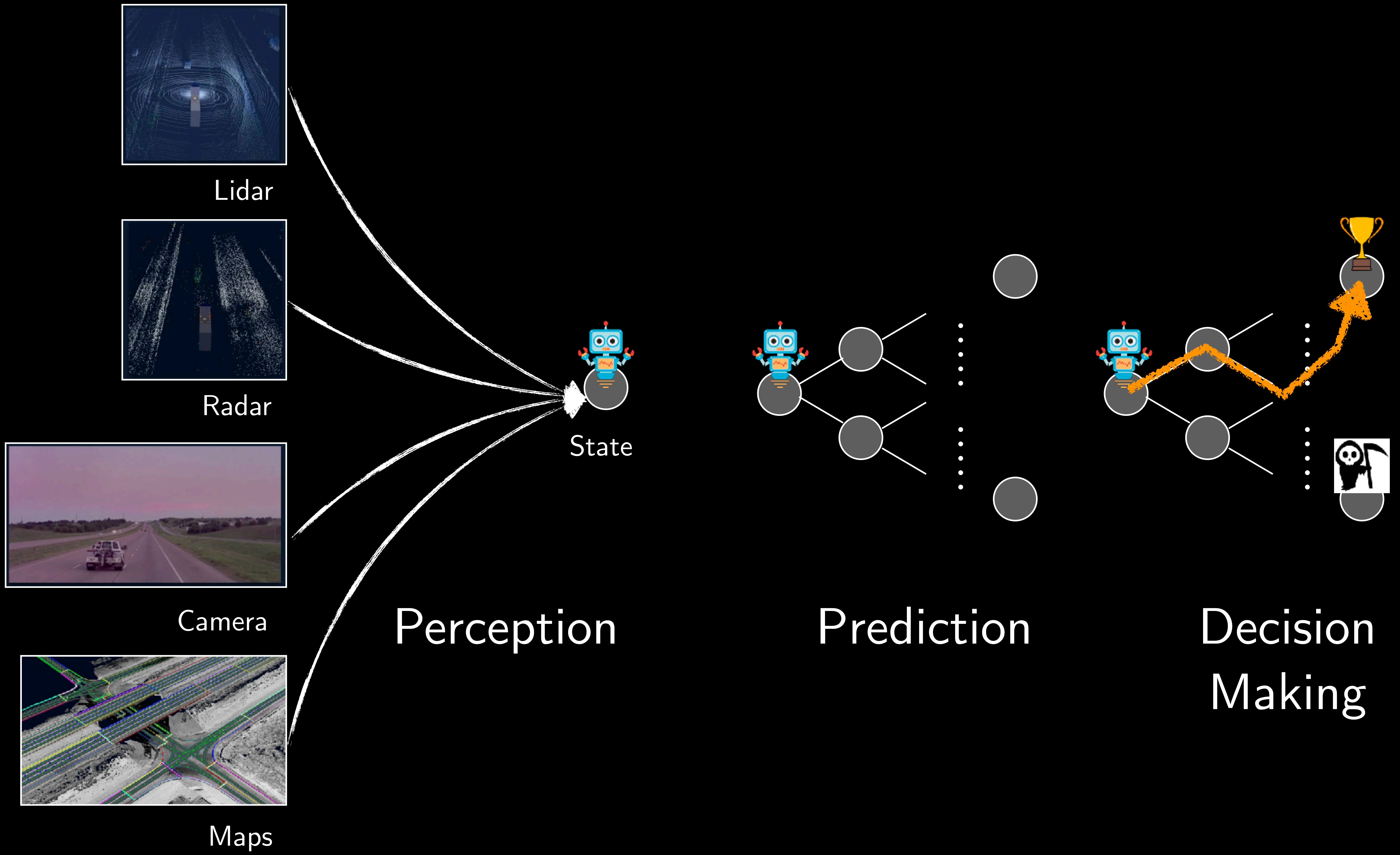
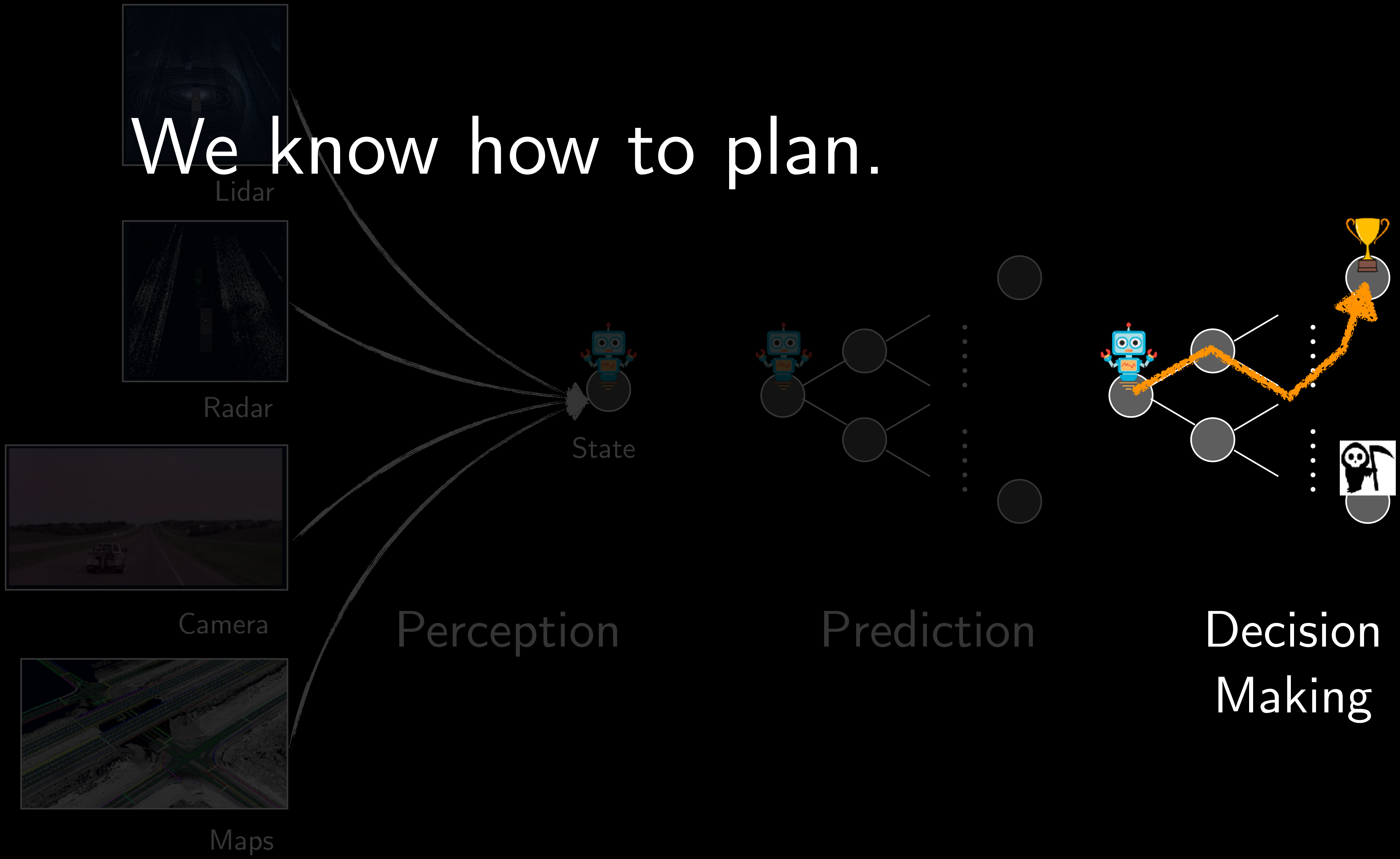
Forecasting in Self-Driving

Sanjiban Choudhury

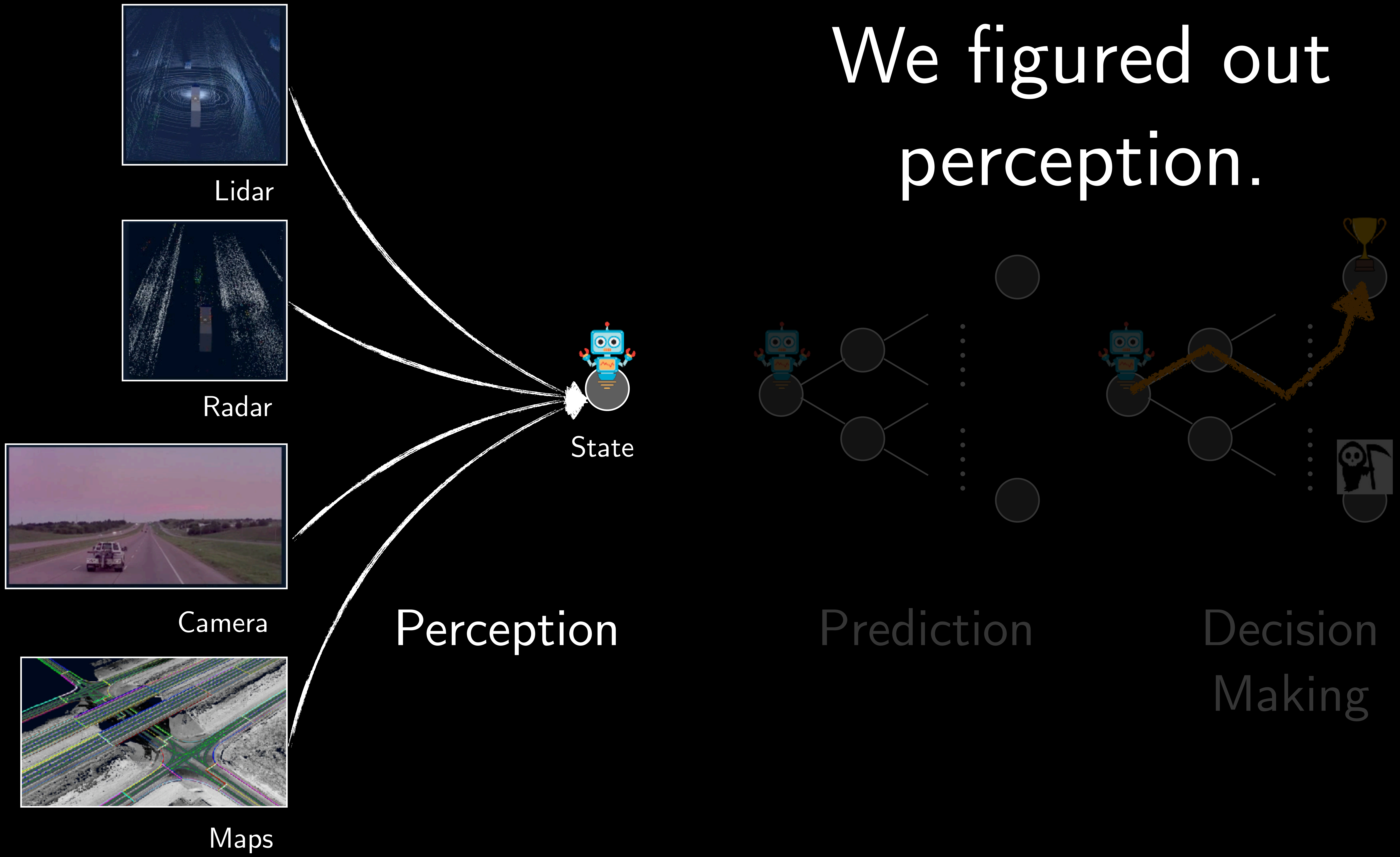
Cornell Bowers CIS
Computer Science


Lidar
Radar
Camera
Maps
State
Perception
Prediction
Decision Making

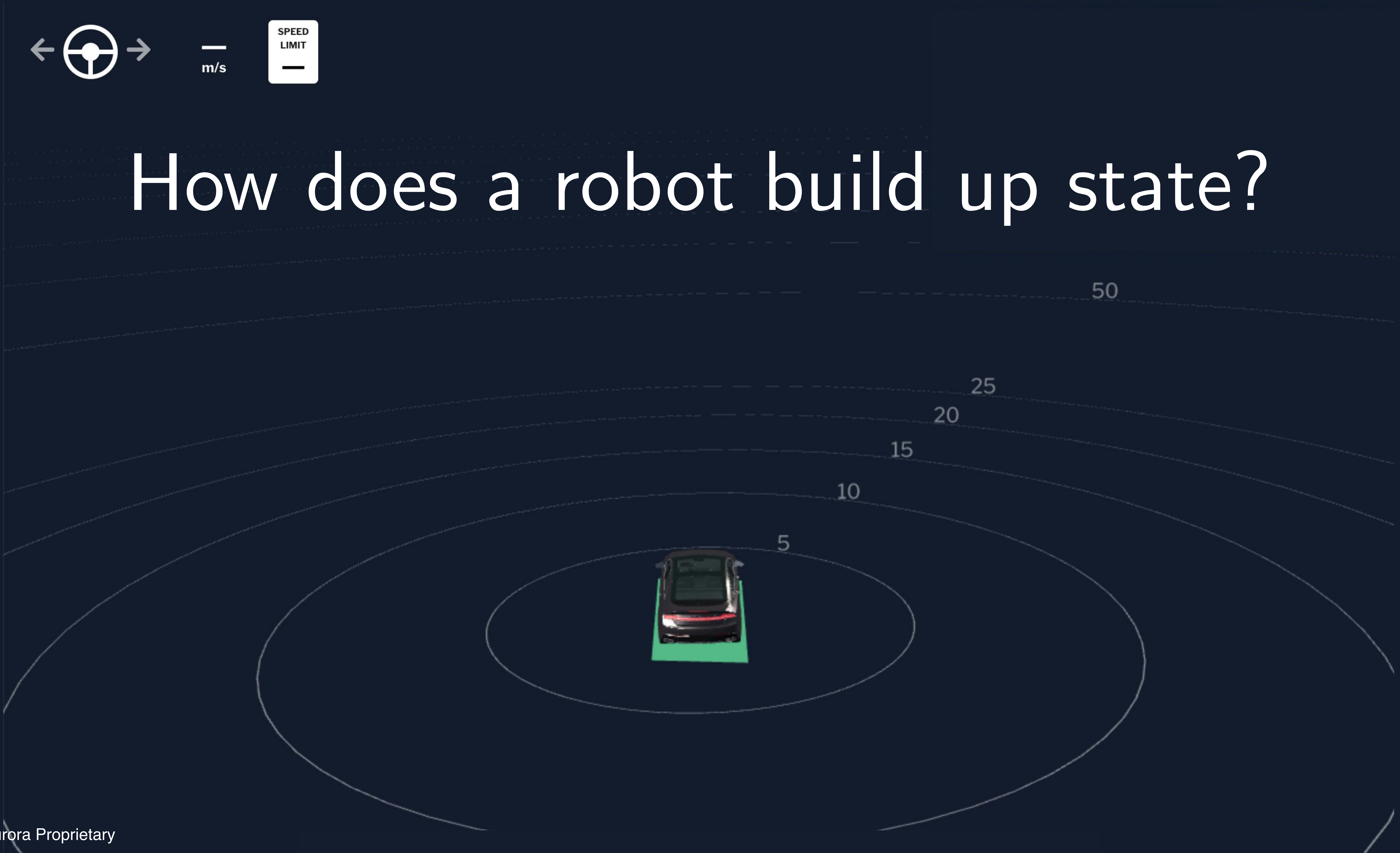
We know how to plan.



We figured out perception.



How does a robot build up state?



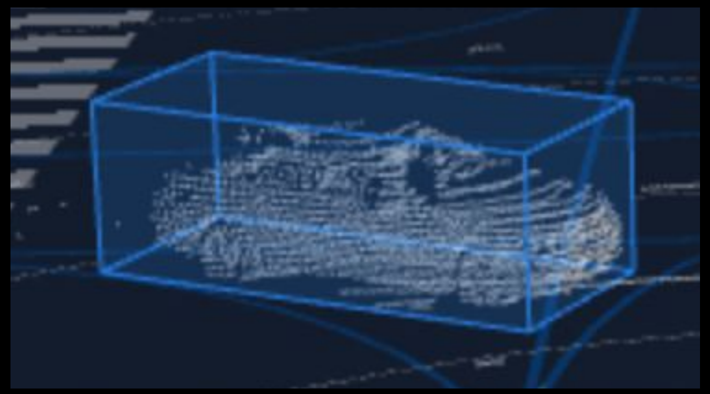

0.0
m/s
SPEED
LIMIT
11
ACTIVE
50
25
20
15
10


0.0
m/s
SPEED
LIMIT
11
ACTIVE
50
25
20
15
10
5


0.0
m/s
SPEED
LIMIT
11
ACTIVE
50
25
20
15
10

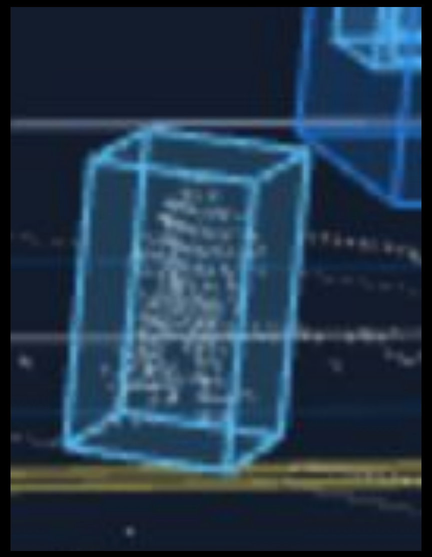
pose

(x, y, ψ)



vel

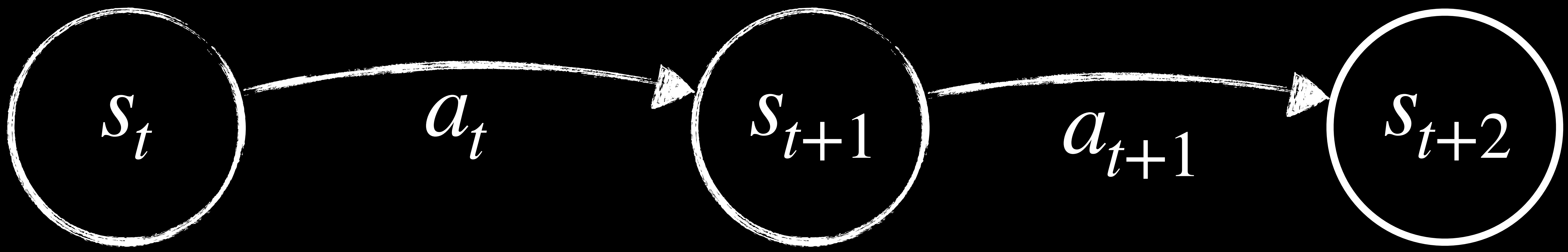
$(\dot{x}, \dot{y}, \dot{\psi})$



type

(pedestrian, car, cyclist)

But we do not observe these directly!



pose

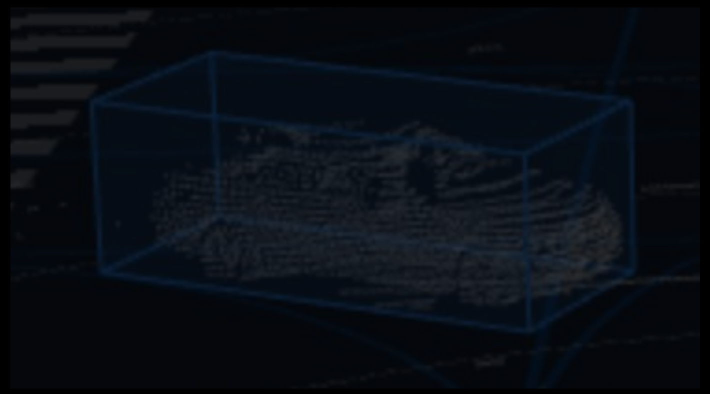
(x, y, ψ)

vel

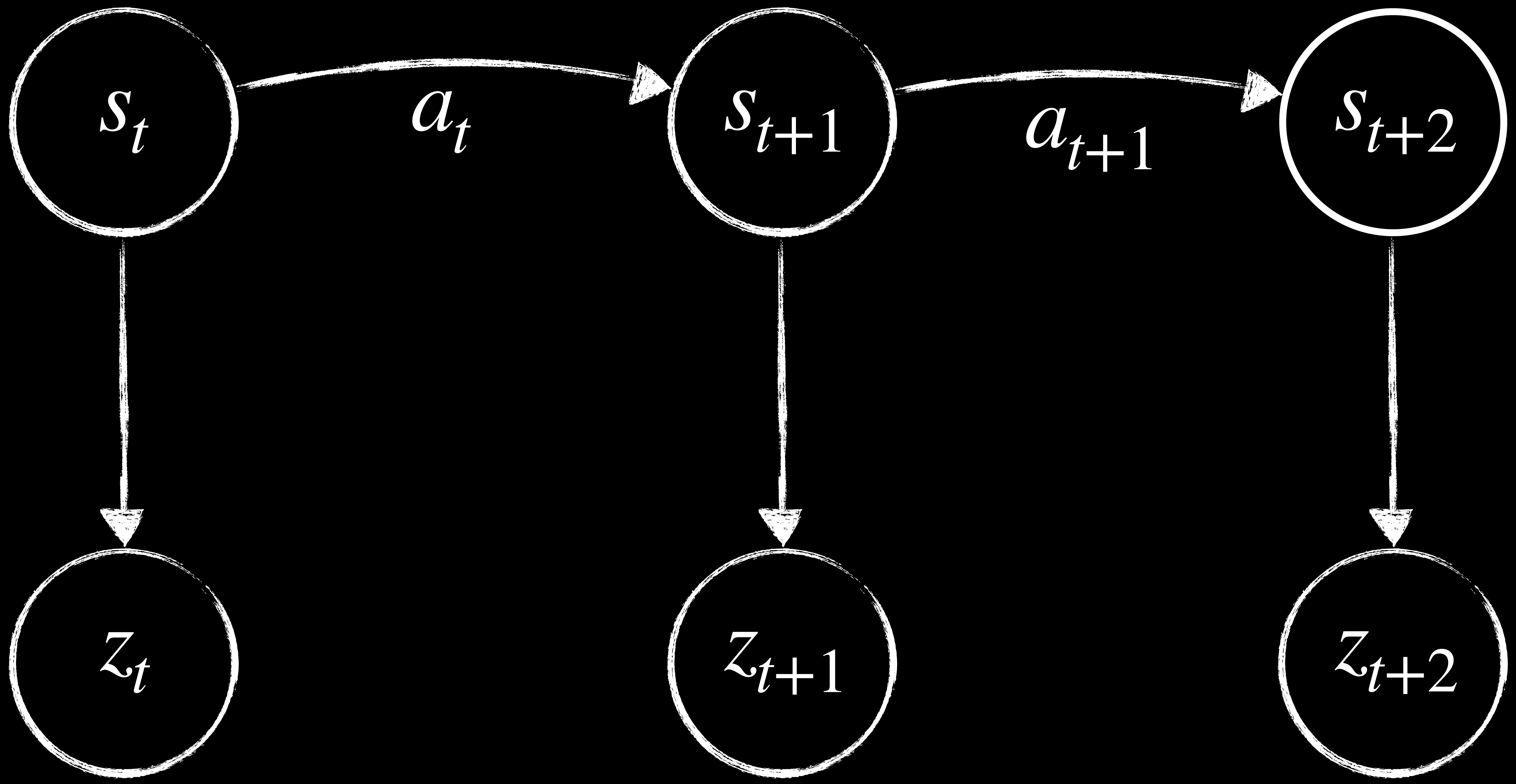
$(\dot{x}, \dot{y}, \dot{\psi})$

type

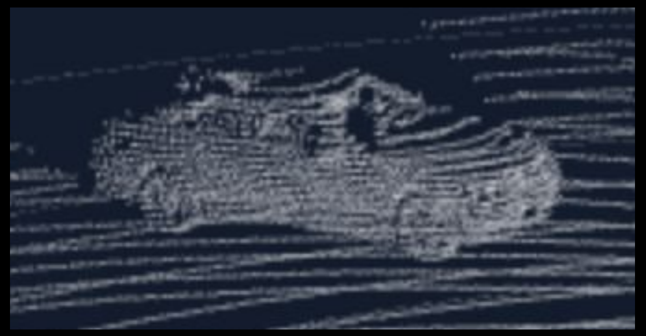
(pedestrian, car, cyclist)



