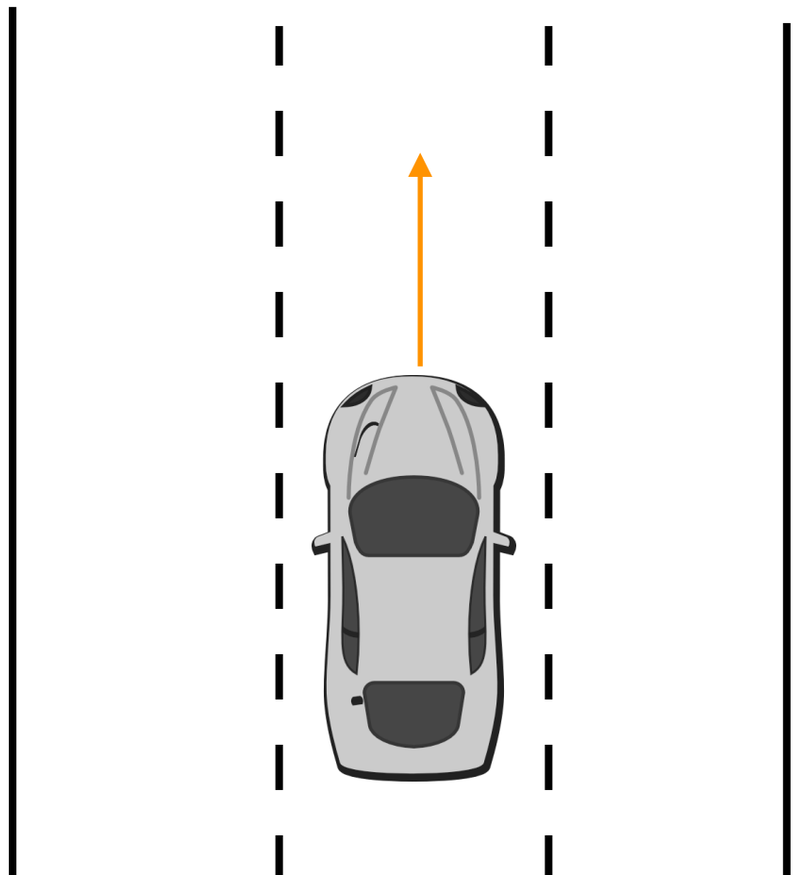


Trajectory Anchors

$p_1 + p_2 + p_3 = 1$

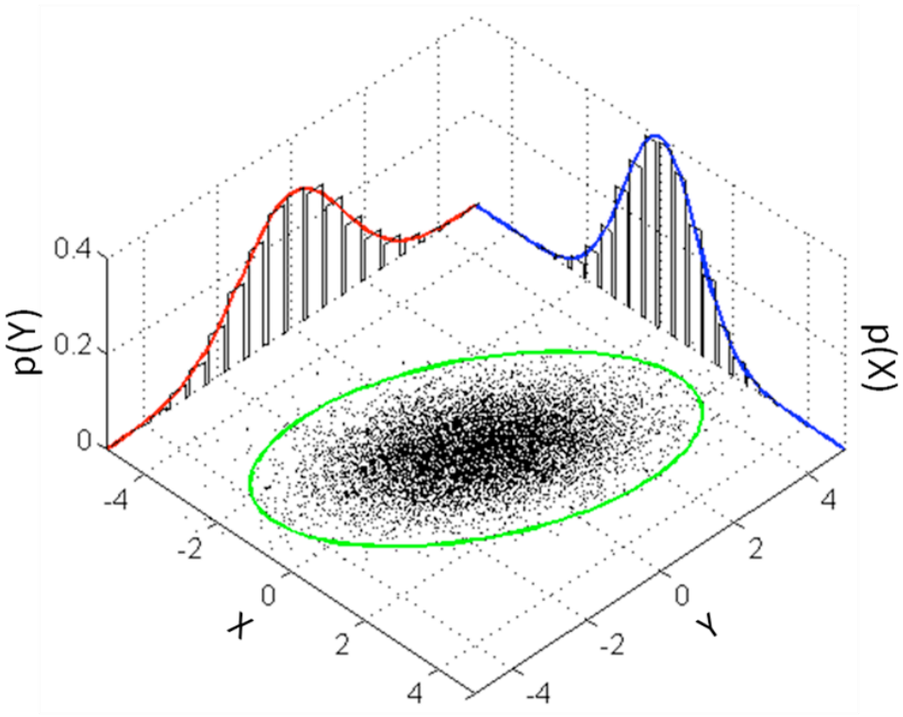
Predicting Offsets

- **Q**: Why is this a good idea?
  - **A**: Lower Variance
- **Q**: Do we just want to predict an offset?
  - **A**: No, also want our network to be able to express *uncertainty*



Detour: Multivariate Gaussian

- Univariate Gaussian: fully determined by mean and variance
  - Maximum entropy distribution with these constraints - makes the fewest further assumptions
- Multivariate Gaussian: mean becomes vector, variance becomes matrix
  - 2D for our use case.



