Cyber-Physical Systems Challenges: Vehicle Automation and Data Acquisition, Part 1

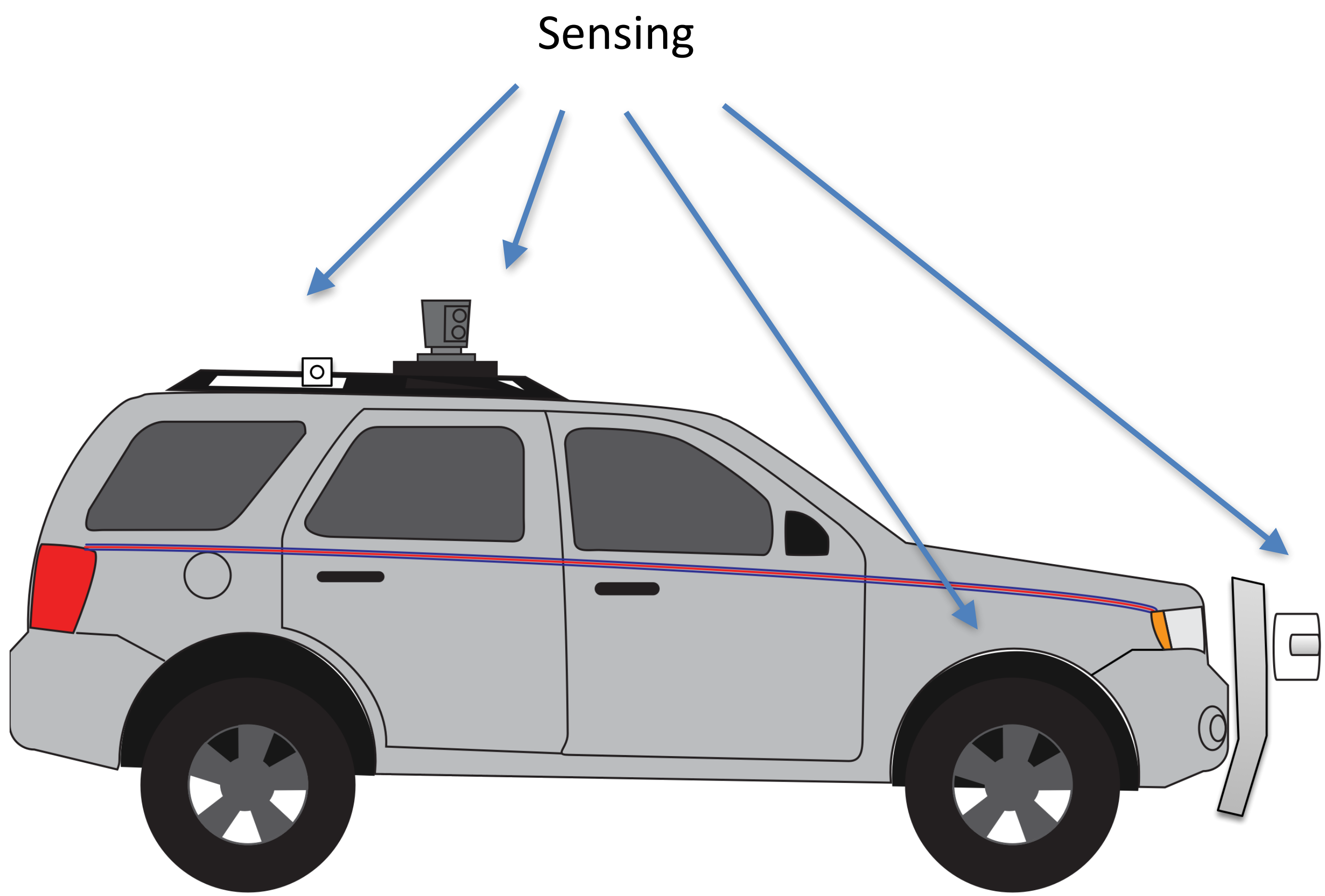
Jonathan Sprinkle, PhD
Litton Industries John M. Leonis
Distinguished Associate Professor
Interim Director, Transportation Research Institute
University of Arizona

With help from:
Rahul Bhadani, Matt Bunting, Nathalie Risso

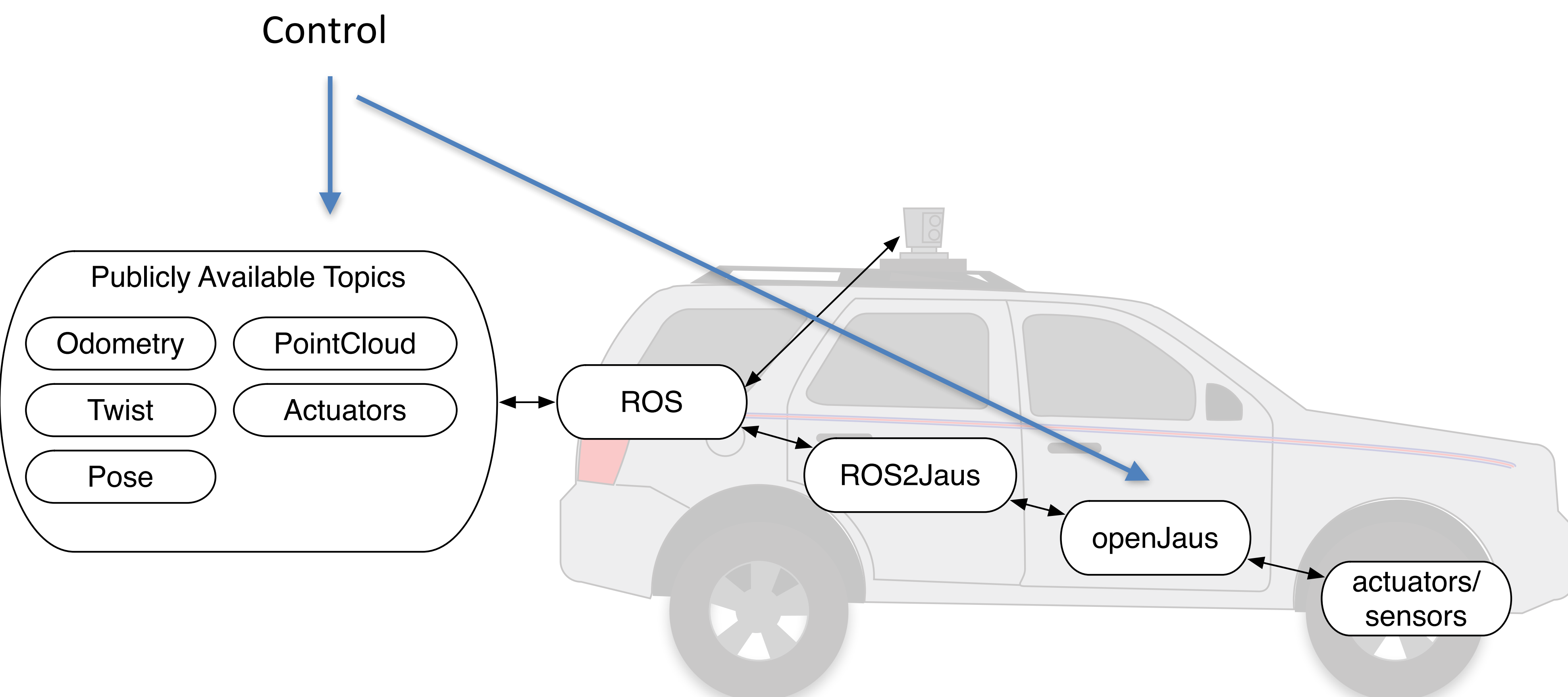
feat: models prepared for collaborations with Maria Laura Delle Monache, Benedetto Piccoli, Benni Seibold, Rafael Stern, Dan Work, and many others!

THE UNIVERSITY OF ARIZONA

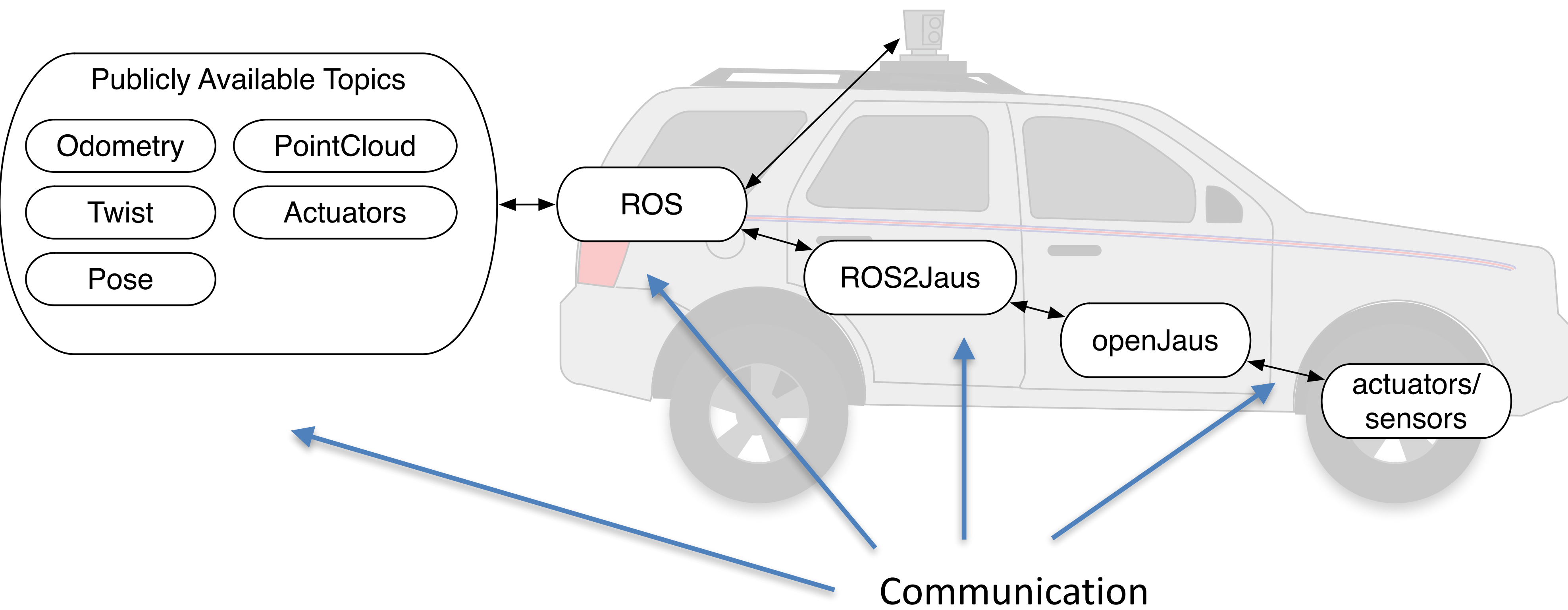
The vehicle automation is very broad…but there are a few fundamentals



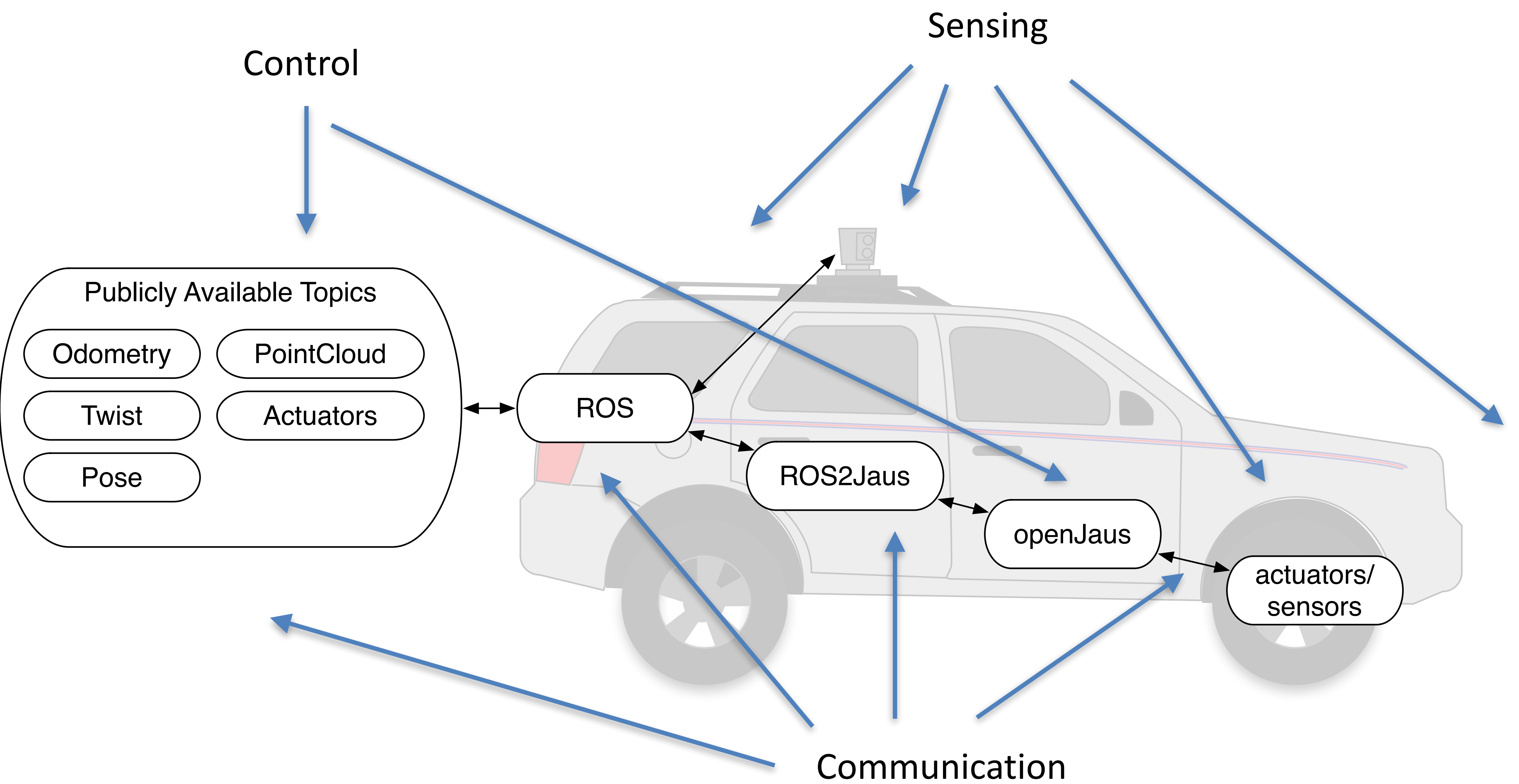
The vehicle automation is very broad…but there are a few fundamentals



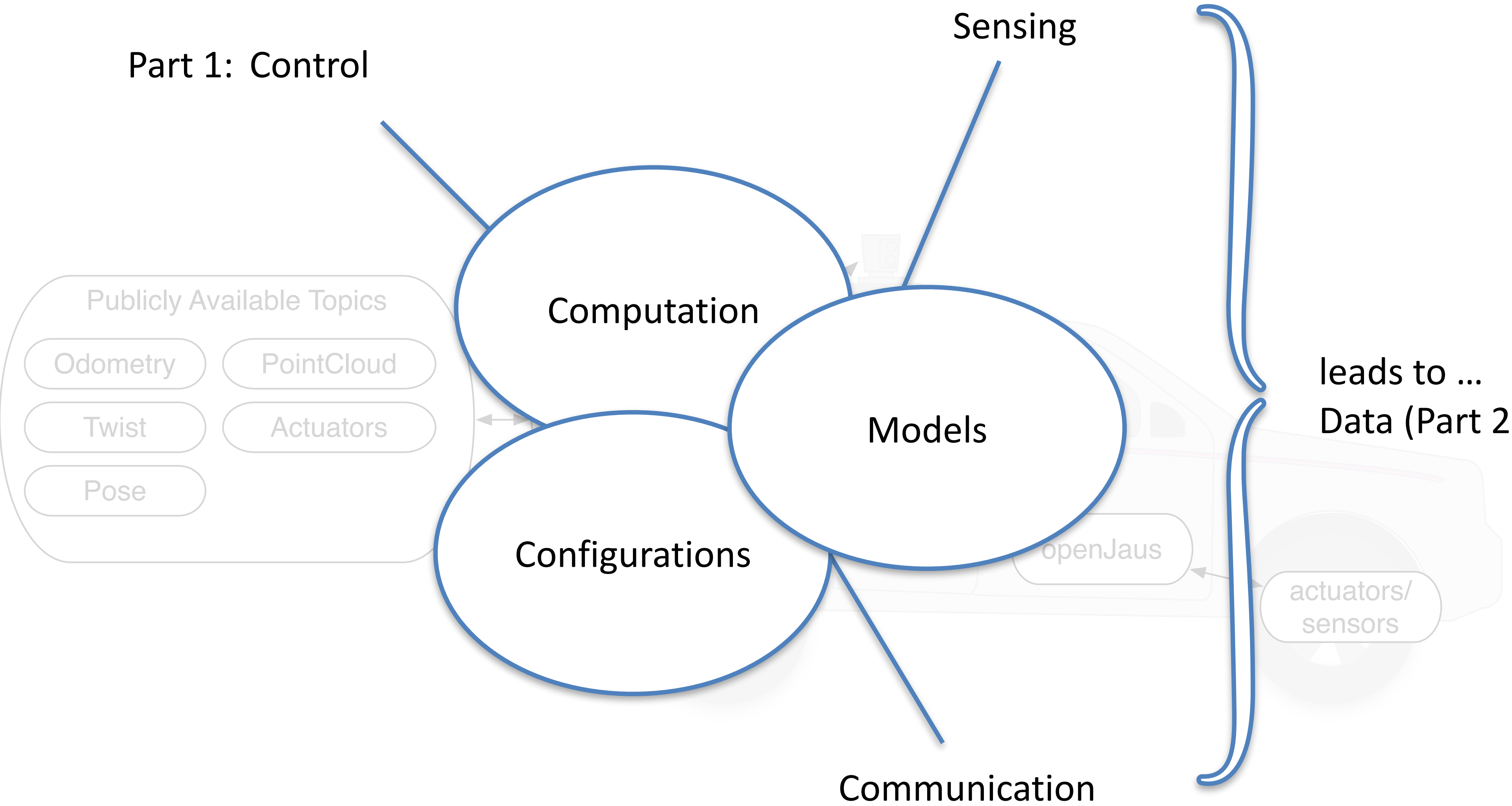
The vehicle automation is very broad…but there are a few fundamentals



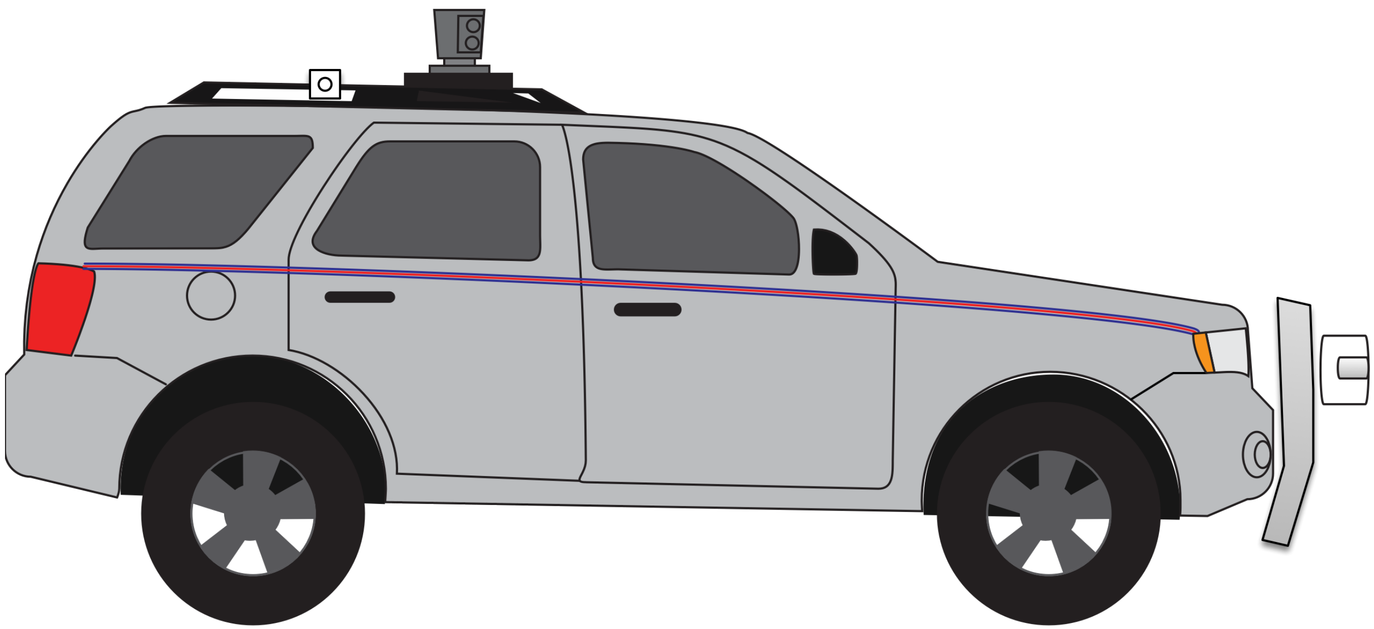
The vehicle automation is very broad…but there are a few fundamentals



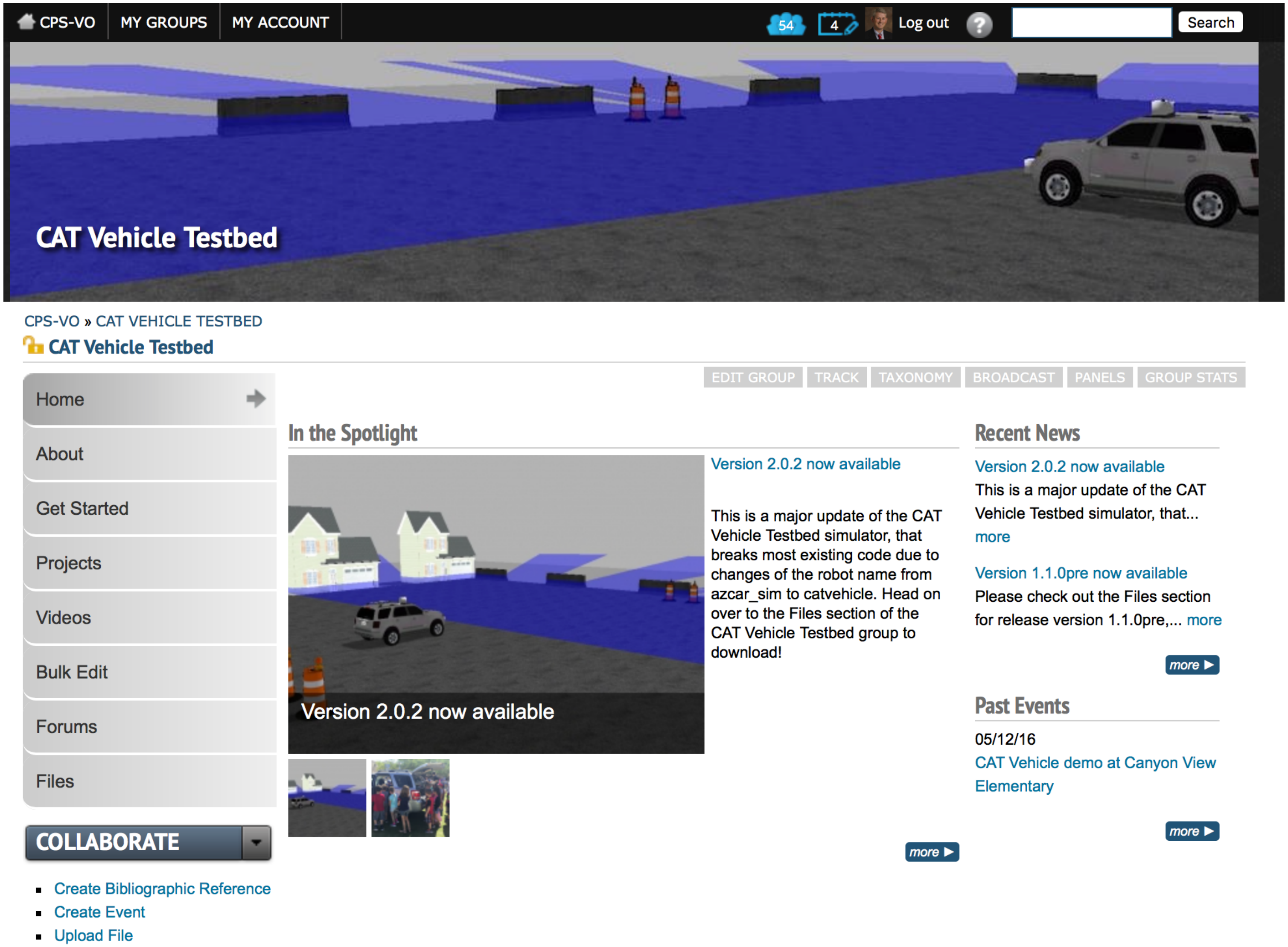
The vehicle automation is very broad…but there are a few fundamentals

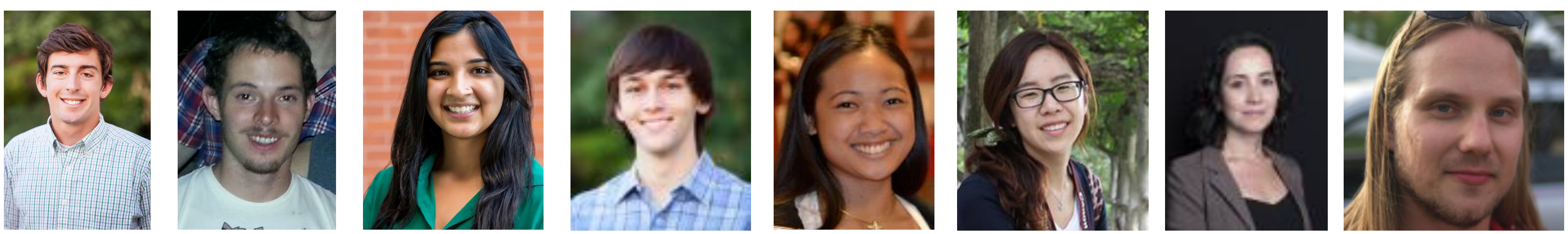


Impossible to cover all vehicles, so we will be primarily using my research testbed as an exemplar



The CAT Vehicle Testbed



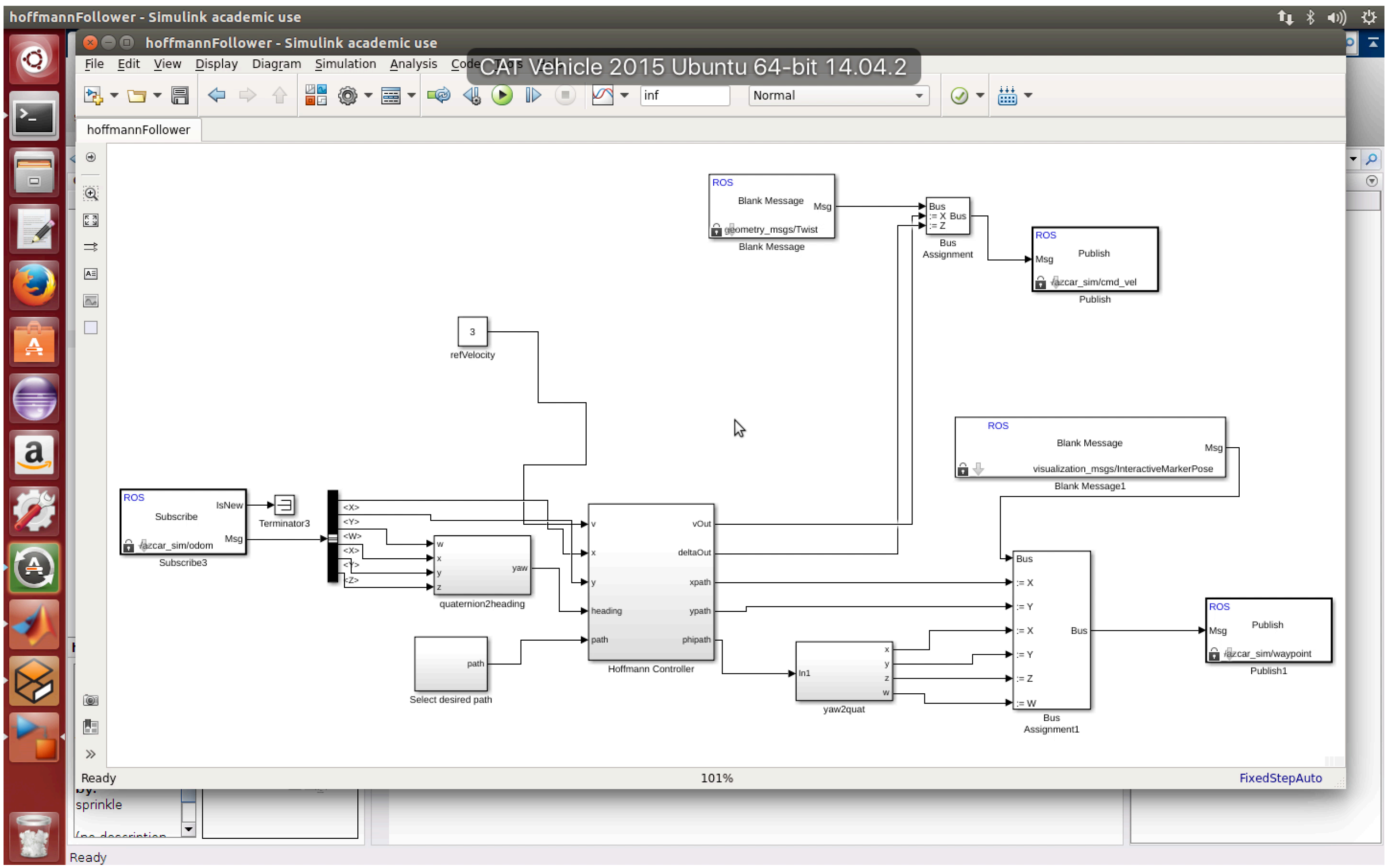


https://cps-vo.org/group/CATVehicleTestbed http://cps-vo.org/group/CATVehicleTestbed