Estimate state from observations

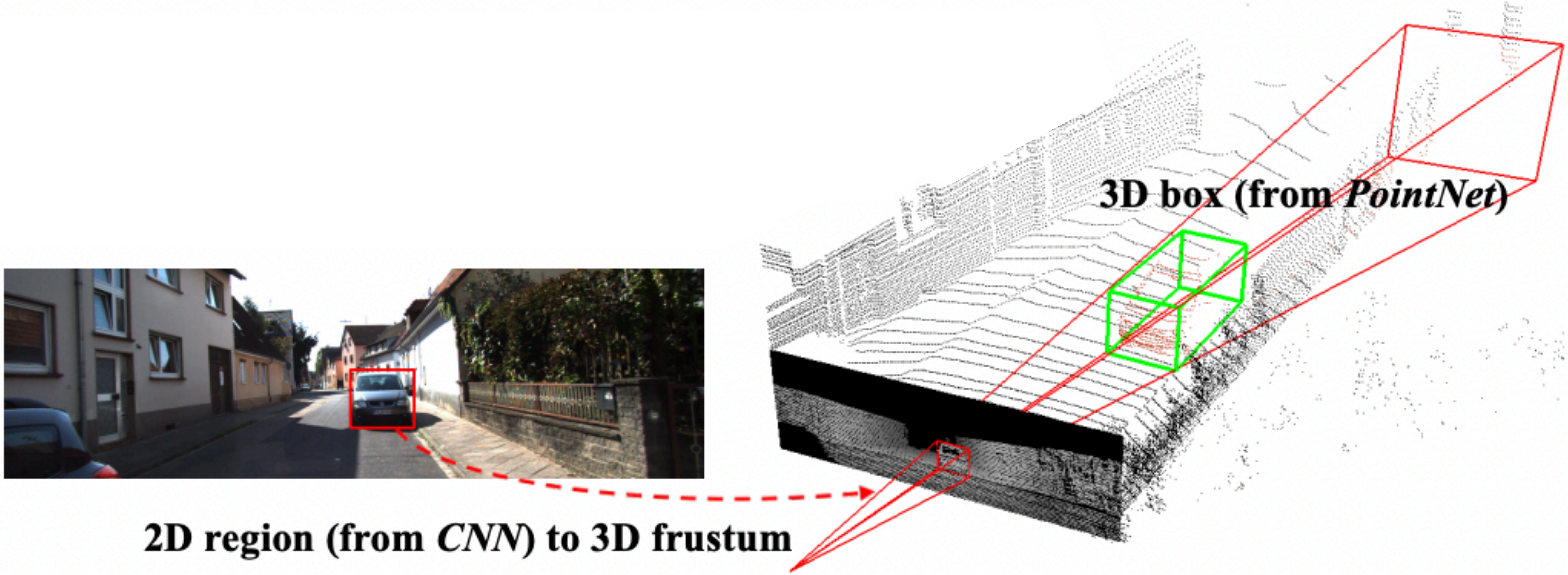


camera

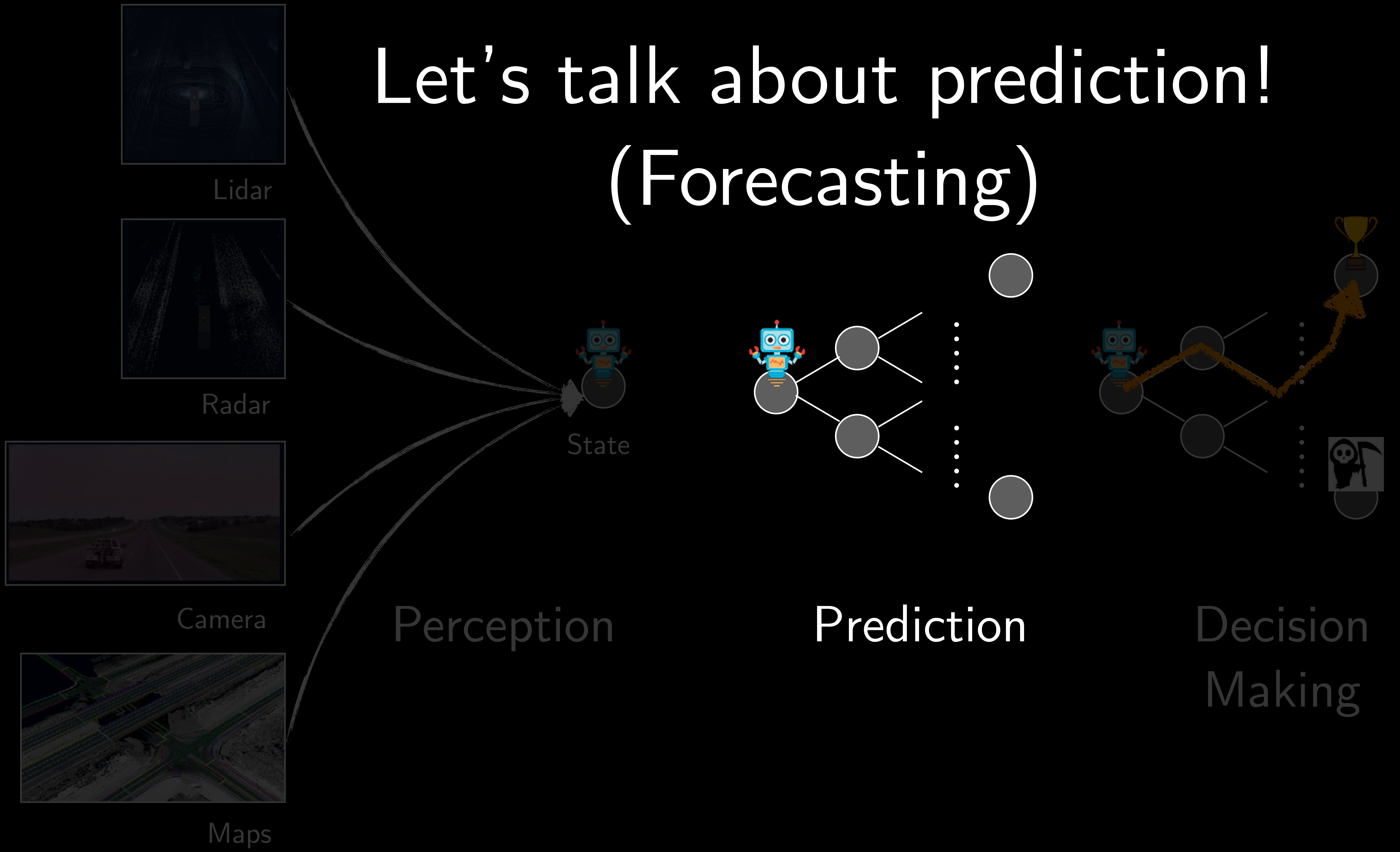


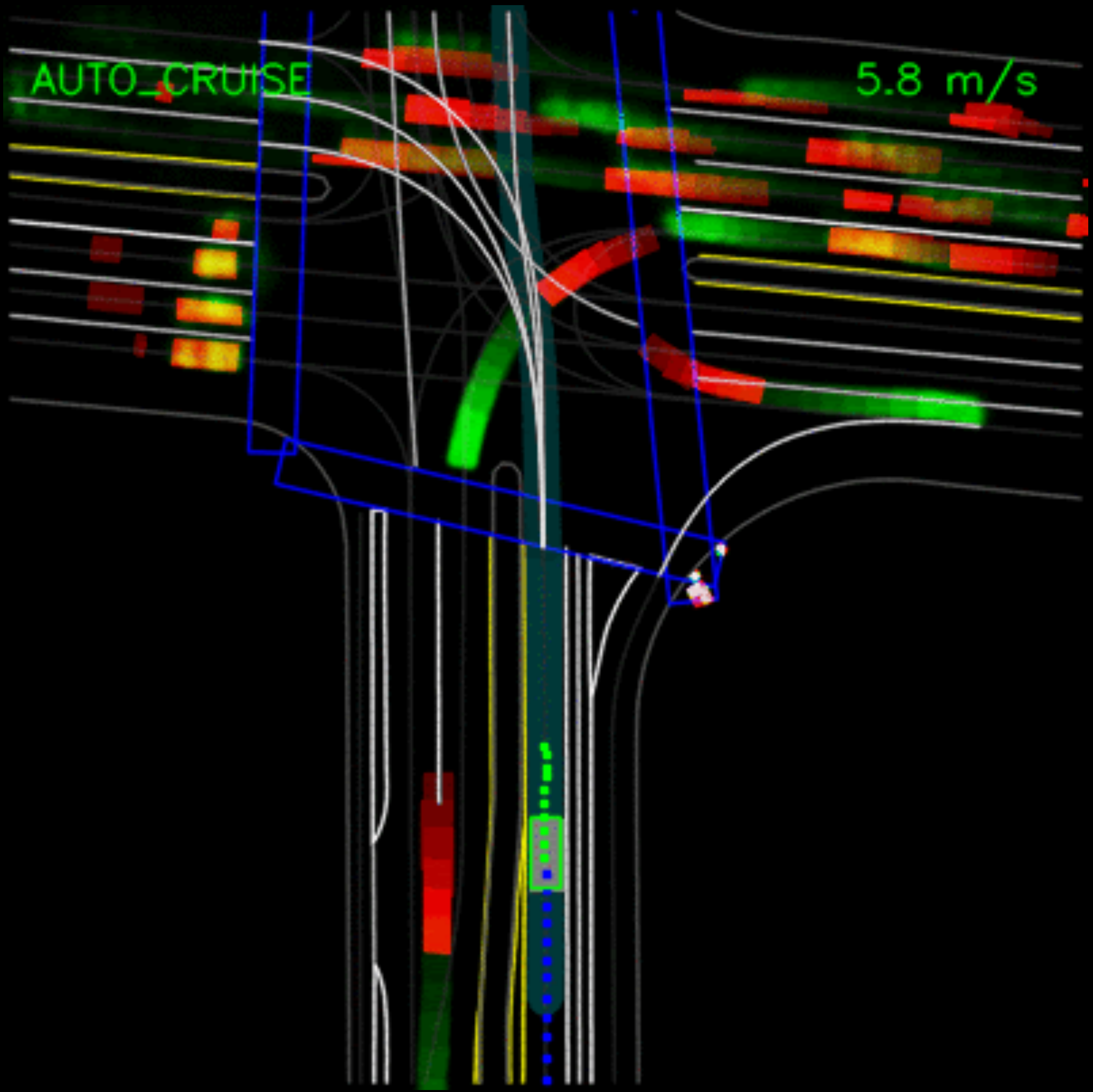
lidar

Frustum PointNets: 3D Instance Segmentation



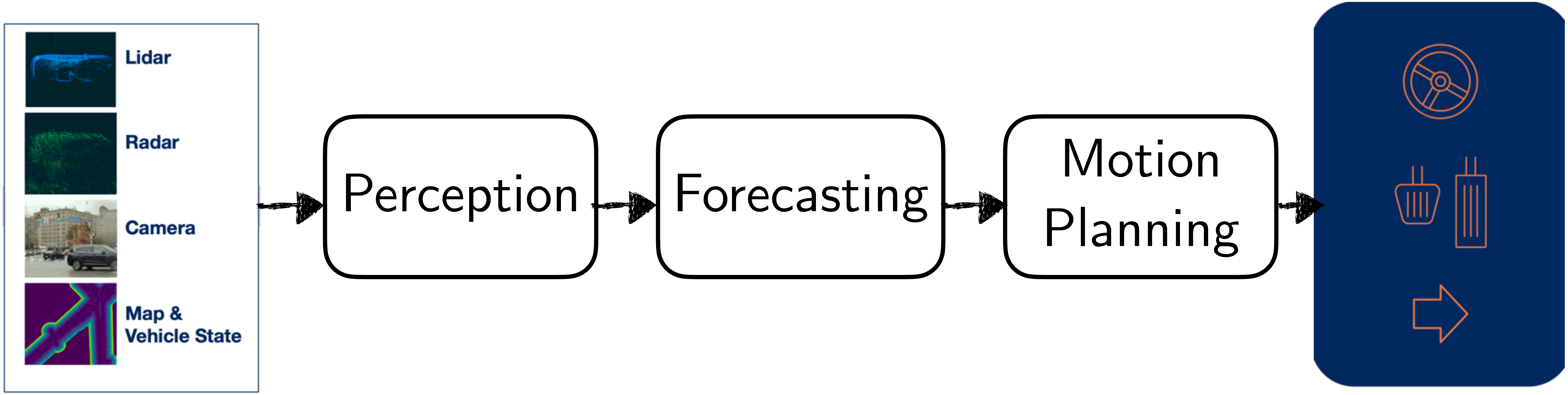
Let's talk about prediction! (Forecasting)




AUTO_CRUISE
5.8 m/s

Much of forecasting for self-driving is built on shaky foundations

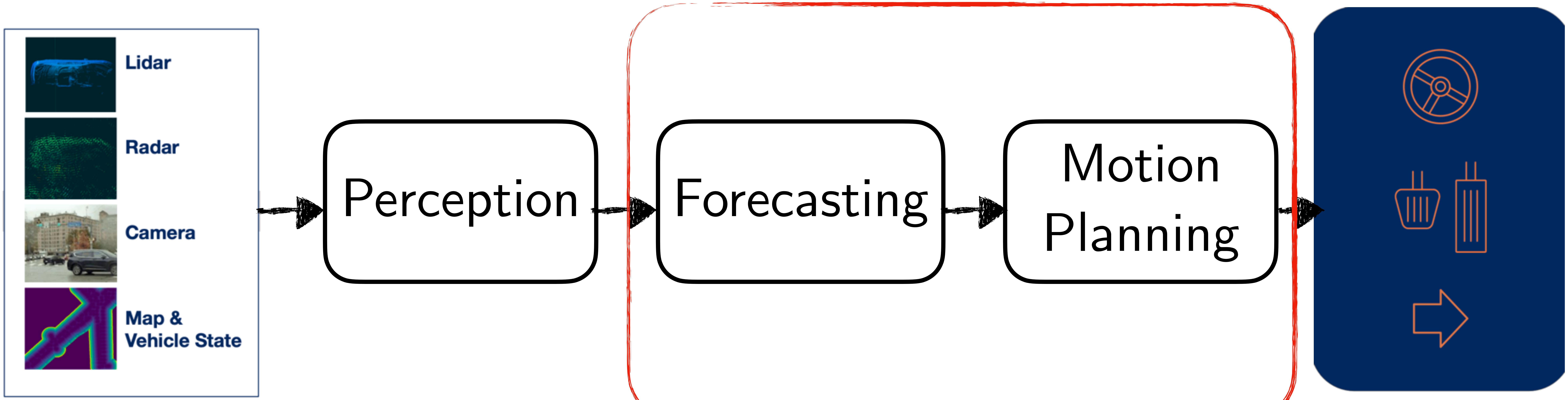
Traditional Architecture



Raw sensor data

Control actions

Should these be decoupled?



Raw sensor data

Control actions

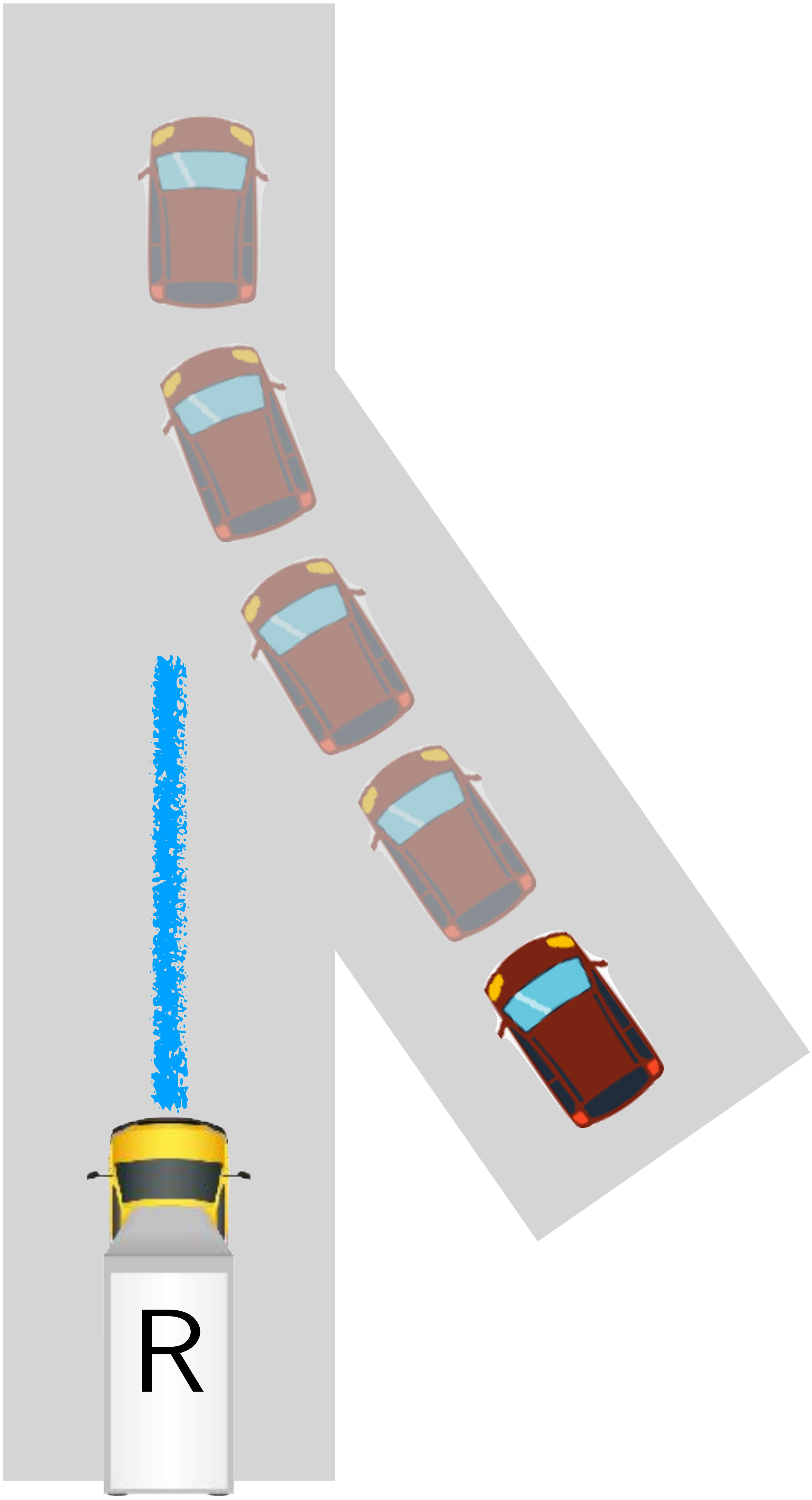
Shaky foundations of forecasting

Are we using the right model?

Are we collecting data correctly?

Are we using the right loss?

Example: Learning forecasts for merging actors



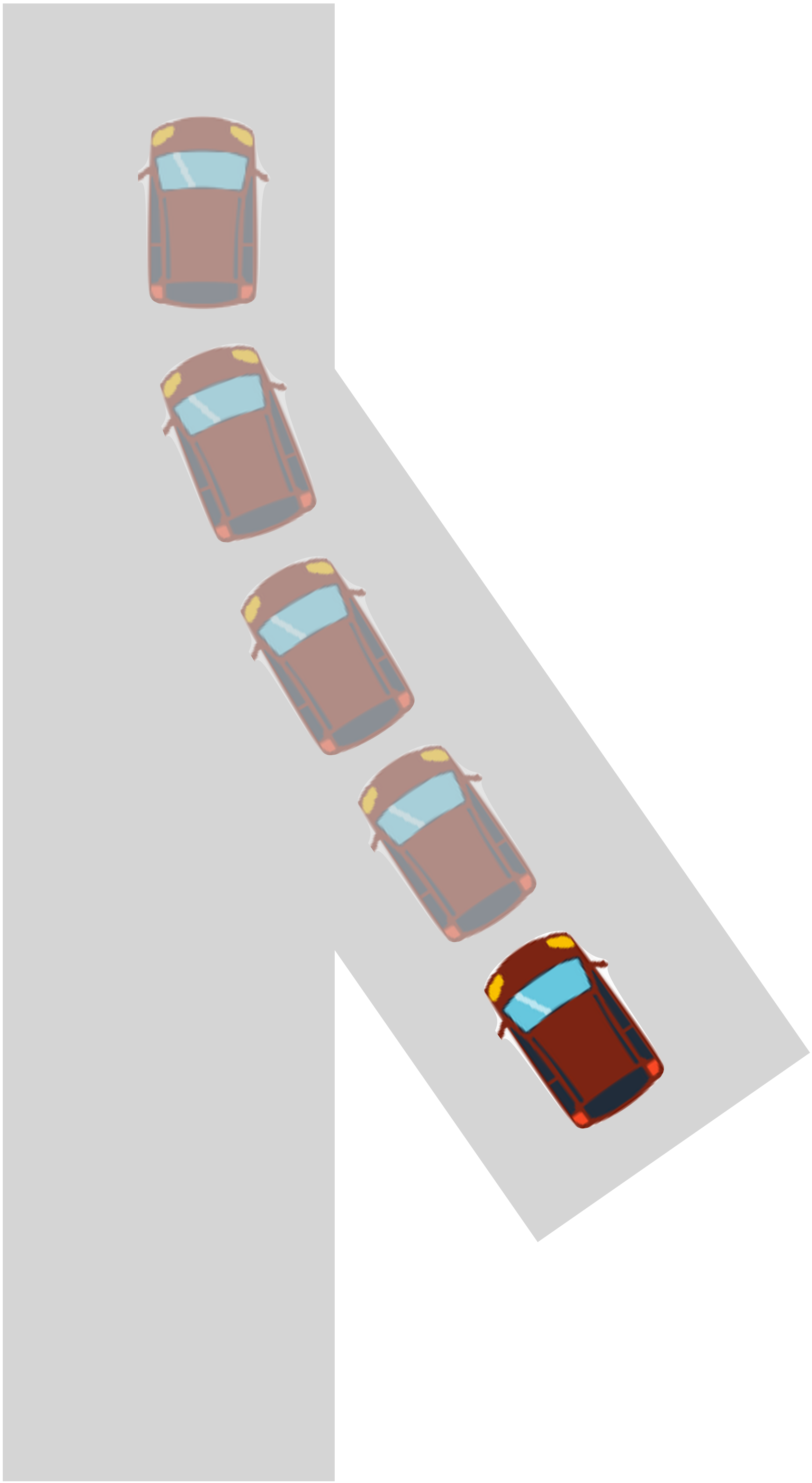
Goal

1. Predict 5s future trajectory

2. Plan with 5s future trajectory

Activity!

Train a learner to predict 5s future.



Model: Input / Output?

Data?