Model Output: Mixture of Gaussians

- Single Trajectory Anchor Distribution (predict μ & $\sum$):

$$\phi(s_t^k|\mathbf{a}^k, \mathbf{x}) = \mathcal{N}(s_t^k|a_t^k + \mu_t^k(\mathbf{x}), \Sigma_t^k(\mathbf{x}))$$

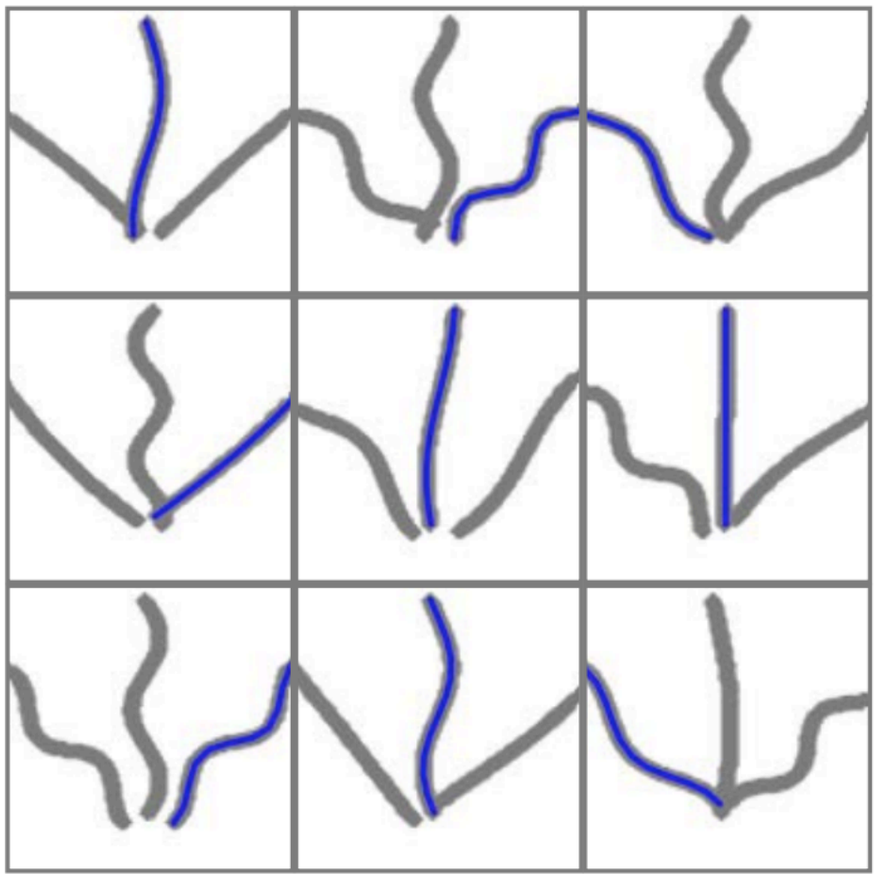
- Full distribution over possible future states (predict π):

$$p(\mathbf{s}|\mathbf{x}) = \sum_{k=1}^{K} \pi(\mathbf{a}^k|\mathbf{x}) \prod_{t=1}^{T} \phi(s_t|\mathbf{a}^k, \mathbf{x})$$

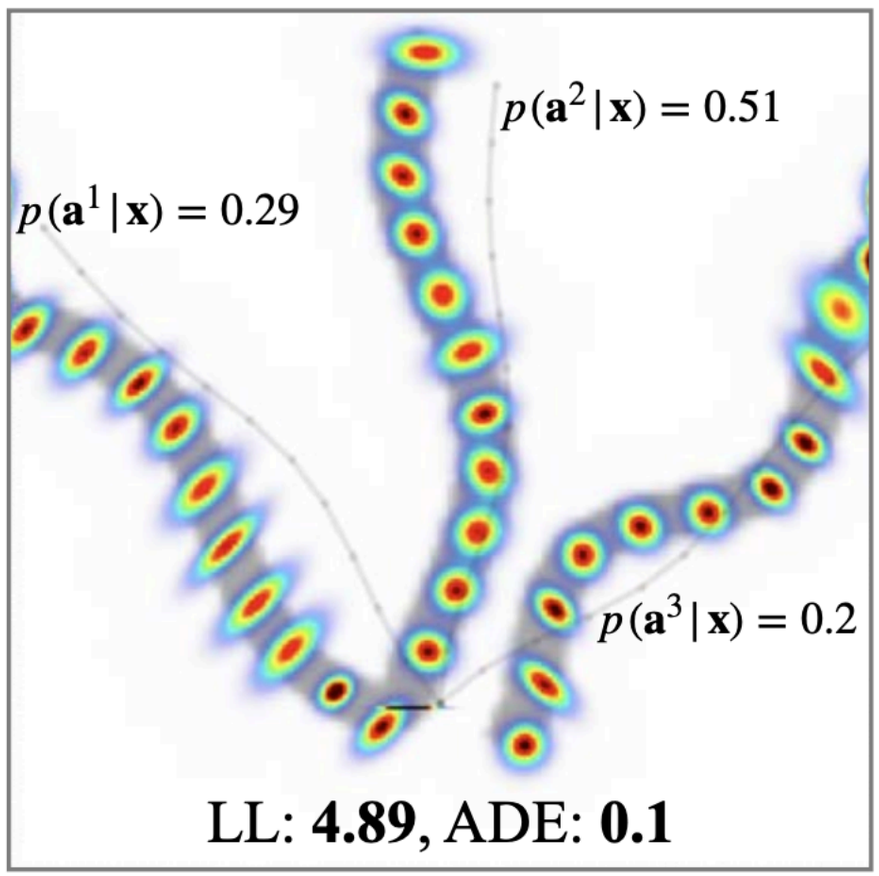
- Loss function:

$$\ell(\theta) = -\sum_{m=1}^{M} \sum_{k=1}^{K} \mathbb{1}(k = \hat{k}^m) \Big[\log \pi(\mathbf{a}^k|\mathbf{x}^m; \theta) + \sum_{t=1}^{T} \log \mathcal{N}(s_t^k|a_t^k + \mu_t^k, \Sigma_t^k; \mathbf{x}^m; \theta)\Big]$$

Model Output Visualized

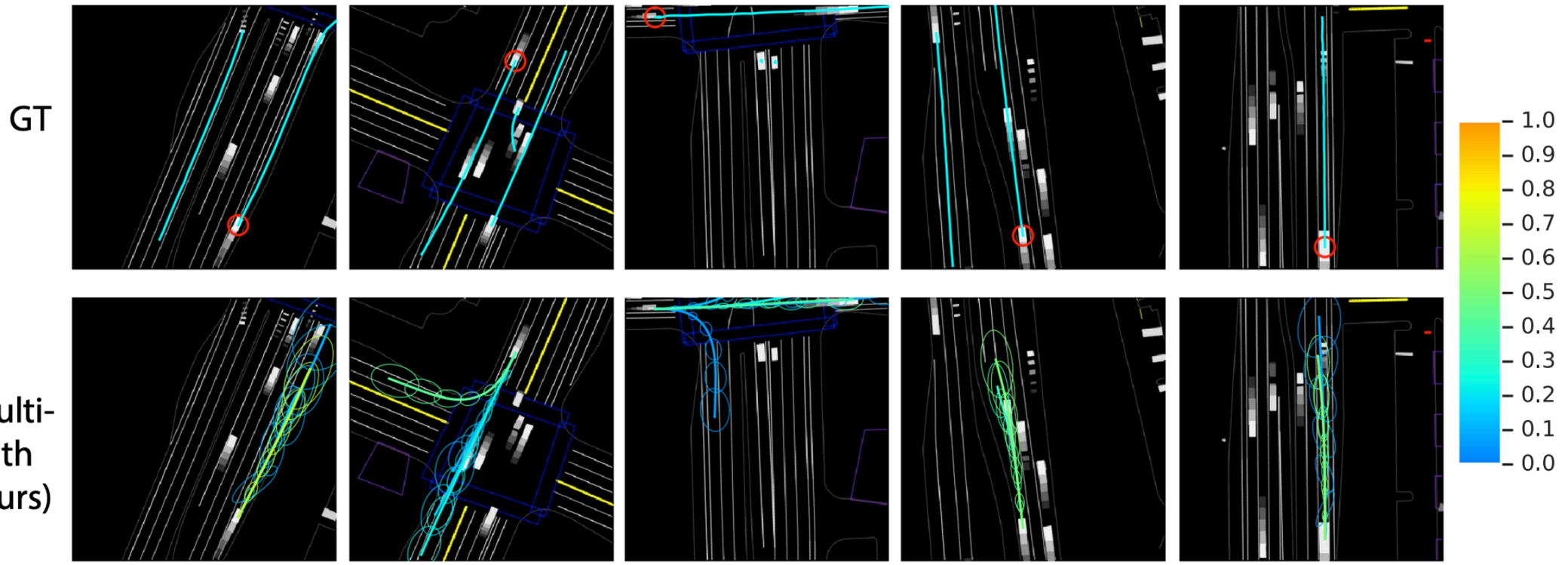