CatVehicle Testbed

(BSD license)

- ROS access to system
- Gazebo simulator with
  - Ackermann steering
  - Lidar (Velodyne and SICK)
  - Realistic (but not "correct") masses and dynamics
- Published topics for odometry, path, laser scans, velocity, steering angles
- Example Simulink Robotics System Toolbox models distributed with the source
- Control inputs through velocity and steering angle (cmd_vel)

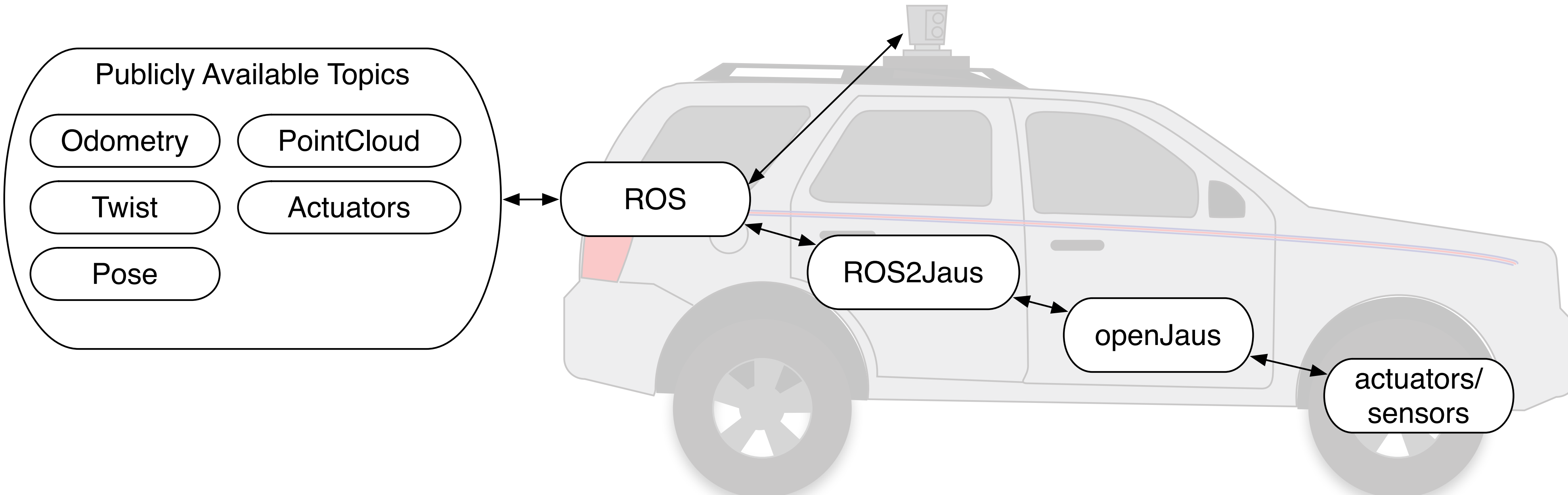


Jonathan Sprinkle,

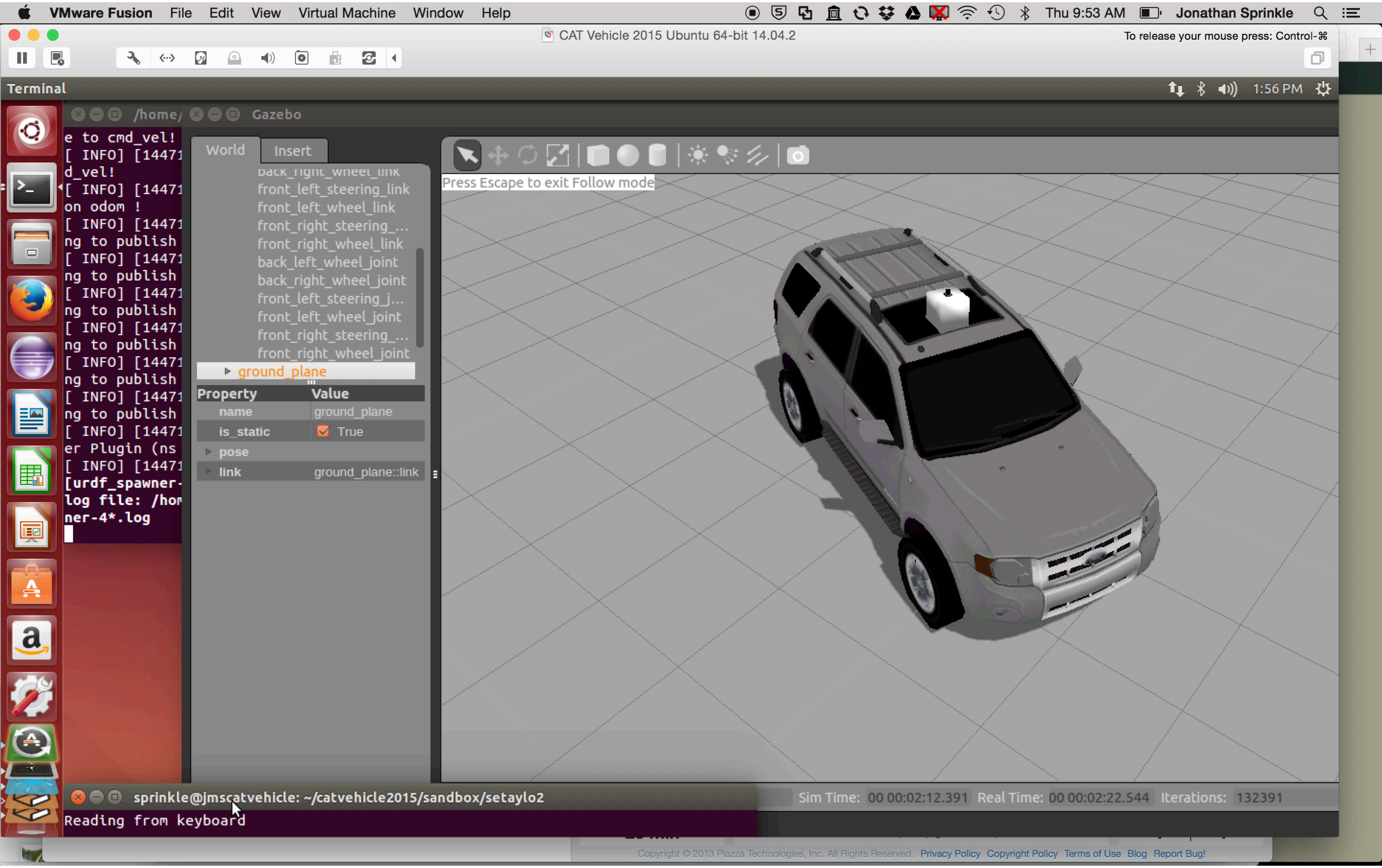
Over-sampled and Over-logged

- 2 MILSPEC machines with dedicated handling for
  - ROS whitepages
  - GPS/INS
- Logging of all potentially useful data to TB++ HDD arrays that rotate out old logs if not claimed
- Dedicated interaction to Velodyne sensor
- 12-18 V DC power supply with 8 output ports (all at same V_0)
- UPS for clean power while driving (uh, clean"ish" power)

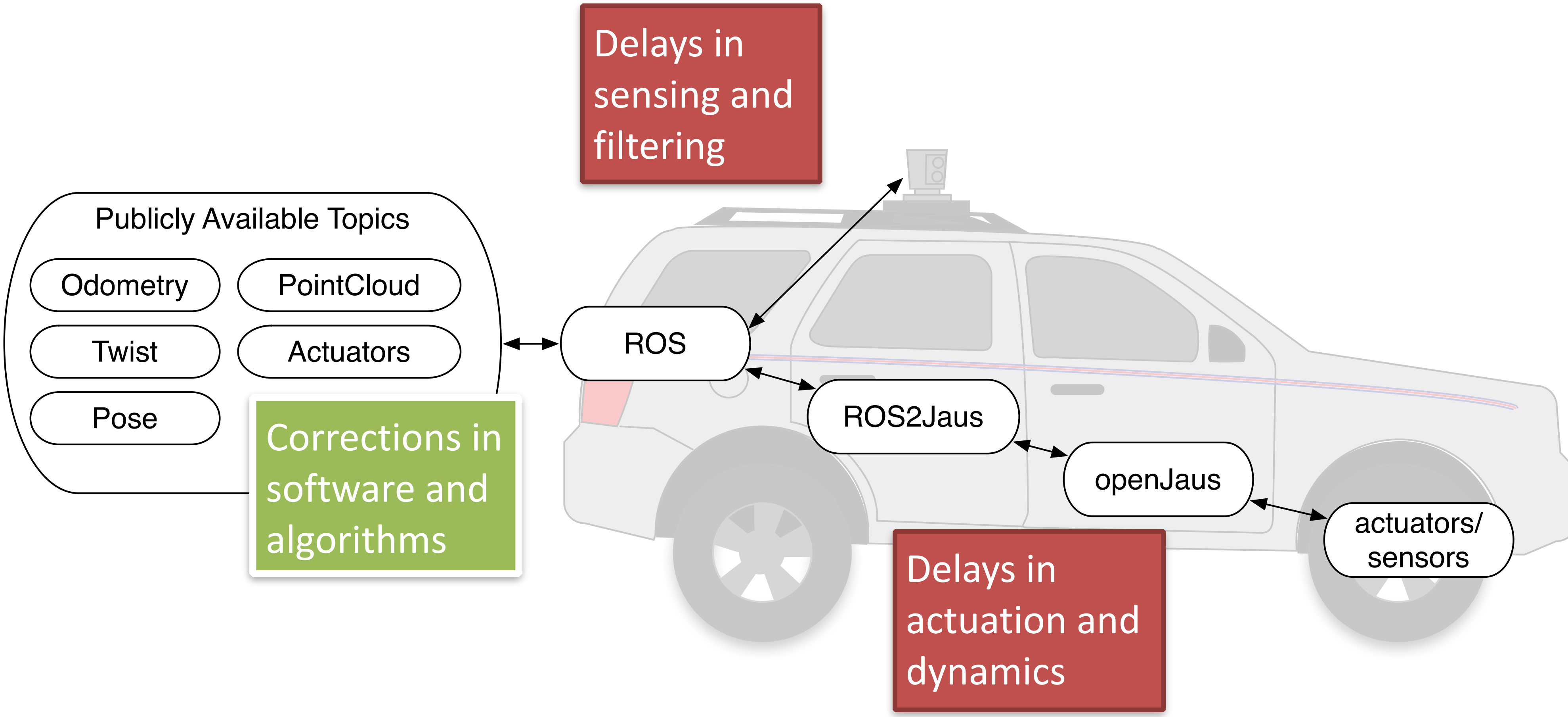
Software Interface Layers



SWIL through Gazebo



Limitations



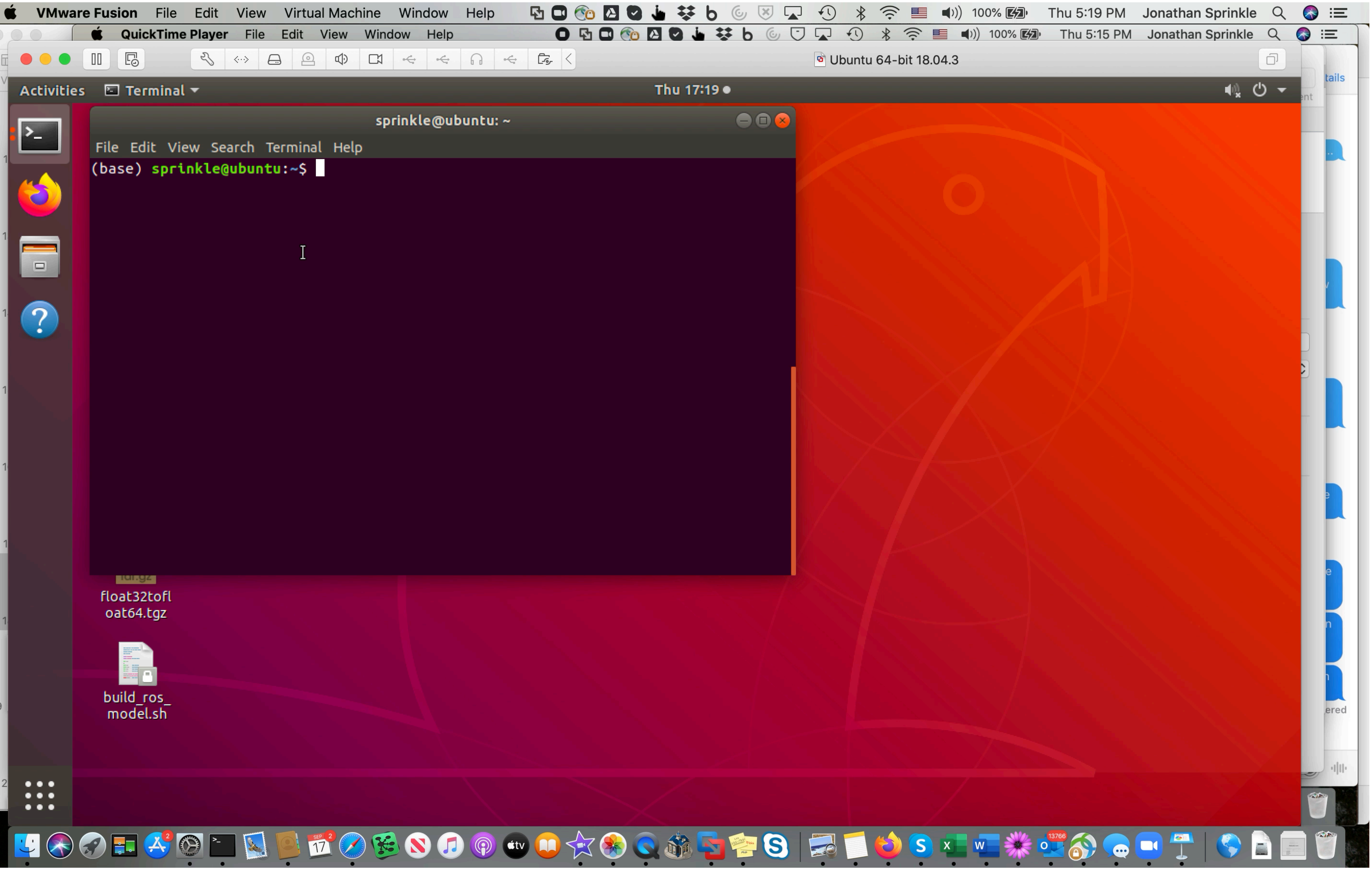
Surrogate: Turtlebot

```
roscore
```

```
rosrun turtlesim turtlesim_node
```

```
rosrun turtlesim turtlesim_teleop_key
```

Surrogate: Turtlebot



Surrogate: Turtlebot

roscore

rosrun turtlesim turtlesim_node

rosrun turtlesim turtlesim_teleop_key

How does a turtle help us with a car?

Inputs: catvehicle

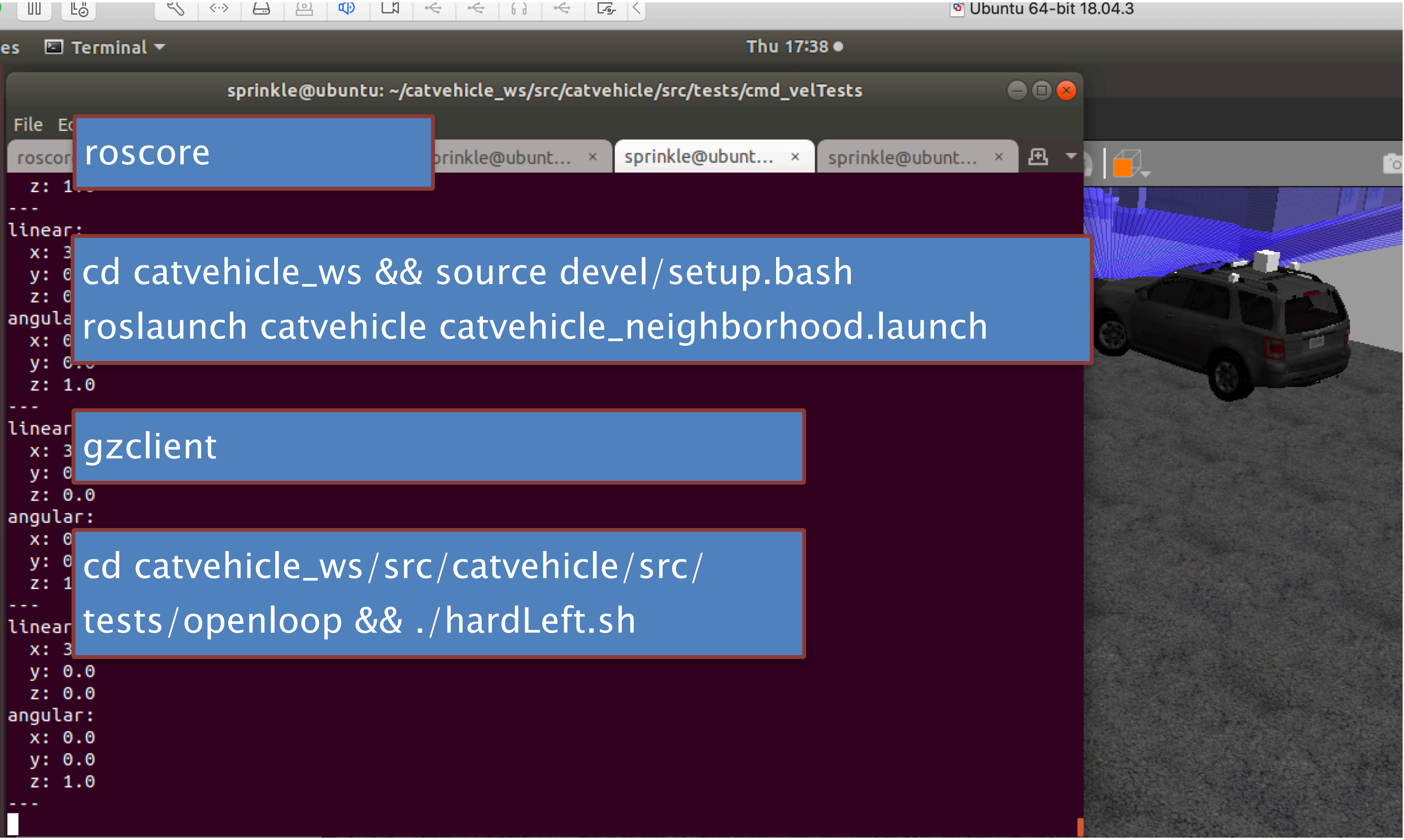
```
roscore
```

```
cd catvehicle_ws && source devel/setup.bash
roslaunch catvehicle catvehicle_neighborhood.launch
```

```
gzclient
```

```
cd catvehicle_ws/src/catvehicle/src/
tests/openloop && ./hardLeft.sh
```

Inputs: catvehicle



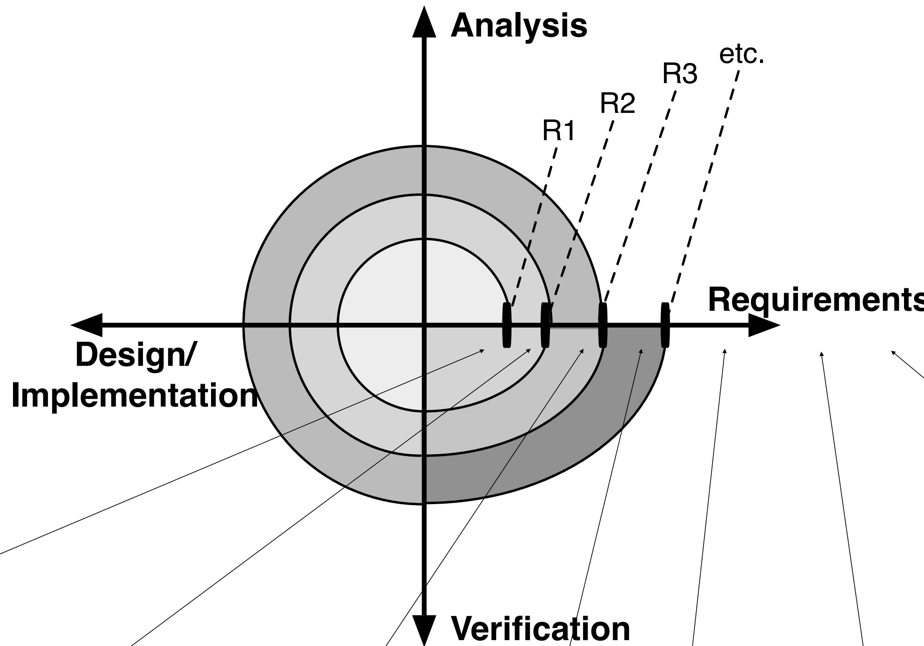
Vehicle Commands through twist message

```
---
linear:
  x: 3.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.0
---
```

x: 3.0 — Desired velocity in m/s

z: 1.0 — Desired steering angle in radians

Project Workflow



Simulate desired input through traditional Simulink blocks

Replace simulink sources with previously recorded data from CAT Vehicle

Validate behavior in Simulink

Generate ROS code from Simulink and run with SWIL testbed

Validate results from SWIL testbed

Transfer code to HWIL testbed (CAT Vehicle) and run isolated vehicle test

Run physical experiment

Vehicle Control Basics

Inputs You May Expect

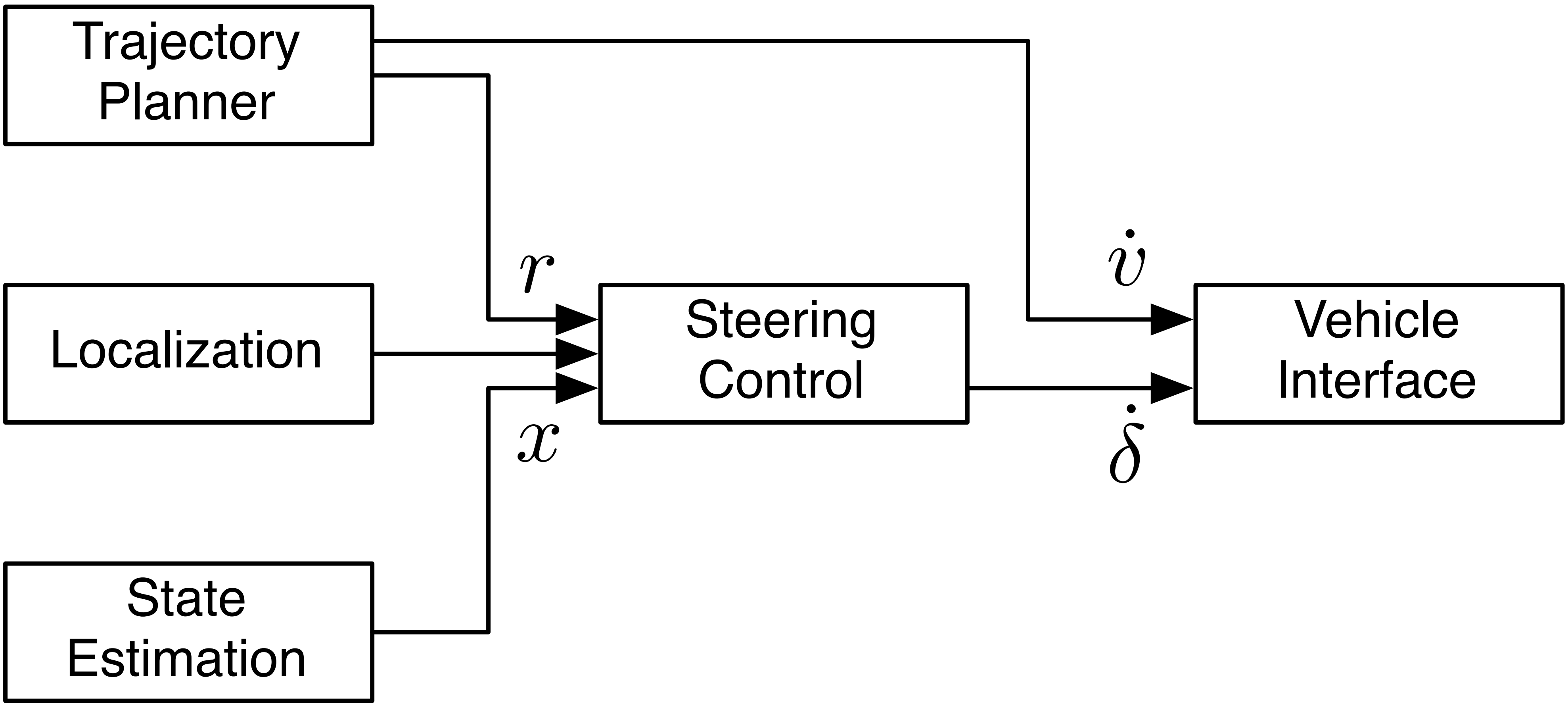
Velocity Control

* Accelerator Angle
* Accelerator Change in angle
* Brake Angle
* Brake change in angle
* Desired Velocity
* Desired Acceleration

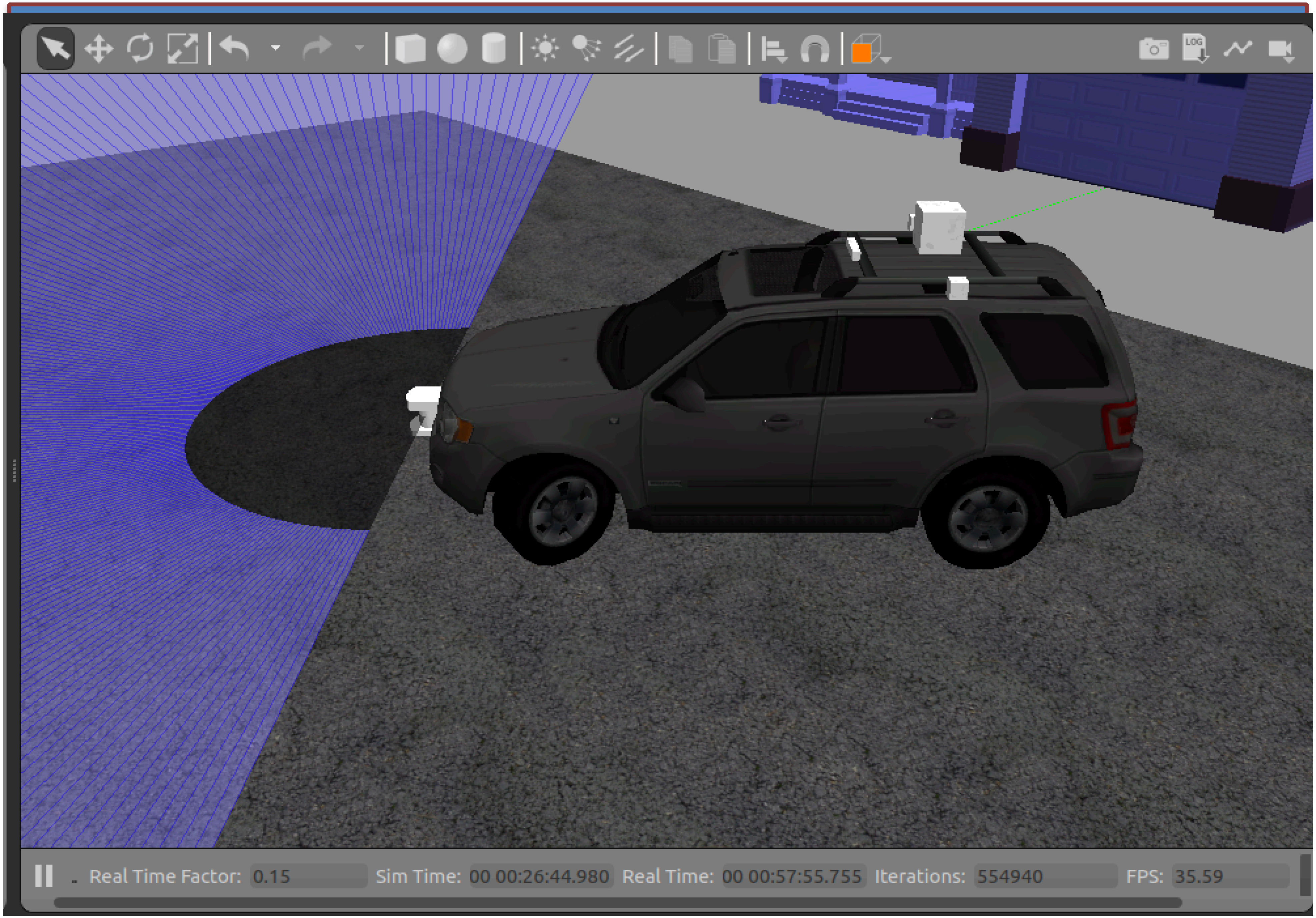
* Cruise Control

Steering Control

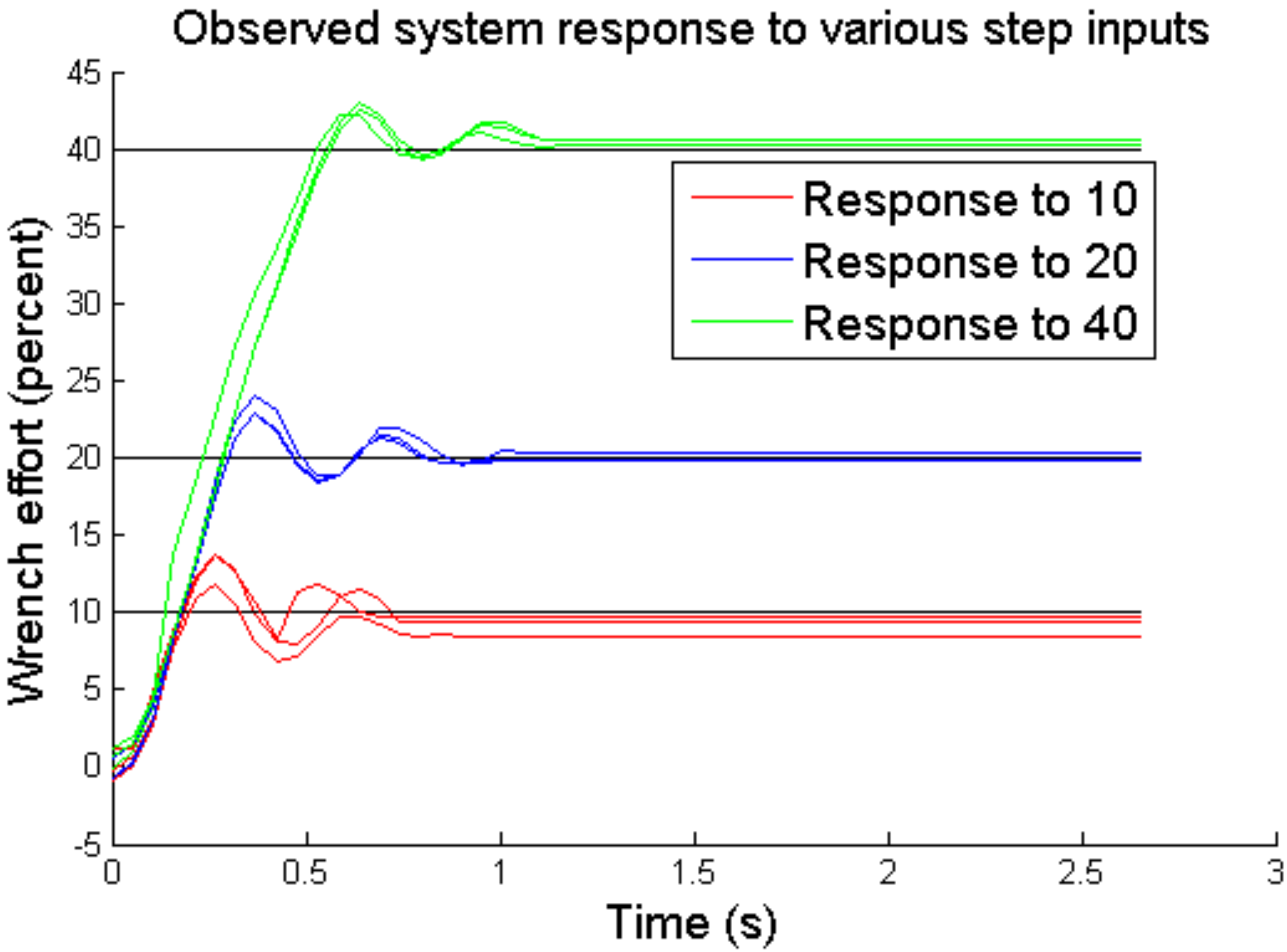
* Steering Angle
* Steering Angle Rate


Trajectory Planner
Localization
State Estimation
r
x
Steering Control
$\dot{v}$
$\dot{\delta}$
Vehicle Interface