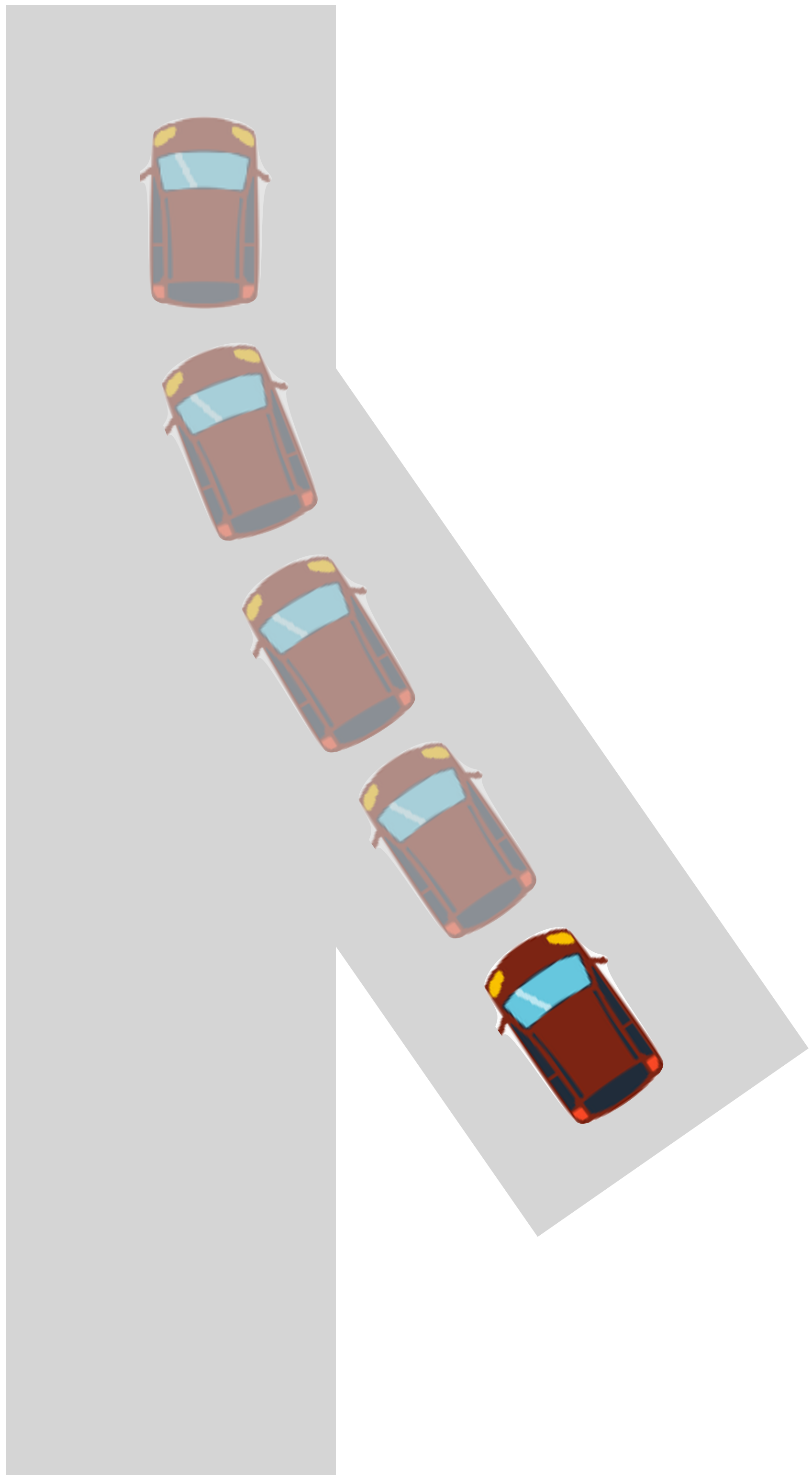
Loss?

Think-Pair-Share!

Think (30 sec): Train a learner to predict 5s future.

Pair: Find a partner

Share (45 sec): Partners exchange ideas



Model: Input / Output?

Data?

Loss?

Why is my current position not sufficient to predict future?

Simple latent variables:

Velocity, Acceleration may not be observable

Complex latent variables:

Intent (turning left, making a lane change) are
not observable and must be inferred from past actions

Sequence Model

(We are just going to use this as a black-box)

A very brief history of sequence prediction in robotics

Kalman Filter + Prediction

Handcraft observation models, apply Bayes rule to figure out latent state, predict. Problem: Tuning it is hard!

RNN, LSTMs

Learn the filter!

Problem: Forget long sequences since only one hidden state vector, vanishing/exploding gradients

Transformers

Retain all hidden state.

Problem: Pay O(H^2) computation

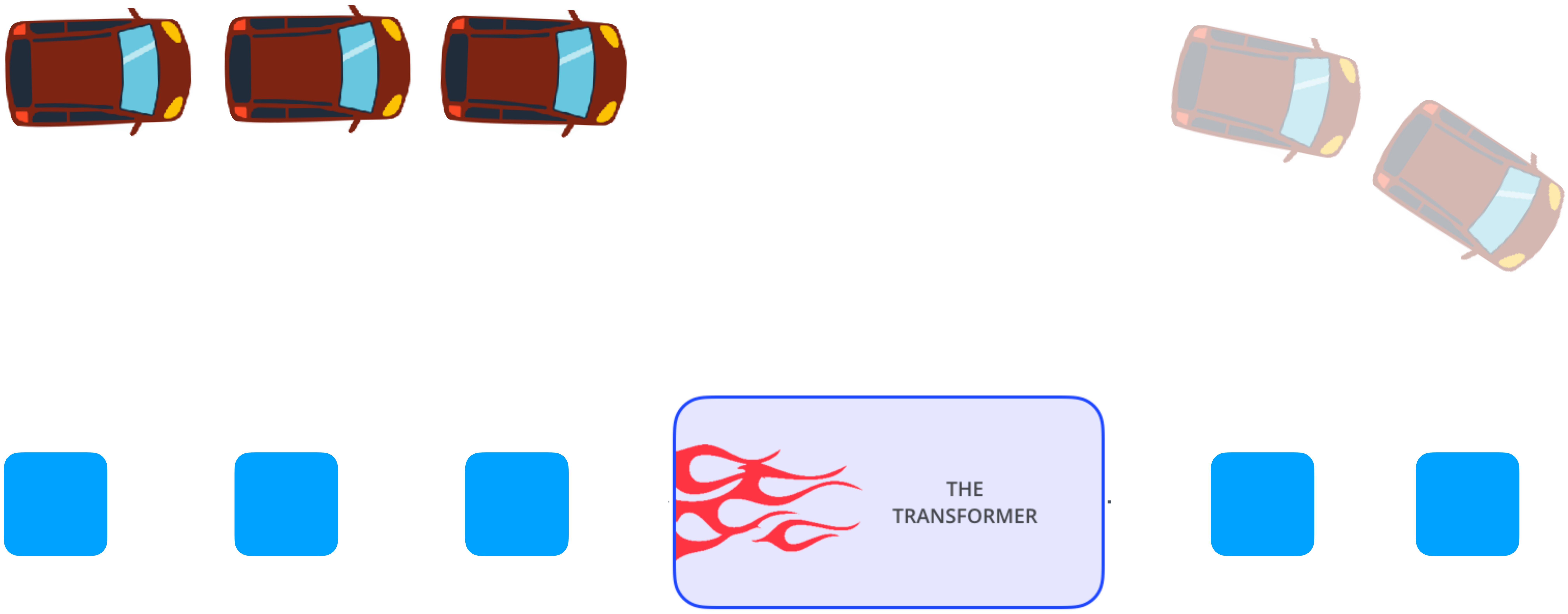
Many good introductory resources on transformers

https://jalammar.github.io/illustrated-transformer/

https://jalammar.github.io/illustrated-gpt2/

Back to forecasting

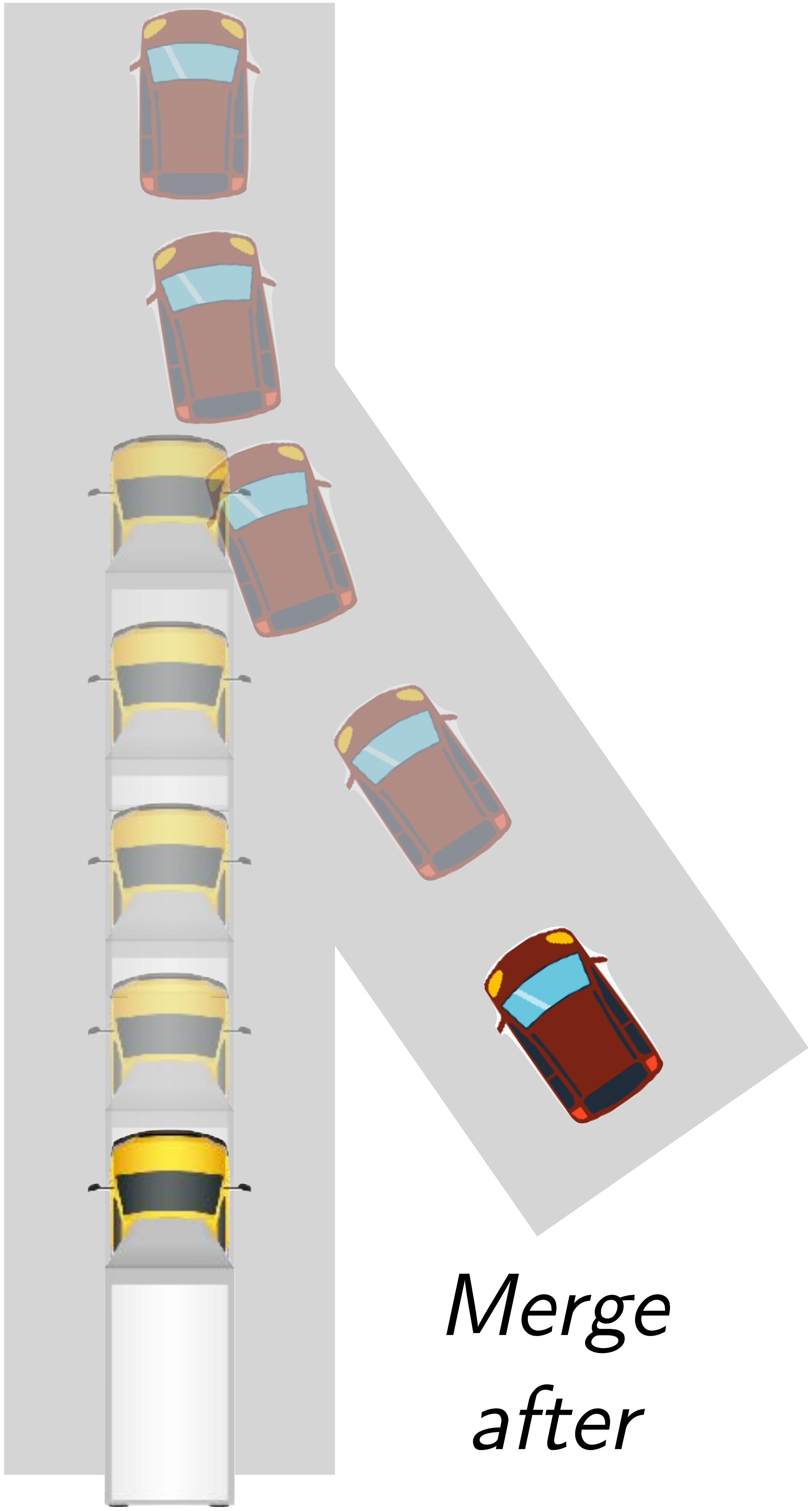
Model: Use a transformer to map history to future

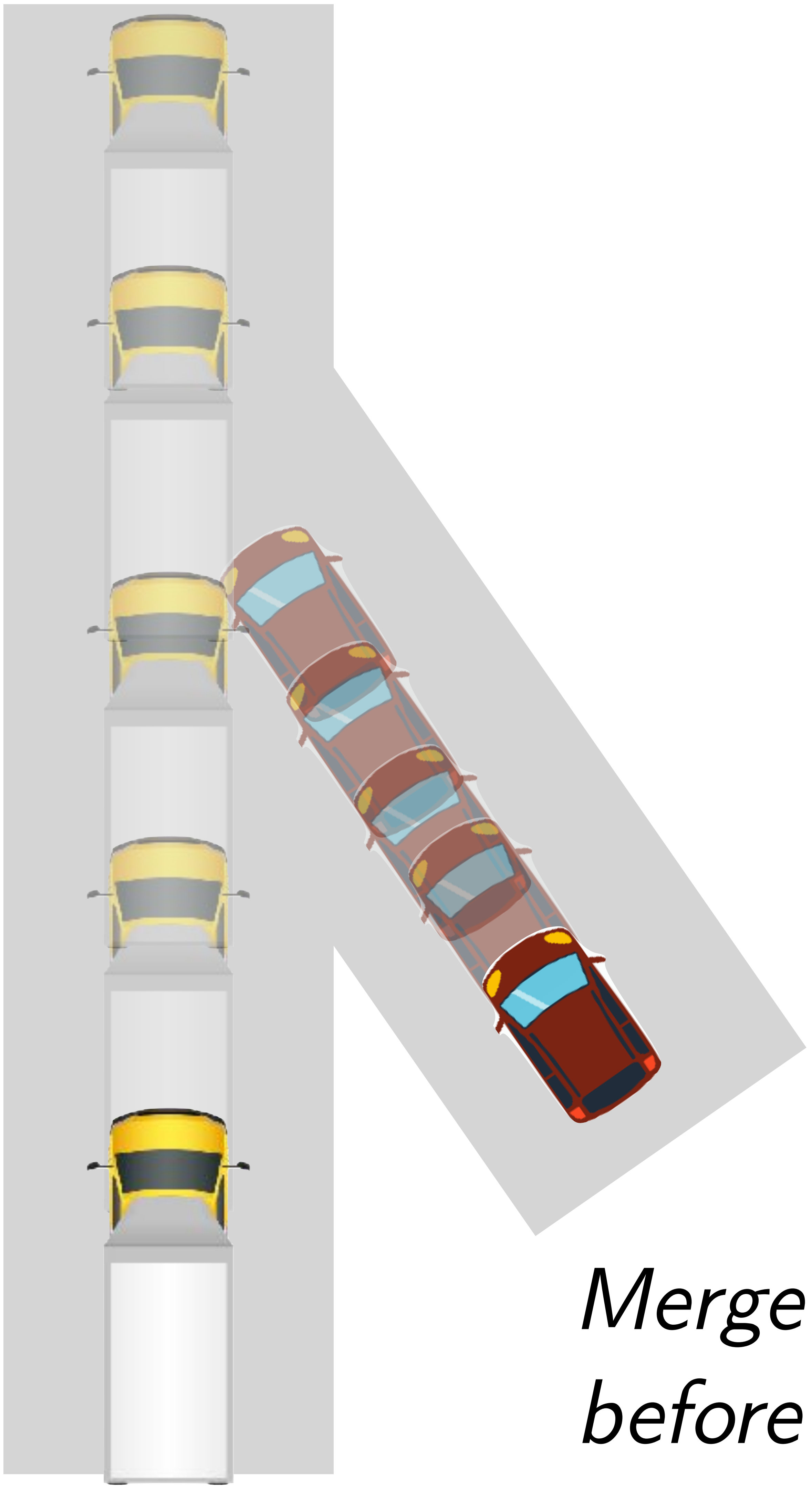


s_{t-2} s_{t-1} s_t s_{t+1} s_{t+2}

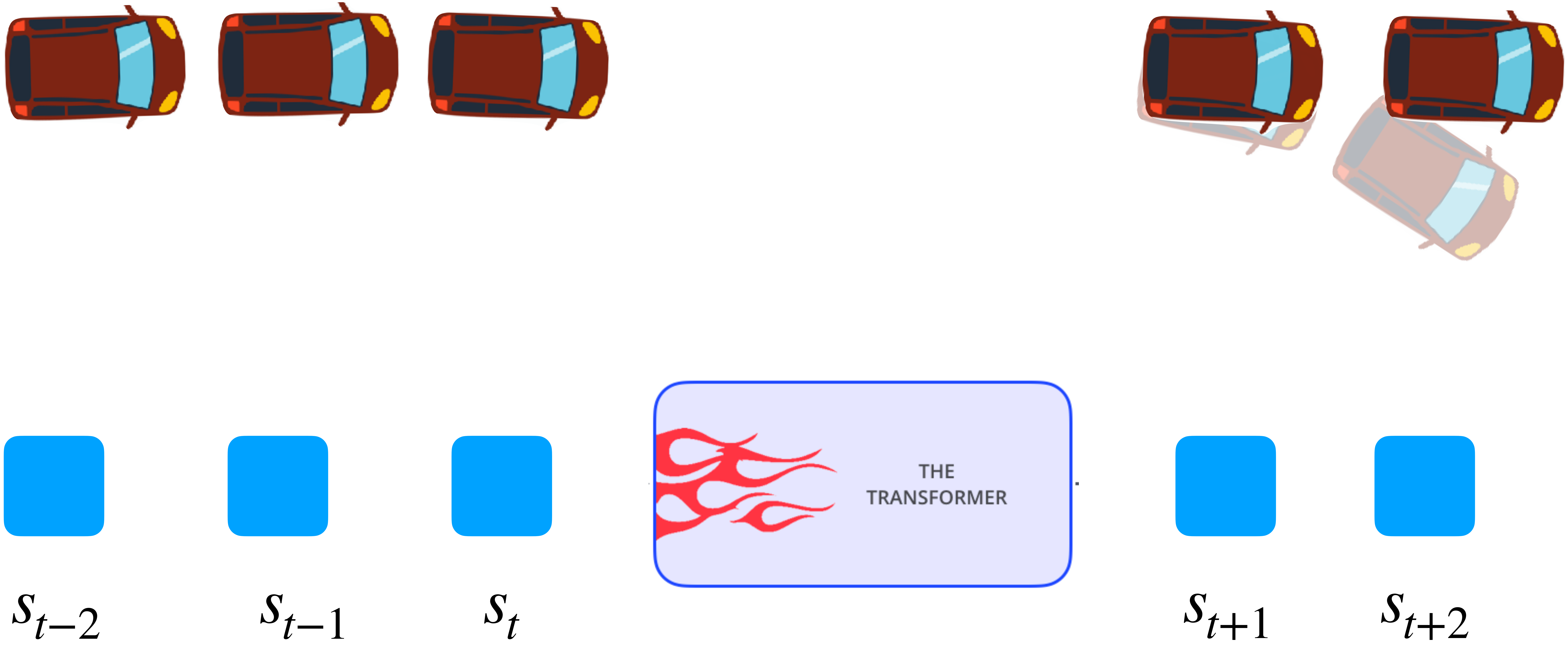
Data: Drive around the car and collect data

Train Data

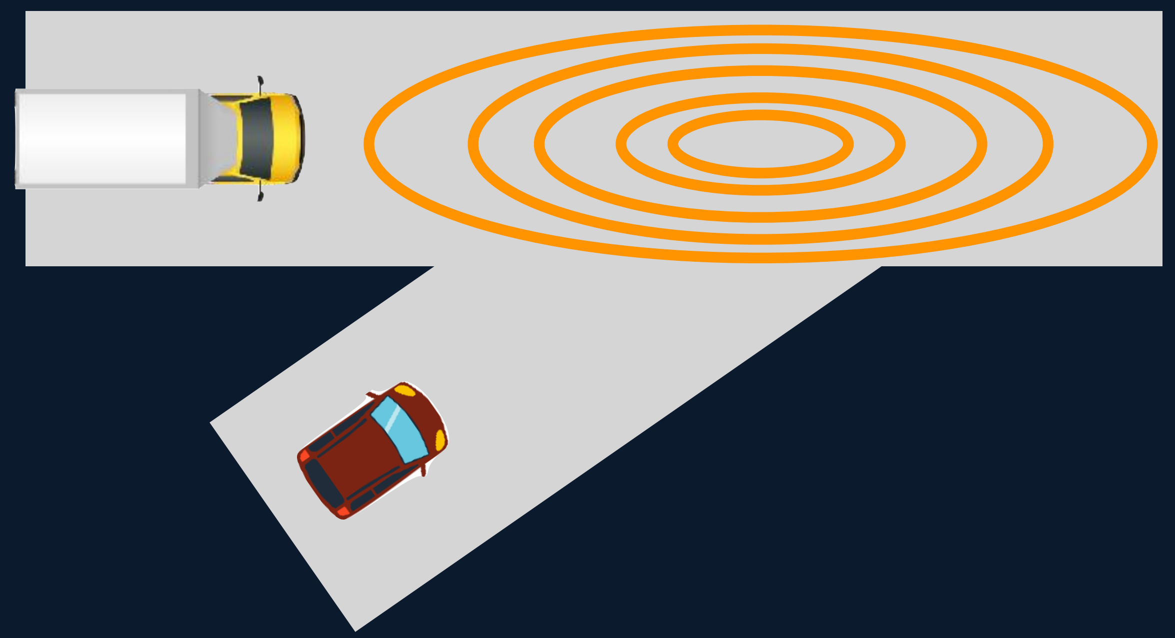




Loss: L2 Loss from Ground Truth



We have model, data, loss.
Let's deploy!



Forecasts have huge variance!

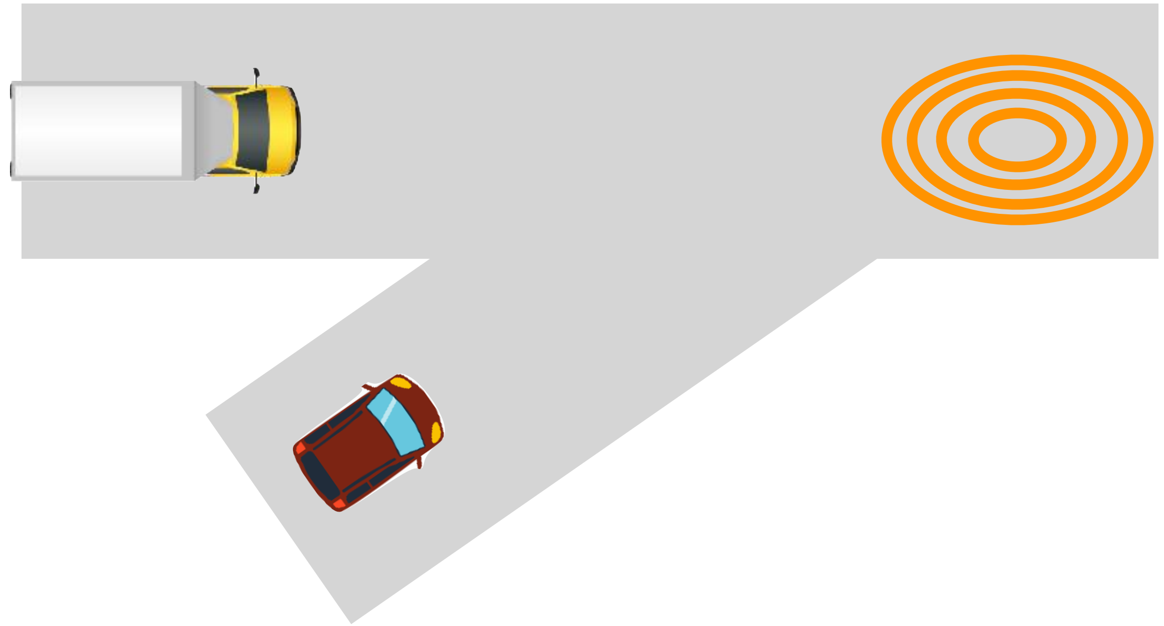
Planner brakes aggressively!

Why is the forecast so whacky?

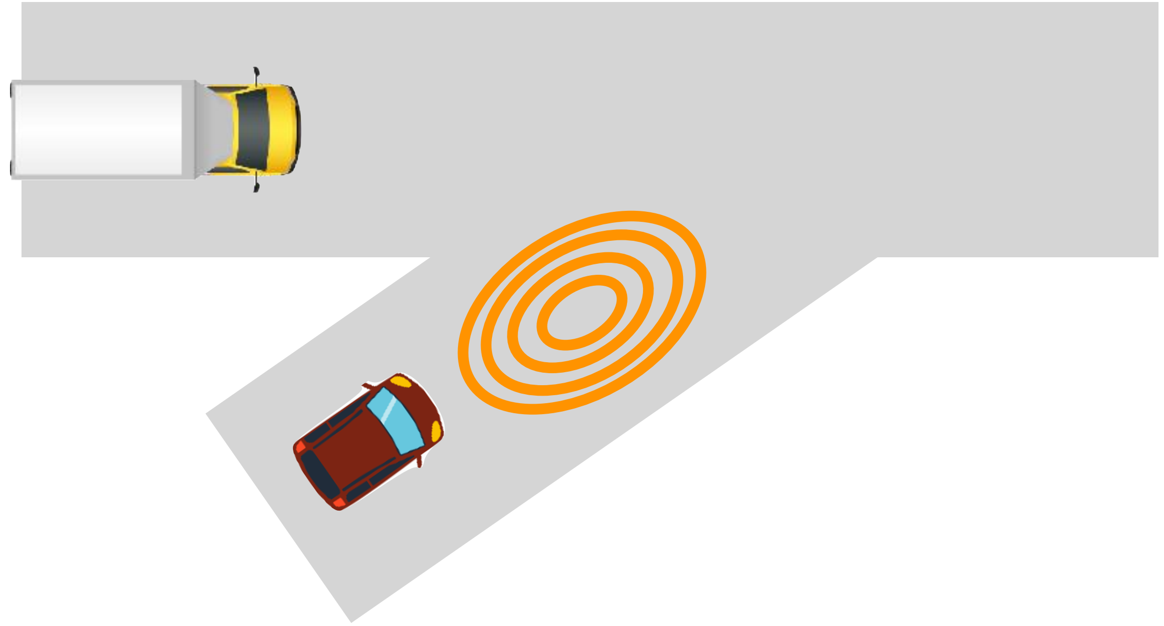
Why is the forecast so whacky?