(a) Dataset samples

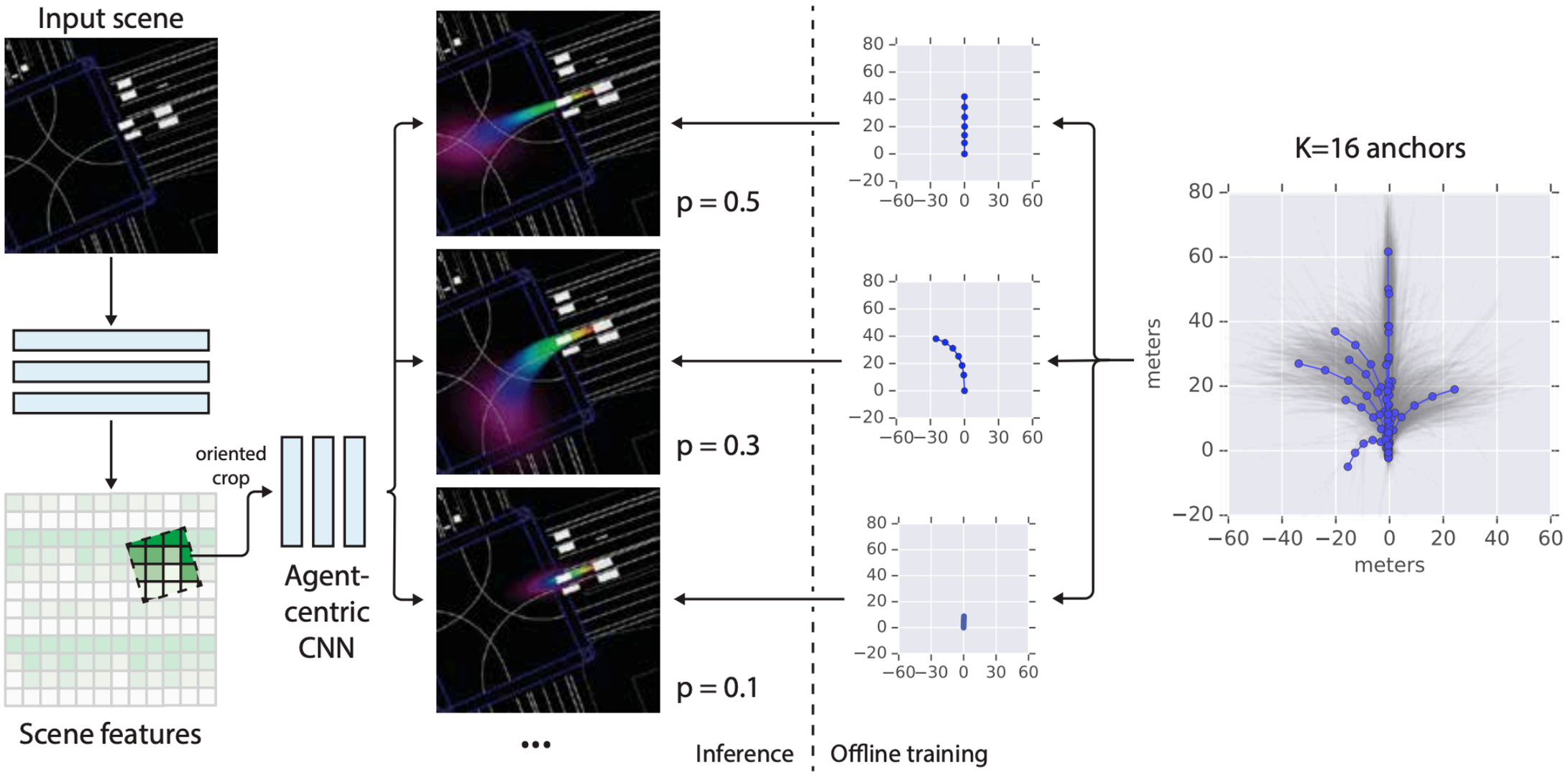


(b) MultiPath K=3, μ, Σ

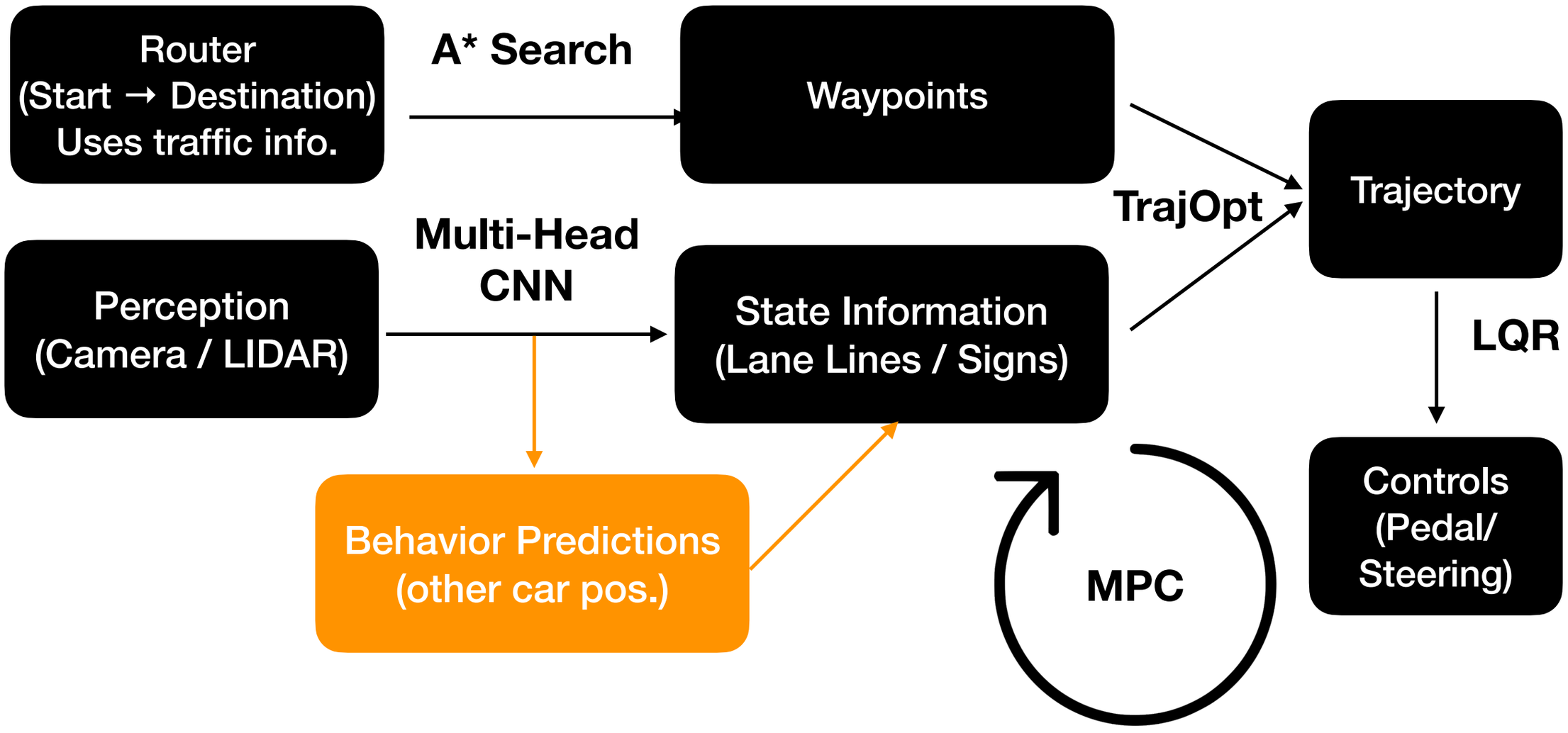
Model Output Snazzily Visualized



Predictive Network Architecture: Waymo's MultiPath



Expanded Self-Driving Stack



Scaled Autonomy