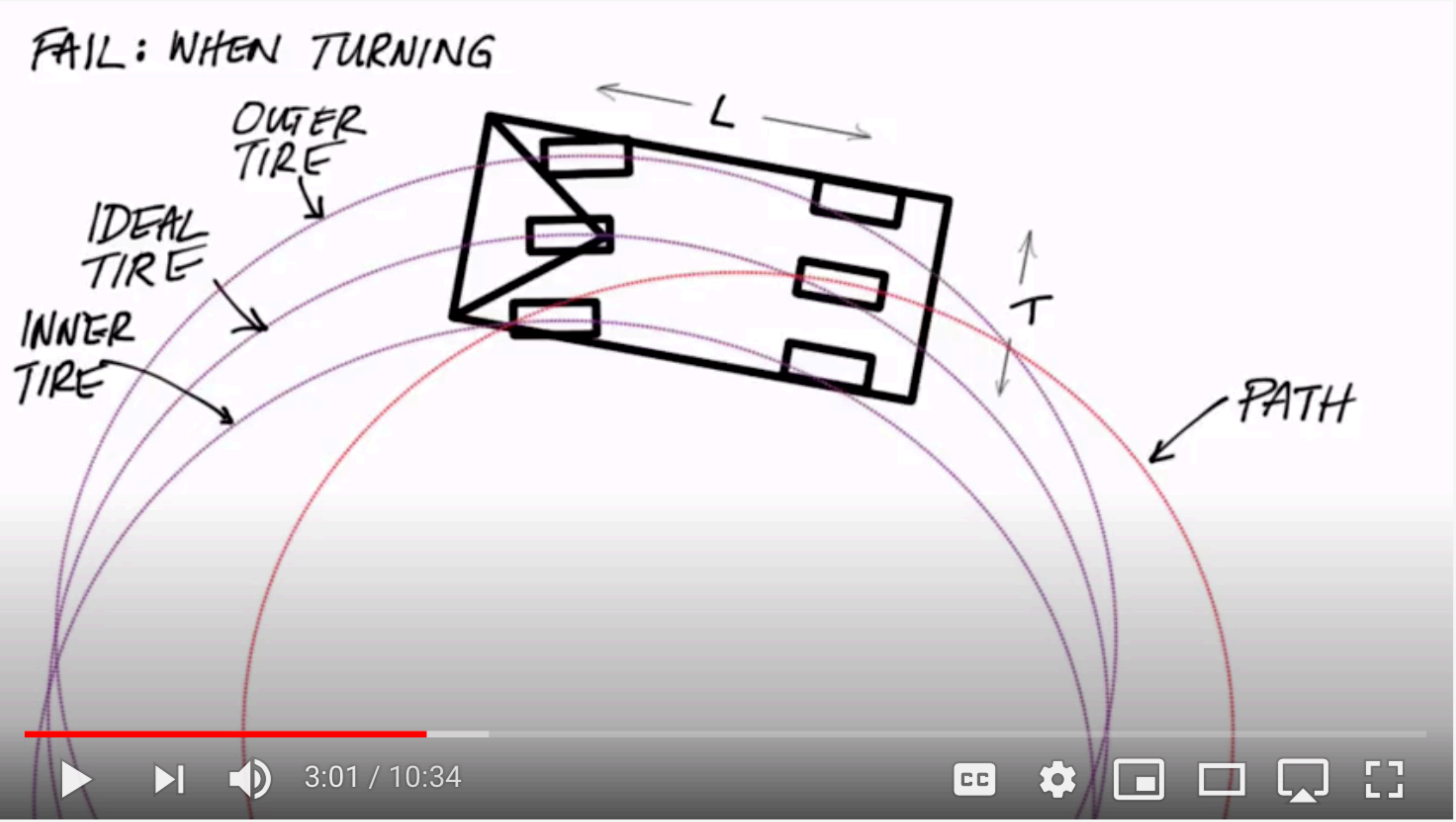
Steering Angle, Steering Rate



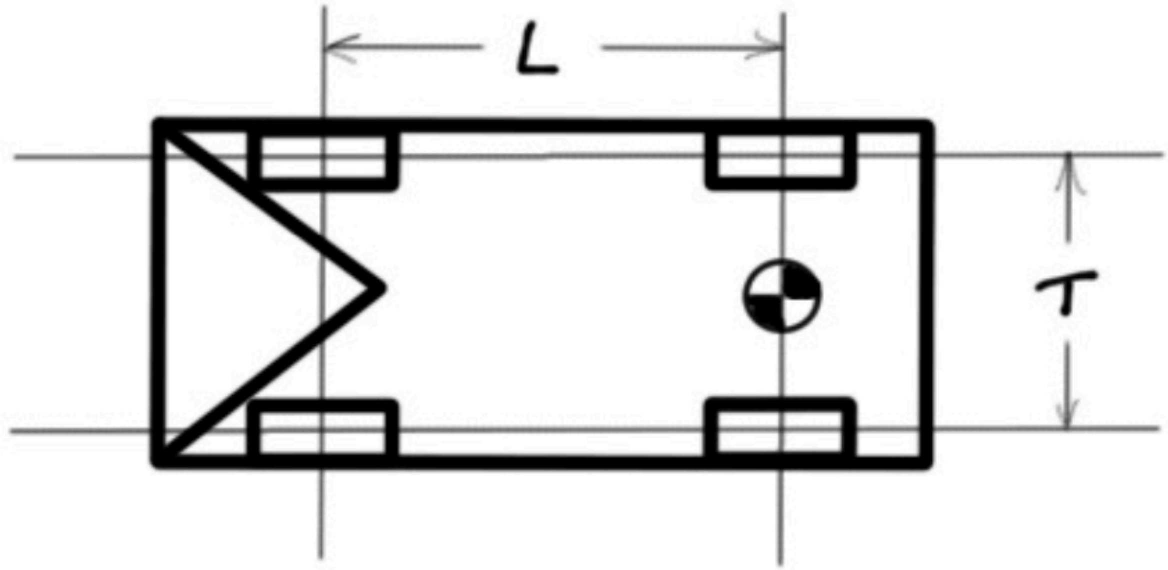
Steering Wheel (Angle) Response to Step Inputs



Trajectory following needs the Ackerman Model

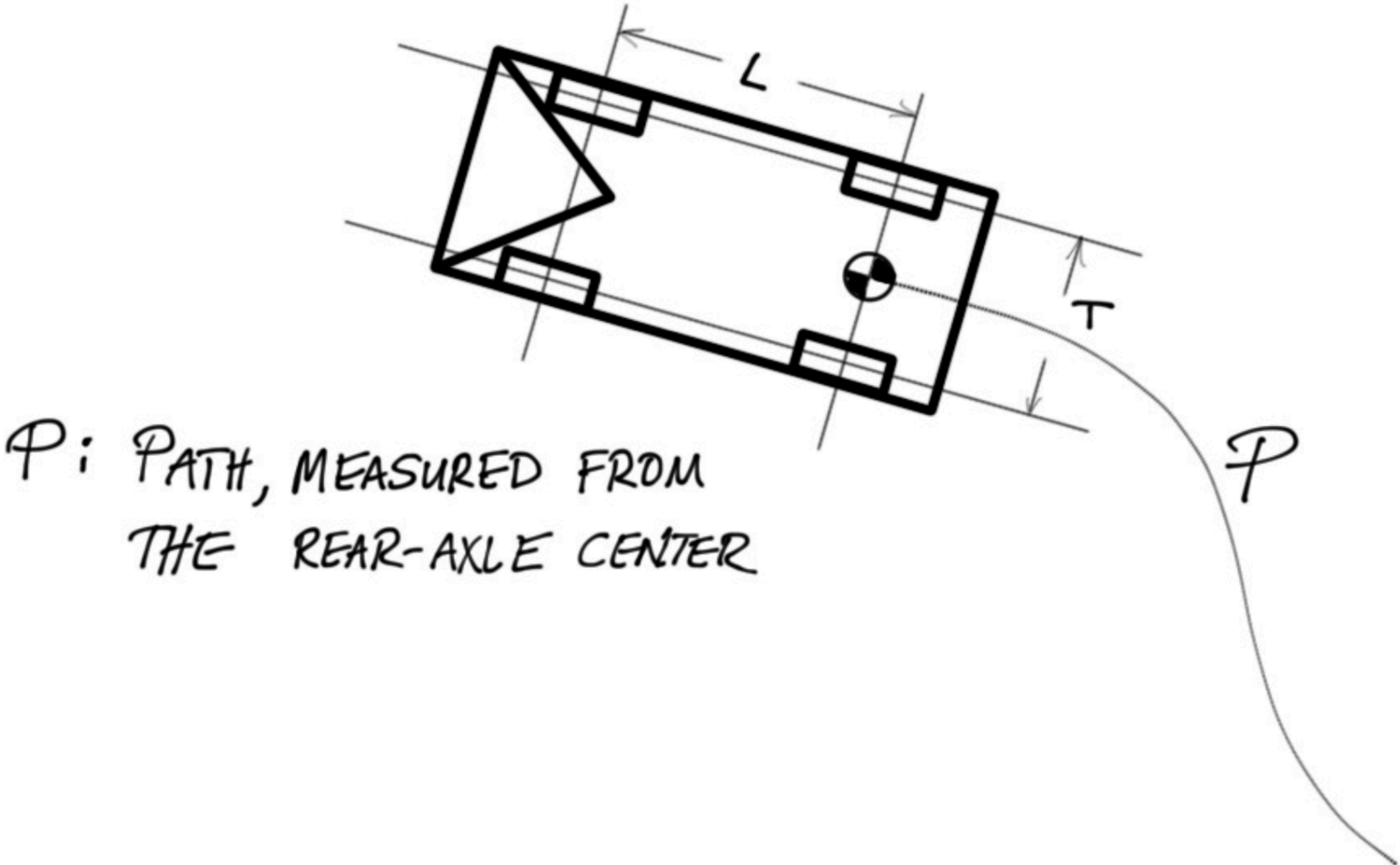


https://www.youtube.com/watch?v=i6uBwudwA5o

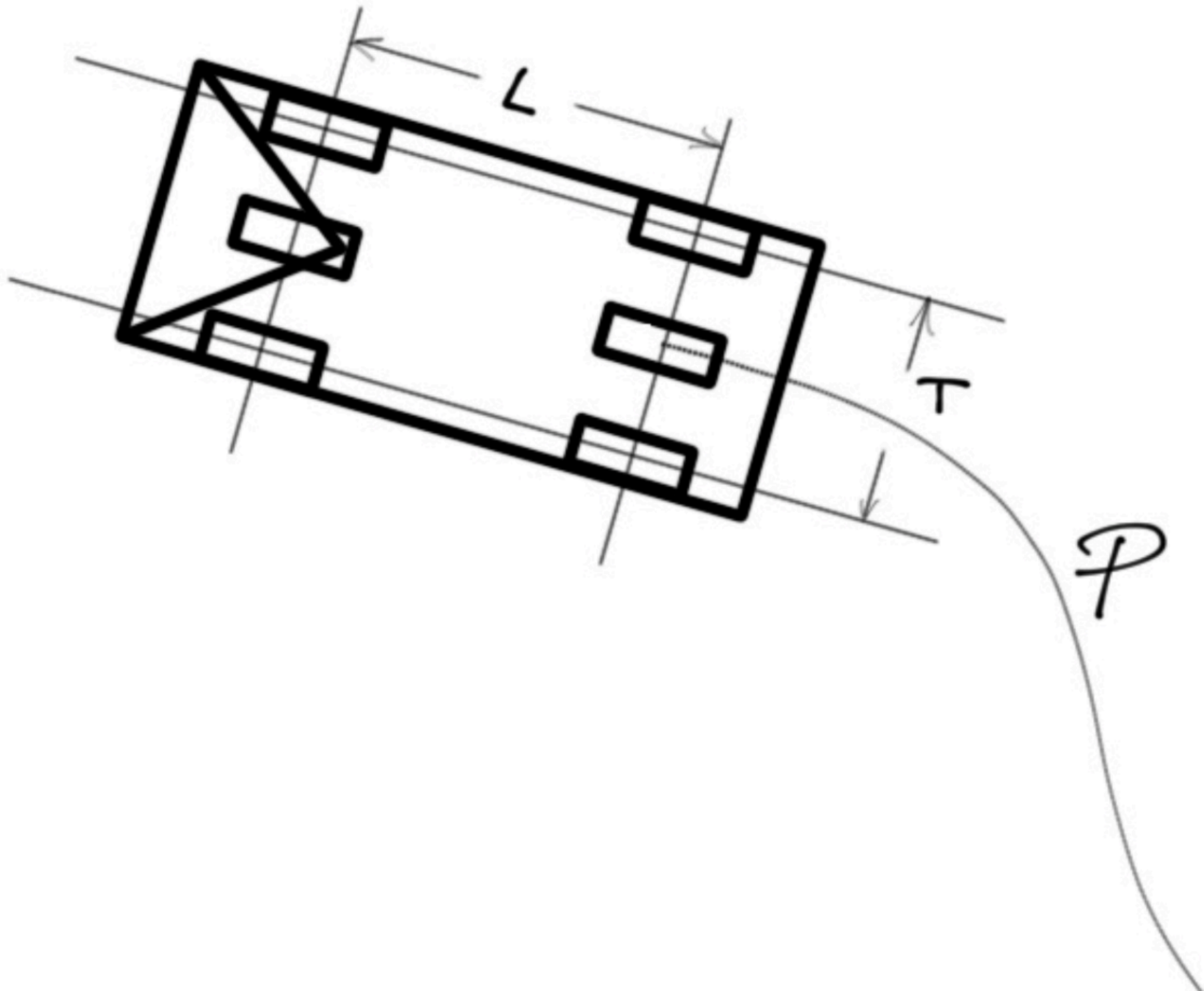


T : TREAD

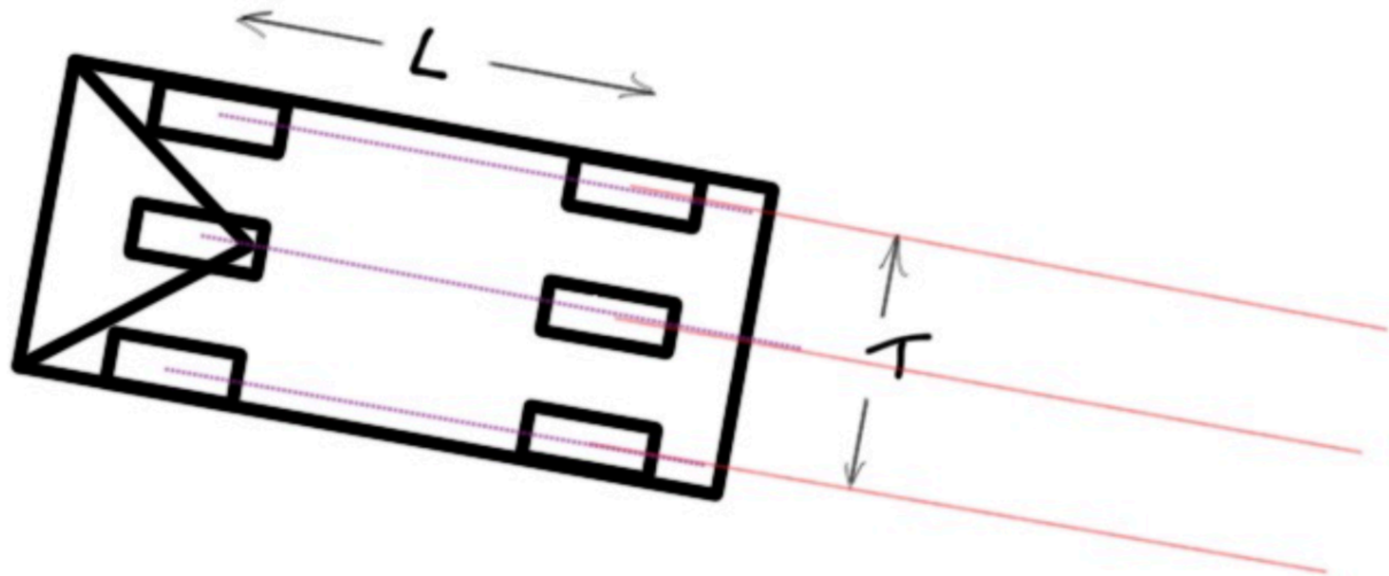
L : WHEELBASE


L
T
P
P: PATH, MEASURED FROM
THE REAR-AXLE CENTER

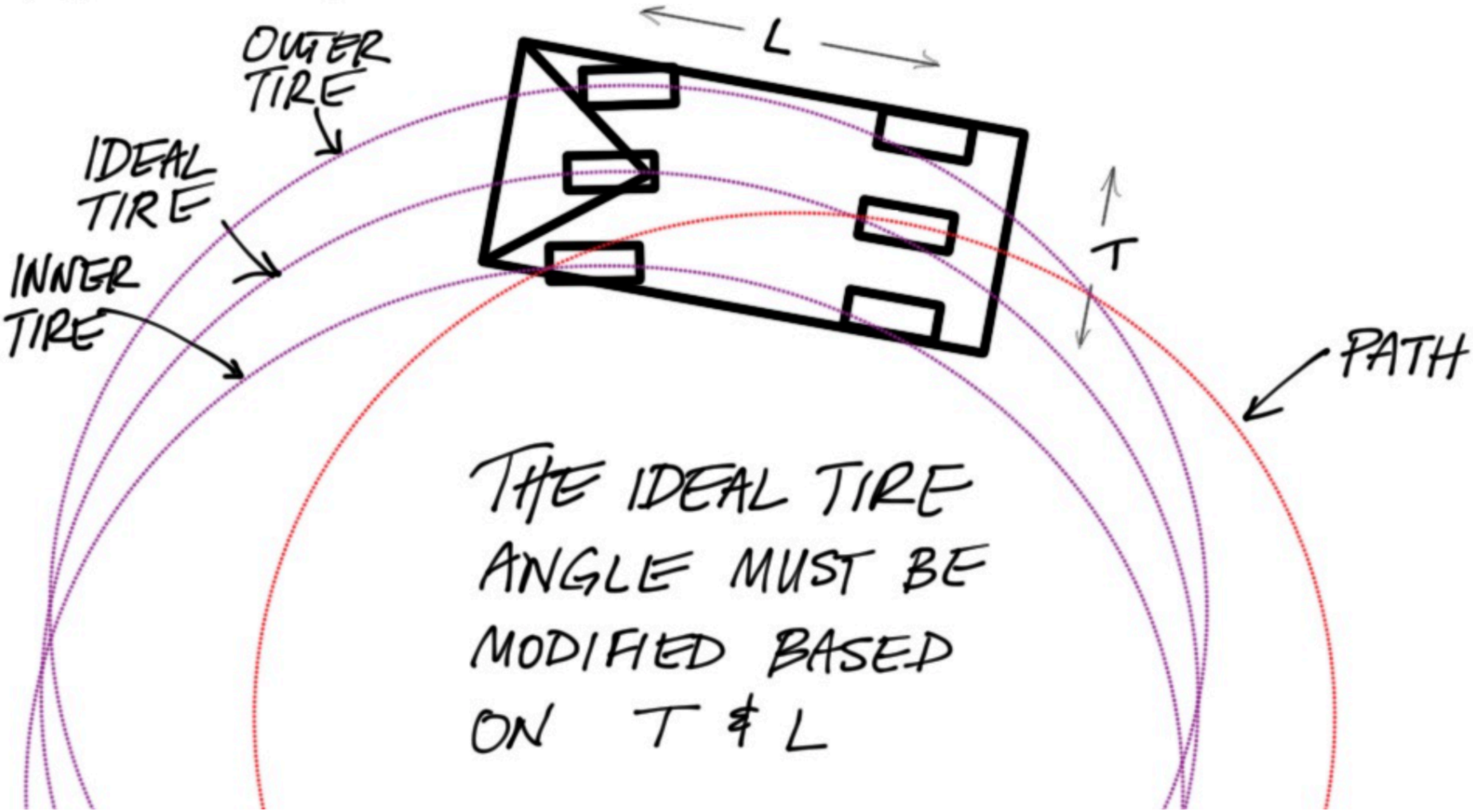
IDEA: MEASURE PATH FROM IMAGINARY "IDEAL" MIDDLE TIRE.