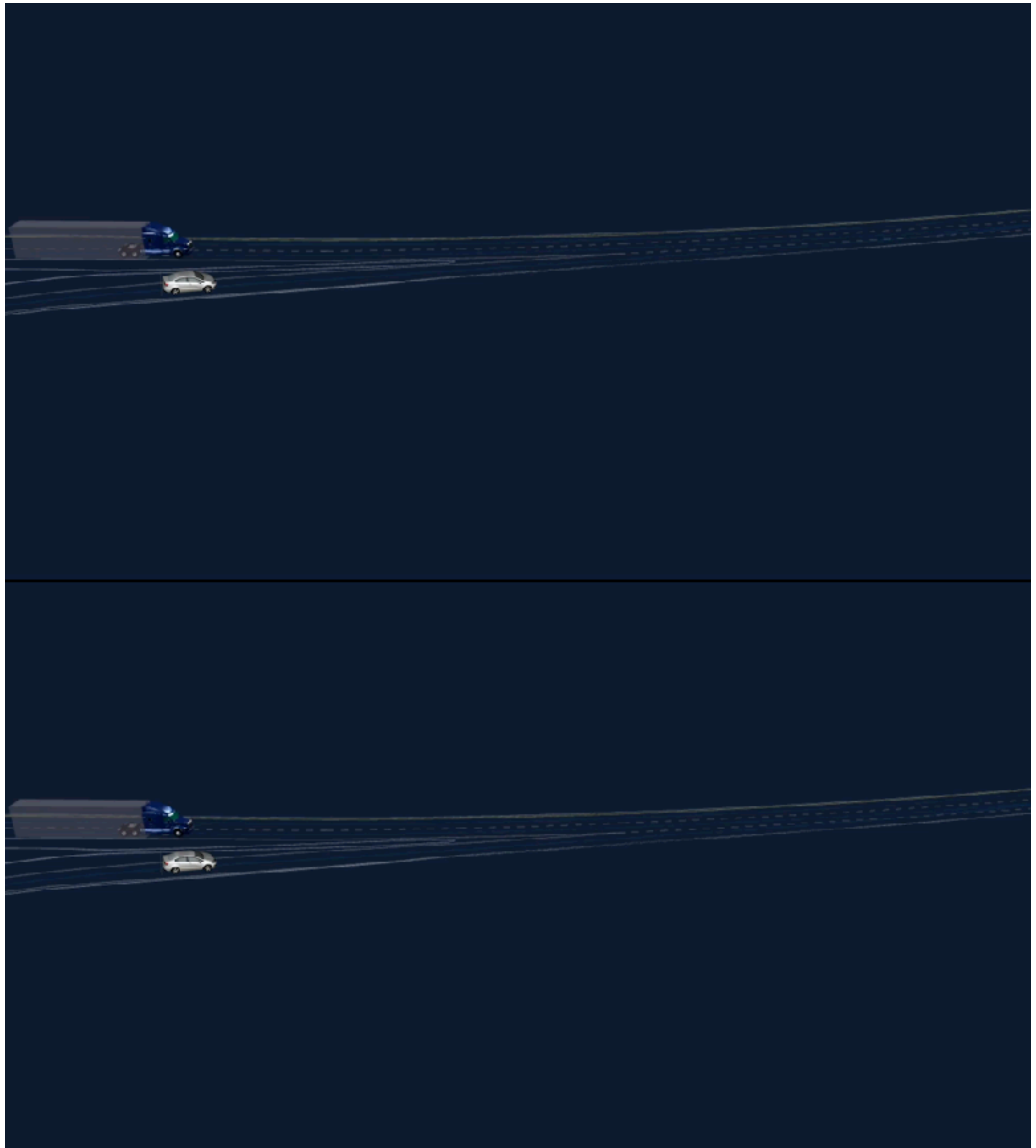
Marginalizing/Averaging
over multiple modes!

Mode A:
Robot merges
after



Mode B:
Robot merges
before



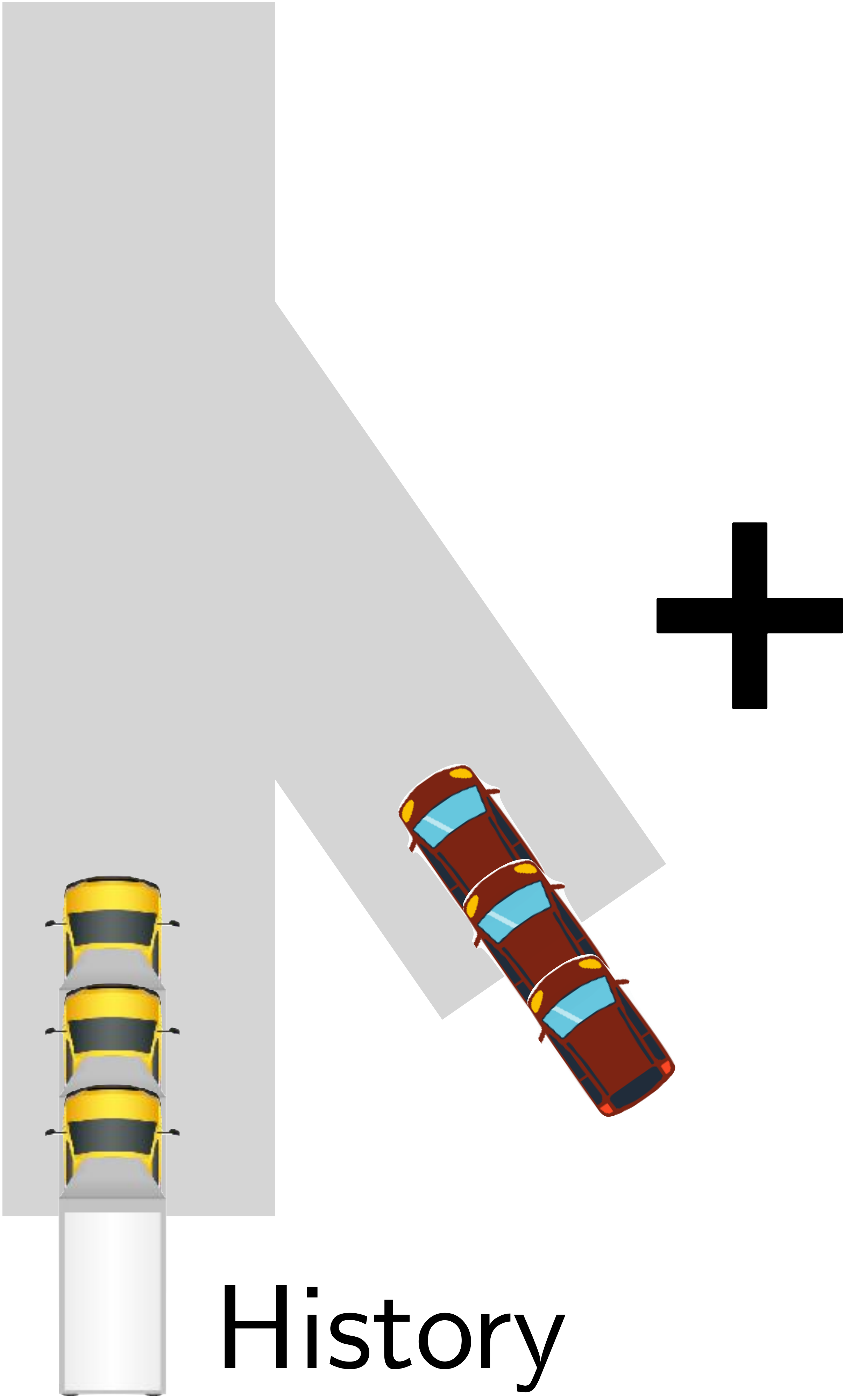


What robot does depends on other humans

What other humans do depends on the robot

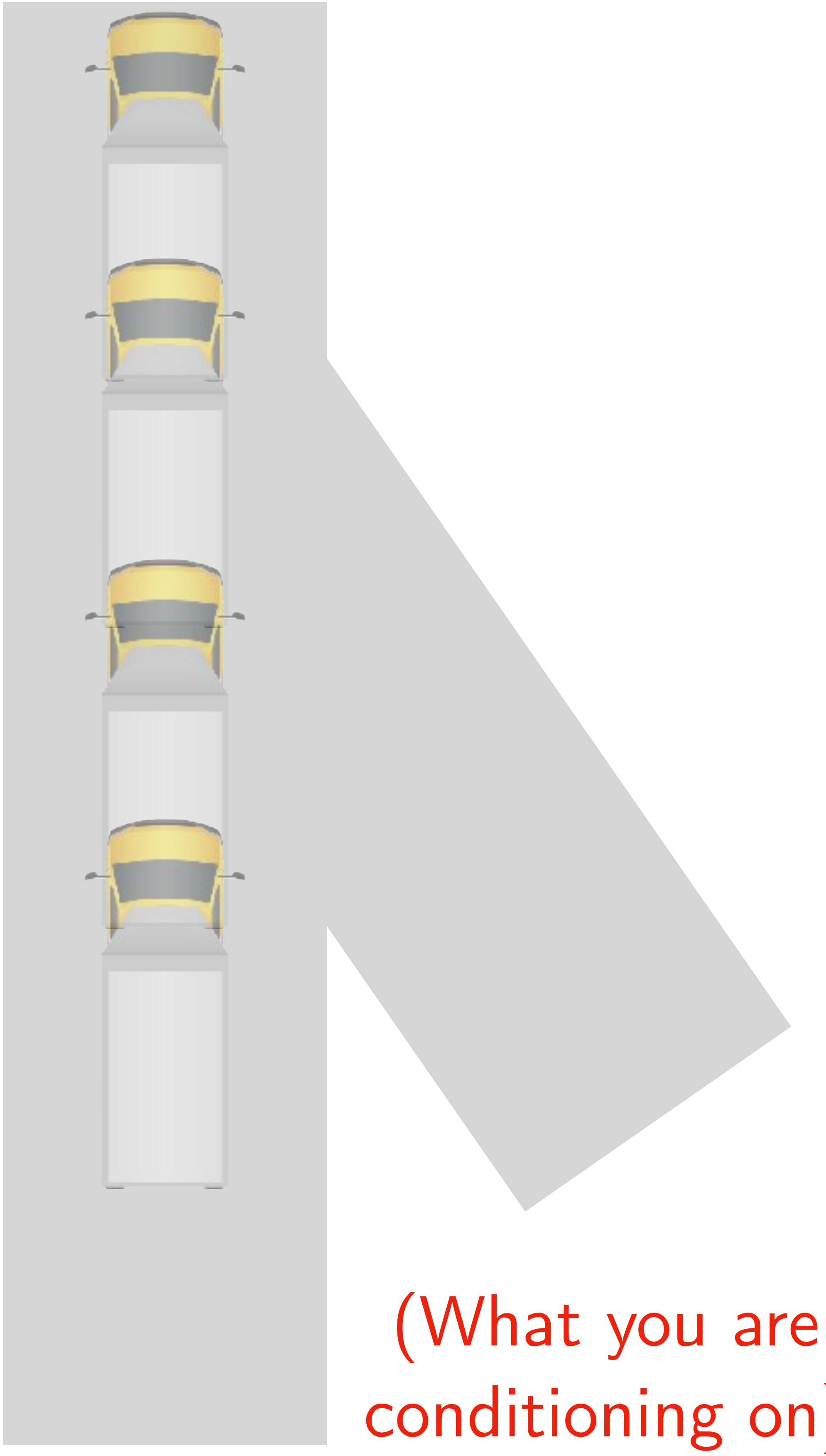
Forecasting-or-planning: a chicken-or-egg problem

Solution: Train Conditional Forecasts



History

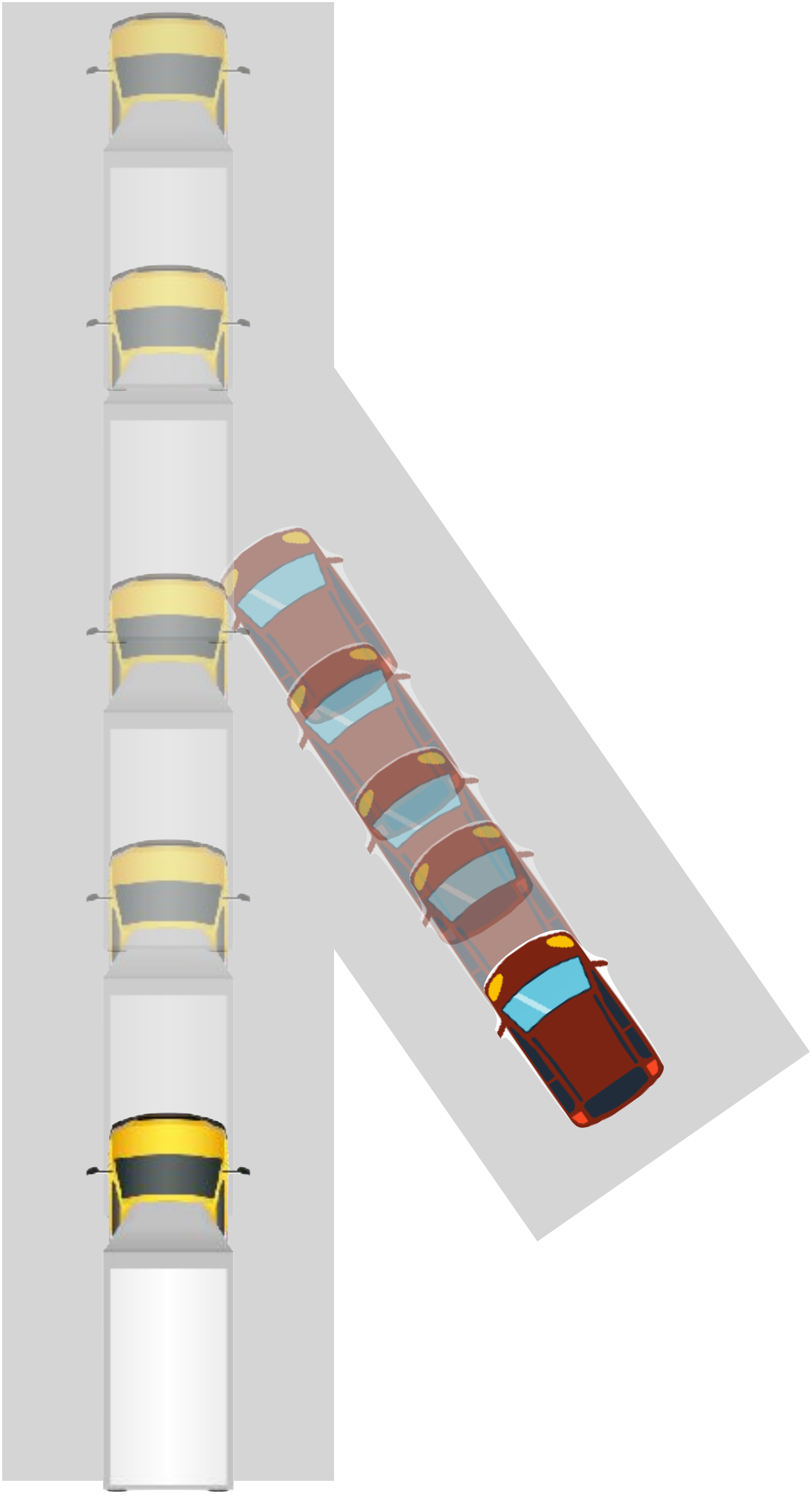
$\ldots, s_{t-2}, s_{t-1}, s_t$



(What you are conditioning on)

Plan

$a_{t_1}, a_{t+2}, a_{t+3}$



Forecast

$s_{t+1}, s_{t+2}, s_{t+3}, s_{t+4}$

How can I use conditional forecasts in practice?

Pseudo code for planning with forecasts

Initialize with a library of candidate trajectories Ξ

For $\xi_{plan} \in \Xi$:

Call conditional forecast with history and ξ_{plan}
to predict $\xi_{forecast}$ for all the agents

Compute cost of ξ_{plan} using $\xi_{forecast}$

Return cheapest plan ξ^*_{plan}

Pseudo code for planning with forecasts

Initialize with a library of candidate trajectories Ξ

For $\xi_{plan} \in \Xi$:

Call conditional forecast with history and ξ_{plan} to predict $\xi_{forecast}$

Compute cost of ξ_{plan} using $\xi_{forecast}$

Return cheapest plan ξ^*_{plan}

Trajectories are continuous sequences of motion. Space of all candidate trajectories is huge!!

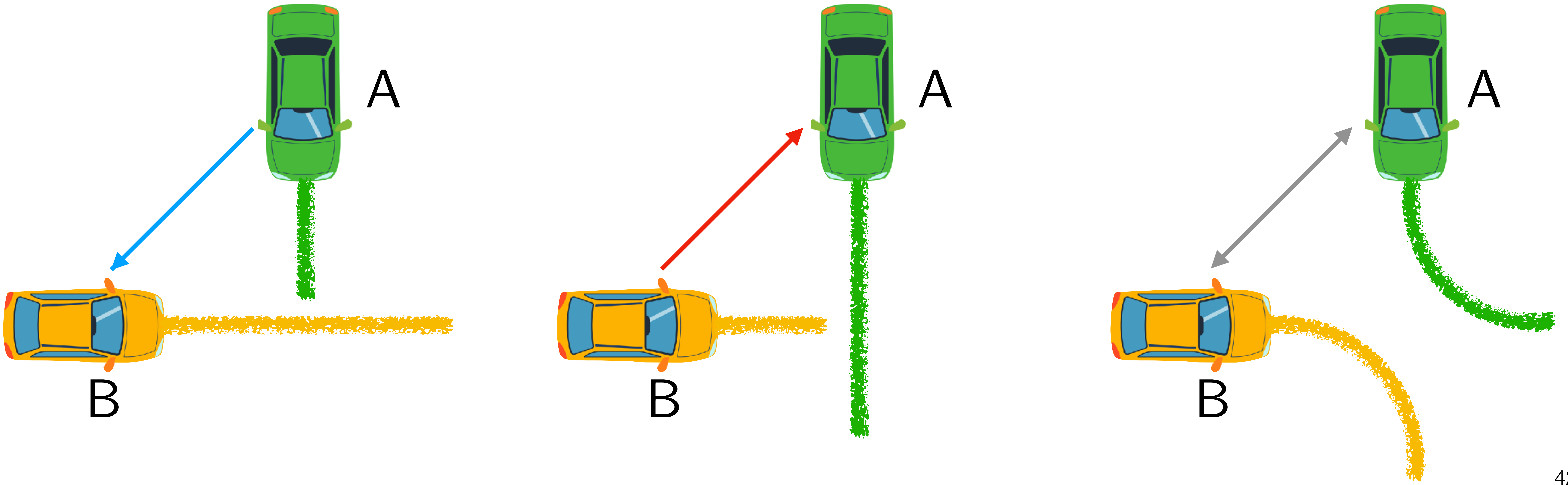
There is a discrete grammar for self-driving …