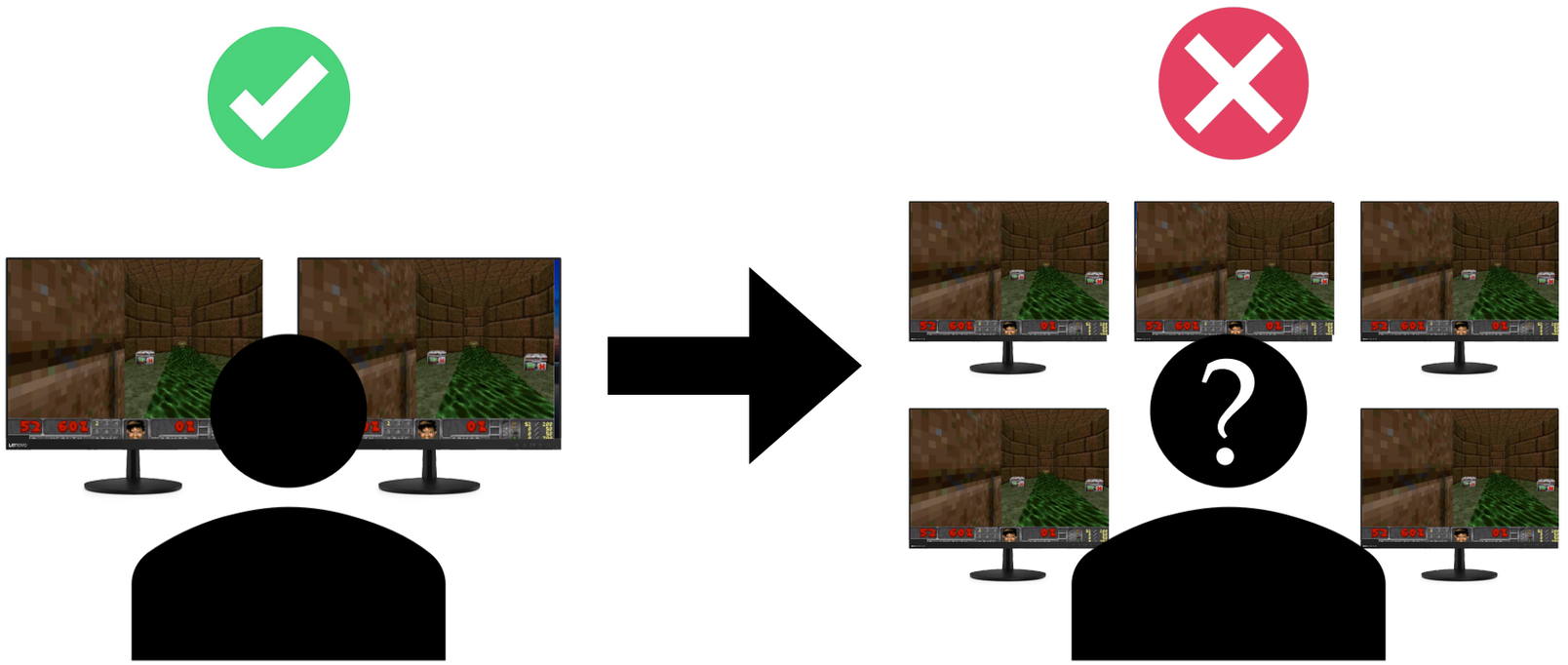
Problem Setup

- Consider a fleet of autonomous robots, each independently executing an identical suboptimal policy

- At any time, the operator can step in and teleoperate a single robot

- Which robot should the operator choose to teleoperate?