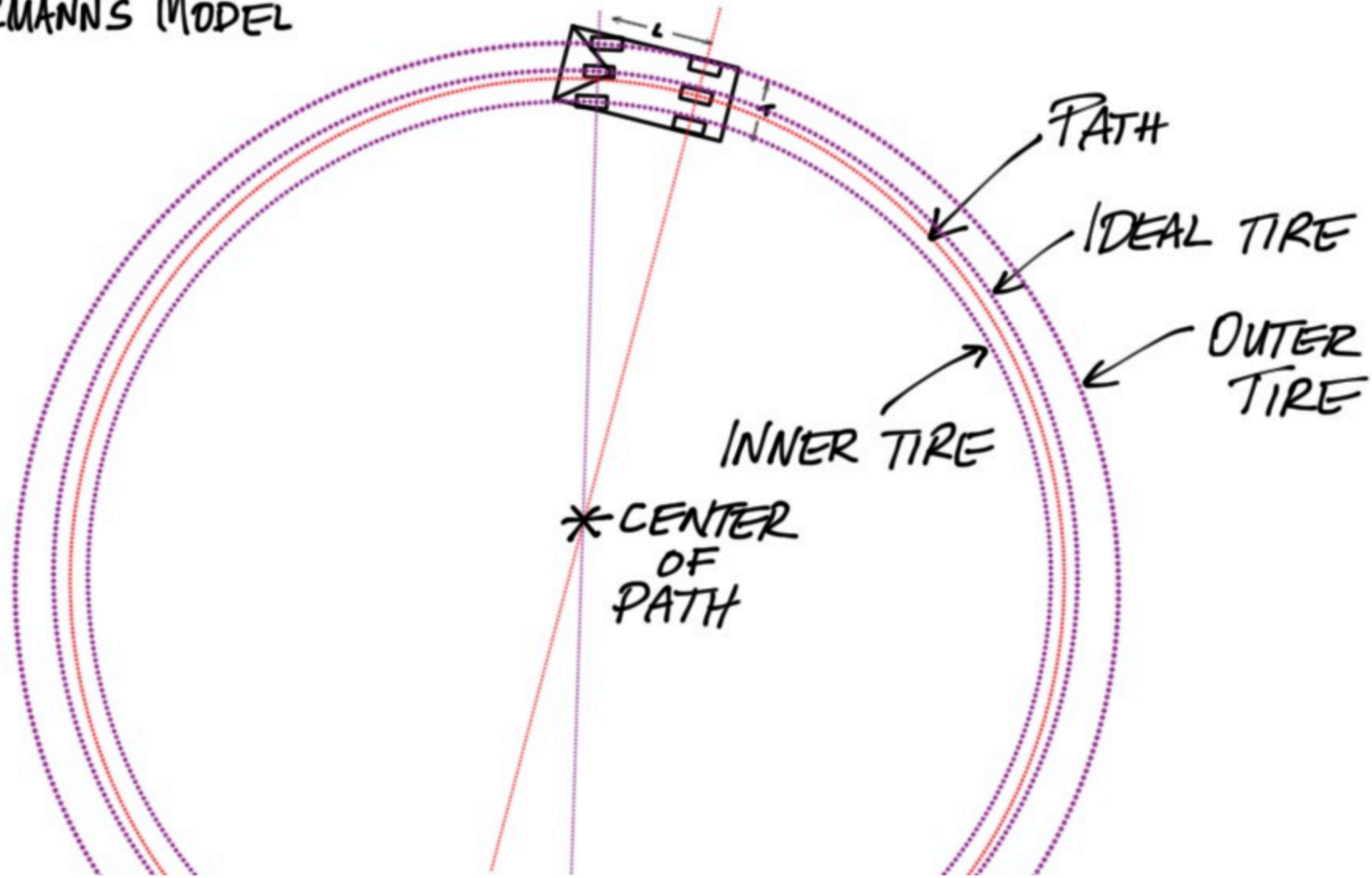
SUCCESS: WHEN GOING STRAIGHT



FAIL: WHEN TURNING

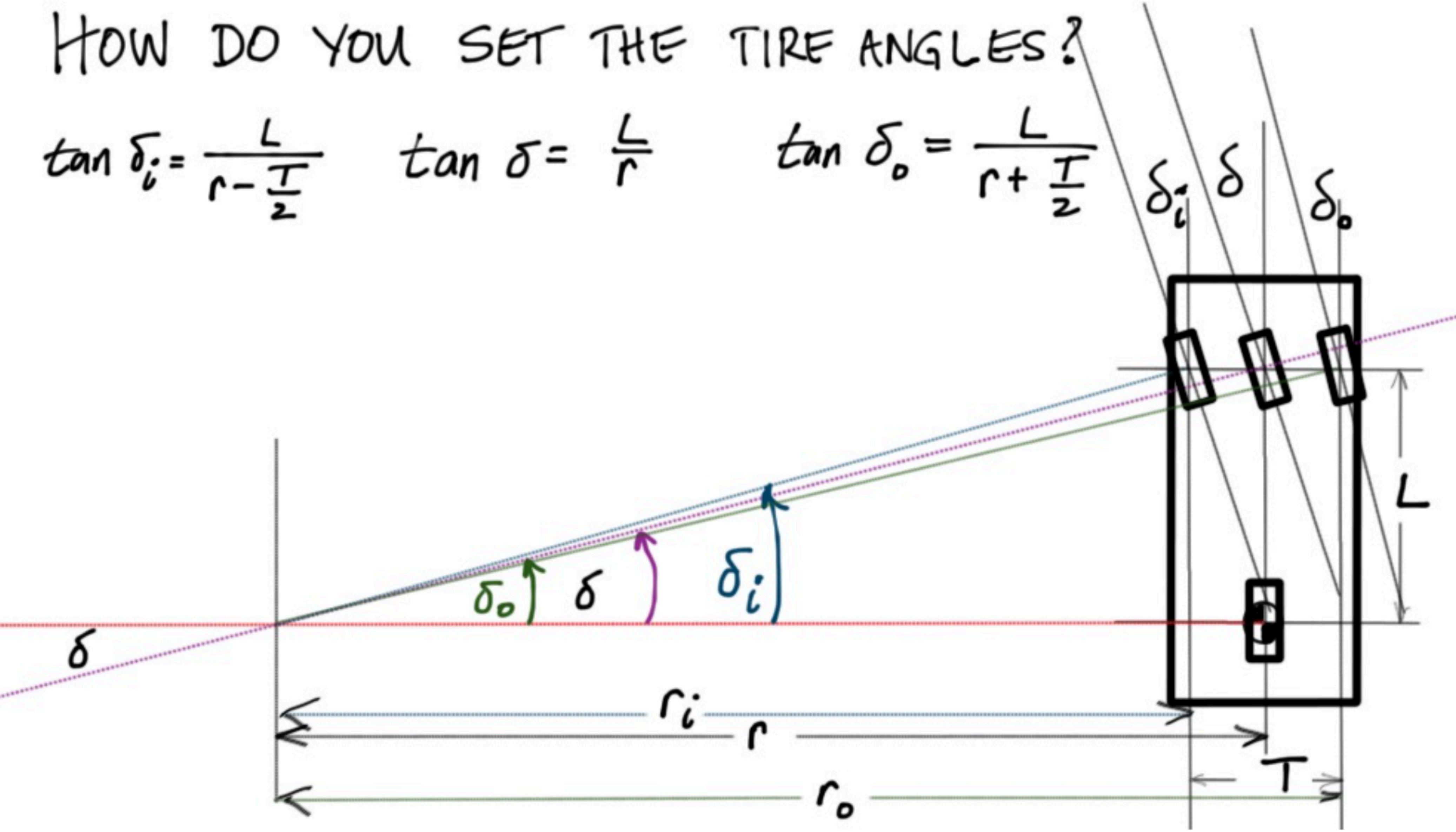


Ackermann's Model



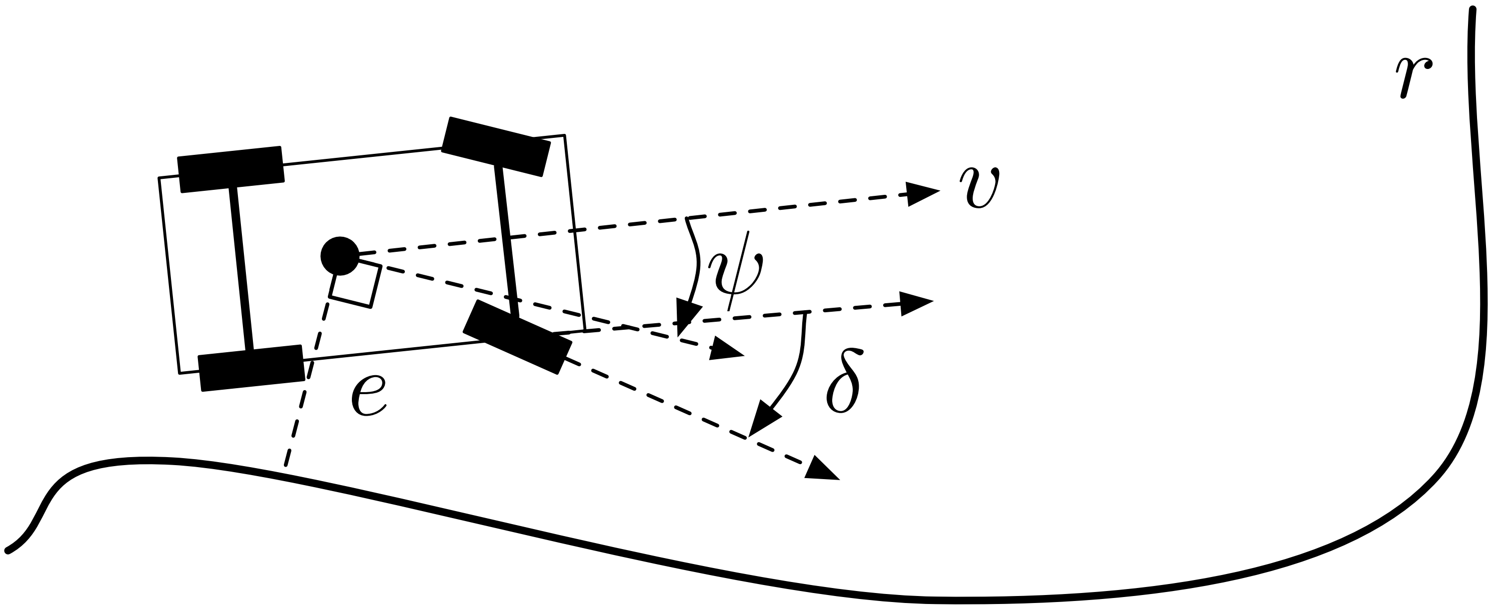
HOW DO YOU SET THE TIRE ANGLES?

$$\tan \delta_i = \frac{L}{r - \frac{T}{2}} \qquad \tan \delta = \frac{L}{r} \qquad \tan \delta_o = \frac{L}{r + \frac{T}{2}}$$



Example: trajectory-following controller [Hoffmann 2006]

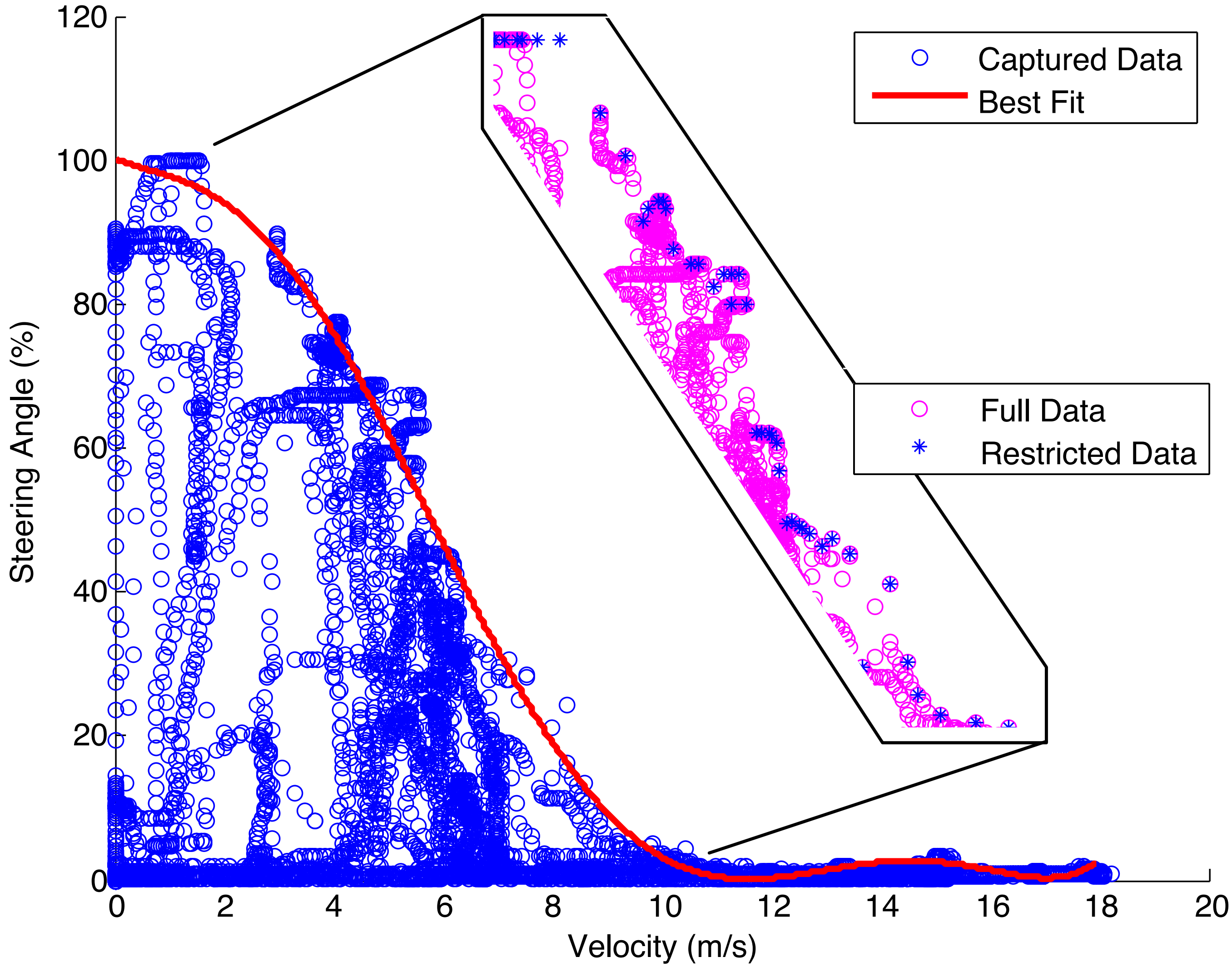
$$\delta(t) = g(e, u, \psi) = \psi(t) + \arctan \frac{ke(t)}{v(t)}$$



Problem: How aggressive to steer, and at what speeds?

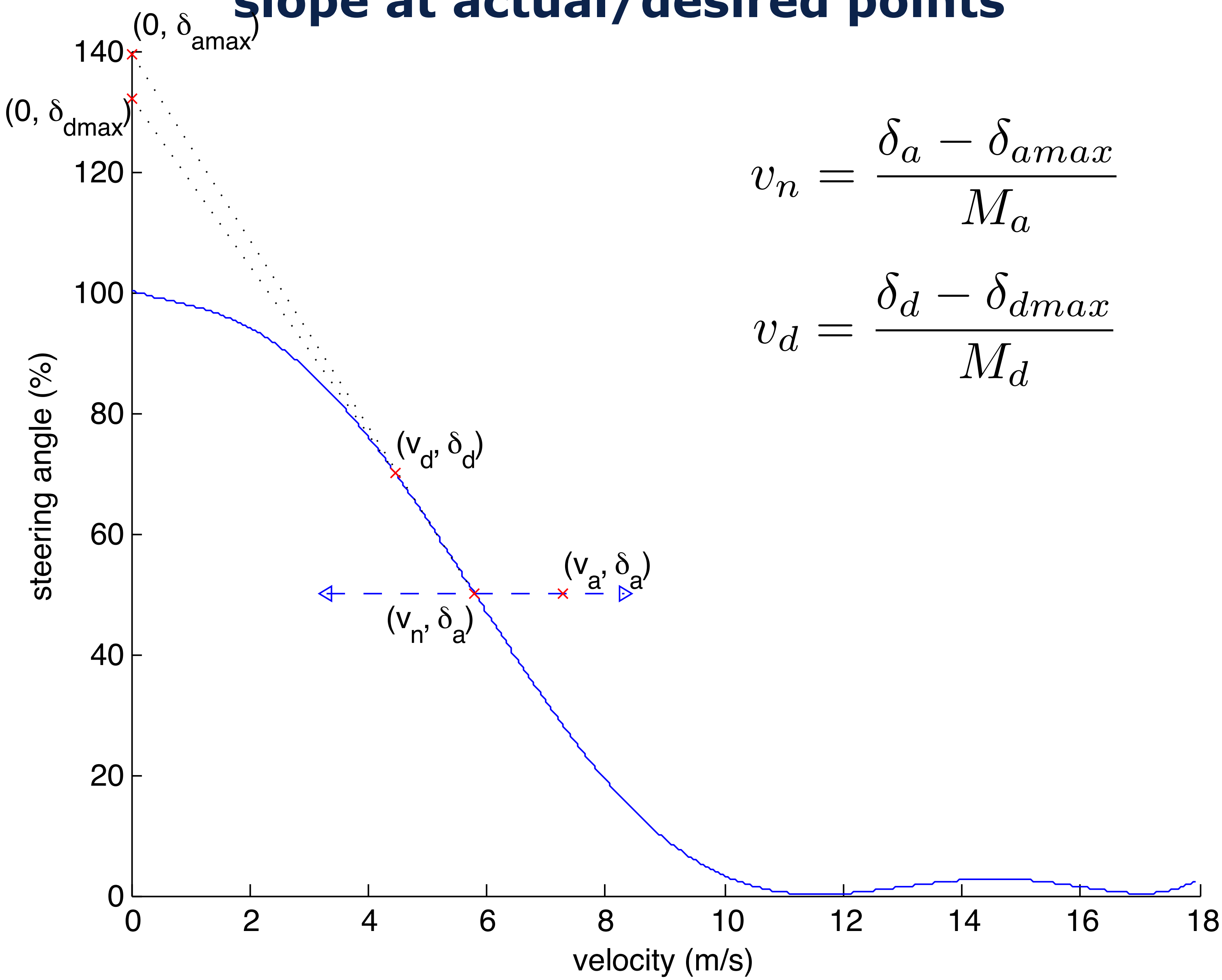
Photo By Twilight Invasion

Best fit



120
100
80
60
40
20
0
Steering Angle (%)
0
2
4
6
8
10
12
14
16
18
20
Velocity (m/s)
Captured Data
Best Fit
Full Data
Restricted Data

Control approach: linearize the velocity/steering slope at actual/desired points



$$v_n = \frac{\delta_a - \delta_{amax}}{M_a}$$

$$v_d = \frac{\delta_d - \delta_{dmax}}{M_d}$$

Slopes of velocity controller

$$d\delta_d = -K_{\delta a}\delta_a - K_{\delta d}\delta_d$$

$$\dot{v} = K_d\left(v_d - v_a\right) + K_a\left(v_n - v_a\right)$$

Let us represent our operating point state vector as

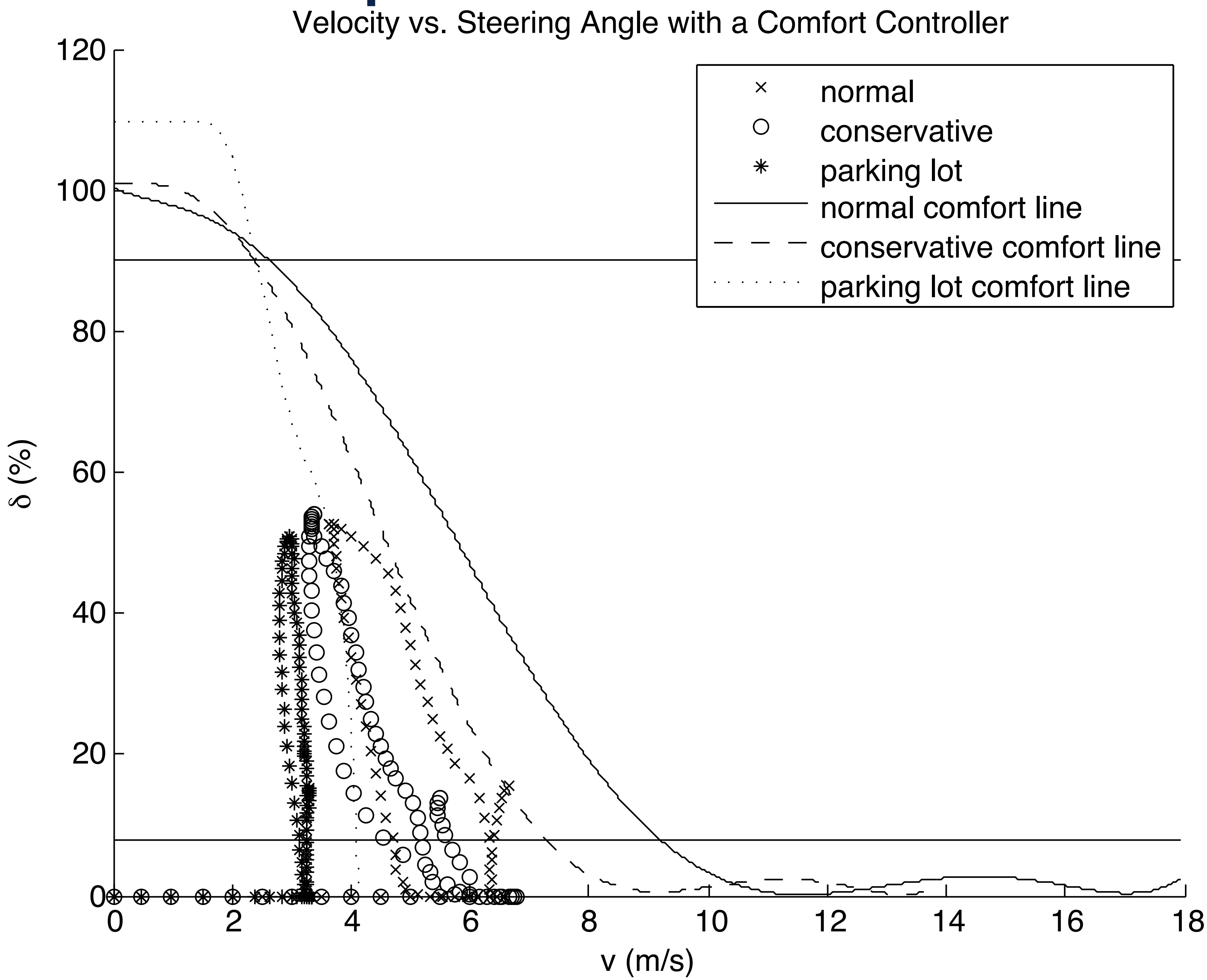
$$x = [\delta_a \;\; v_a]^T$$

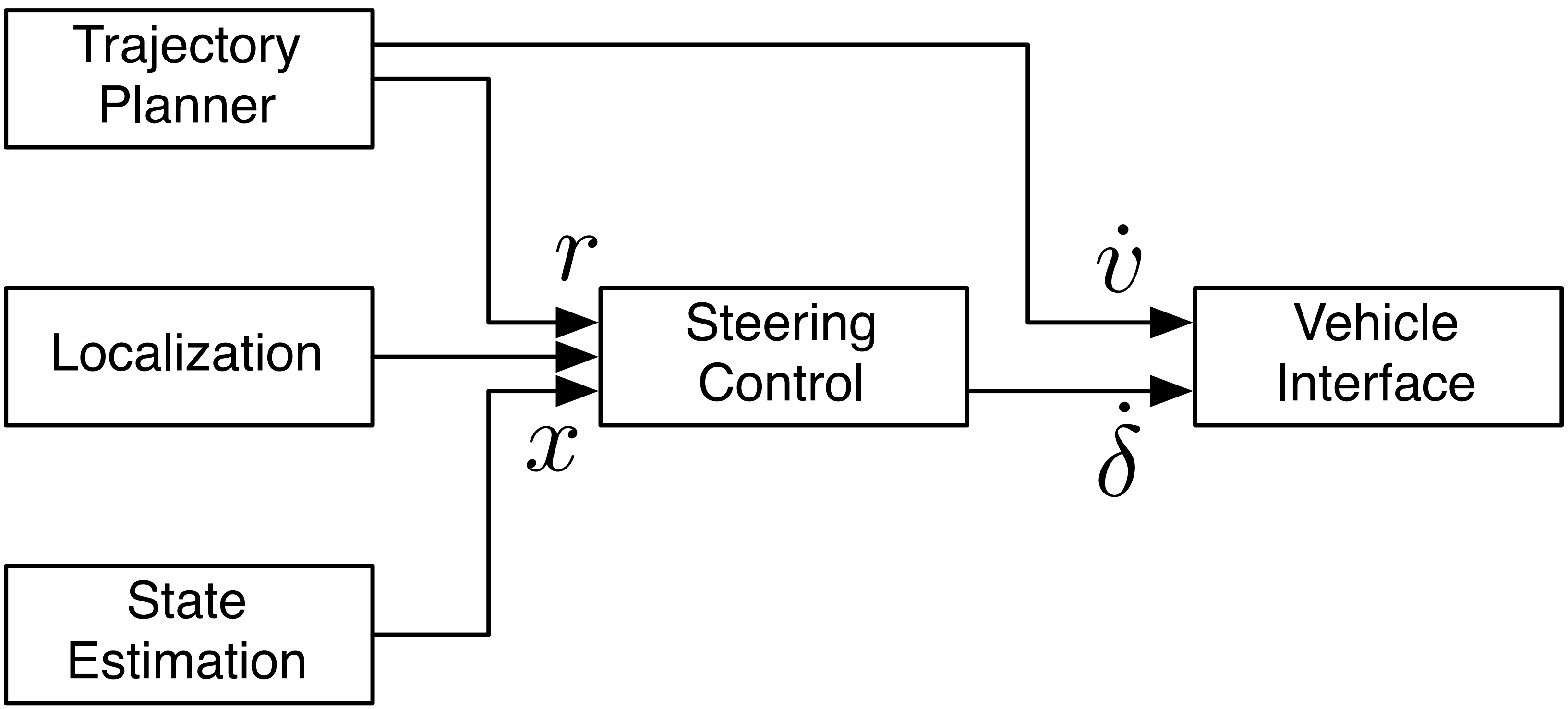
Then, we can define the following linearized state space for an operating point near *x*:

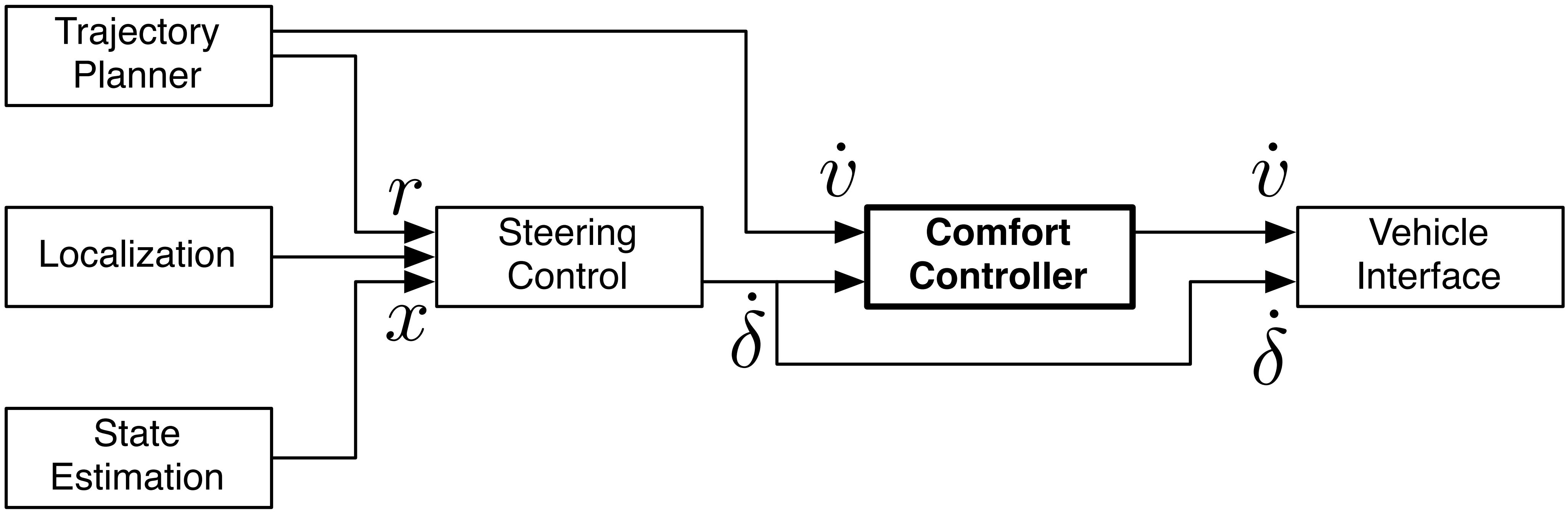
$$y = \begin{bmatrix} \frac{K_{va}}{M_a} & -\left(K_{va} + K_{vd}\right) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \left[\frac{K_{vd}}{M_d}\right]\delta_d$$

$$dx = \begin{bmatrix} -K_{\delta a} & 0 \\ \frac{K_{va}}{M_a} & -\left(K_{va} + K_{vd}\right) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \begin{bmatrix} -K_{\delta d} \\ \frac{K_{vd}}{M_d} \end{bmatrix}\delta_d$$

Scatter plot with comfort controller




Trajectory Planner
Localization
State Estimation
r
x
Steering Control
$\dot{v}$
$\dot{\delta}$
Vehicle Interface

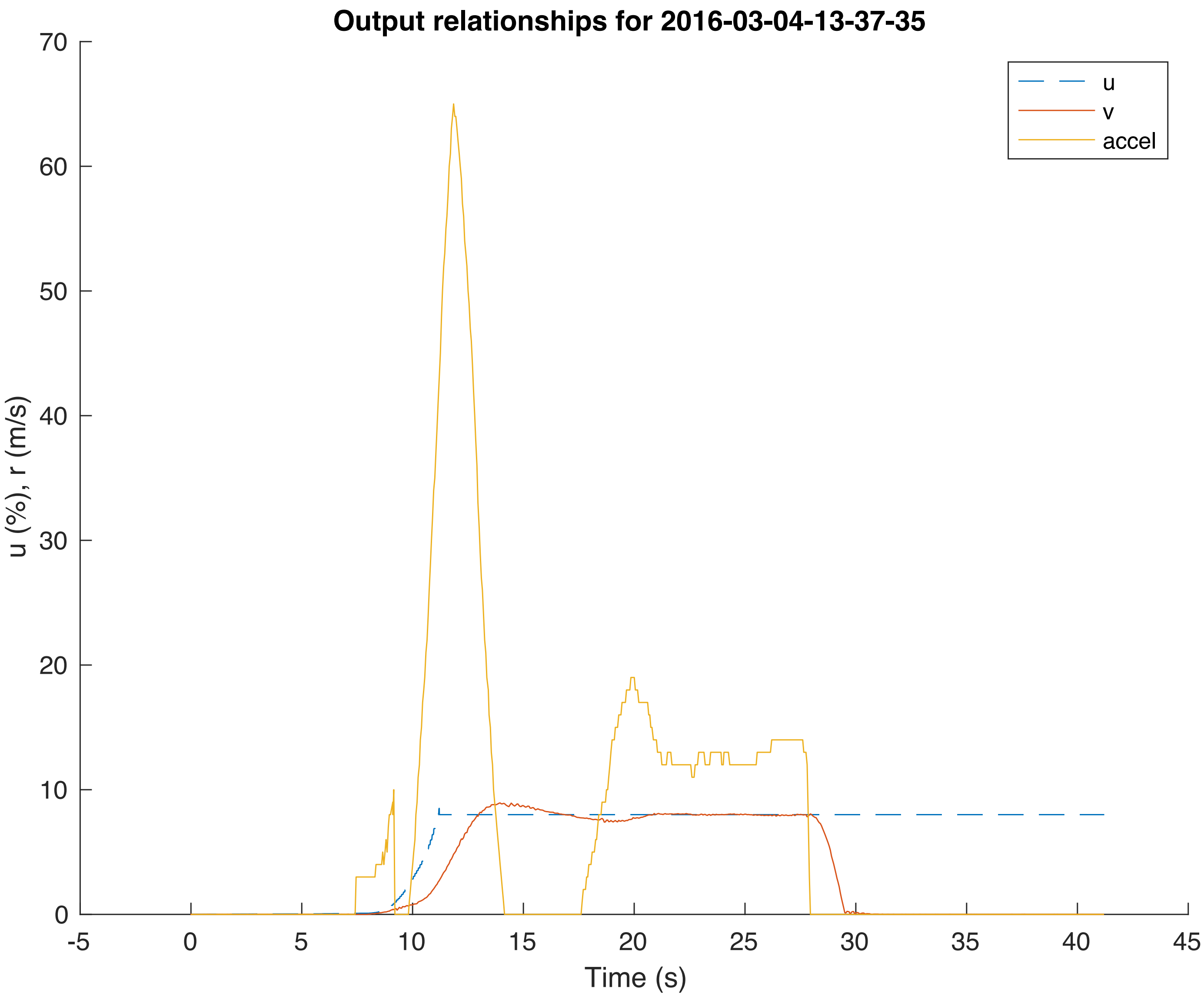


Trajectory Planner
Localization
State Estimation
r
x
Steering Control
$\dot{v}$
$\dot{\delta}$
Comfort Controller
$\dot{v}$
$\dot{\delta}$
Vehicle Interface