3 fundamental modes of space-time paths

A Yields to B

B Yields to A

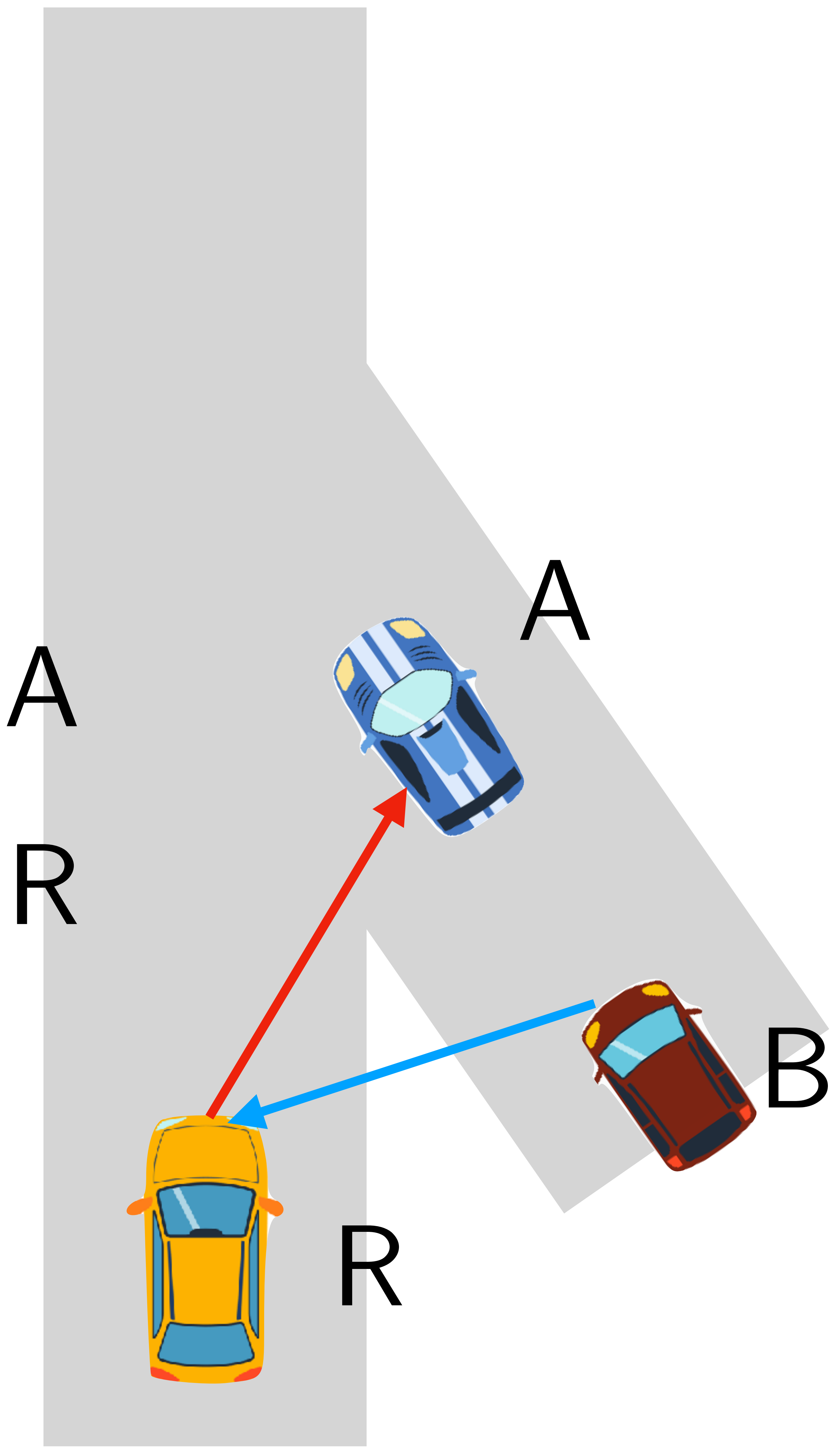
Not Yield



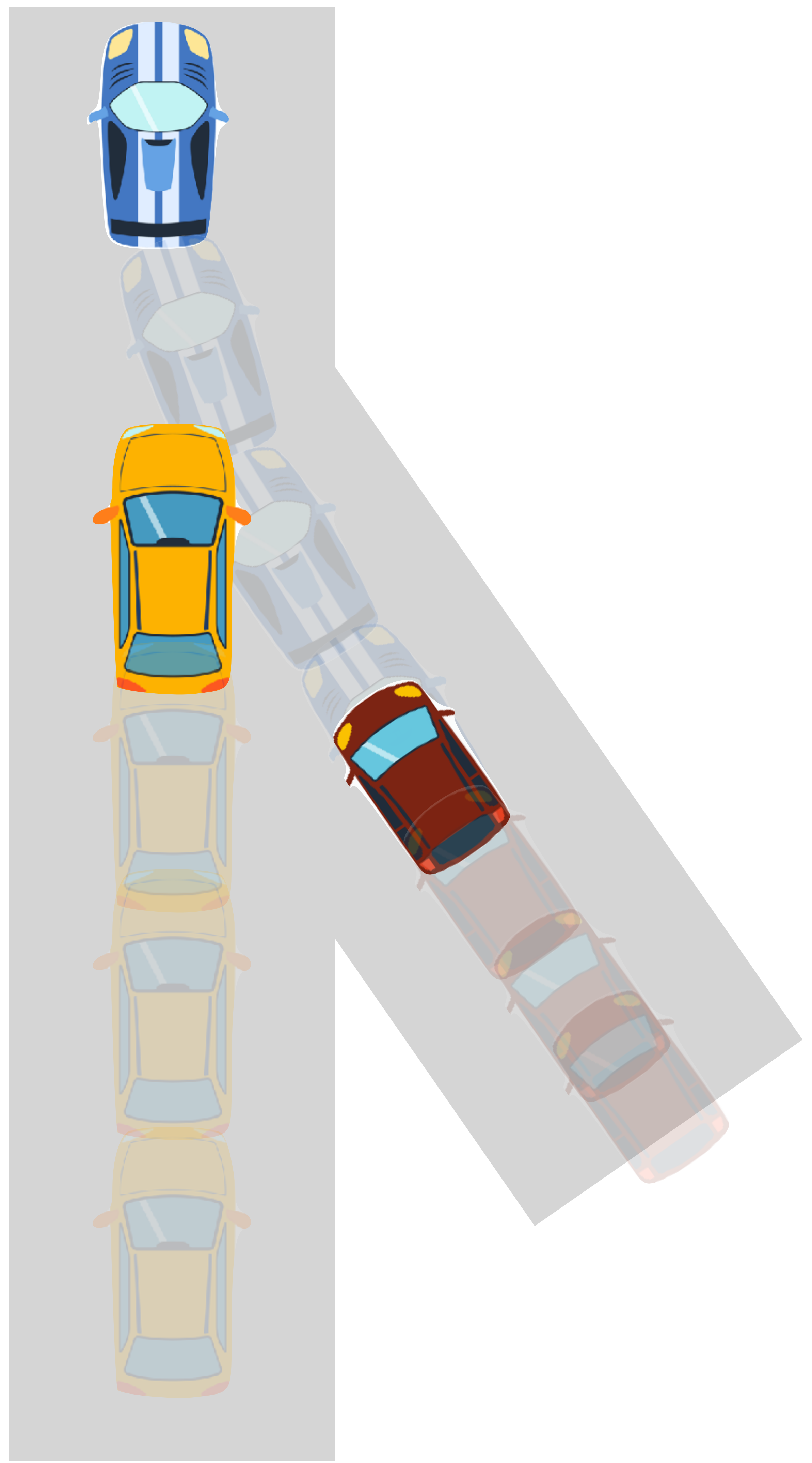
Mode ≡ A single basin of forecast

R Yields to A

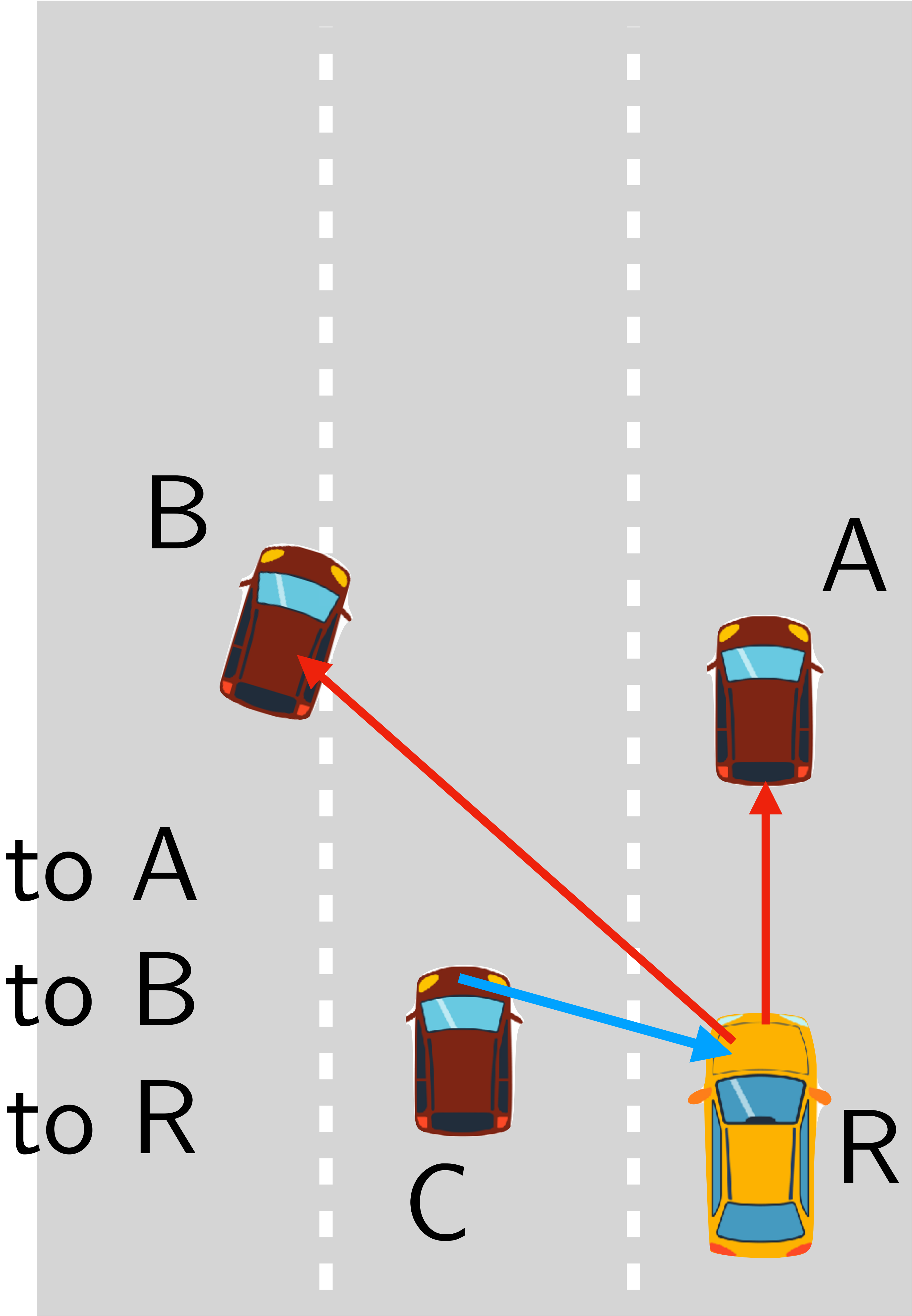
B Yields to R



≡

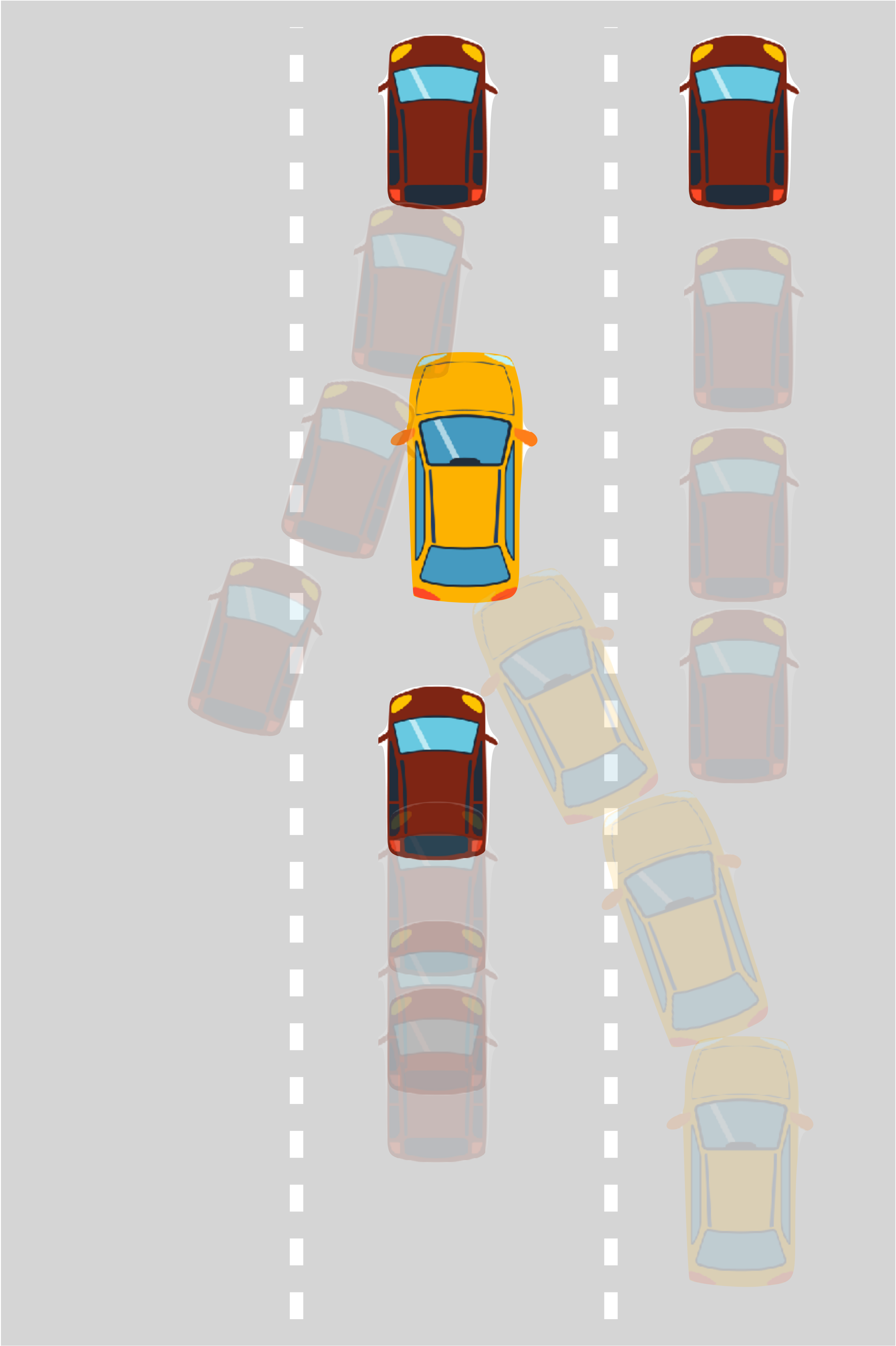


Mode ≡ A single basin of forecast



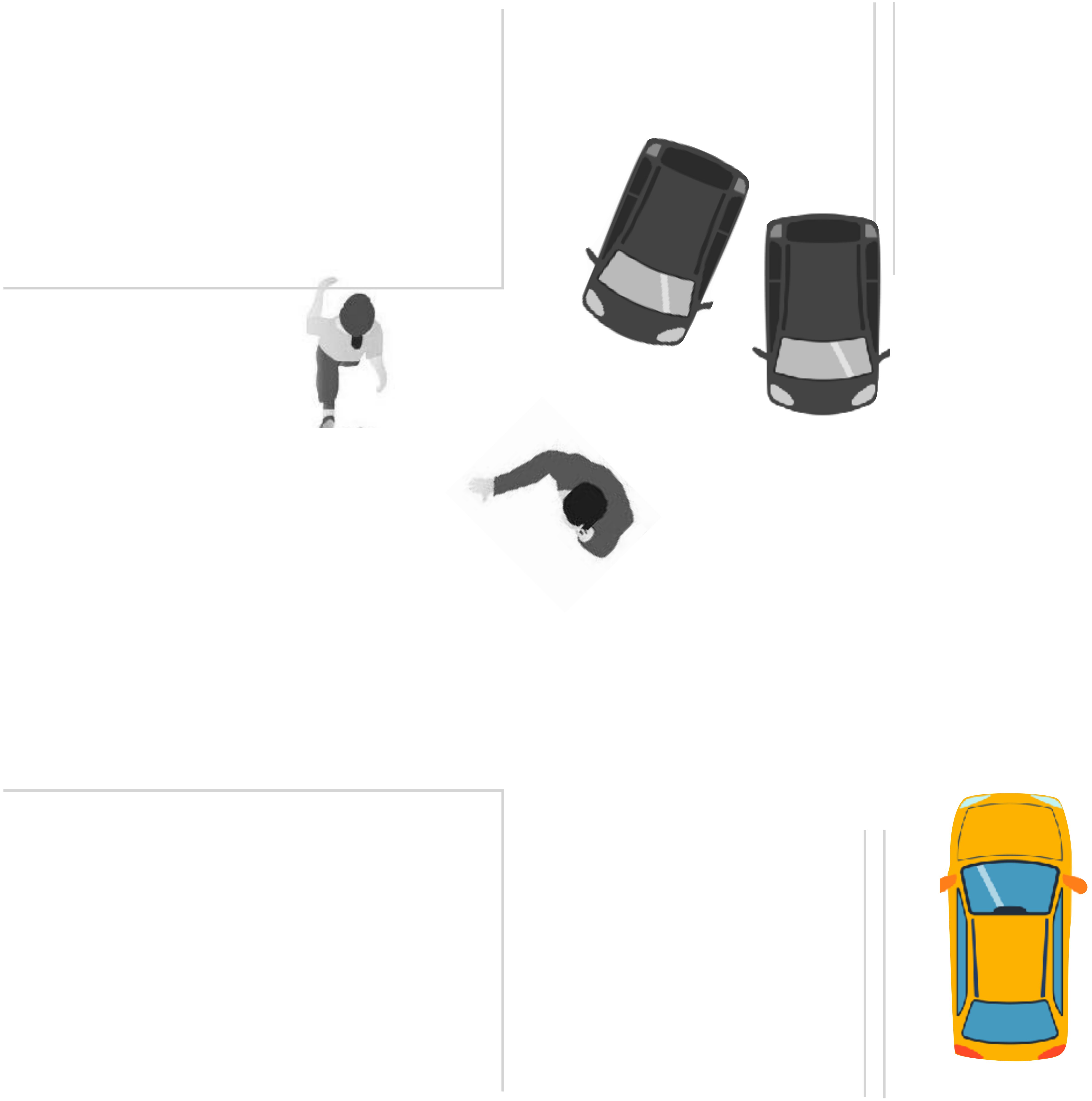
R Yields to A
R Yields to B
C Yields to R