Core Problem: Teleoperator Performance Degrades as Fleet Sizes Grow



When presented with a small fleet of robots, skilled teleoperators can supervise them effectively and step in where their assistance is needed most.

However, teleoperators do not have limitless attention which makes it difficult for them to supervise large fleets.

Key Idea

Learn from user interventions *given a few robots* ***to anticipate*** *which of many robots need help the most.*

1) *This is easier for people to provide good data for because they can simultaneously focus on all robots.*

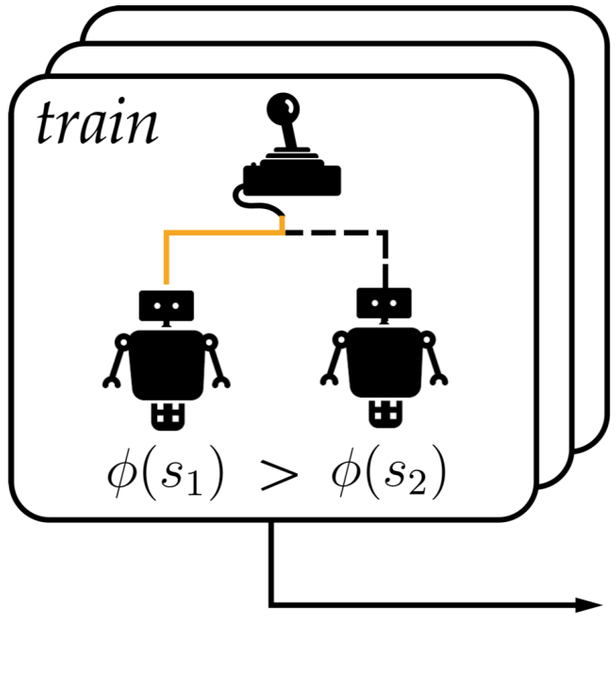
2) *Imitating these switches puts the person in situations where they can best provide demonstrations to improve the robot policy.*

Method

Go back to Step 1

(Optional) Step 0: *Train baseline policy or learn from demos.*

Step 1*: Let users freely choose which of a few robots to teleop.*



Step 2*: To learn a function that generalizes to an arbitrary number of robots, we model the user as making rational choices to maximize utility across all robots:*

$$\phi = \arg\max_{\phi \in \Phi} \mathbb{E}\left[\sum_{i=1}^{n}\sum_{t=0}^{T-1} R(s_i^t, a_i^t) \mid \pi_H, \pi_R\right] \qquad \mathbb{P}[i_H^t = i] = e^{\phi(s_i^t)} / \sum_{j=1}^{n} e^{\phi(s_j^t)}$$

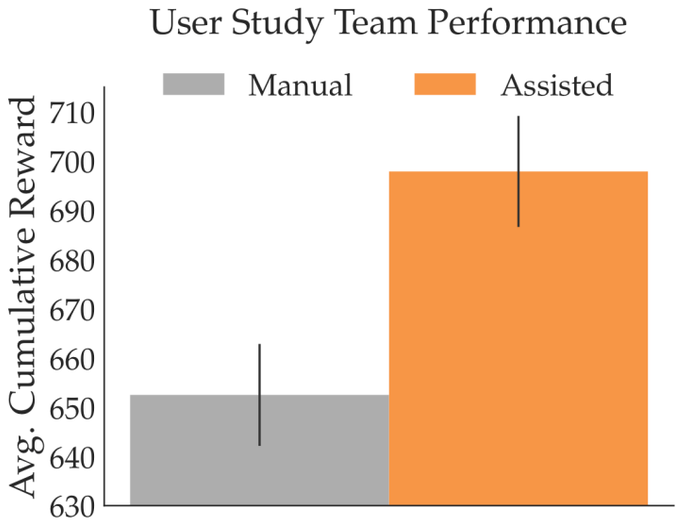
Then, we train a network to mimic these choices by maximizing the log likelihood:

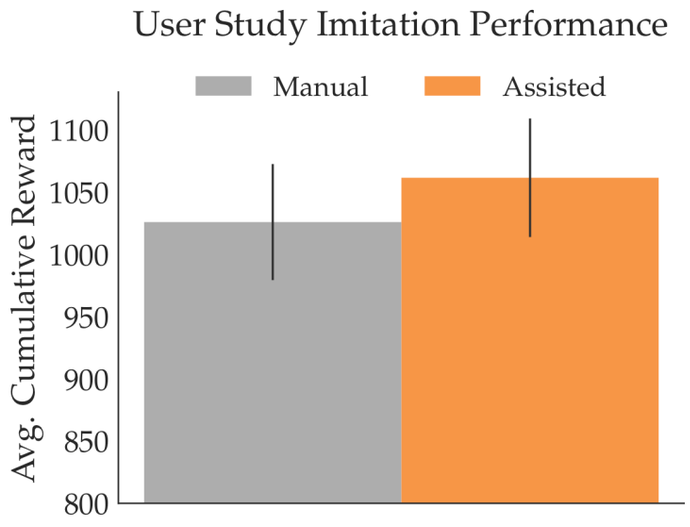
$$\ell(\boldsymbol{\theta}; \mathscr{D}) = \sum_{(s_1^t, \ldots, s_n^t, i_H^t) \in \mathscr{D}} -\phi_{\boldsymbol{\theta}}\left(s_{i_H^t}\right) + \log\left(\sum_{j=1}^{n} e^{\phi_{\boldsymbol{\theta}}\left(s_j^t\right)}\right)$$