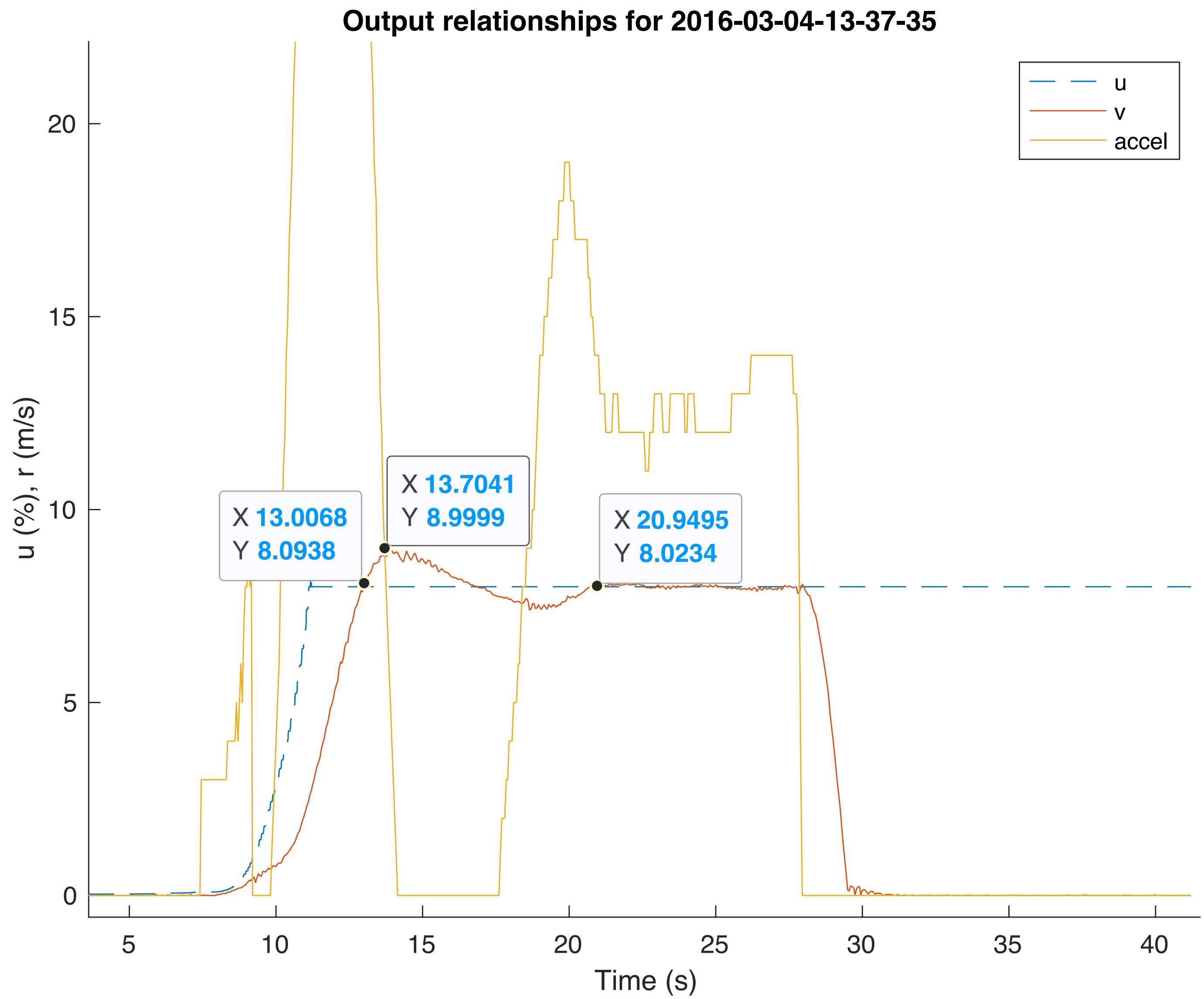
Vehicle control limitations

The University of Arizona

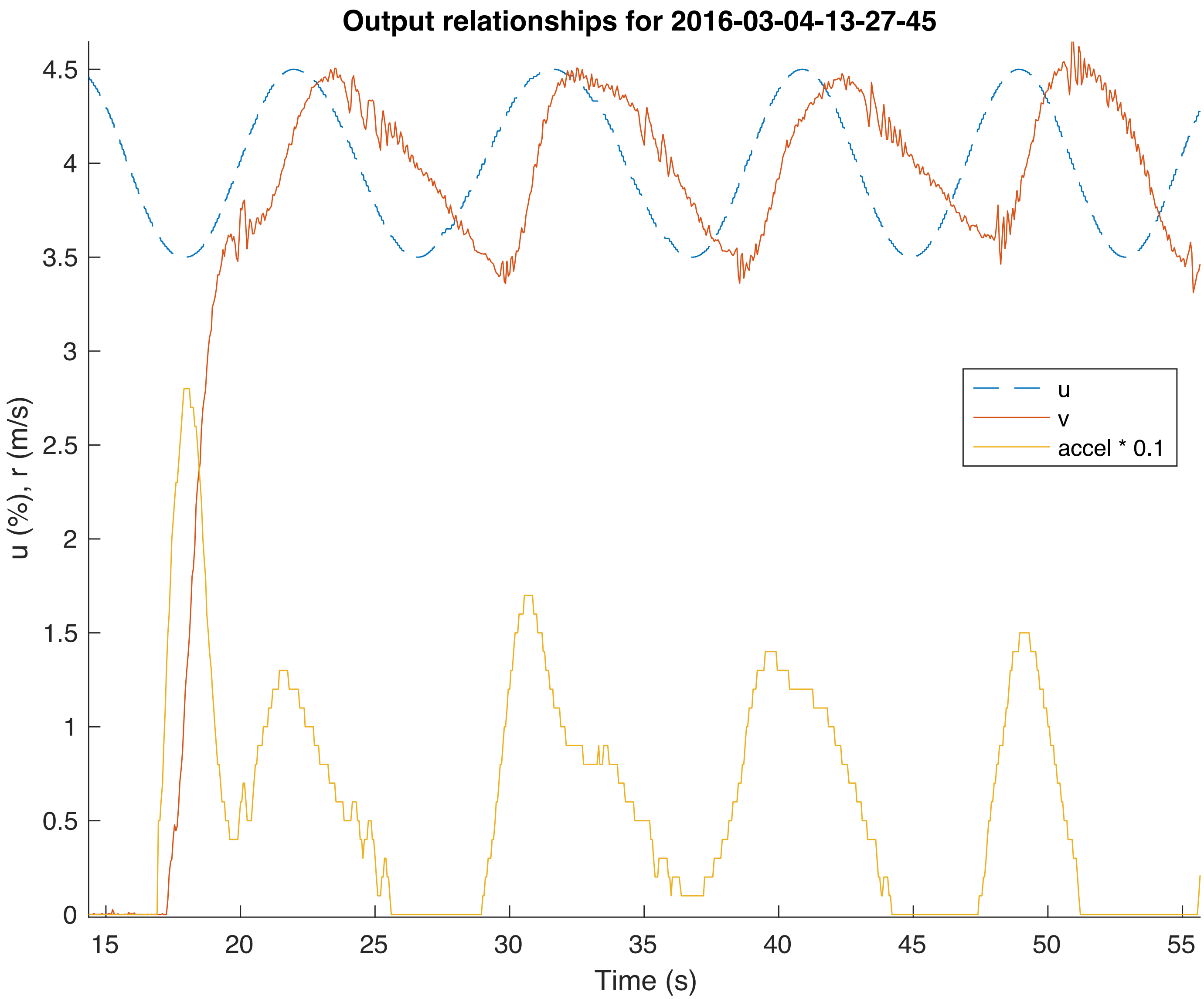
Basic Velocity Following (Cruise Controller)



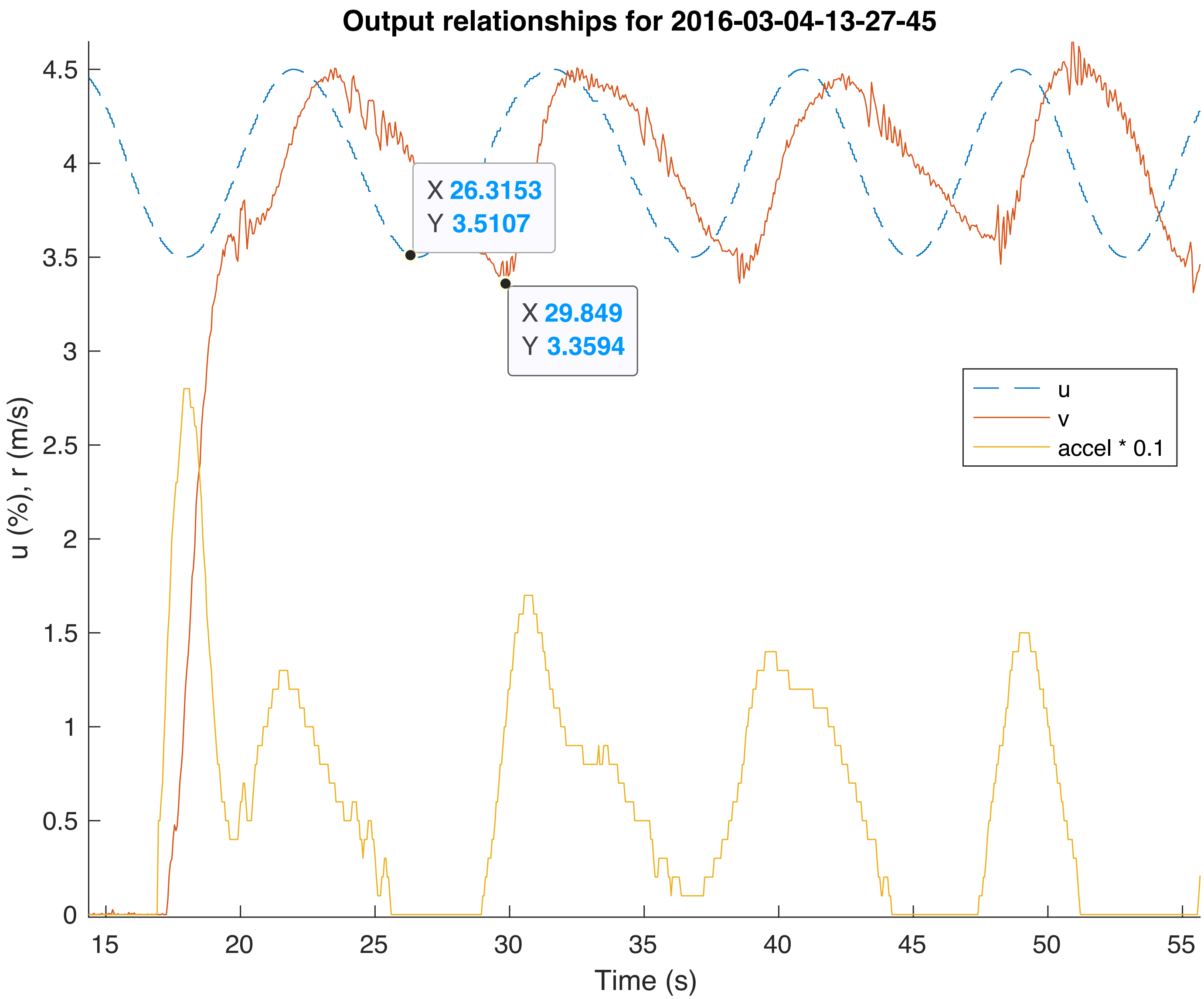
Basic Velocity Following (Cruise Controller)



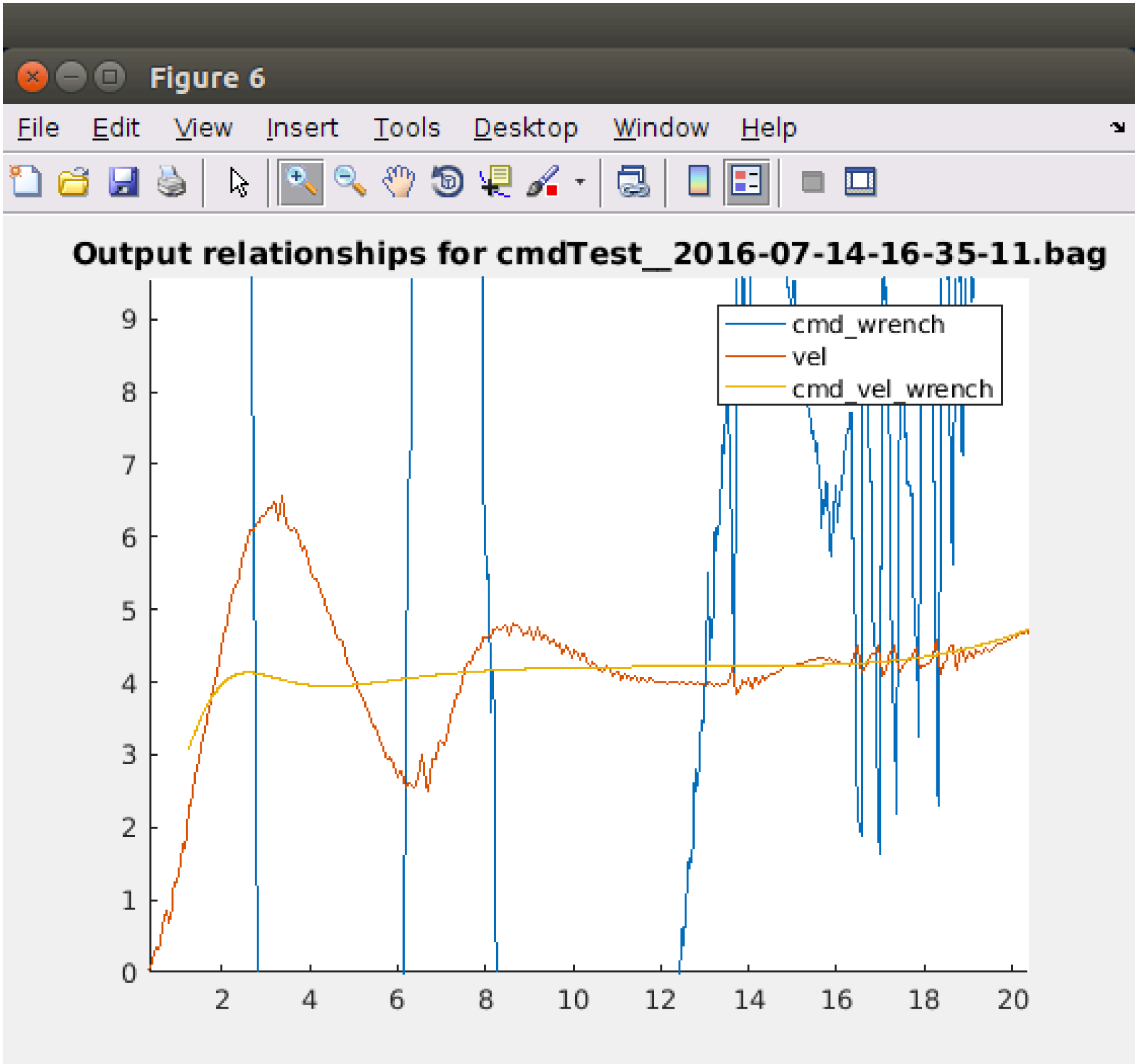
Maybe it would do OK on periodic inputs



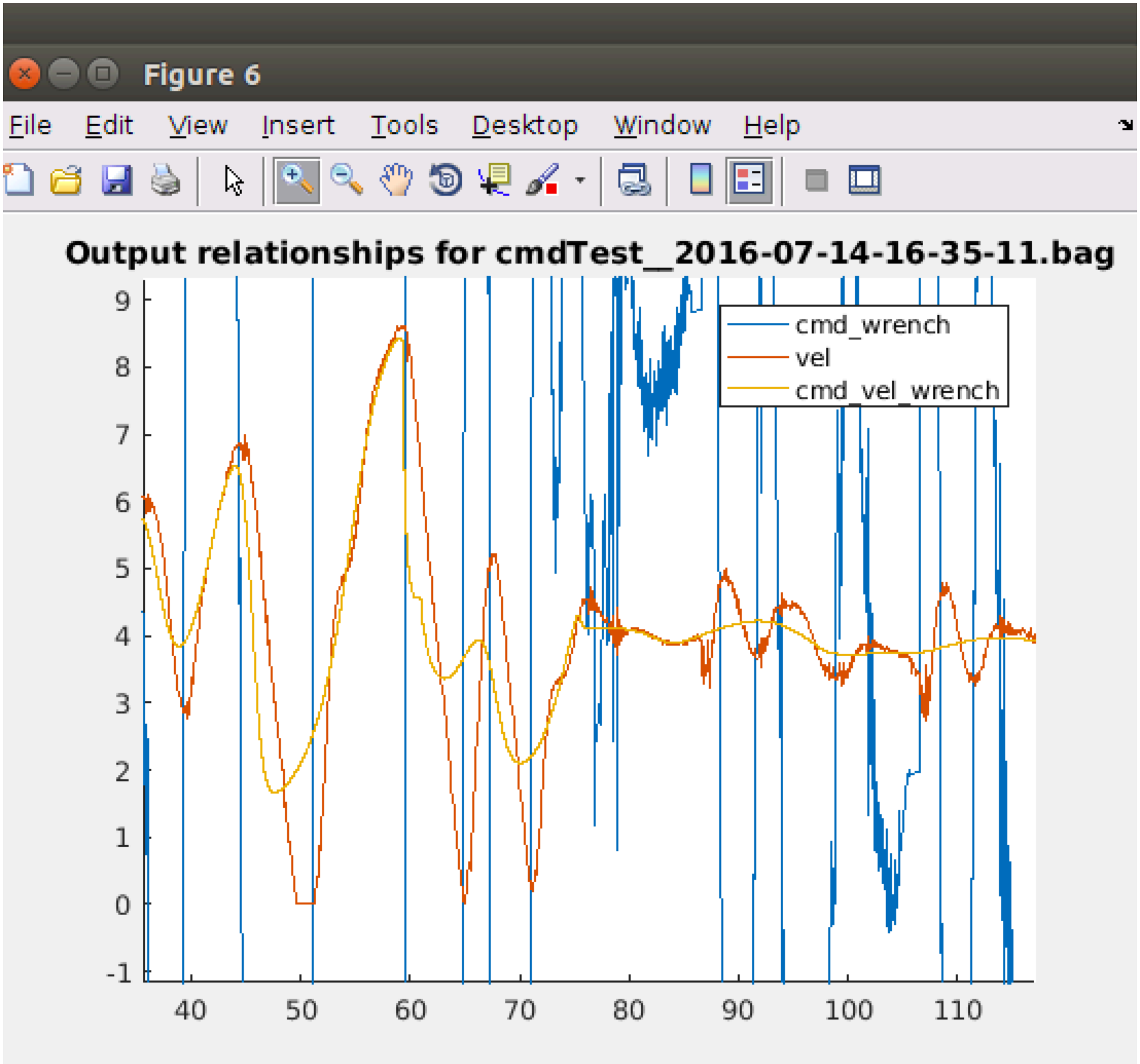
Maybe it would do OK on periodic inputs



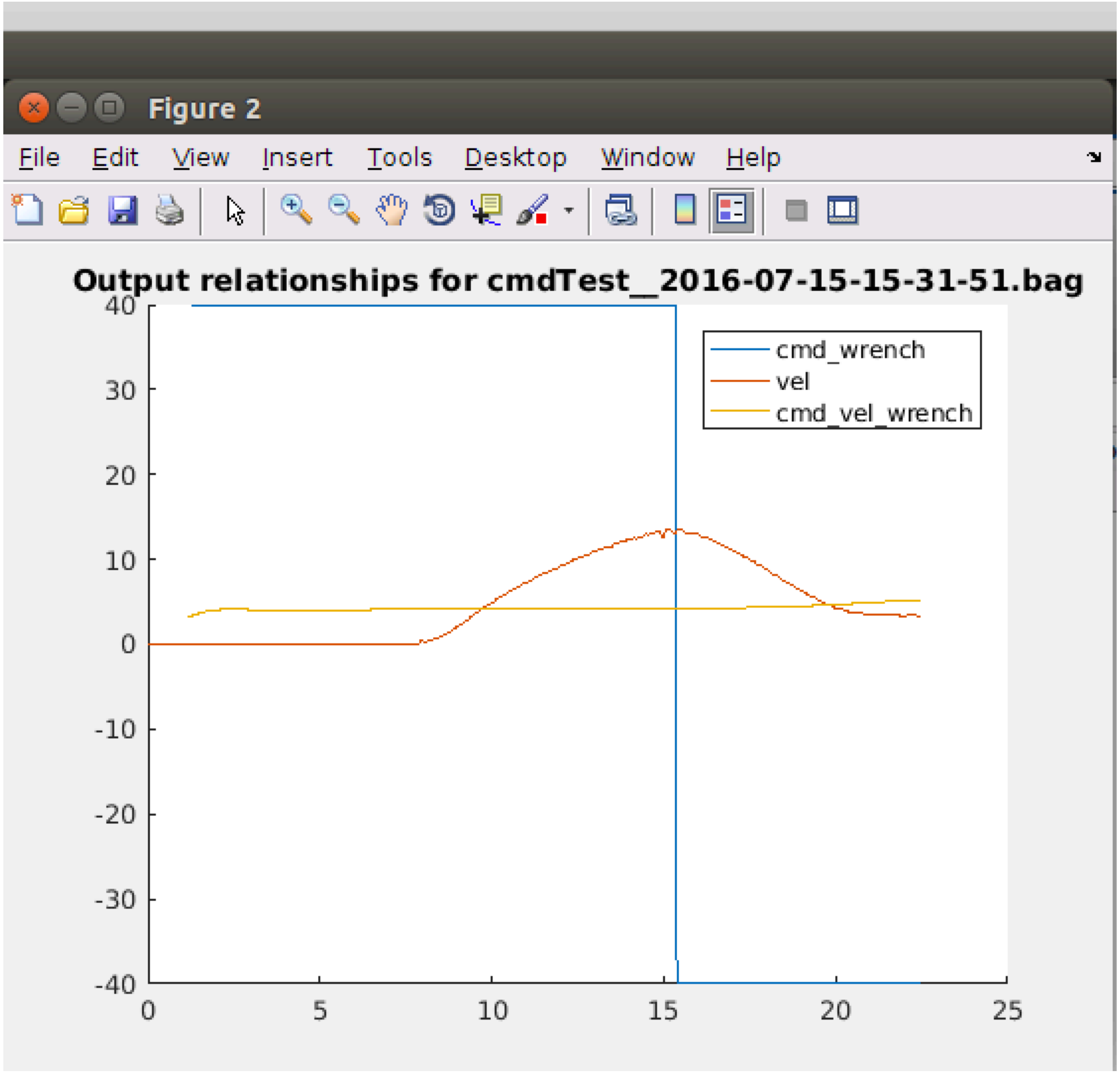
Follow a curve—FAST



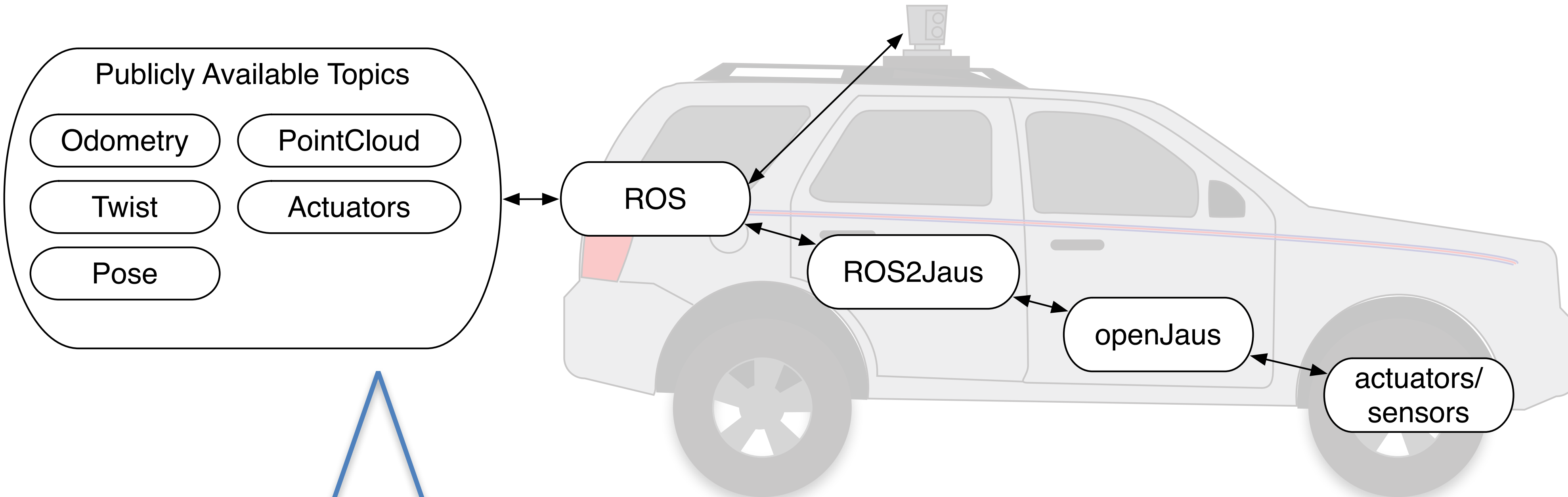
Follow a time-varying velocity

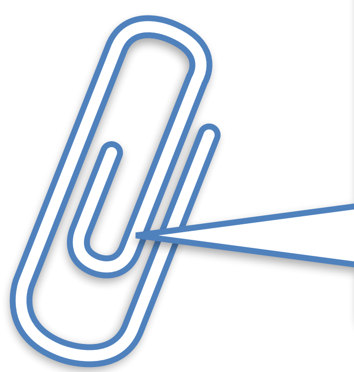


PID Windup is bad



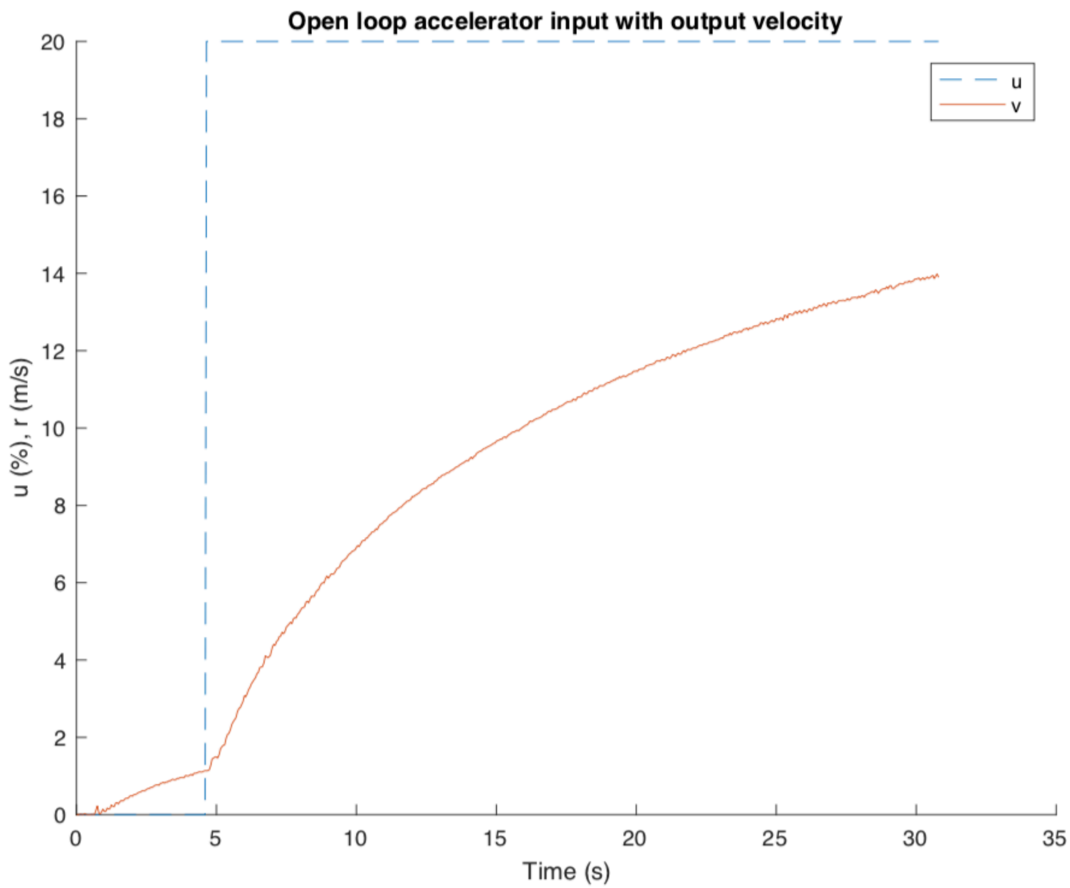
Software Interface Layers





It looks like you are trying to generate code from models.

System ID: 1 morning's worth of work



The expected range of use for the controller is $0 \leq v \leq 10$ m/s. The approach to system ID was taken from [3], and reference inputs ranging from $u_1 \in \{0.2, 0.3, 0.4\}$ were used to develop the following first-order model for velocity based on $u_1 > 0$:

$$G(s) = \frac{0.1333}{s + 0.5} \tag{1}$$

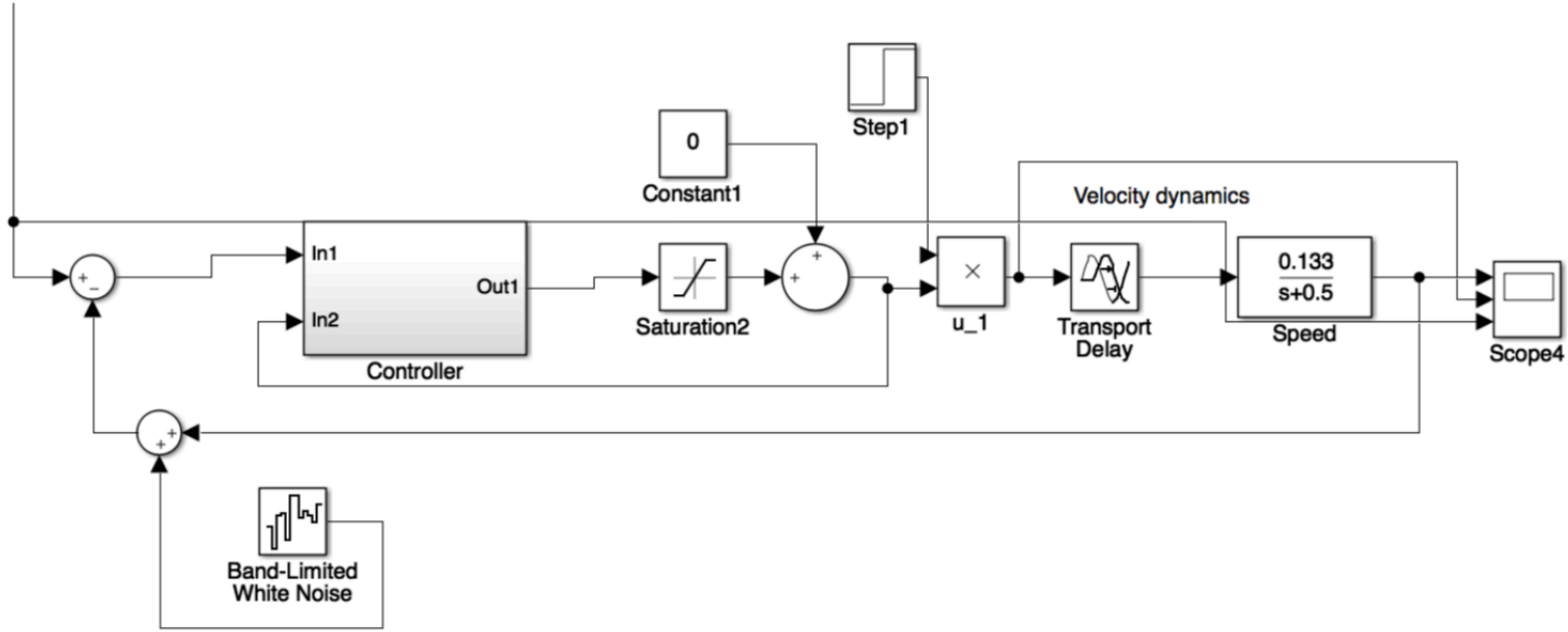


Fig. 4. Simulation of the open-loop transfer function under control by the switched system.