≡

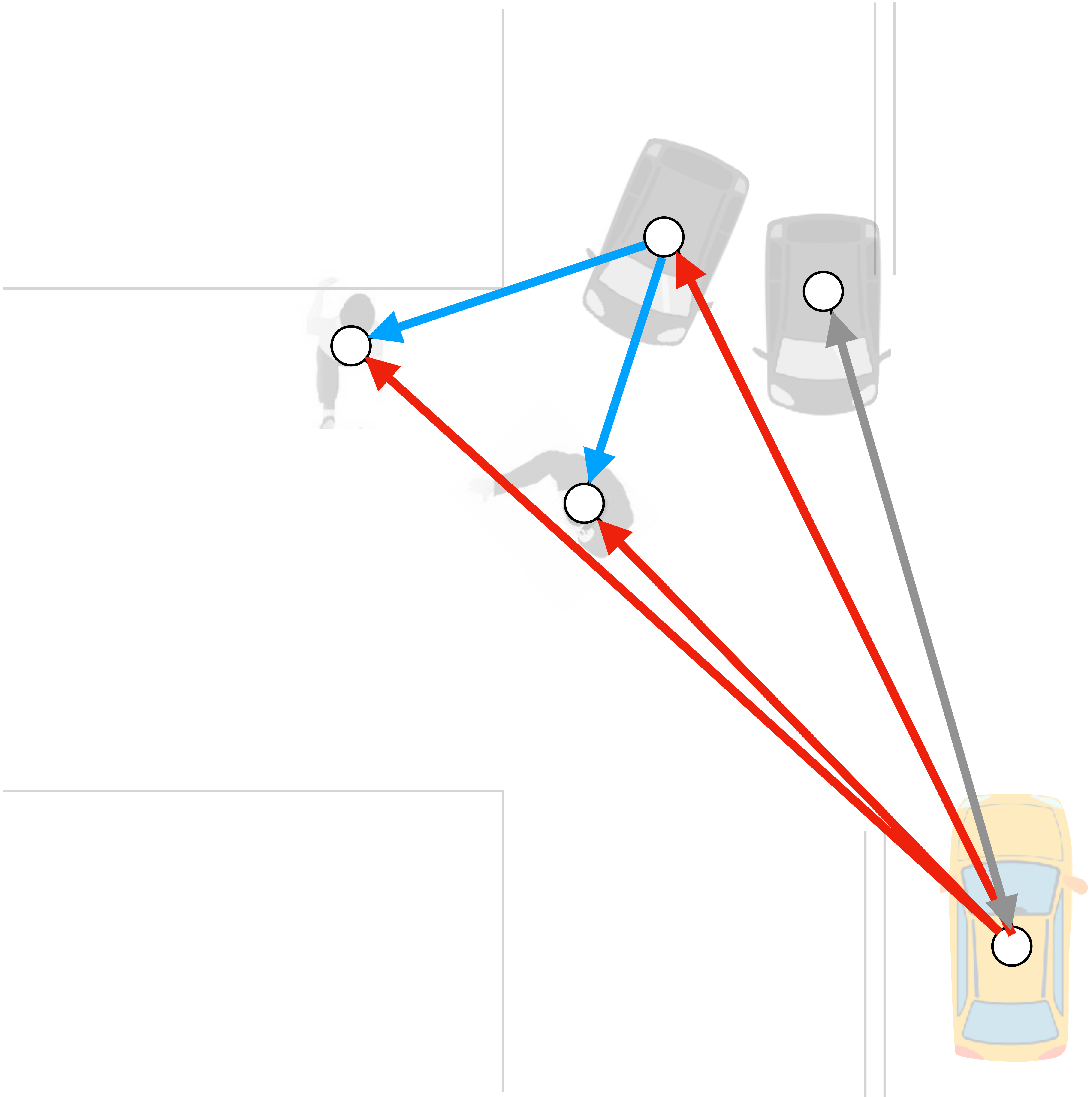


Instead of containing on plans, just condition on modes

Back to the scene



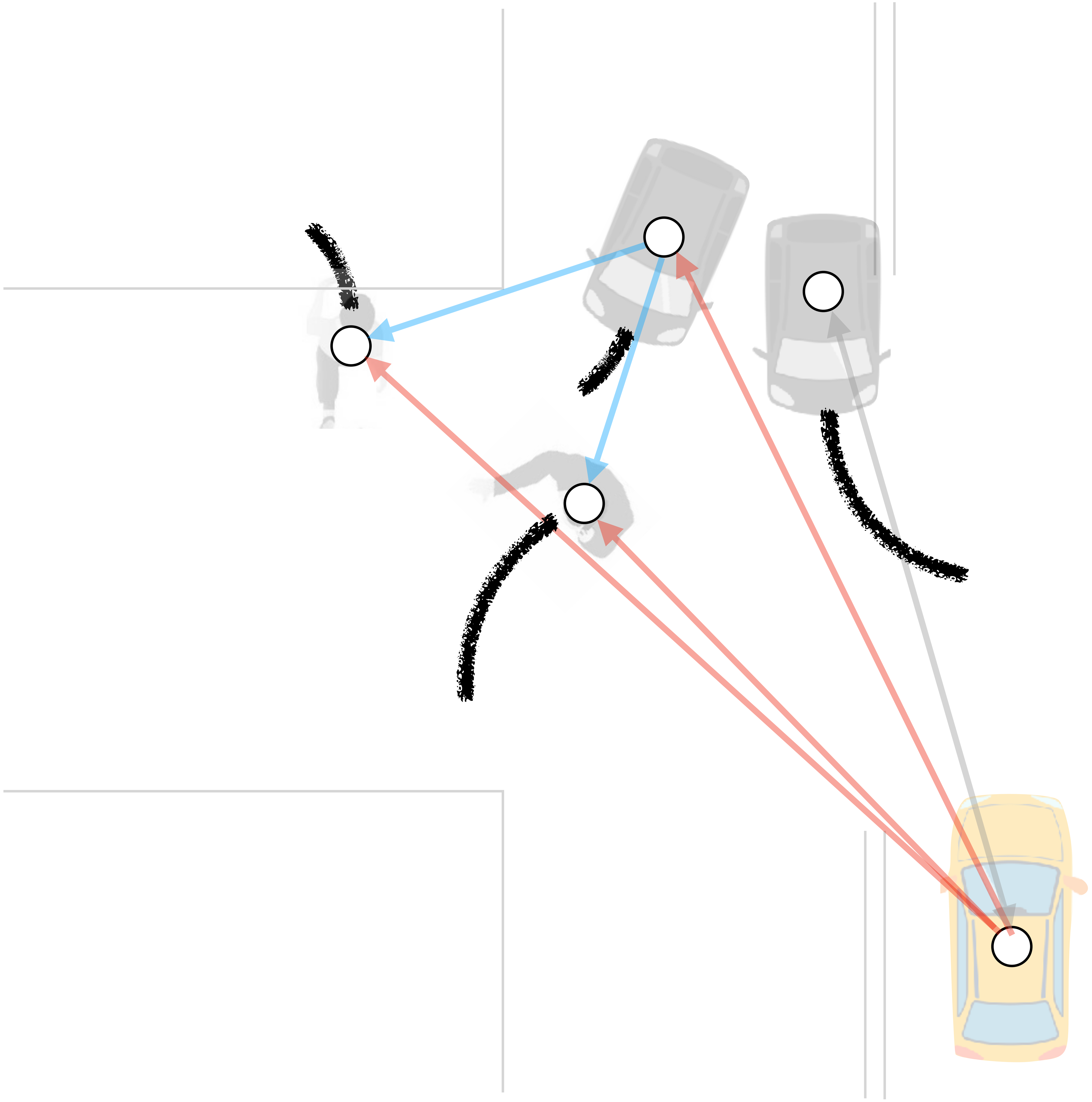
Message Passing on a Graph



Given a set of modes chosen by the robot

Infer what modes others are likely to choose

Message Passing on a Graph

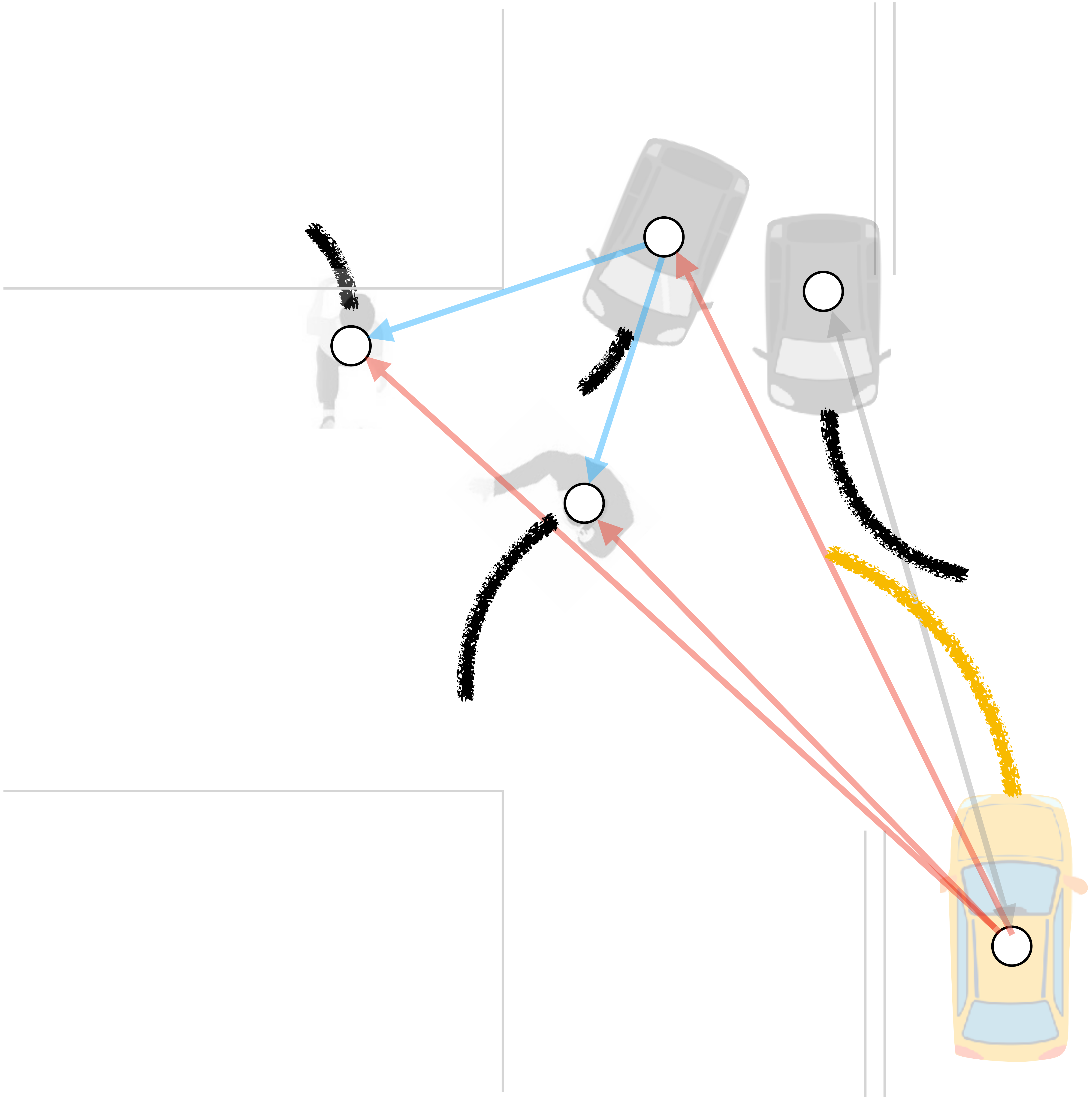


Given a set of modes chosen by the robot

Infer what modes others are likely to choose

Forecast actors given modes

Message Passing on a Graph

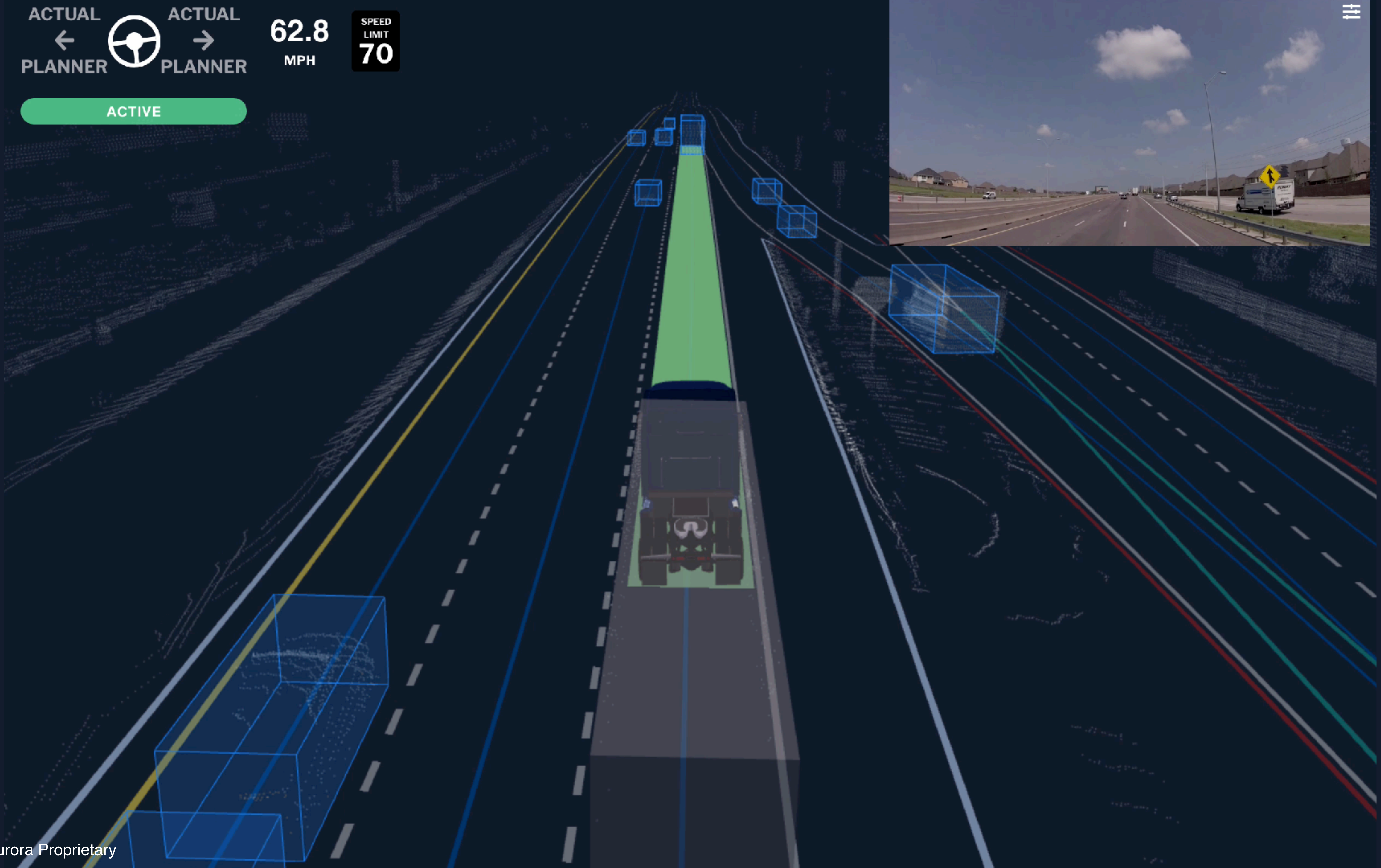


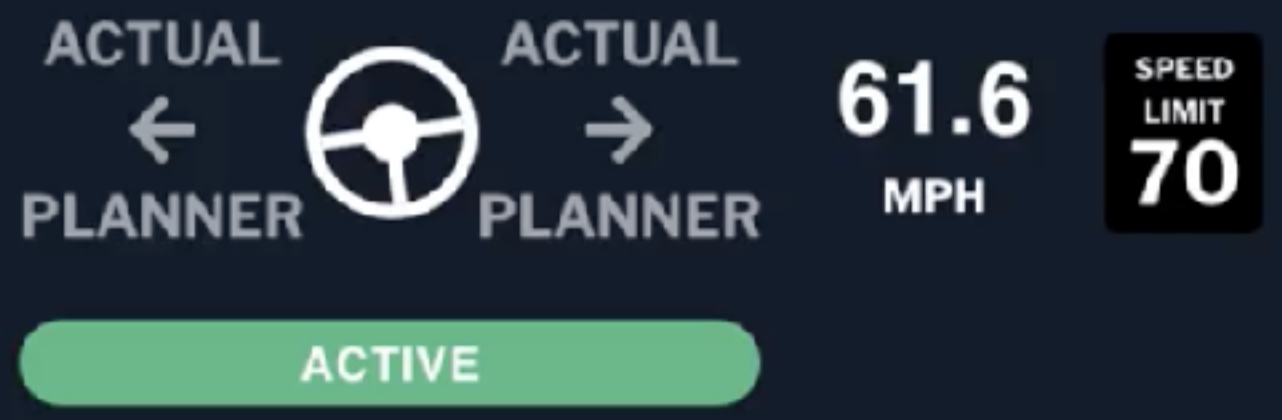
Given a set of modes chosen by the robot

Infer what modes others are likely to choose

Forecast actors given modes

Plan given forecast

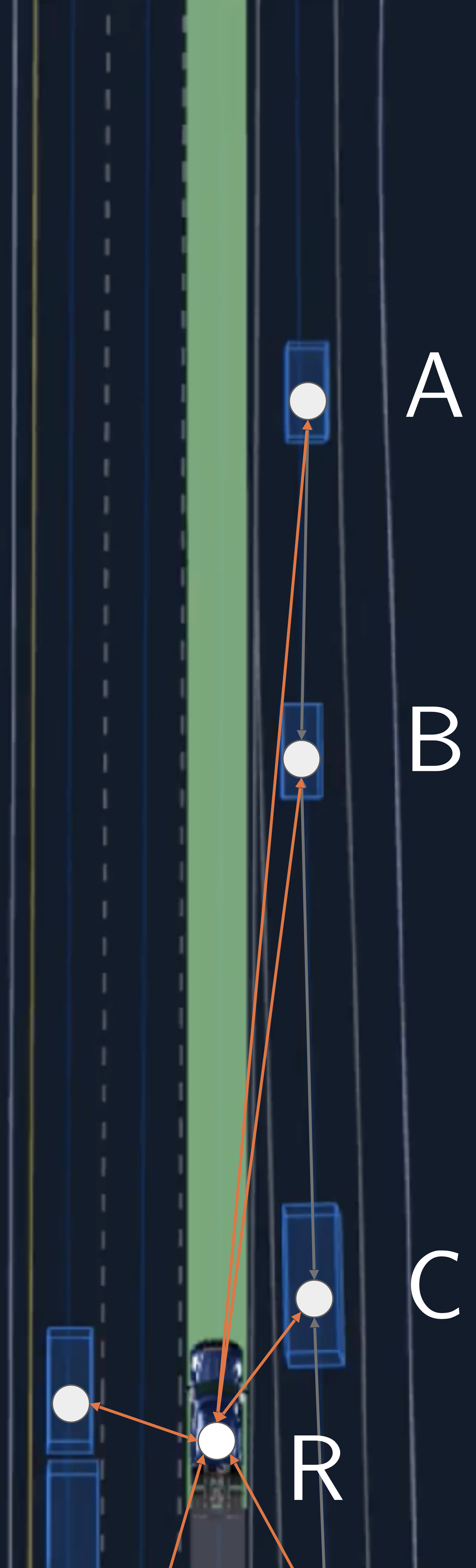

ACTUAL
PLANNER
ACTUAL
PLANNER
62.8
MPH
SPEED
LIMIT
70
ACTIVE

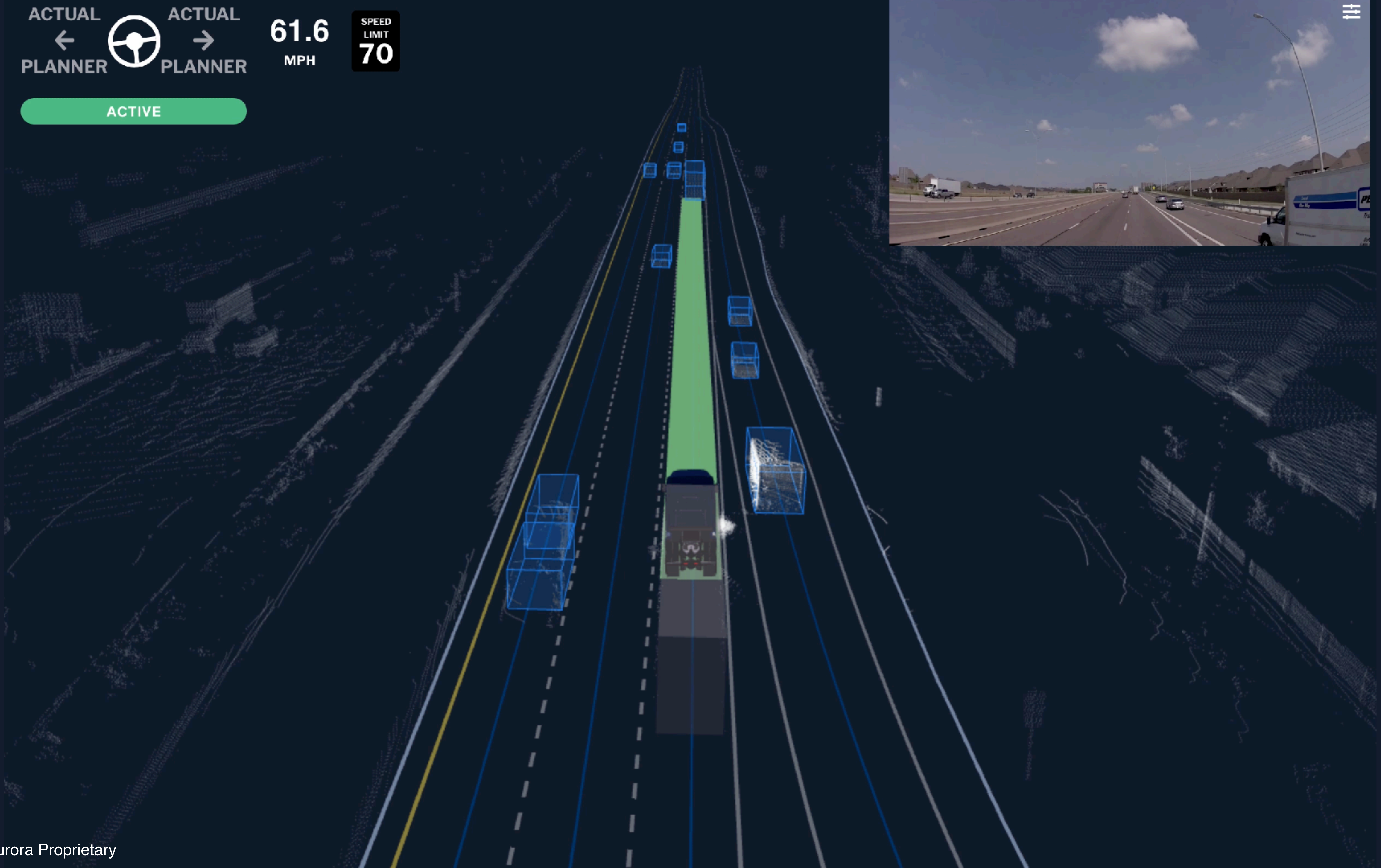


R Yields to A

R Yields to B

C Yields to R




ACTUAL
PLANNER
ACTUAL
PLANNER
61.6
MPH
SPEED
LIMIT
70
ACTIVE

Shaky foundations of forecasting

Are we using the right model?

Conditional forecasting

Are we collecting data correctly?

Are we using the right loss?

Shaky foundations of forecasting

Are we using the right model?

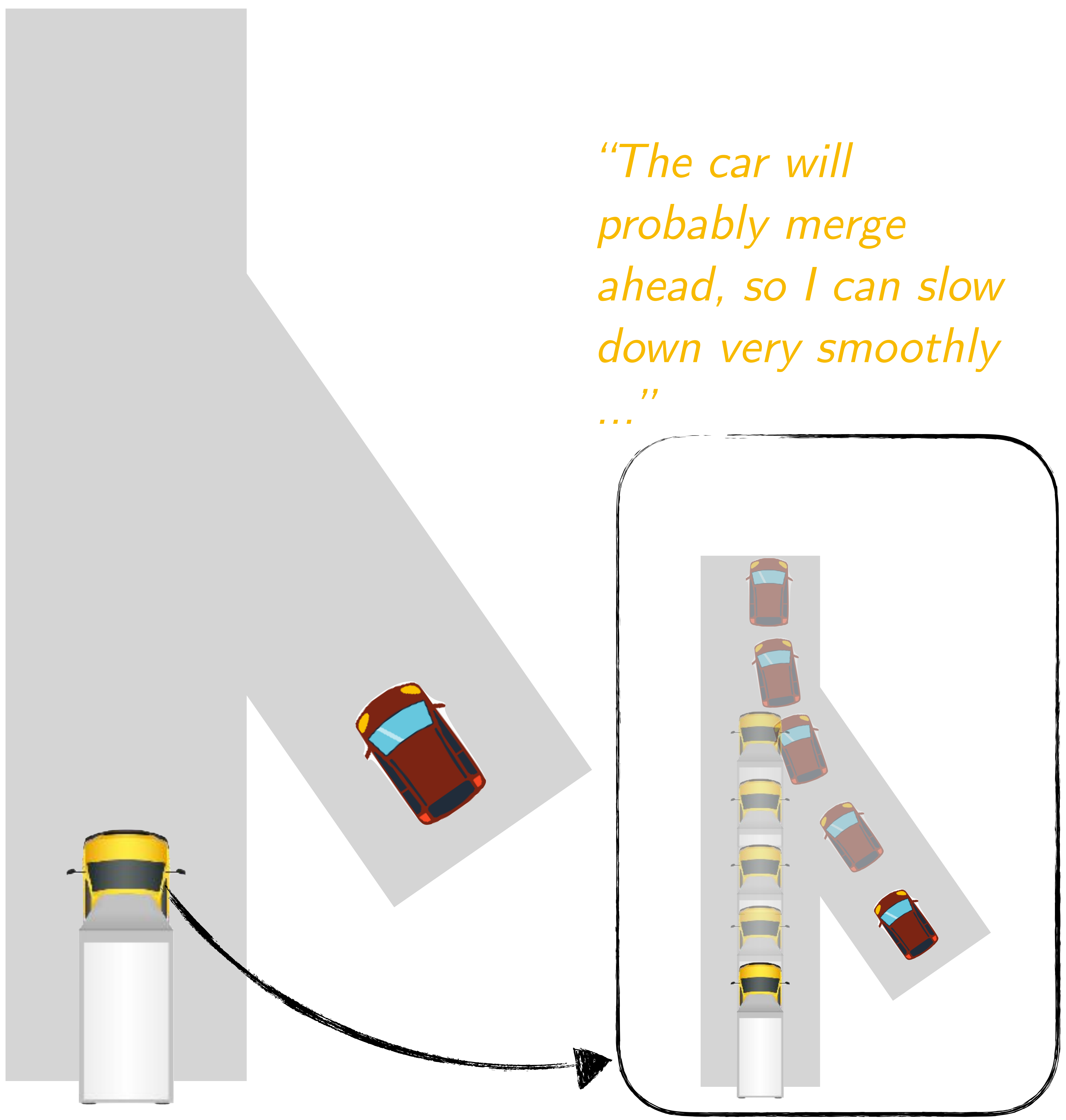
Conditional forecasting

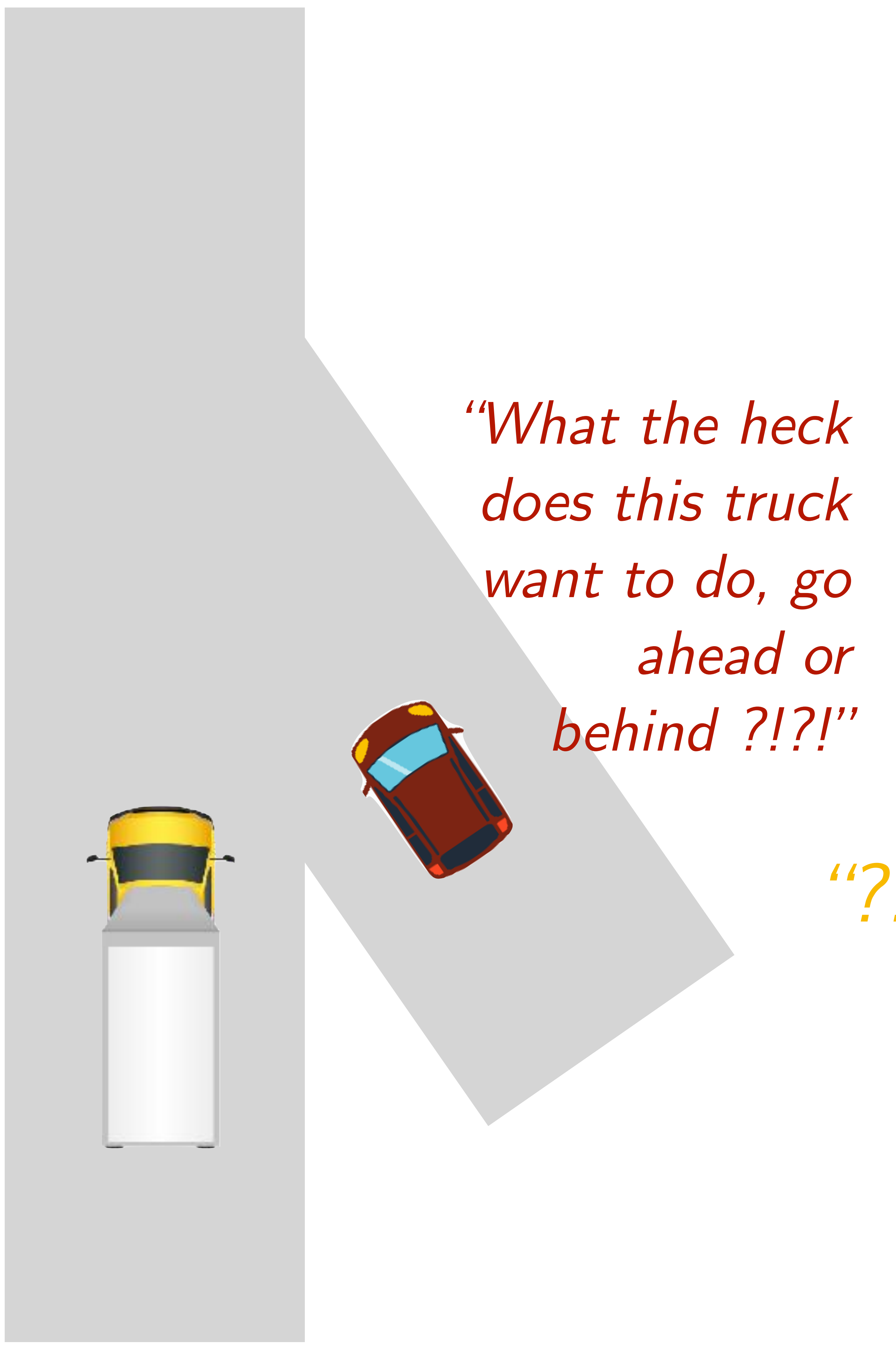
Are we collecting data correctly?

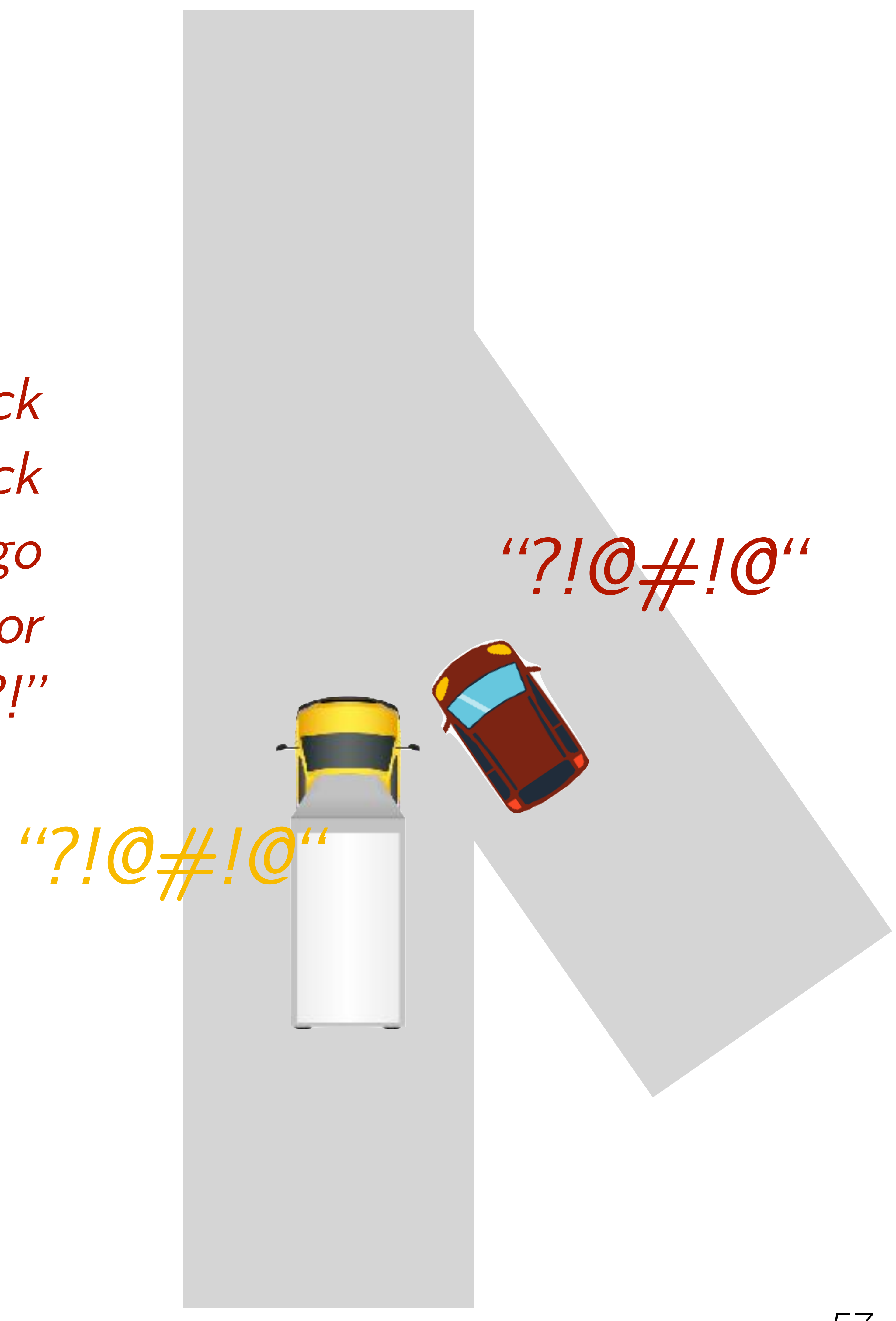
Are we using the right loss?