Step 3*: Take the argmax over the learned function to automatically choose a robot for the user.*

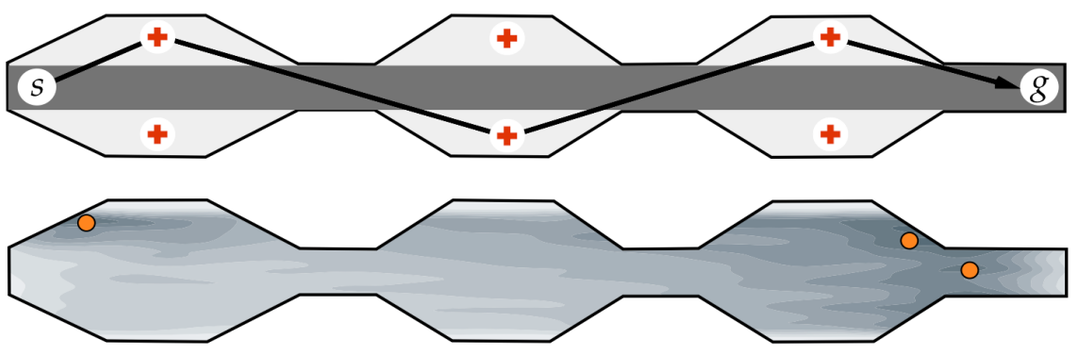
Step 4*: Use collected data to improve robot policy.*