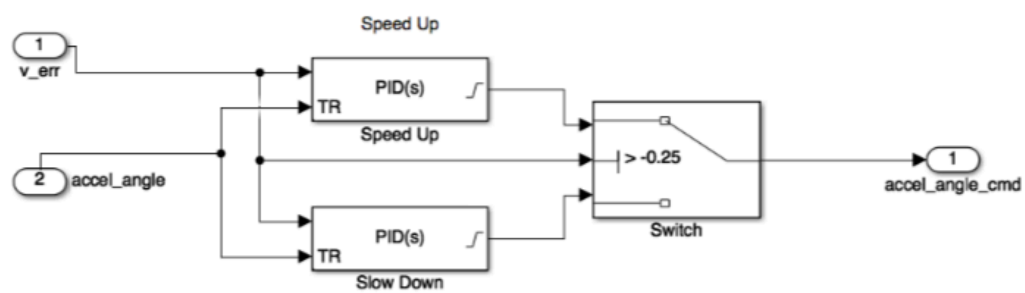
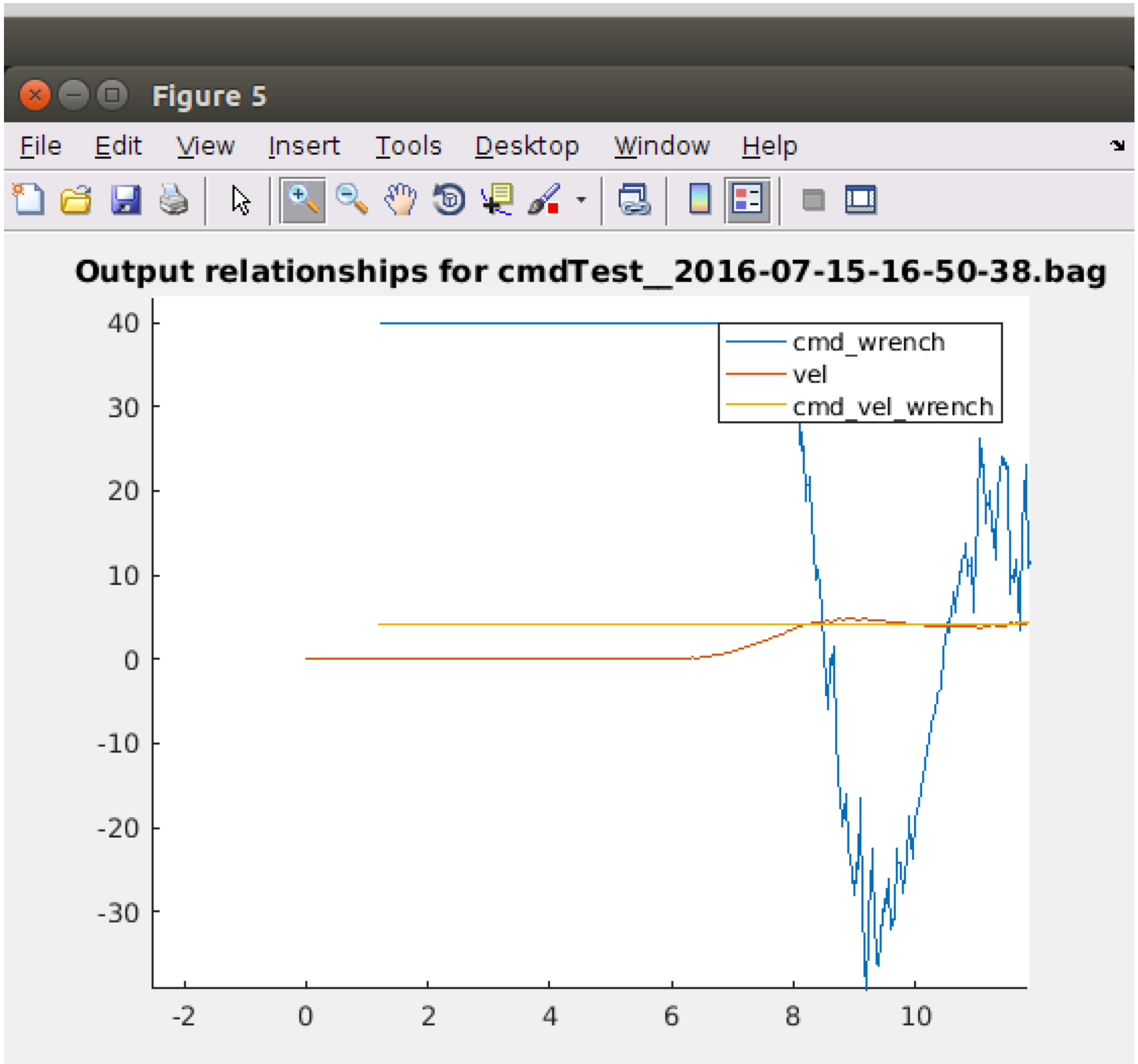


Fig. 2. A two-mode switched controller permits faster reaction to slow down, if the reference error is significant.

After clicking the "PID Windup" box in Simulink



Simulated response: SWIL

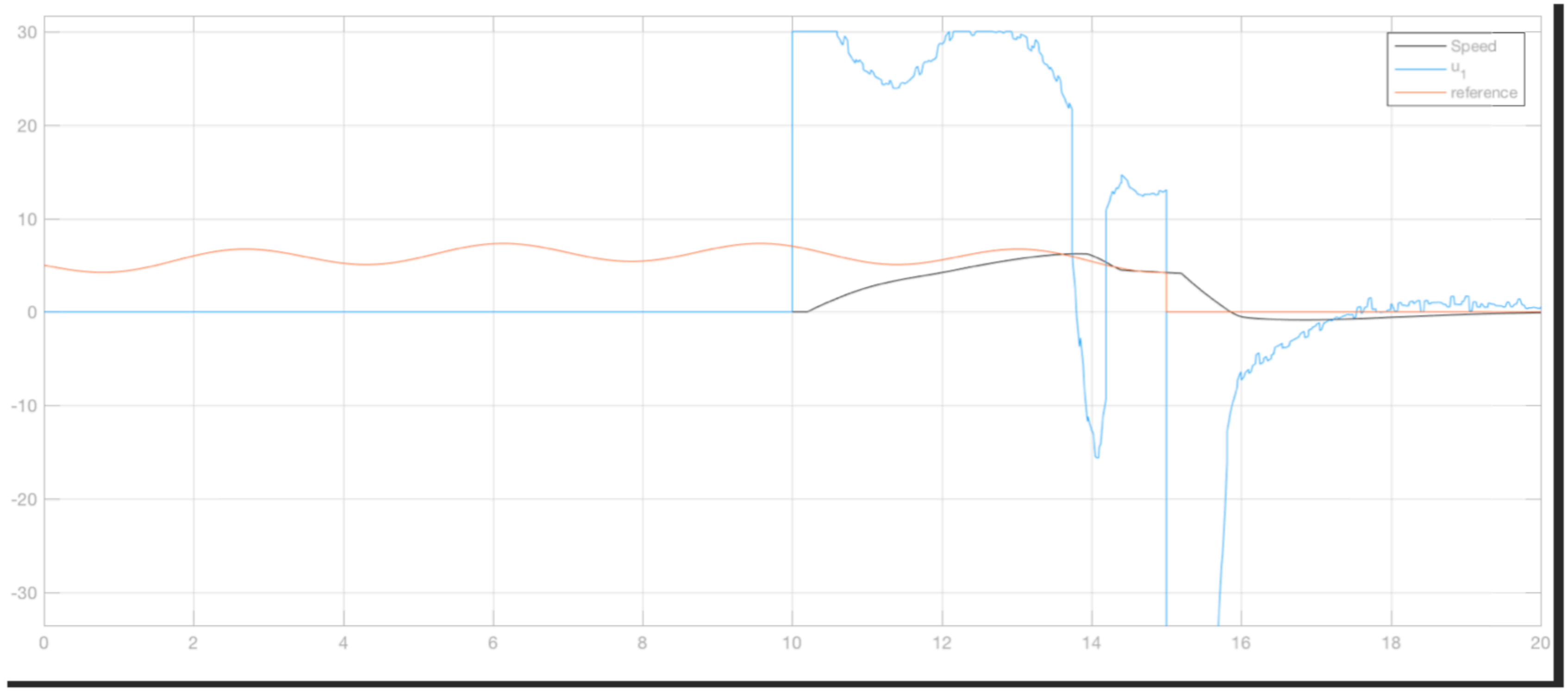


Fig. 5. The simulated system's tracking response demonstrates rejection of both sensor noise and delay, as well as avoidance of windup due to inactivity in the plant model if the controller is not active even though reference inputs are being received.

HWIL

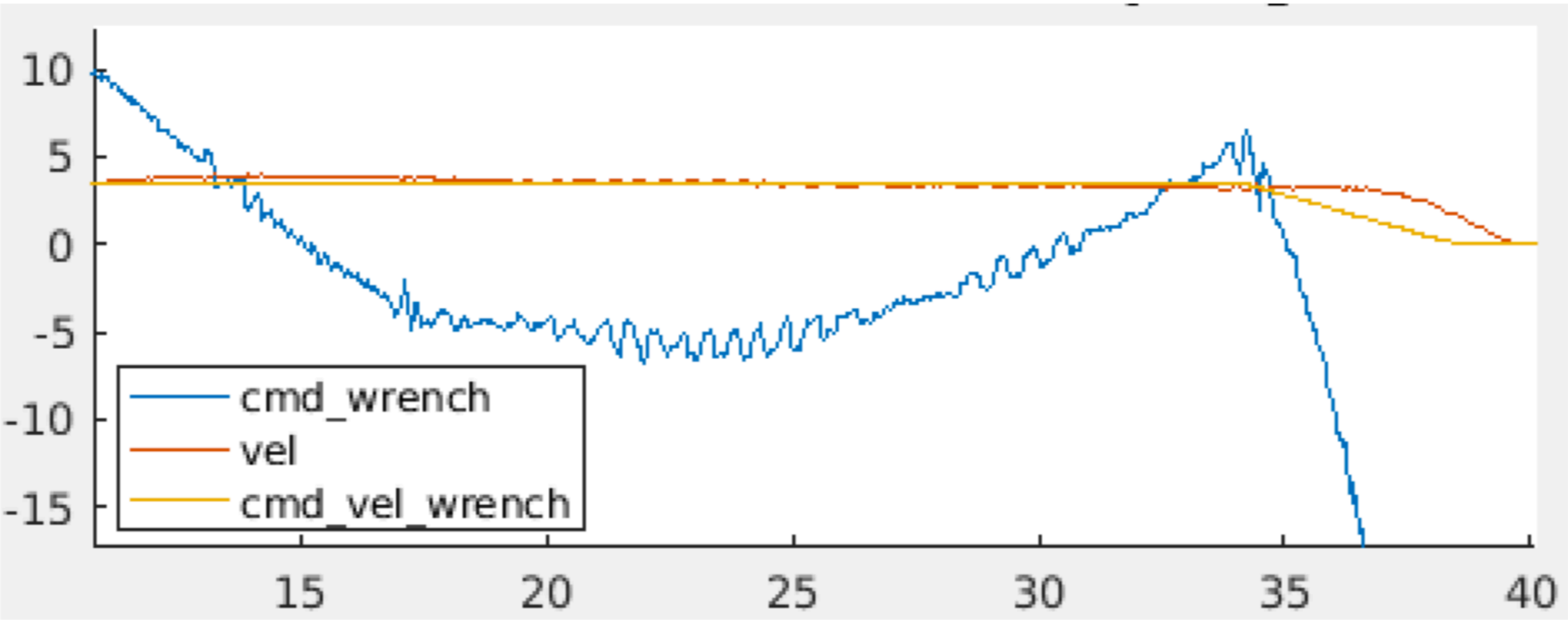
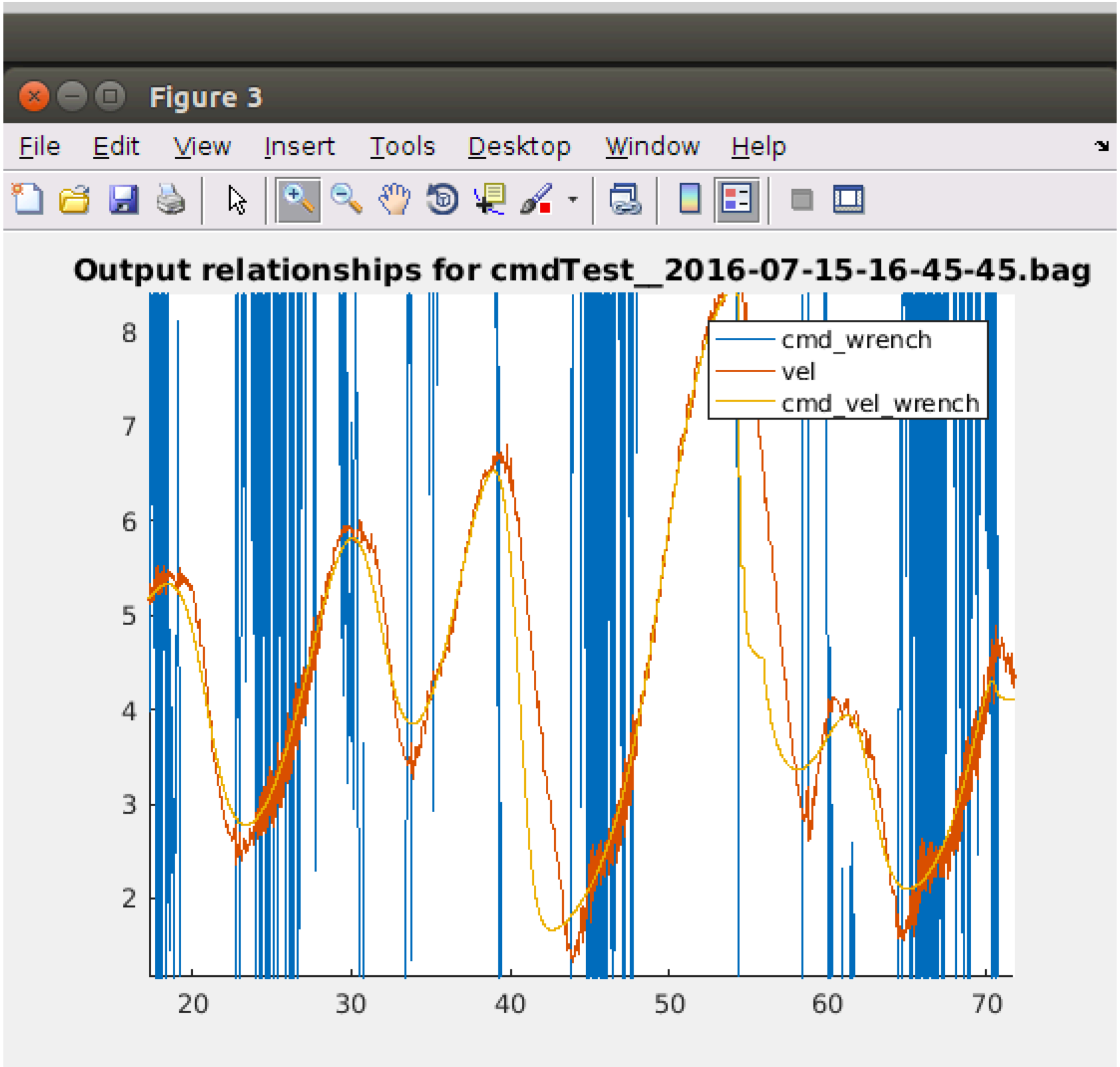


Fig. 6. Tracking a velocity at $3m/s$

Following a representative curve



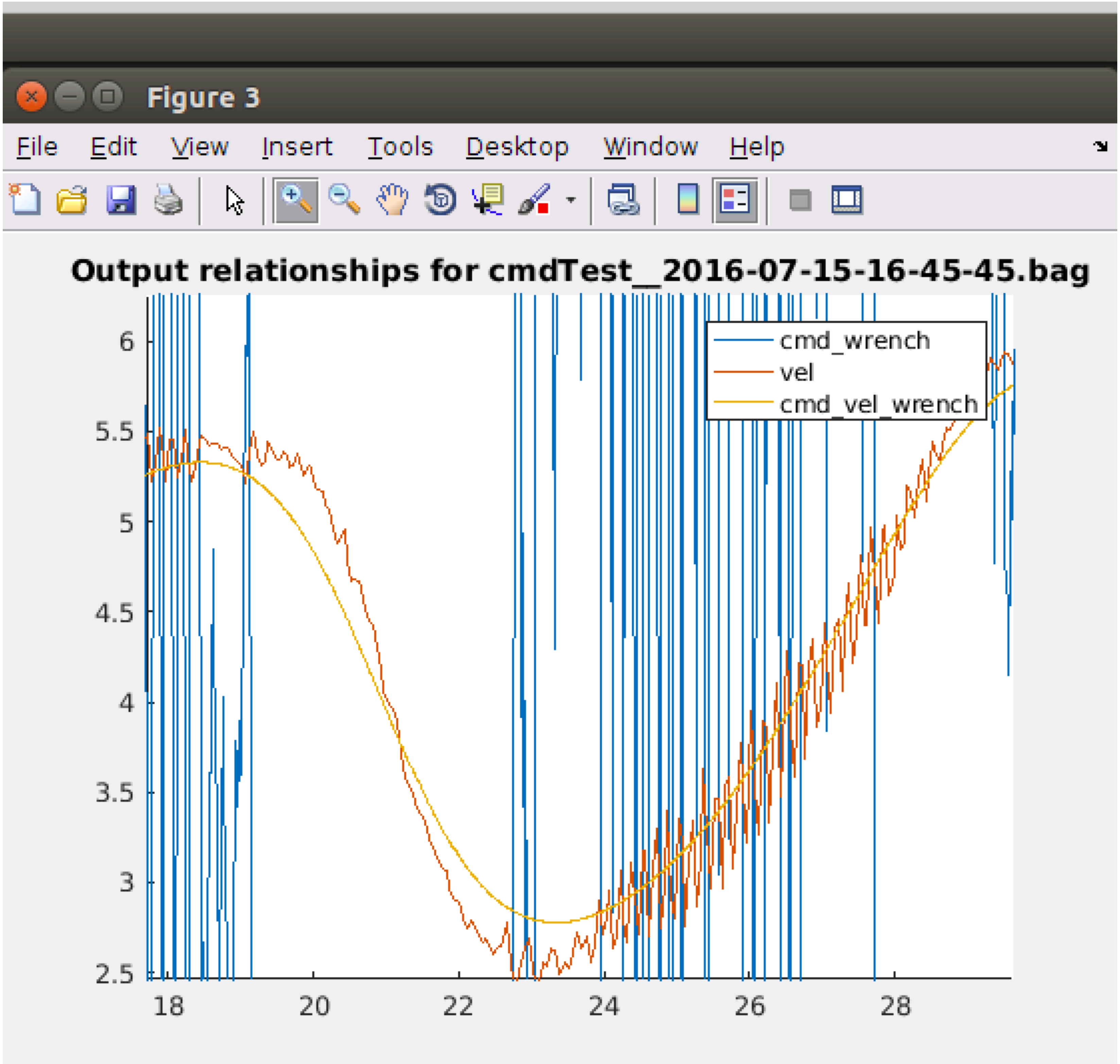
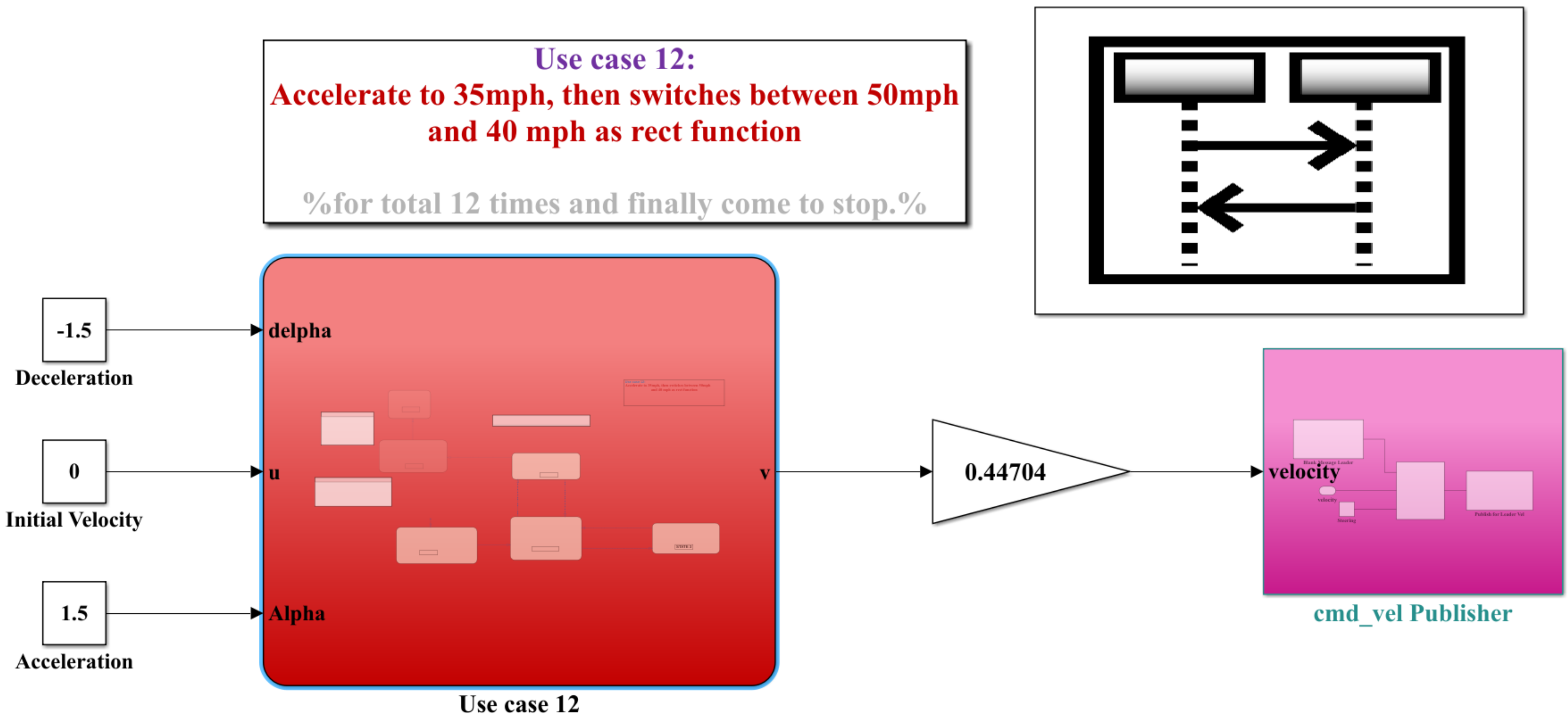
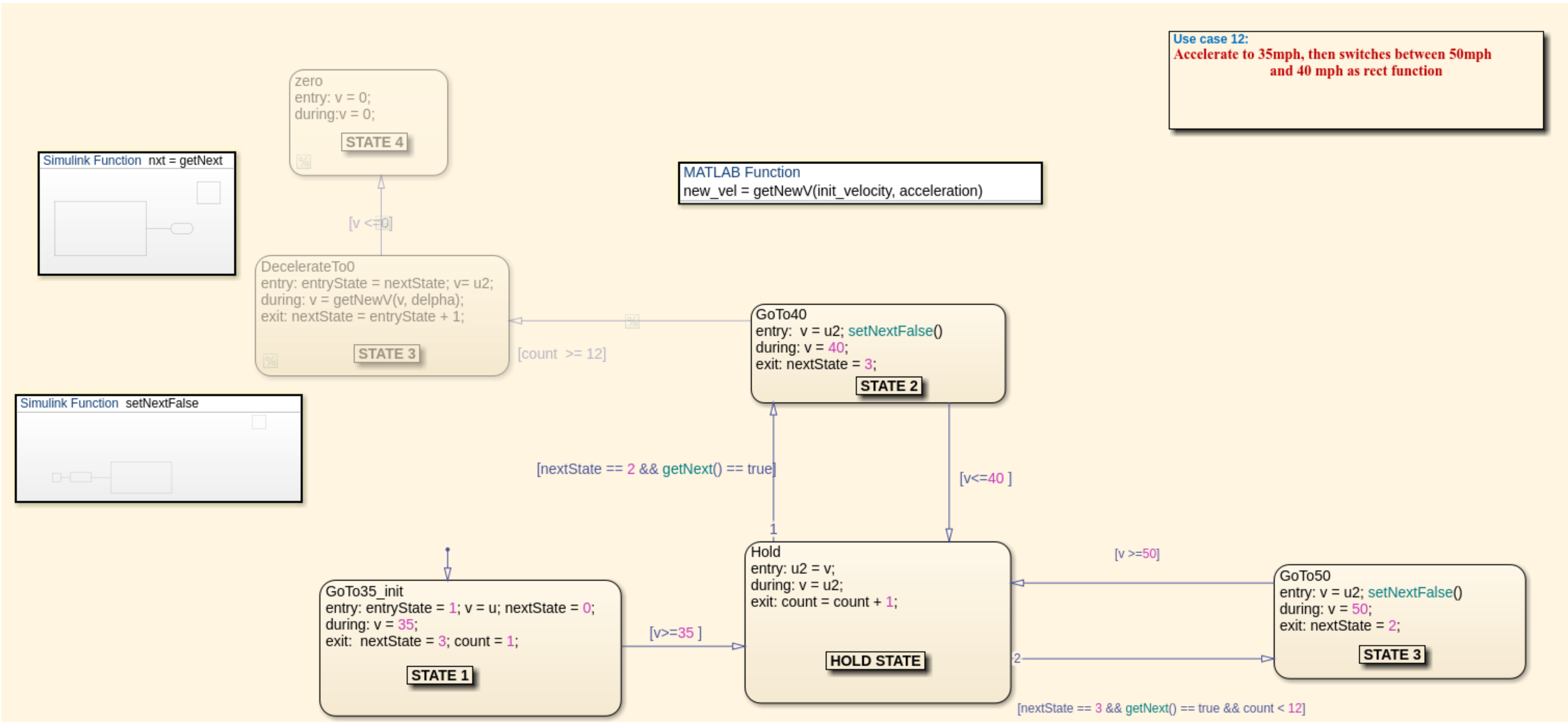


Figure 3
File
Edit
View
Insert
Tools
Desktop
Window
Help
Output relationships for cmdTest__2016-07-15-16-45-45.bag
cmd_wrench
vel
cmd_vel_wrench
6
5.5
5
4.5
4
3.5
3
2.5
18
20
22
24
26
28

High speed = High risk

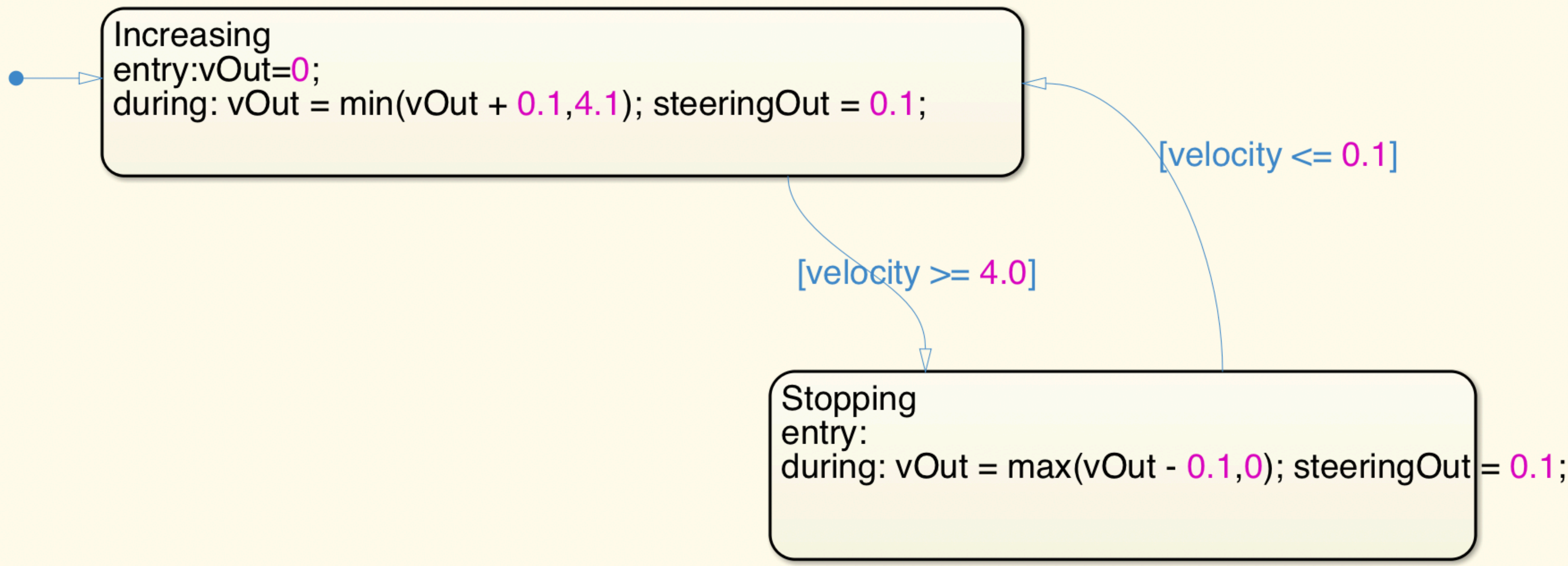
How would you choose to update the velocity in a lead vehicle, for consistency between experiments?