What happens when we deploy the forecast at test time?

What happens when we deploy model?

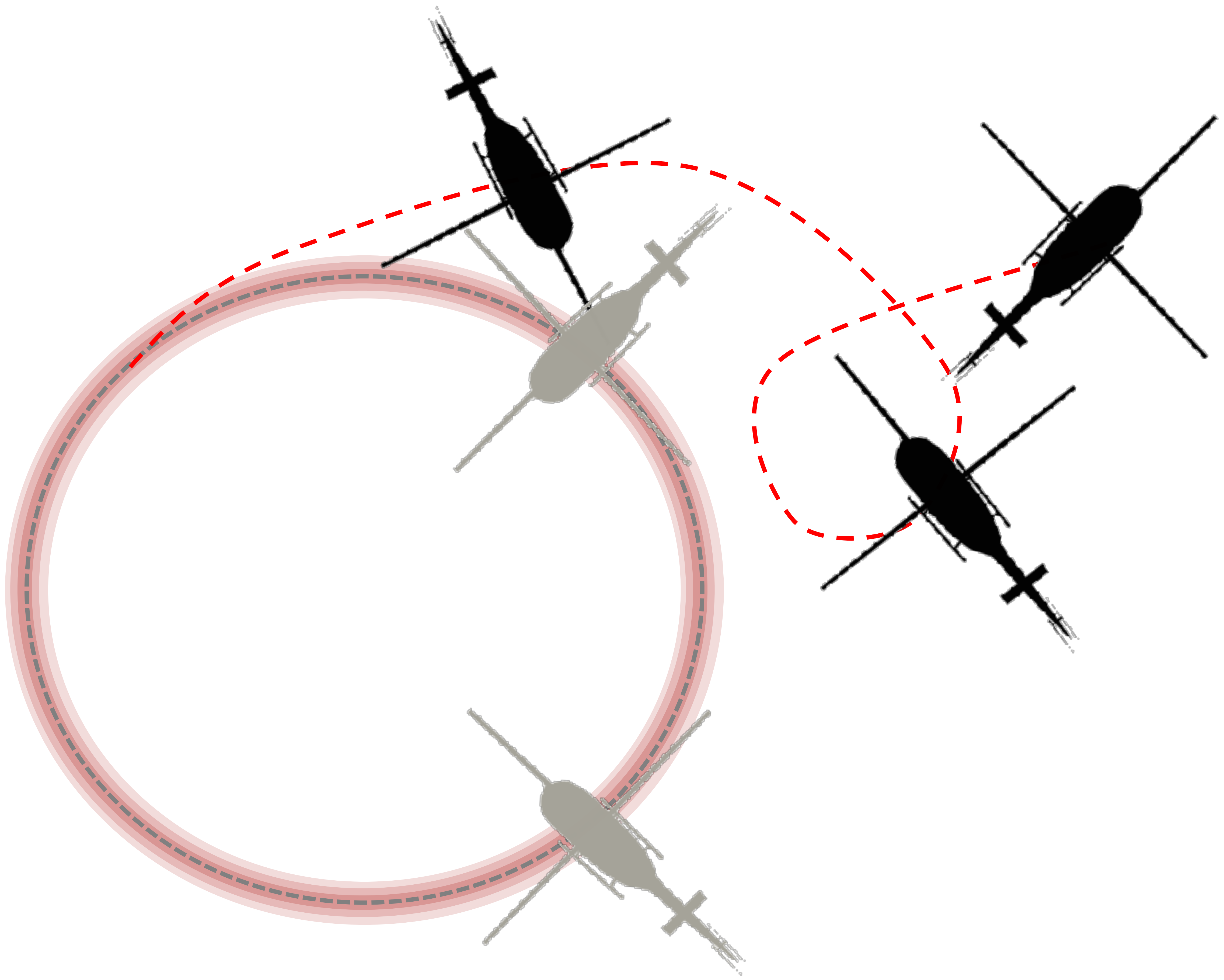




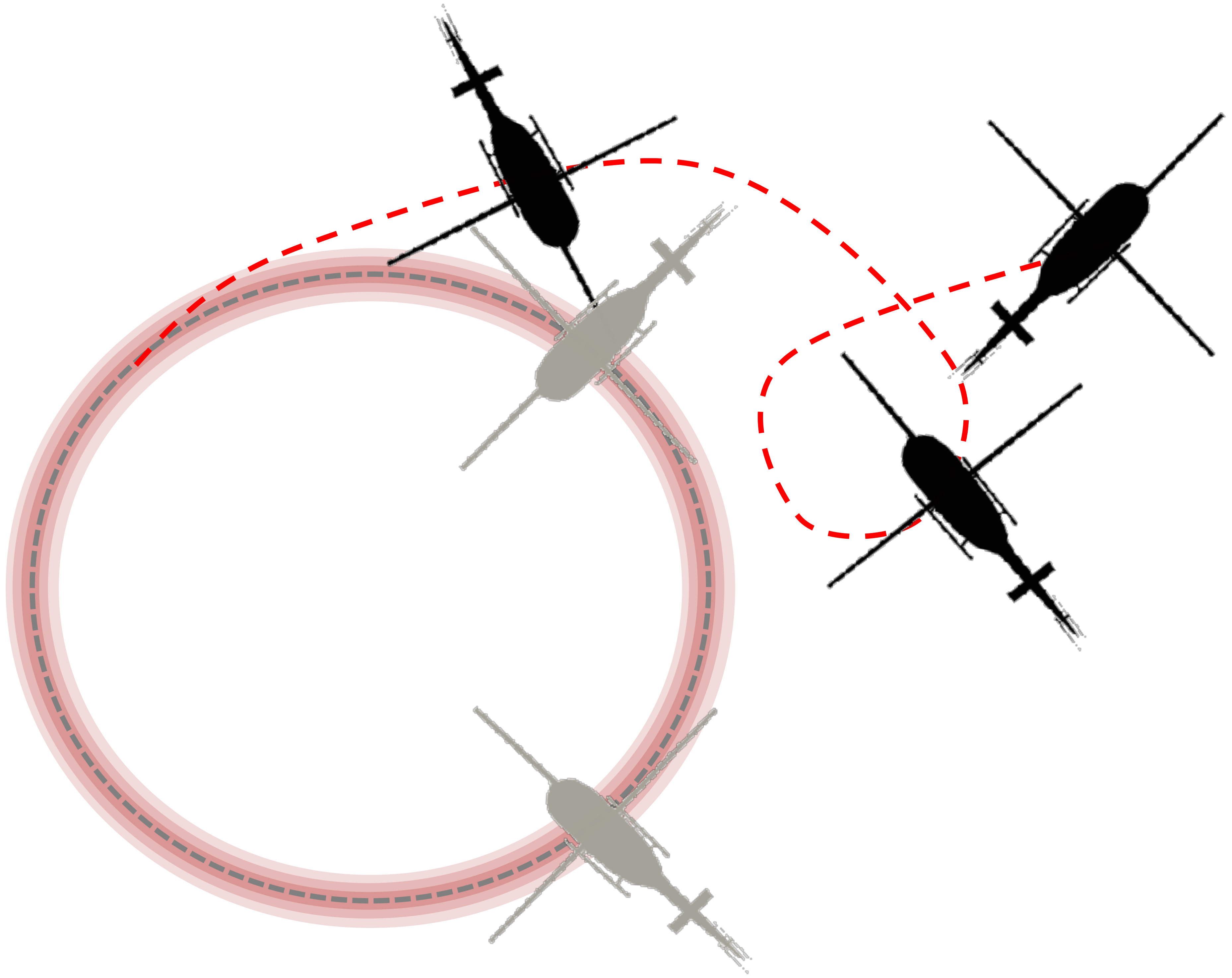


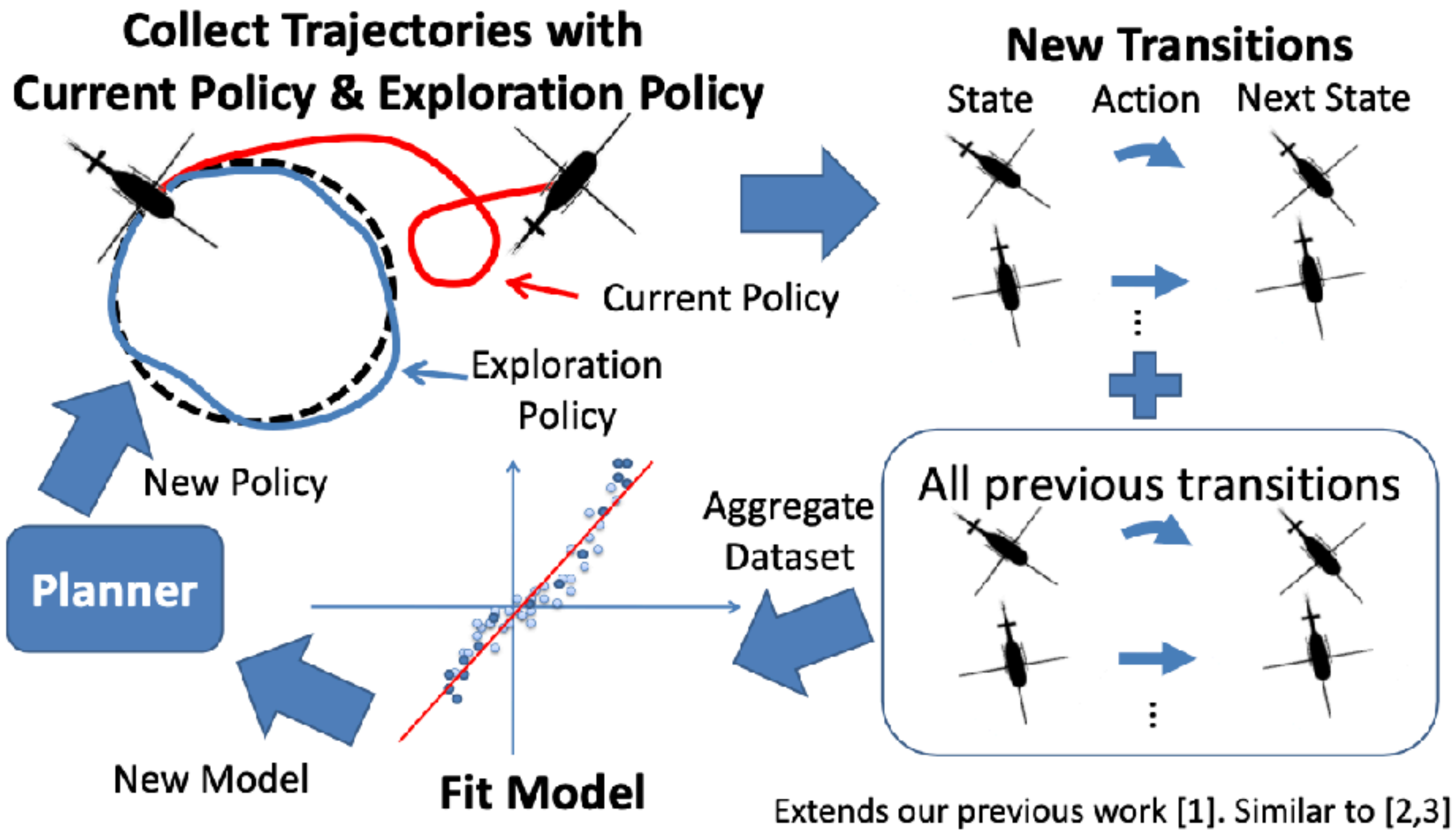
We gathered data when the human was driving the AV

We have seen this problem before!



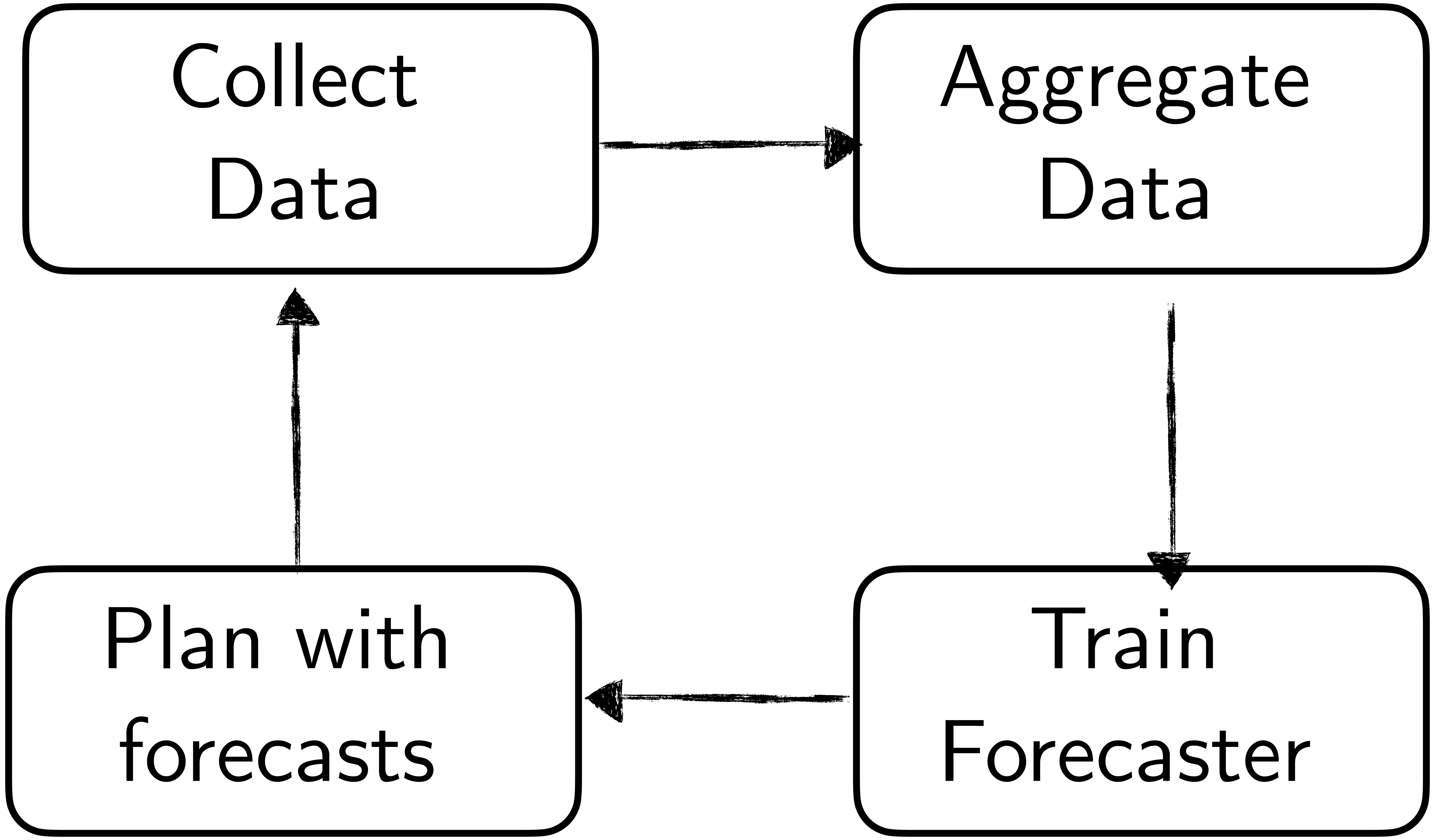
Solution: DAGGER for SysID





Extends our previous work [1]. Similar to [2,3]

DAGGER for Forecasting!



Shaky foundations of forecasting

Are we using the right model?

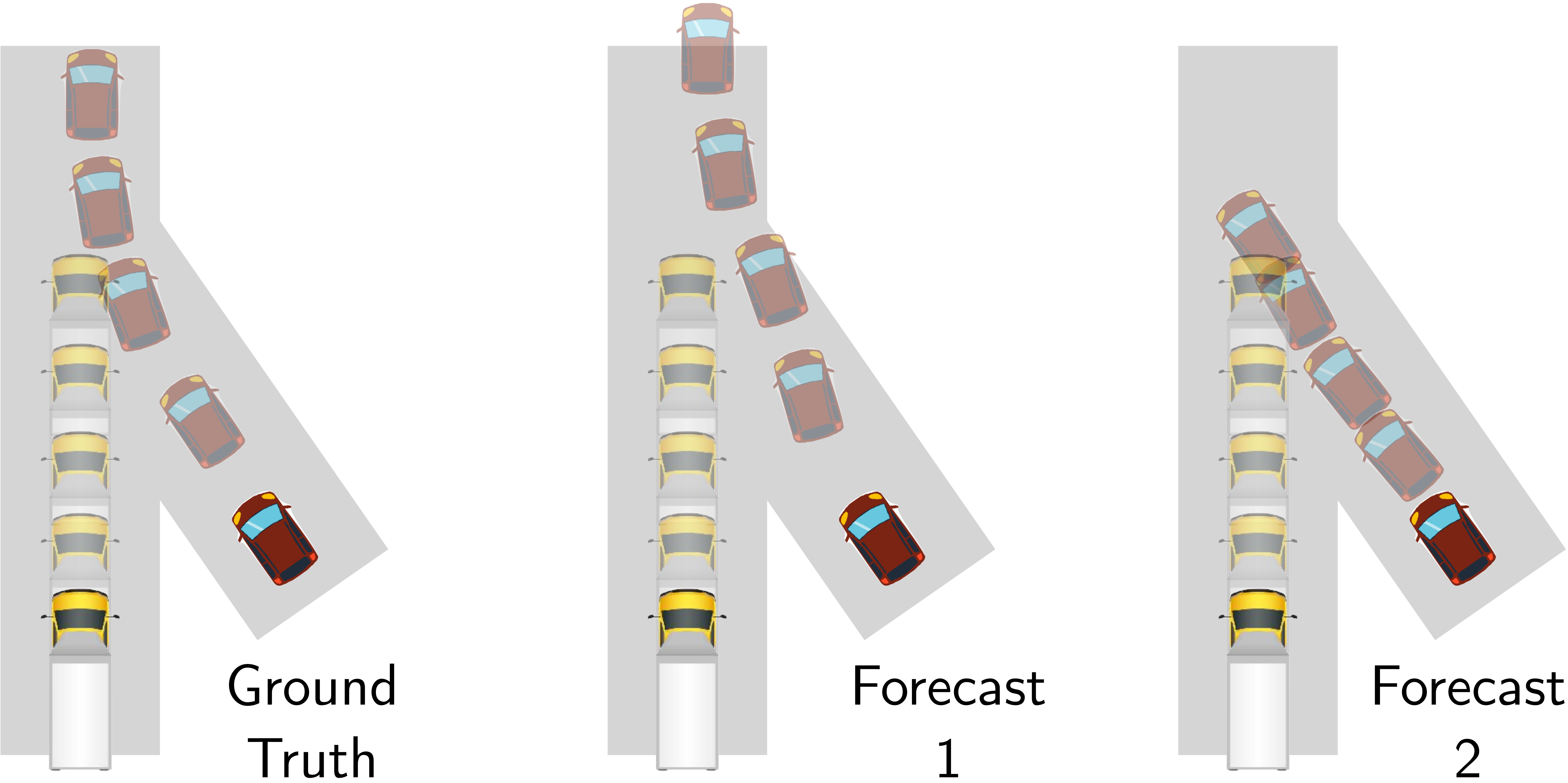
Conditional forecasting

Are we collecting data correctly?

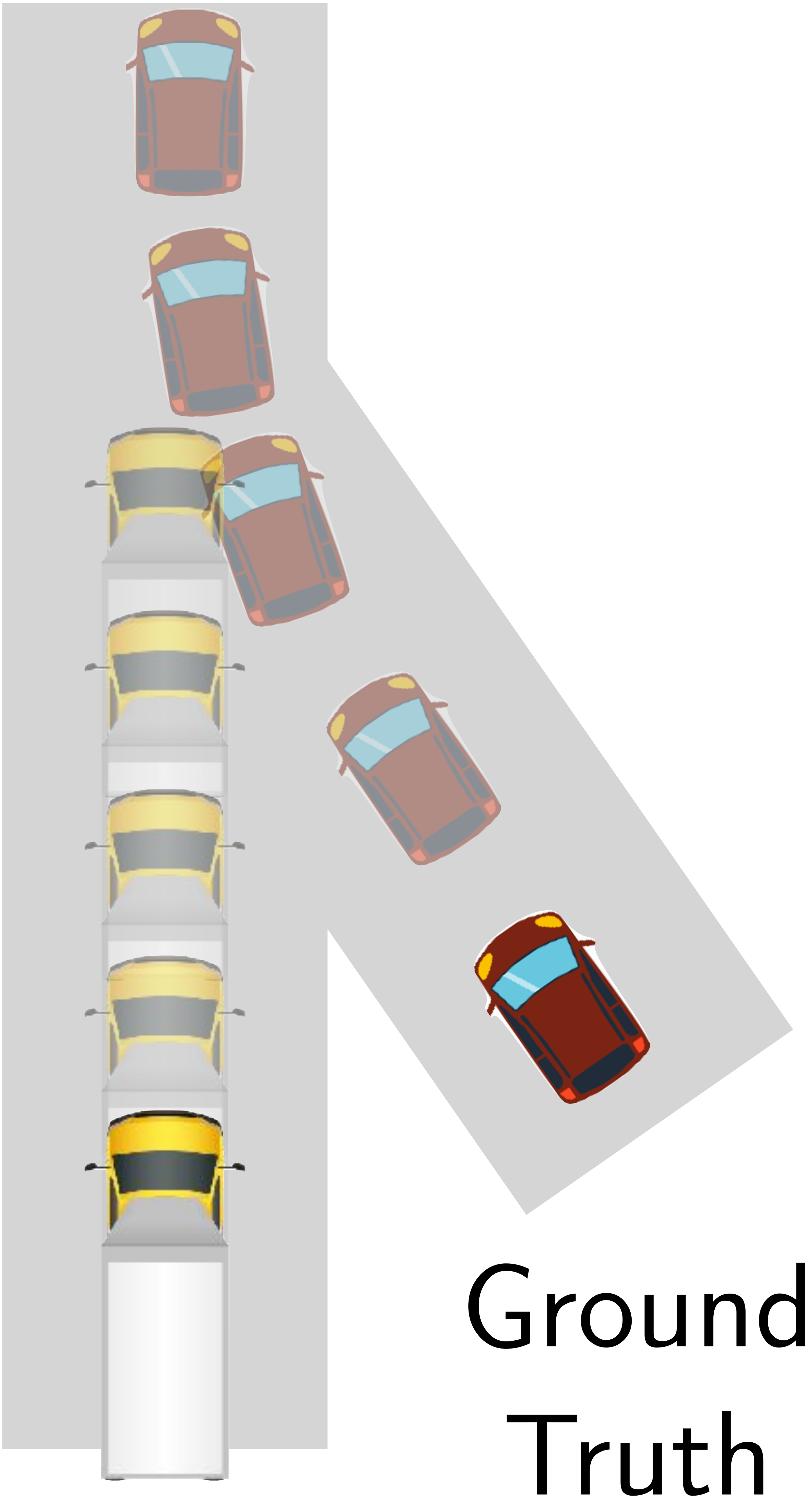
Interactively collect data

Are we using the right loss?