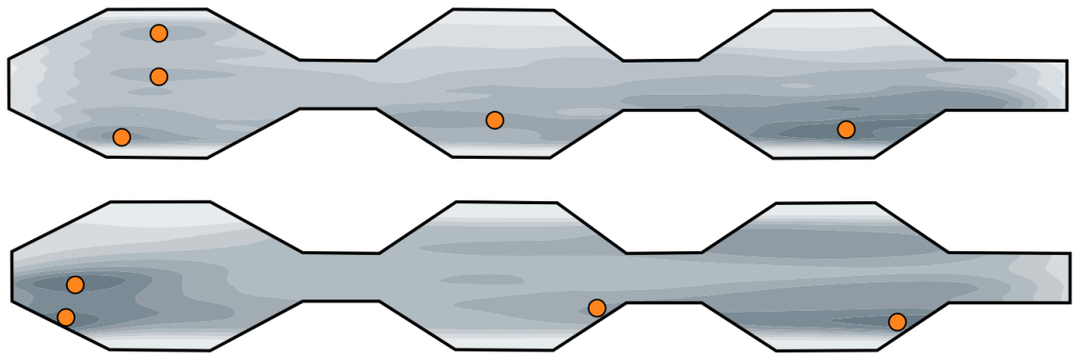
Results: User Study



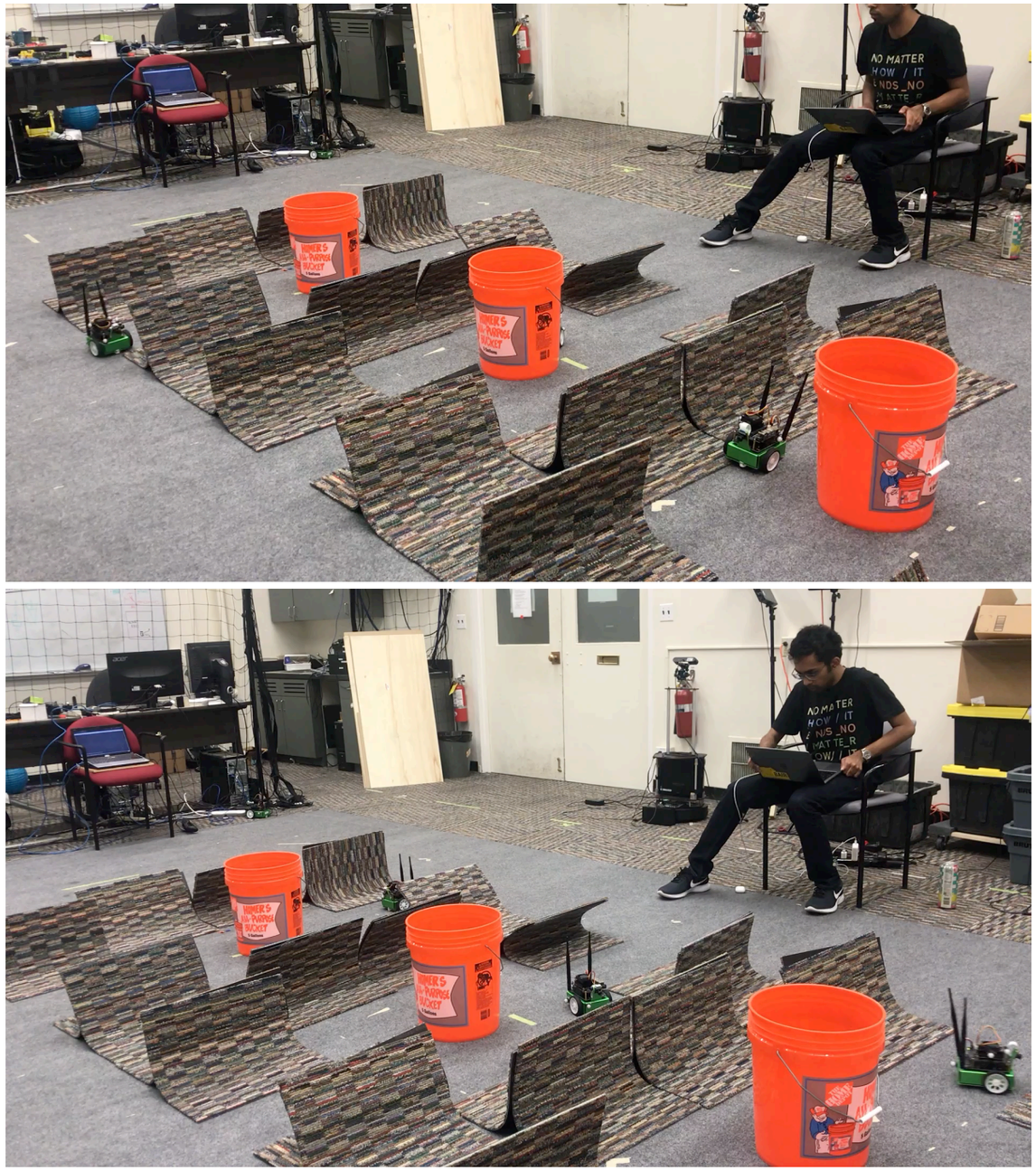


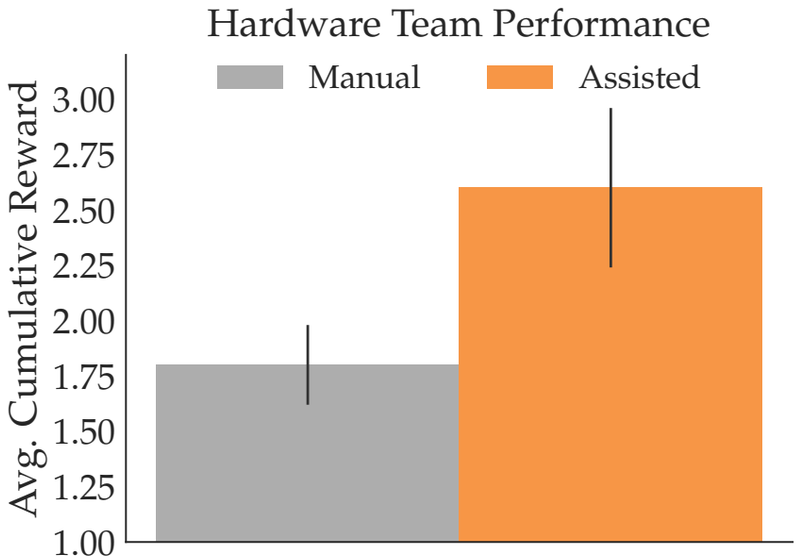
	Unassisted	**Assisted**	F(1,11)	*p*-value
Q1: On average, it was easy to guide the robots to their goals.	2.92	4.50	17.49	$<$ **0.01**
Q2: I was successful at guiding the robots.	2.25	3.92	13.75	$<$ **0.01**
Q1, after objective measures revealed	2.67	4.17	17.47	$<$ **0.01**
Q2, after objective measures revealed	2.92	3.25	0.88	0.37

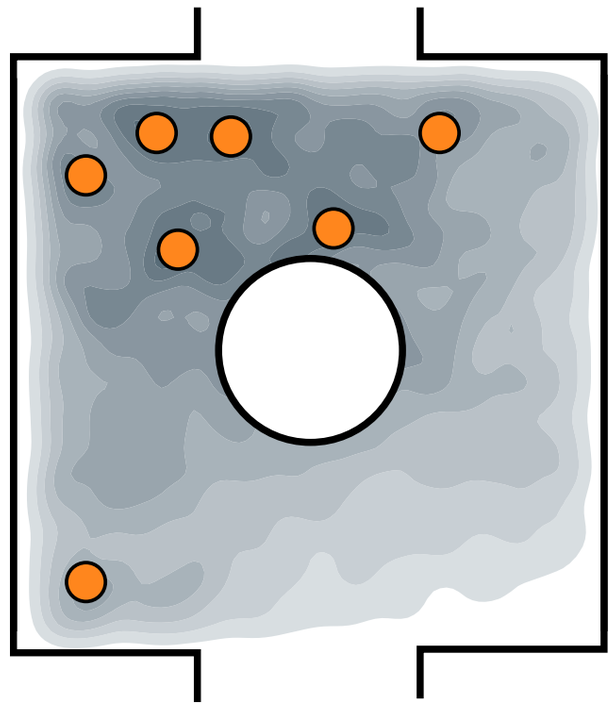




Results: Real Robots







Questions?