Use case 12:
Accelerate to 35mph, then switches between 50mph and 40 mph as rect function
%for total 12 times and finally come to stop.%
-1.5
Deceleration
0
Initial Velocity
1.5
Acceleration
delpha
u
Alpha
v
Use case 12
0.44704
velocity
cmd_vel Publisher

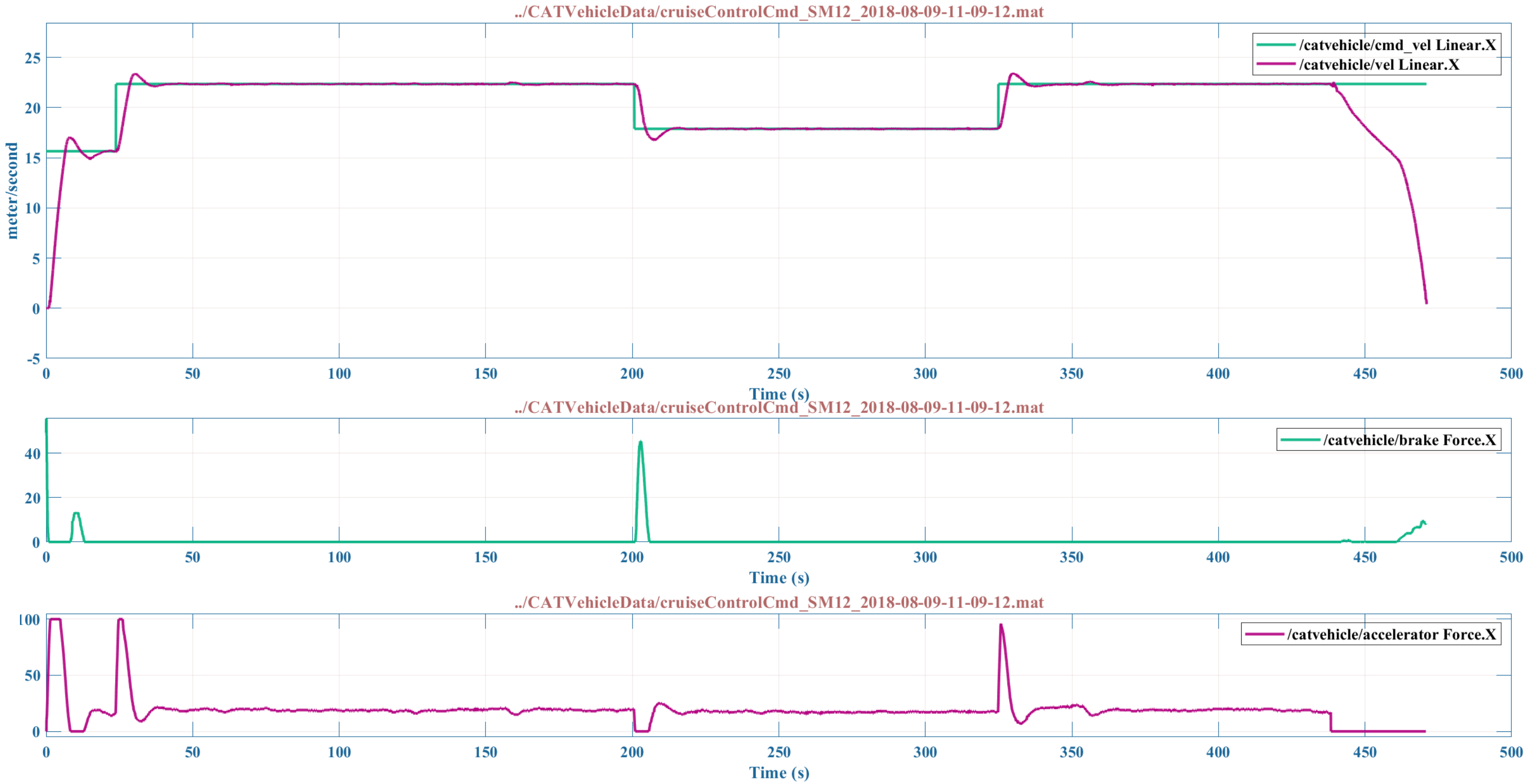


Use case 12:
Accelerate to 35mph, then switches between 50mph and 40 mph as rect function
Simulink Function nxt = getNext
Simulink Function setNextFalse
zero
entry: v = 0;
during:v = 0;
STATE 4
[v <= 0]
DecelerateTo0
entry: entryState = nextState; v= u2;
during: v = getNewV(v, delpha);
exit: nextState = entryState + 1;
STATE 3
[count >= 12]
MATLAB Function
new_vel = getNewV(init_velocity, acceleration)
GoTo40
entry: v = u2; setNextFalse()
during: v = 40;
exit: nextState = 3;
STATE 2
[nextState == 2 && getNext() == true]
[v<=40]
1
Hold
entry: u2 = v;
during: v = u2;
exit: count = count + 1;
HOLD STATE
2
[v >=50]
GoTo50
entry: v = u2; setNextFalse()
during: v = 50;
exit: nextState = 2;
STATE 3
[nextState == 3 && getNext() == true && count < 12]
GoTo35_init
entry: entryState = 1; v = u; nextState = 0;
during: v = 35;
exit: nextState = 3; count = 1;
STATE 1
[v>=35]

Close up example: model-based approaches



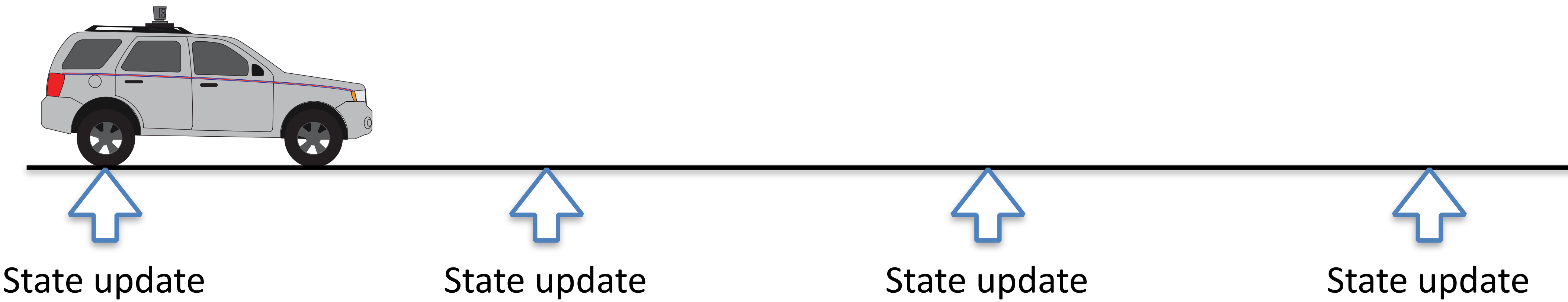
HWIL tests at high speed, made possible with models



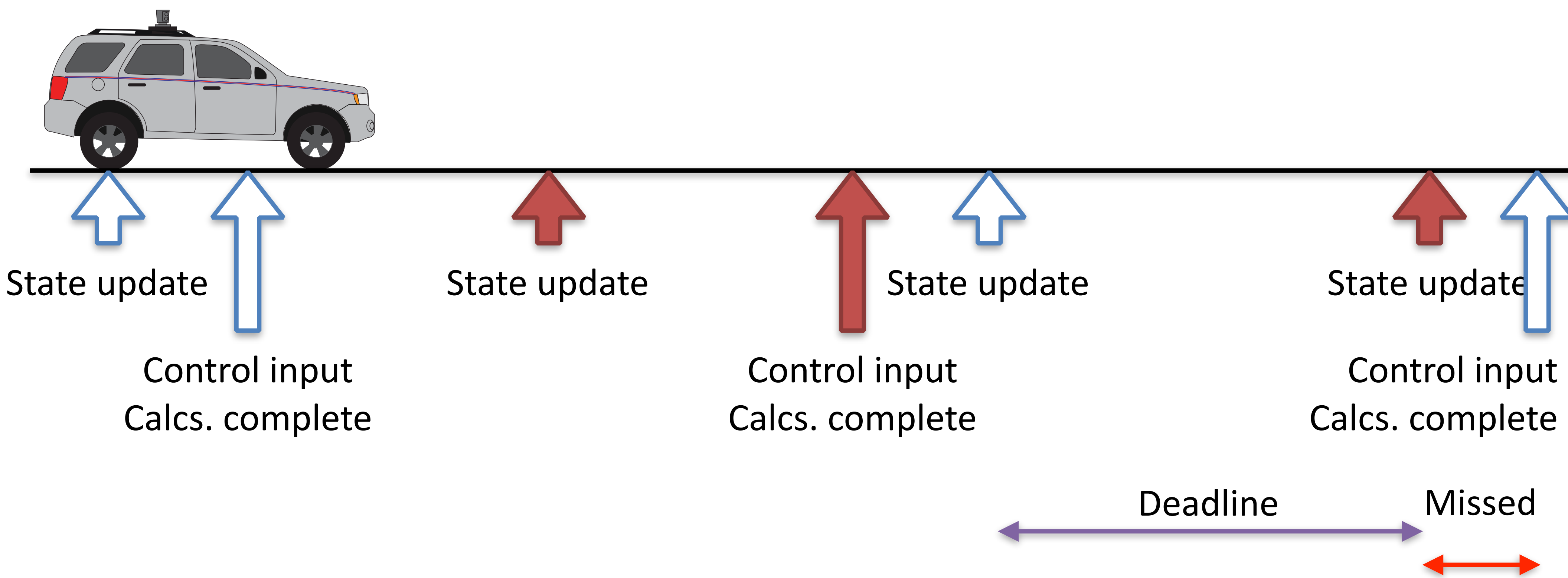
Challenges with Integration and advanced control

THE UNIVERSITY OF ARIZONA

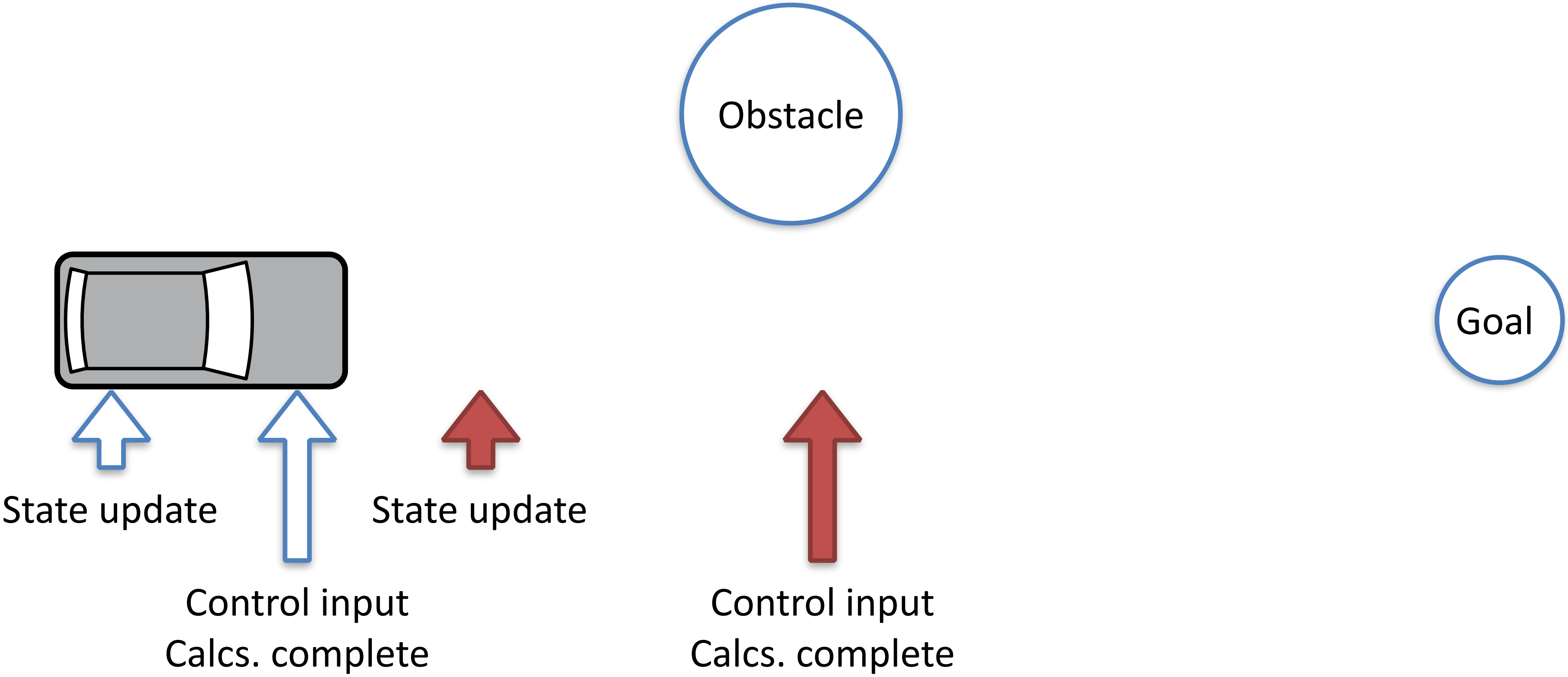
Consider a moving ground vehicle



When computing a control decision...



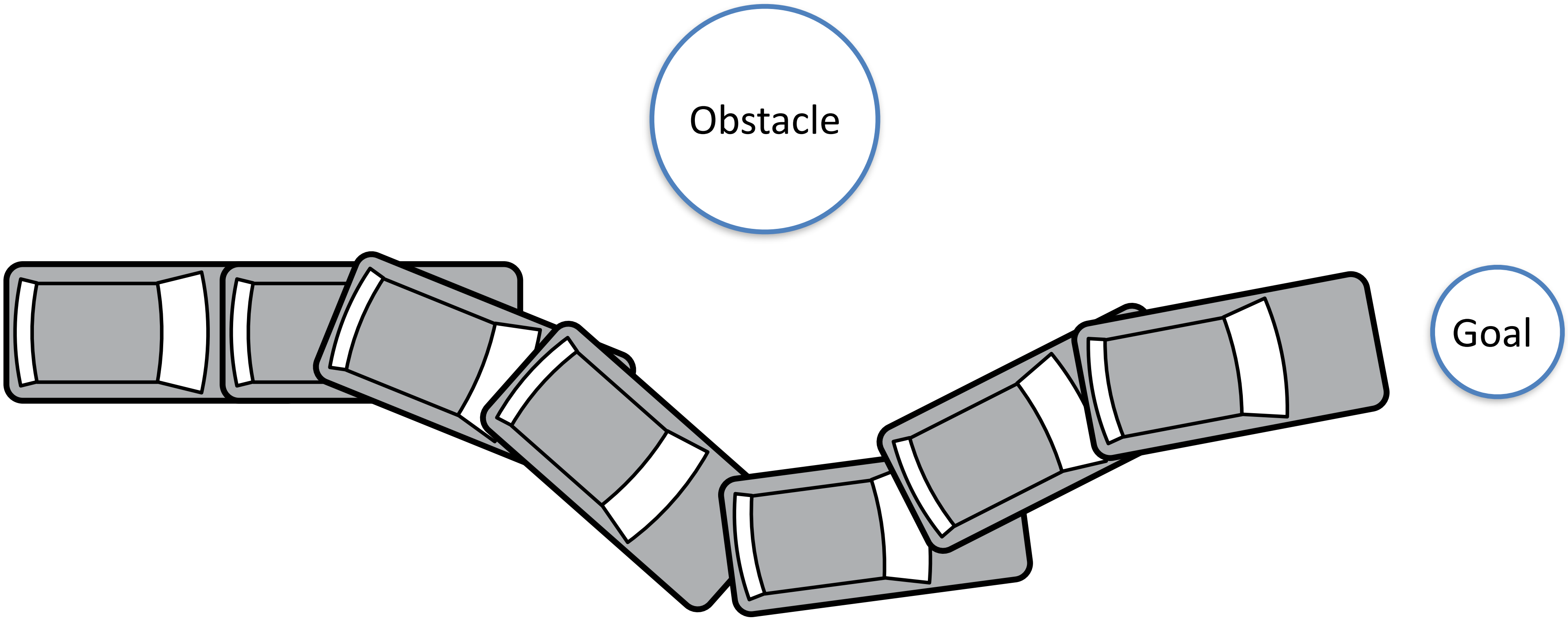
What if an obstacle moves while we compute?



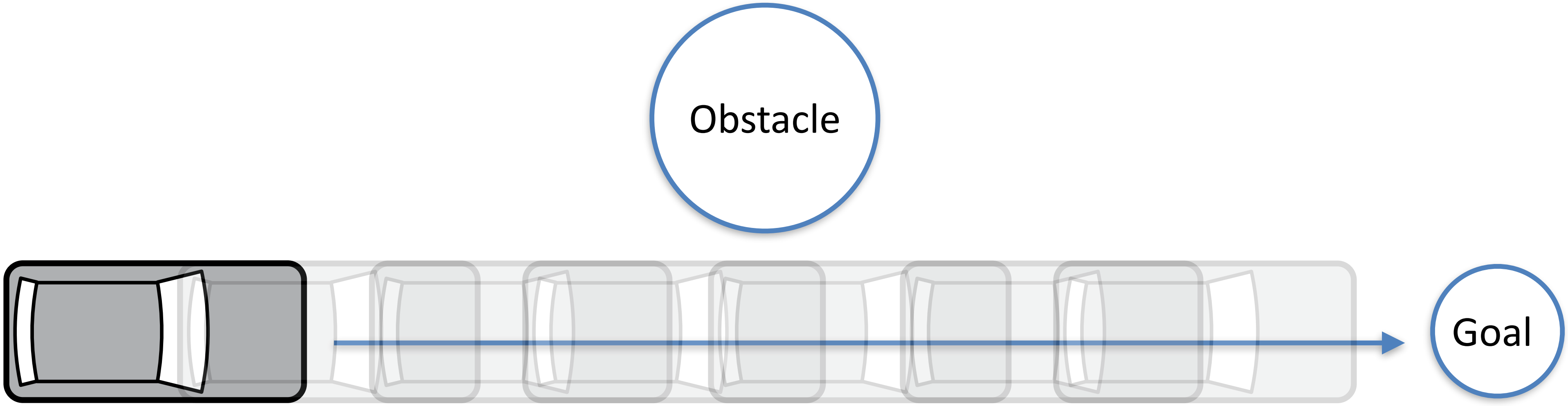
Obvious design goal: Reduce the period for state update/control input as much as possible.

This is a motivating animation, not a kinematic simulation.

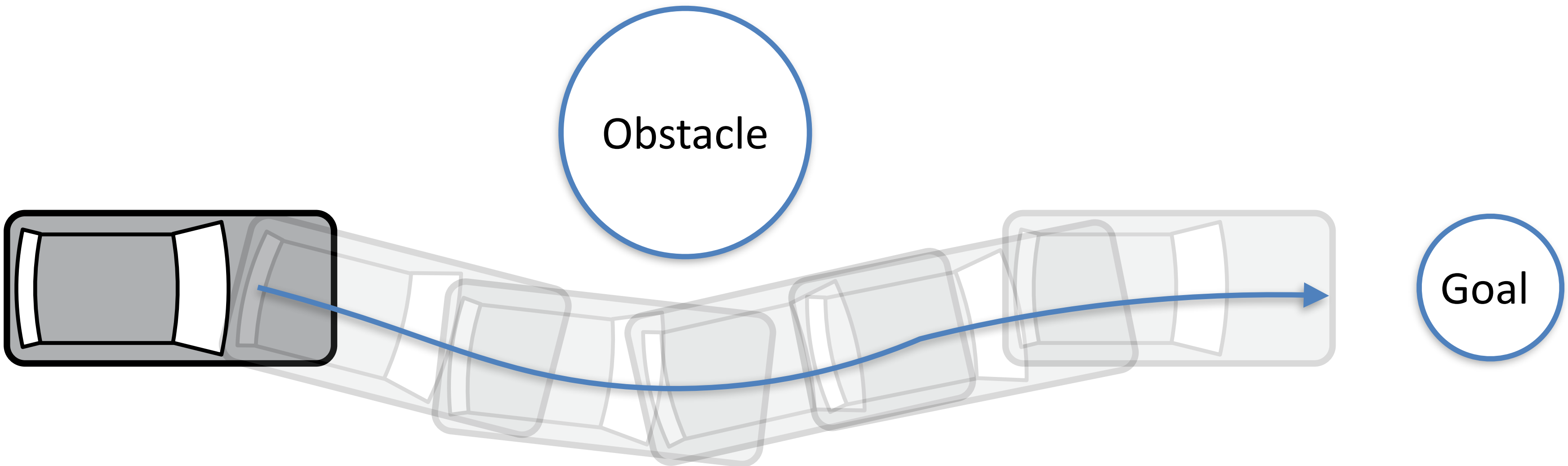
Model-Predictive Control: Plan Trajectories at Runtime



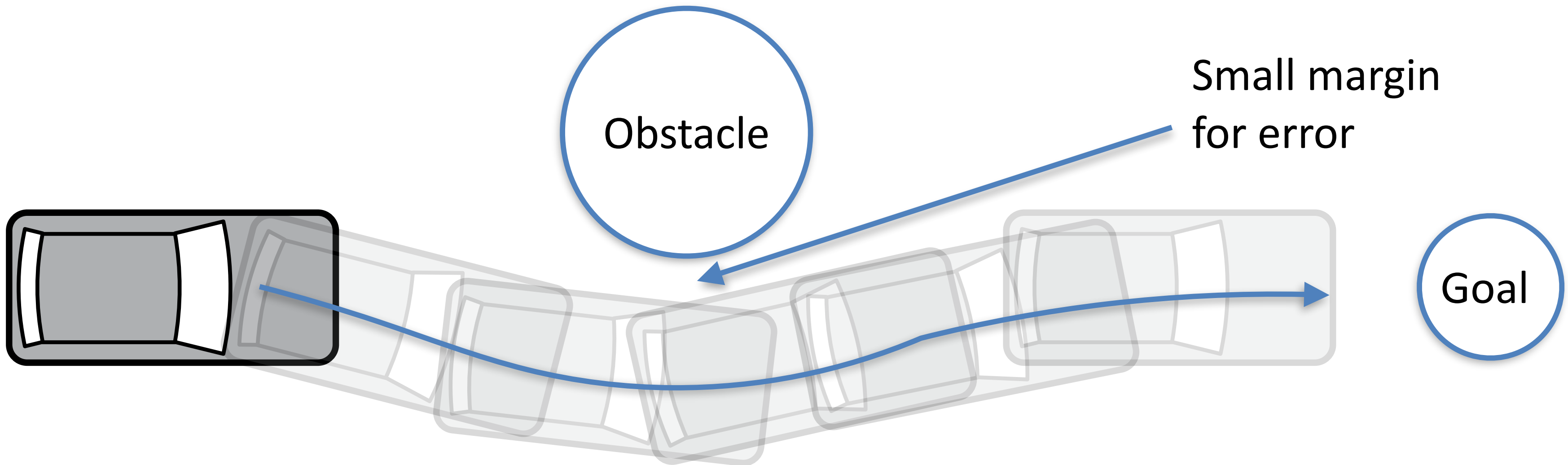
Model-Predictive Control: Plan Trajectories at Runtime



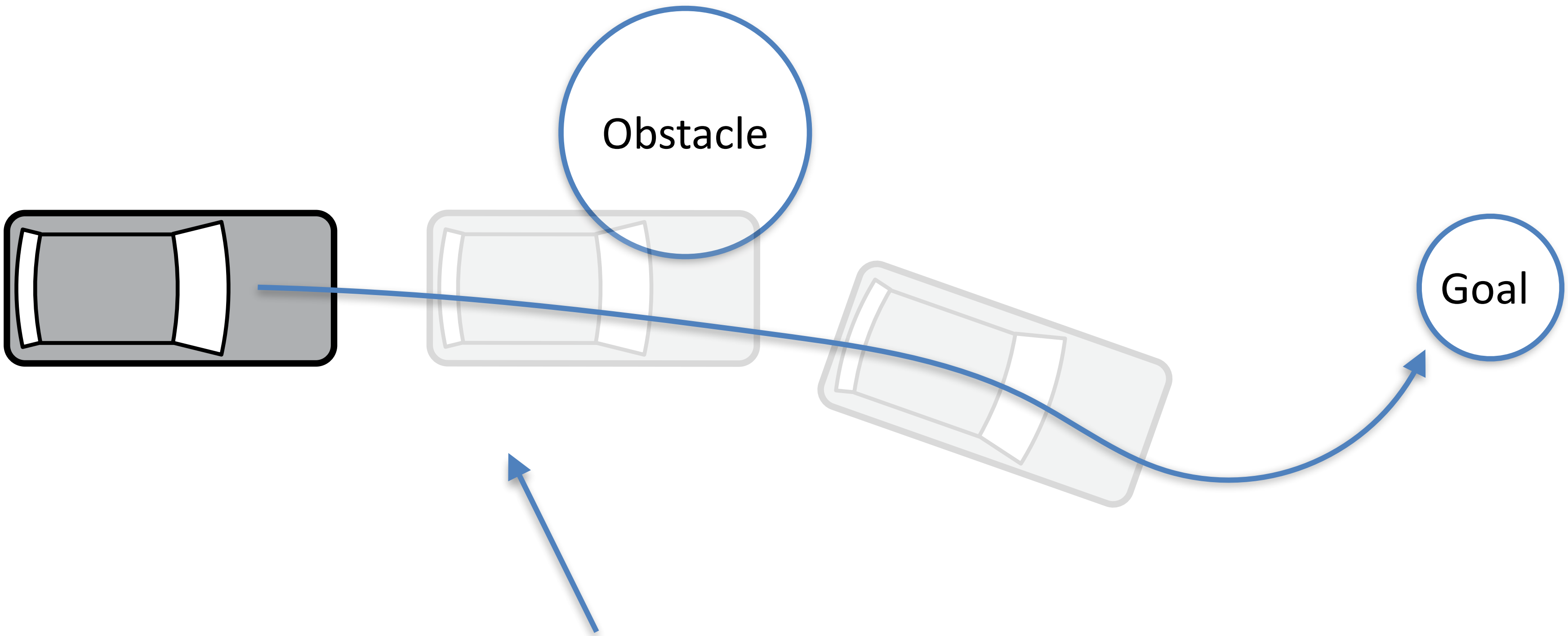
Model-Predictive Control: Plan Trajectories at Runtime



Model-Predictive Control: Plan Trajectories at Runtime

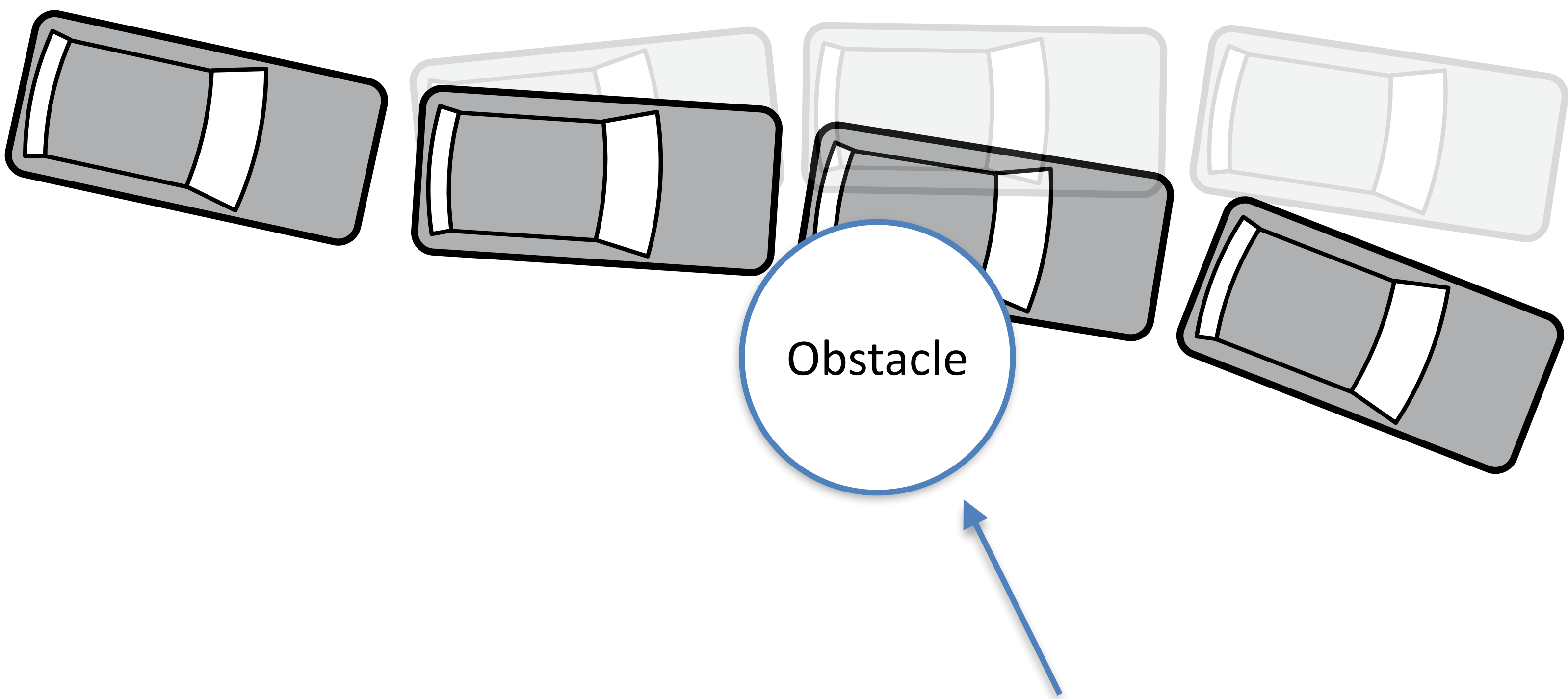


Easy…I'll use an accurate vehicle model to predict the trajectory and avoid the obstacle.