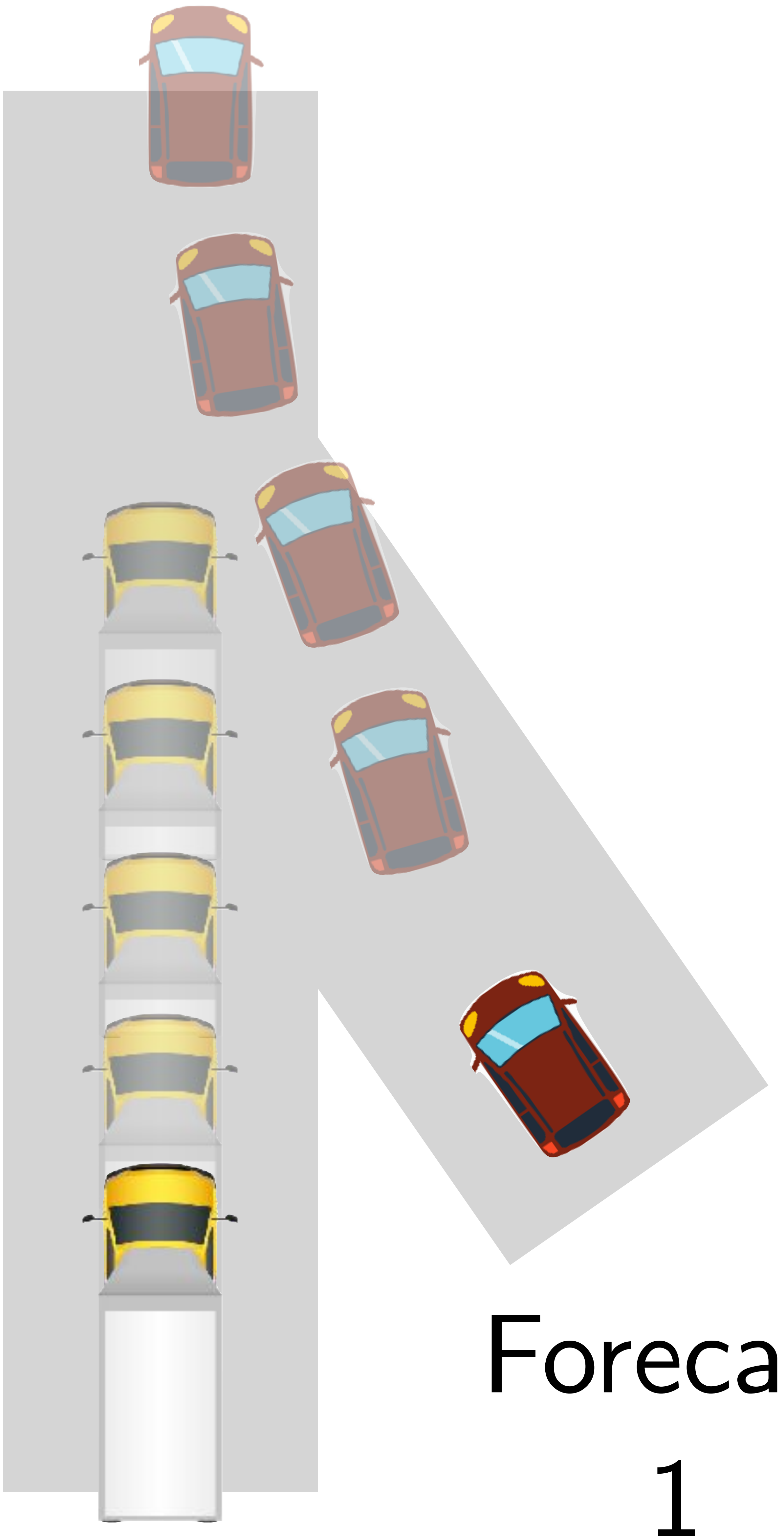
Take a look at the two potential forecasts



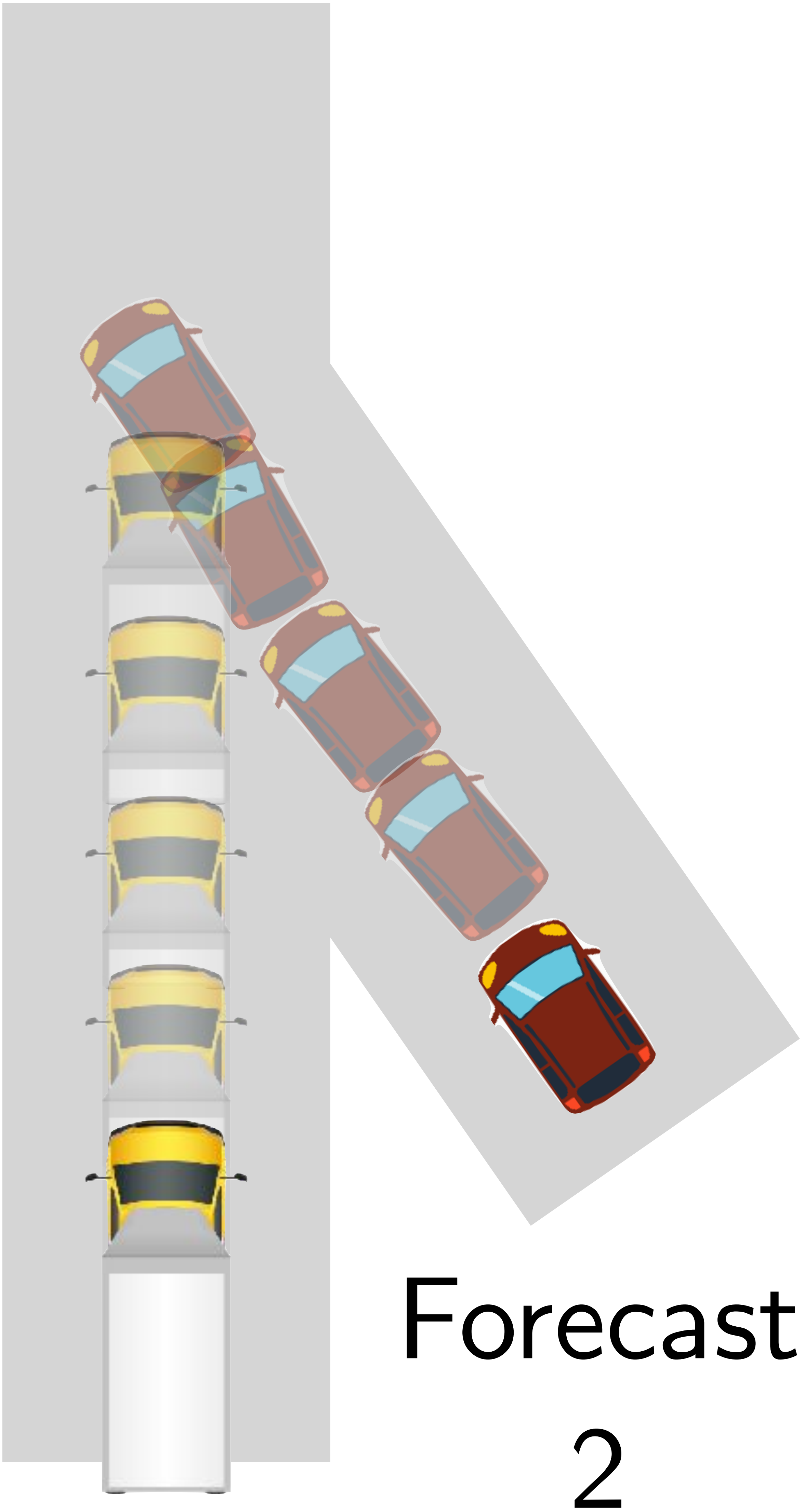
They both have the same L2 loss



Ground Truth

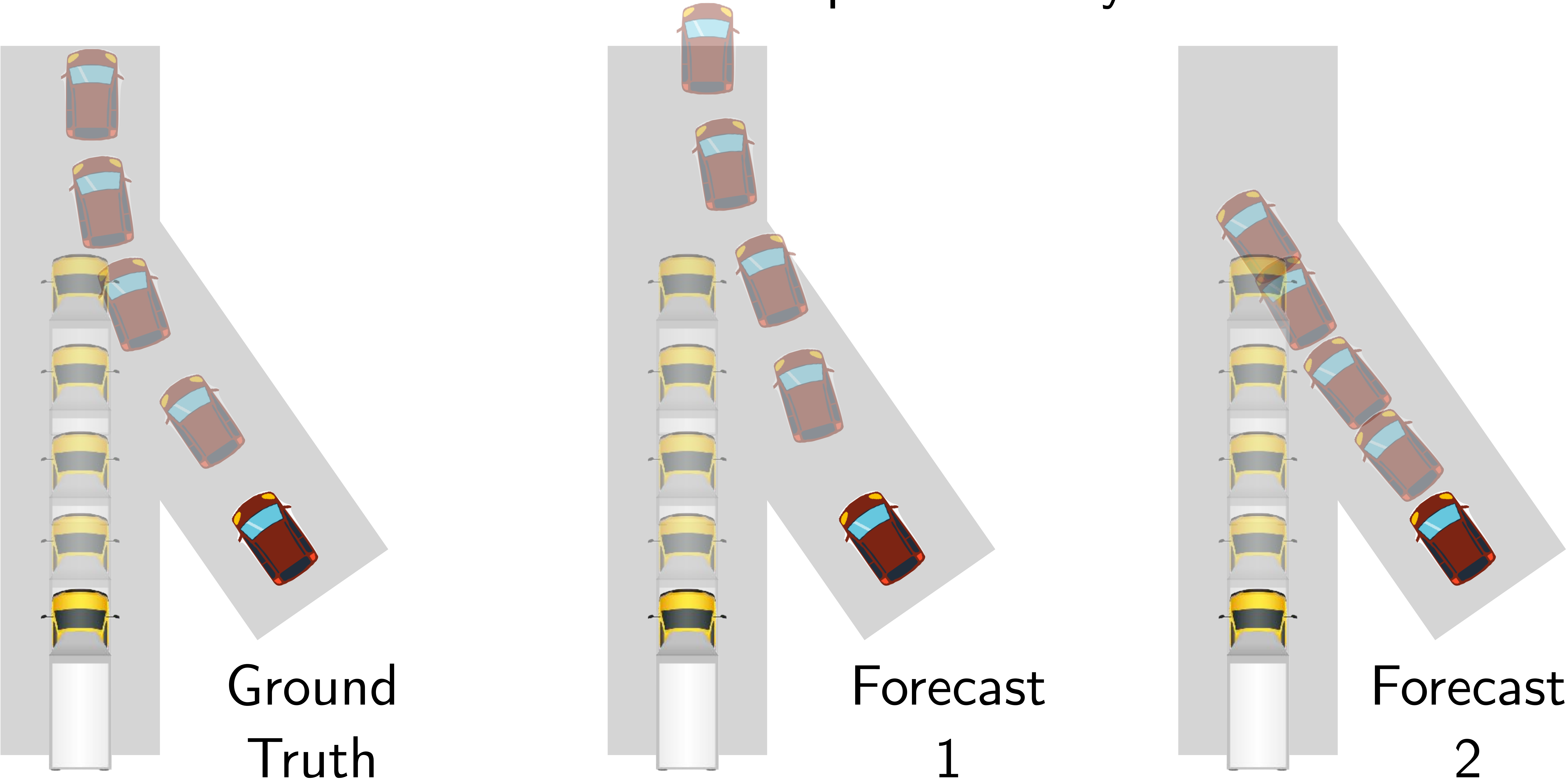


Forecast 1



Forecast 2

They both have the same L2 loss.
Which one do we prefer? Why?



Shaky foundations of forecasting

Are we using the right model?

Conditional forecasting

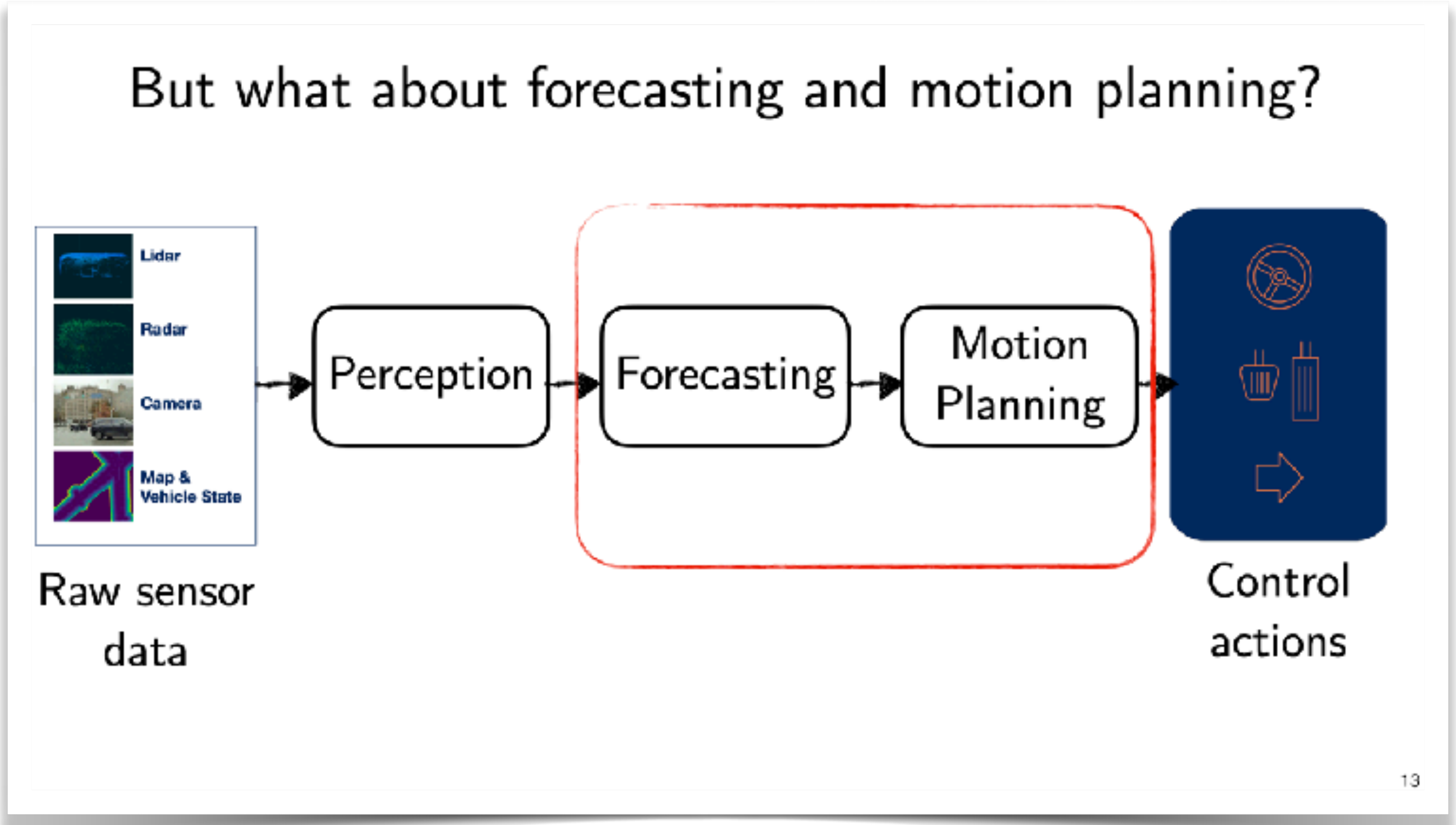
Are we collecting data correctly?

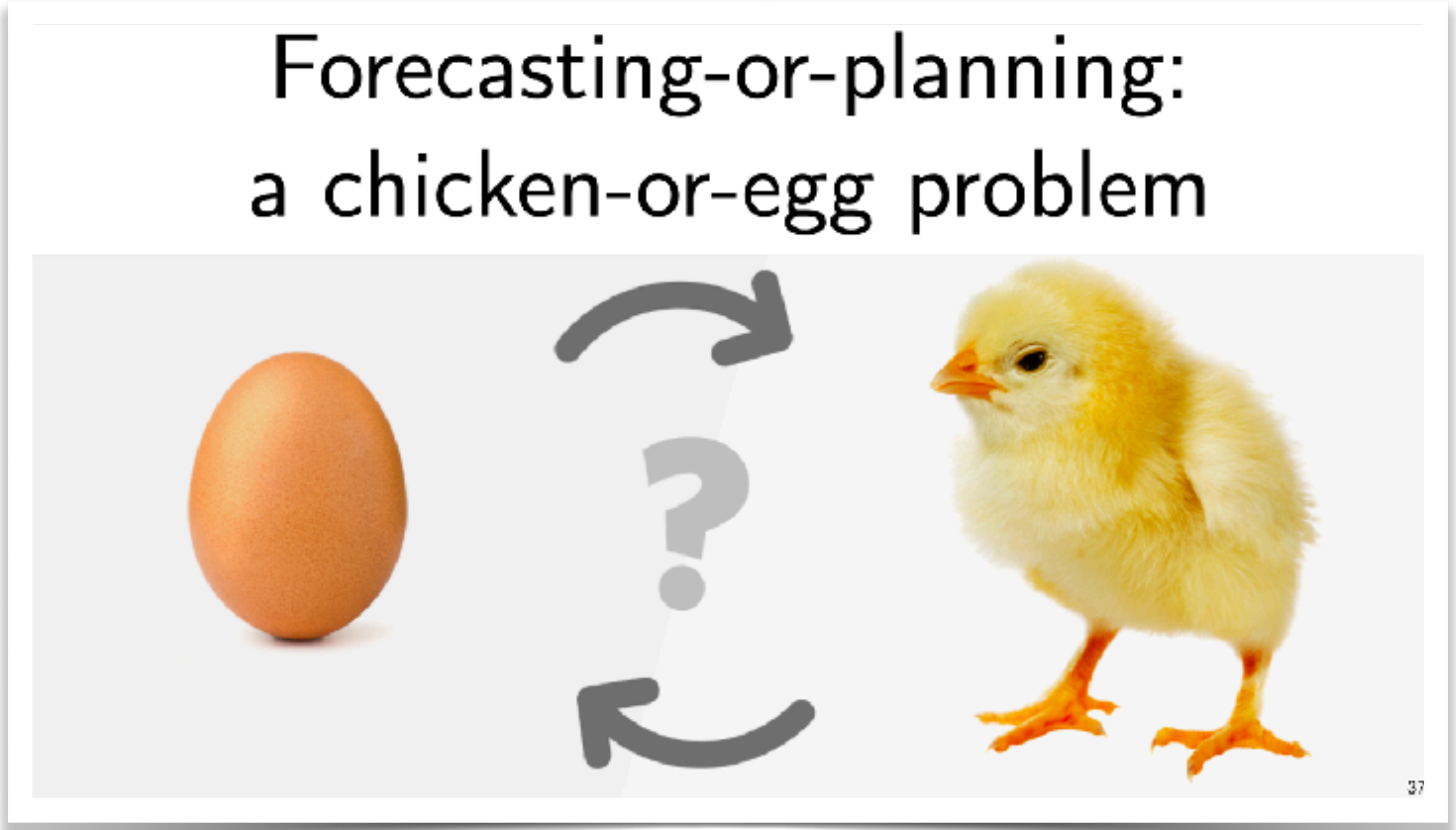
Interactively collect data

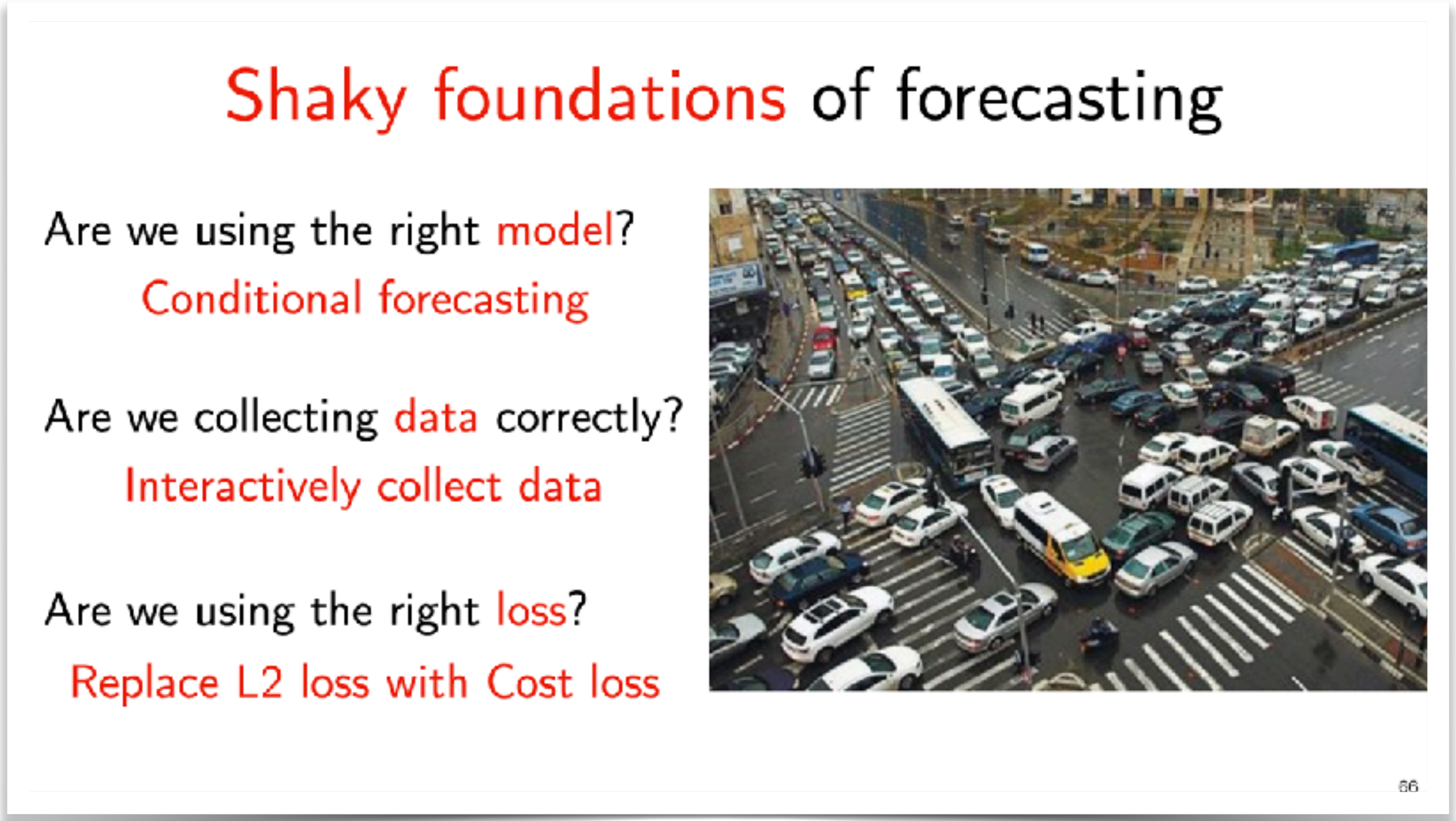
Are we using the right loss?

Replace L2 loss with Cost loss

tl;dr


But what about forecasting and motion planning?
Lidar
Radar
Camera
Map & Vehicle State
Raw sensor data
Perception
Forecasting
Motion Planning
Control actions
13


Forecasting-or-planning:
a chicken-or-egg problem
37


Shaky foundations of forecasting
Are we using the right model?
Conditional forecasting
Are we collecting data correctly?
Interactively collect data
Are we using the right loss?
Replace L2 loss with Cost loss
66