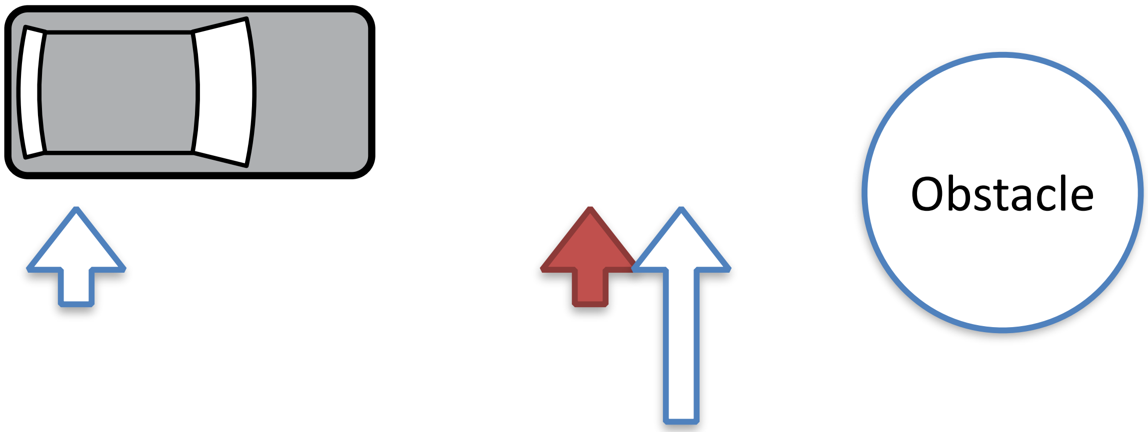
Takes longer to compute control inputs

Easy…I'll pick a simpler vehicle model to make it more likely to return control inputs in time

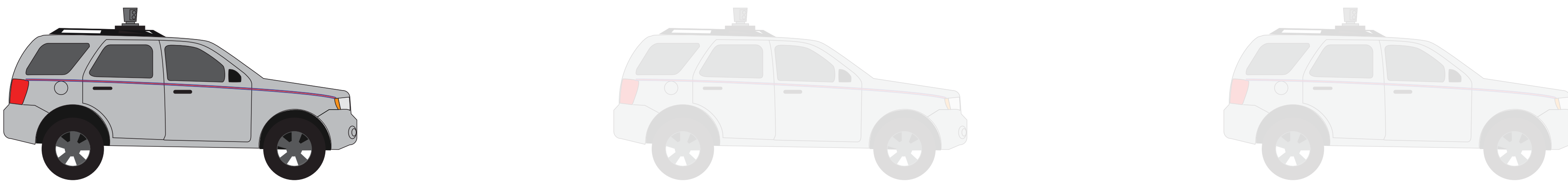


That control input does not mean what you think it means.

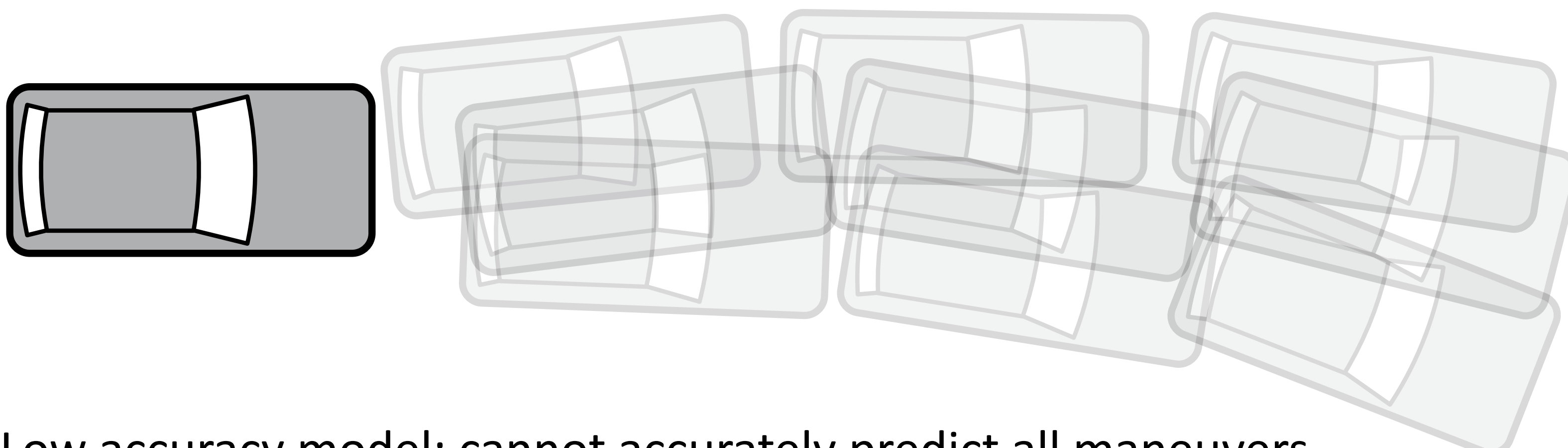
Competing constraints



High accuracy model: takes longer to optimize.



High vehicle speed: cannot tolerate slow return time.



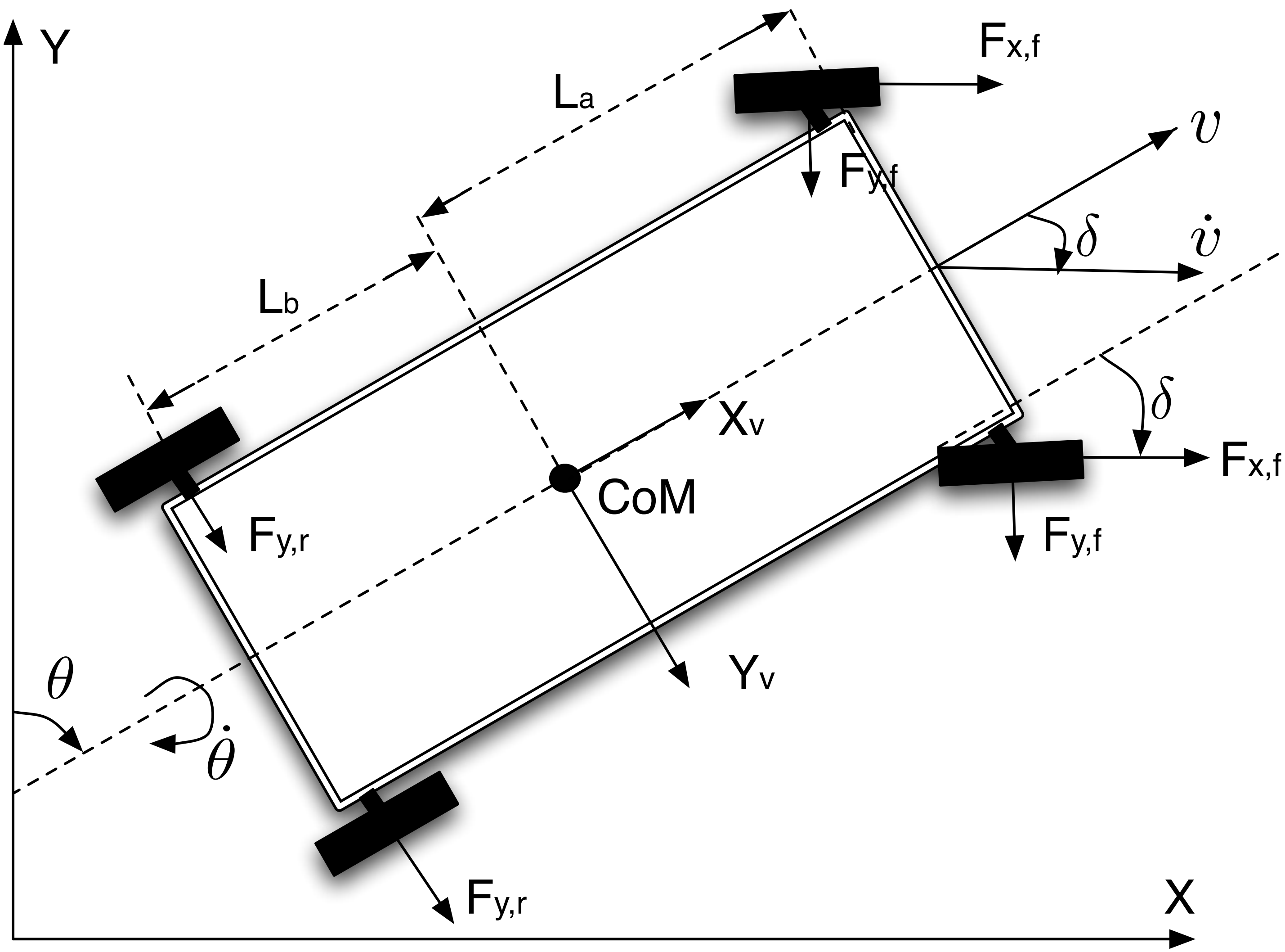
Low accuracy model: cannot accurately predict all maneuvers.

Kinematic/Dynamical Models

$$\dot{\xi} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

$$\dot{\xi} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$

$$\dot{\xi} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta)+2F_{y,f}\cos(\delta)\right)-2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$



$$\xi_{k,t+1} = \hat{f}_q(\xi_{k,t}, u_{k,t})$$
$$t \in \{k, k+1, \cdots, k+N-1\}$$

MPC solves the optimization problem $\mathbf{P}^q(\xi_k)$ at time k by using the model $\hat{f}_q$. We denote the input sequence $\{u^q_{k,k}, u^q_{k,k+1}, \cdots, u^q_{k,k+N-1}\}$ by U^q_k, and formulate the following problem:

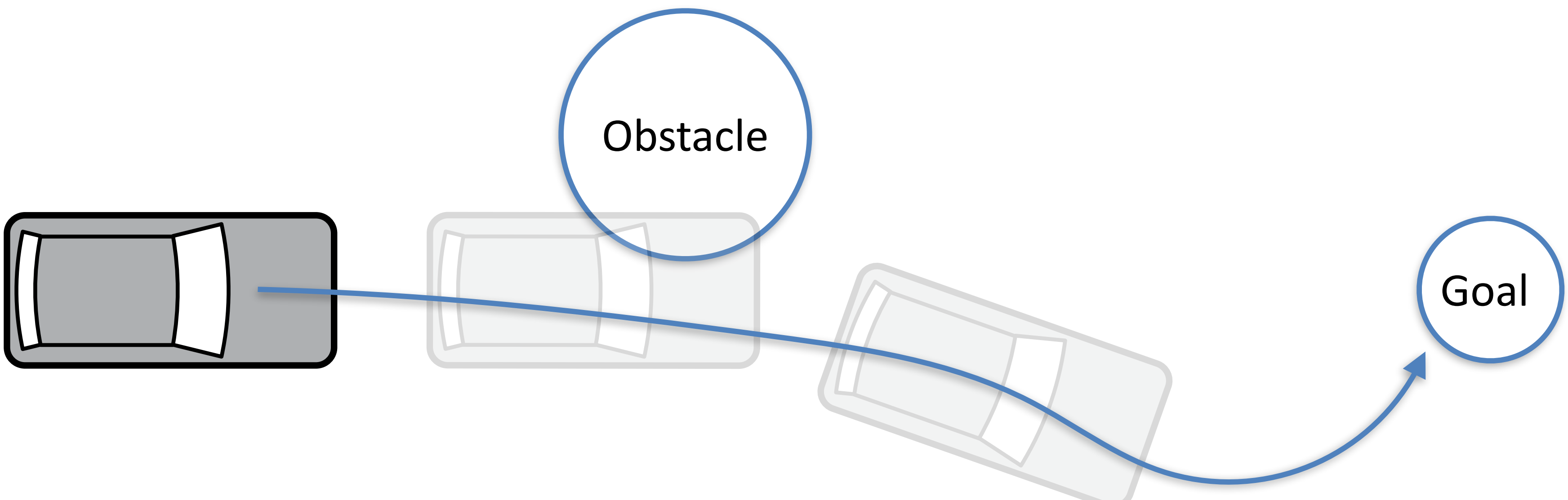
$$\mathbf{P}^q(\xi_k) : \underset{U^q_k}{\operatorname{argmin}} \{J_N(\xi_k, U^q_k) : U^q_k \subset \mathbb{R}^m\}$$

$$J_N(\xi_k, U^q_k) = \sum_{t=k}^{k+N-1} \ell(\xi^q_{k,t}, u^q_{k,t}) + F(\xi^q_{k,k+N})$$

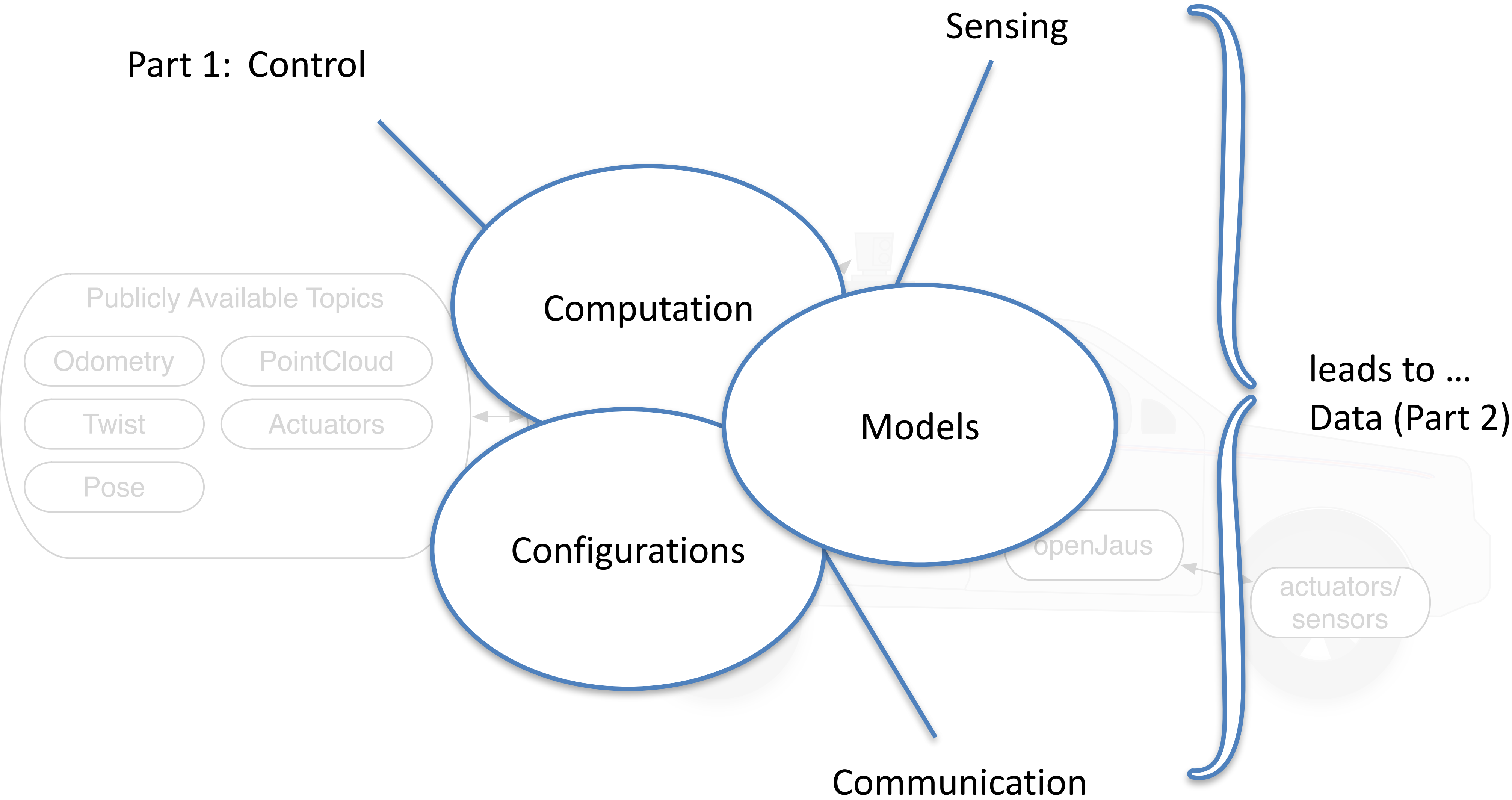
$$U^{q*}_k = \{u^{q*}_{k,k}, u^{q*}_{k,k+1}, \cdots, u^{q*}_{k,k+N-1}\}$$

Receding Horizon Control/MPC

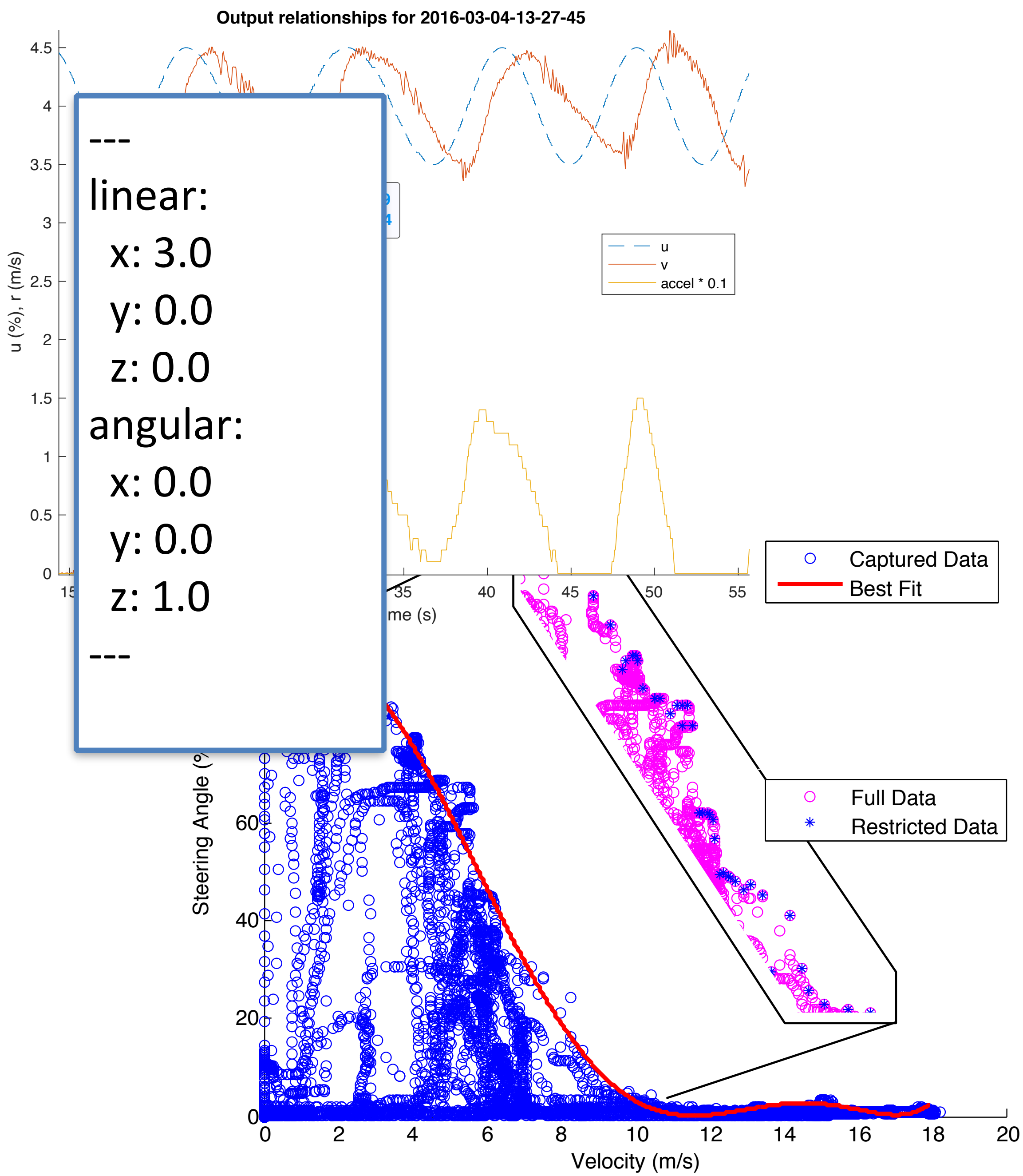
- Useful when looking for optimal(ish) solutions to competing constraints
- Can be challenging if nonconvex in the optimization function, or if competing constraints result in bifurcation of the solution
- May not return a value in time!!



What's Next

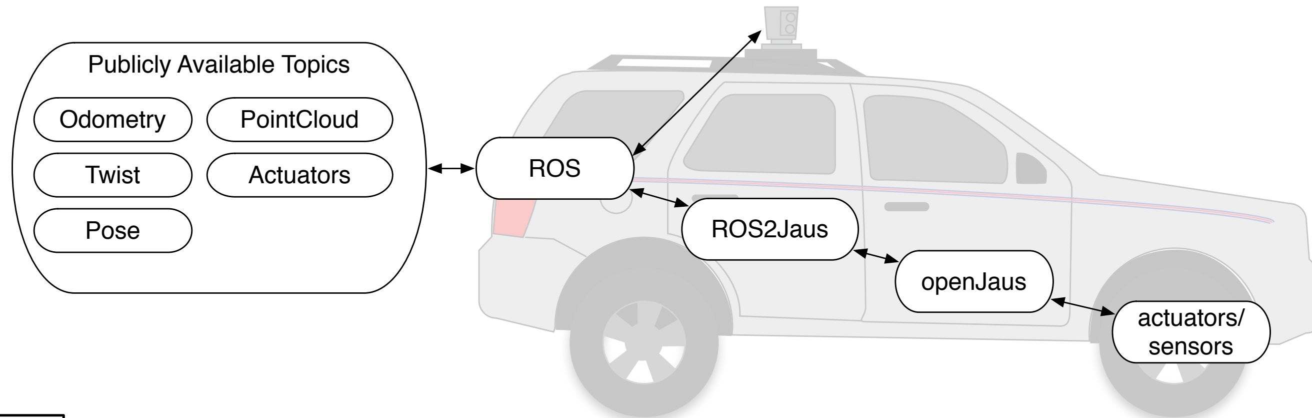


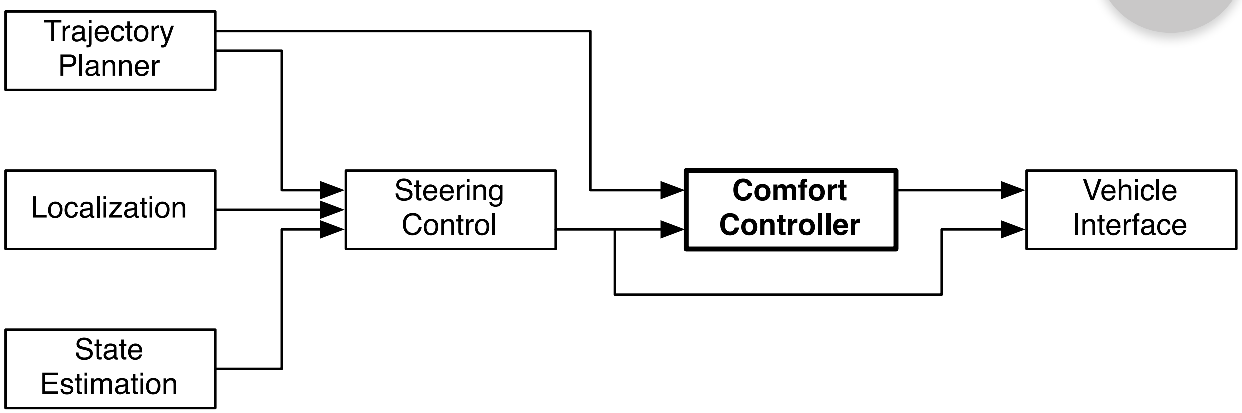
Part 1: Control
Sensing
Computation
Models
Configurations
Communication
leads to …
Data (Part 2)
Publicly Available Topics
Odometry
PointCloud
Twist
Actuators
Pose
openJaus
actuators/
sensors



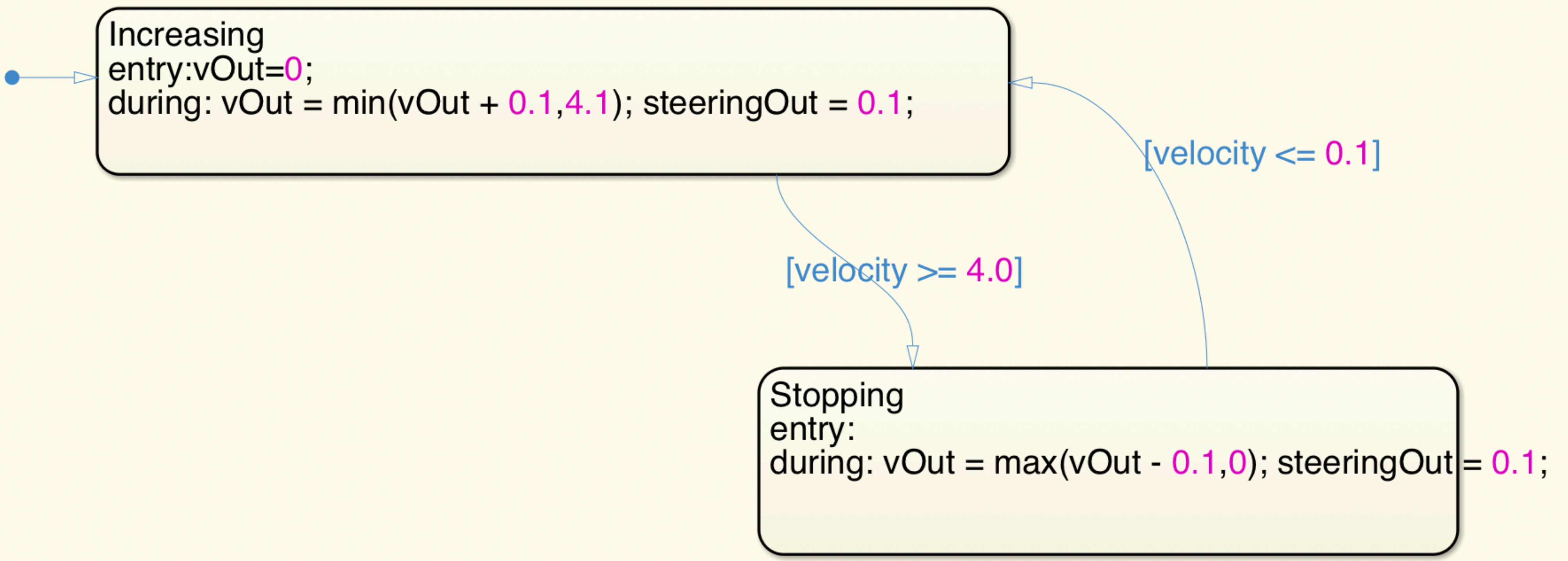
Extracting data, understanding how it can be interpreted and used in learning and validation.

Looking at interface layers and architectures for where to insert control





Exploring how to use models to build new abstractions that lead to new properties



Thanks for the Support

"CAREER: Domain-Specific Modeling Techniques for Cyber-Physical Systems" NSF CNS-1253334

"REU Site: CatVehicle: Cognitive and Autonomous Test Vehicle" NSF IIS-1262960, NSF CNS-1659428

Additional support for awards CNS-1253334 and IIS-1262960 provided by the Air Force Office of Scientific Research

"CPS: Synergy: Control of Vehicular Traffic Flow via Low Density Autonomous Vehicles" NSF CNS-1446435, 1446690, 1446702, 1446715.

Generous support for the CatVehicle Challenge was provided by MathWorks

Amazing photography by John de Dios and Alan Davis, at Davis de Dios Media

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or AFOSR.

More Awesome stuff

jmscslgroup

http://catvehicle.arizona.edu/

http://csl.arizona.edu