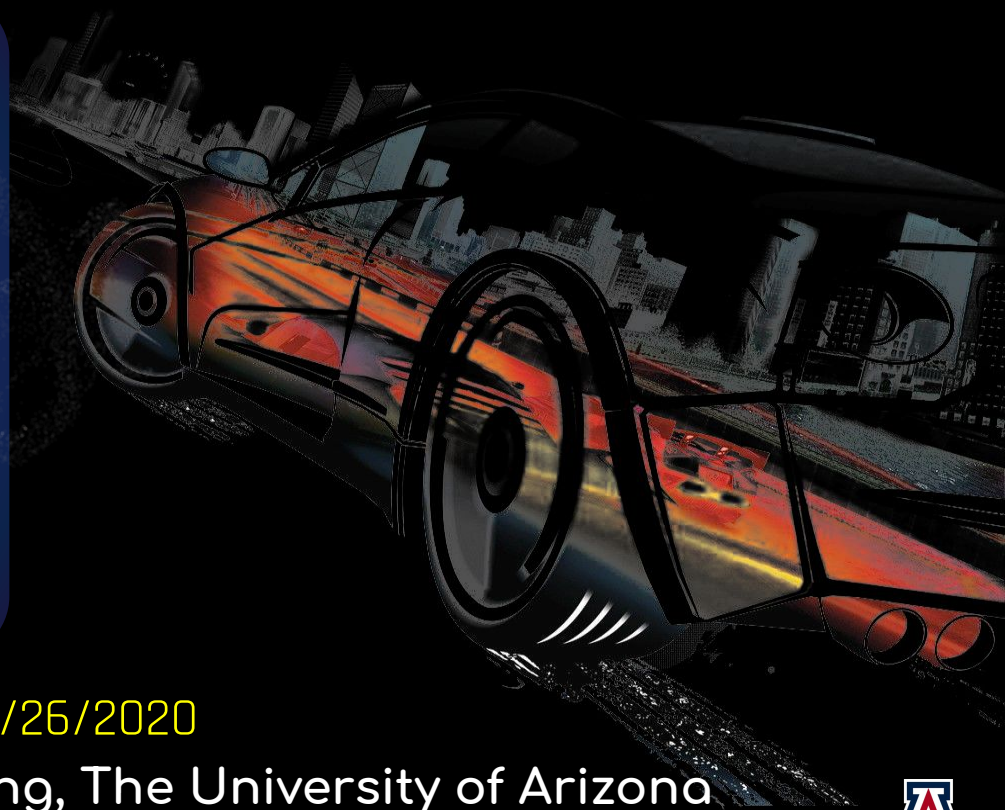
Large-scale simulation for multi-vehicle applications



Rahul Bhadani, Jonathan Sprinkle 10/26/2020

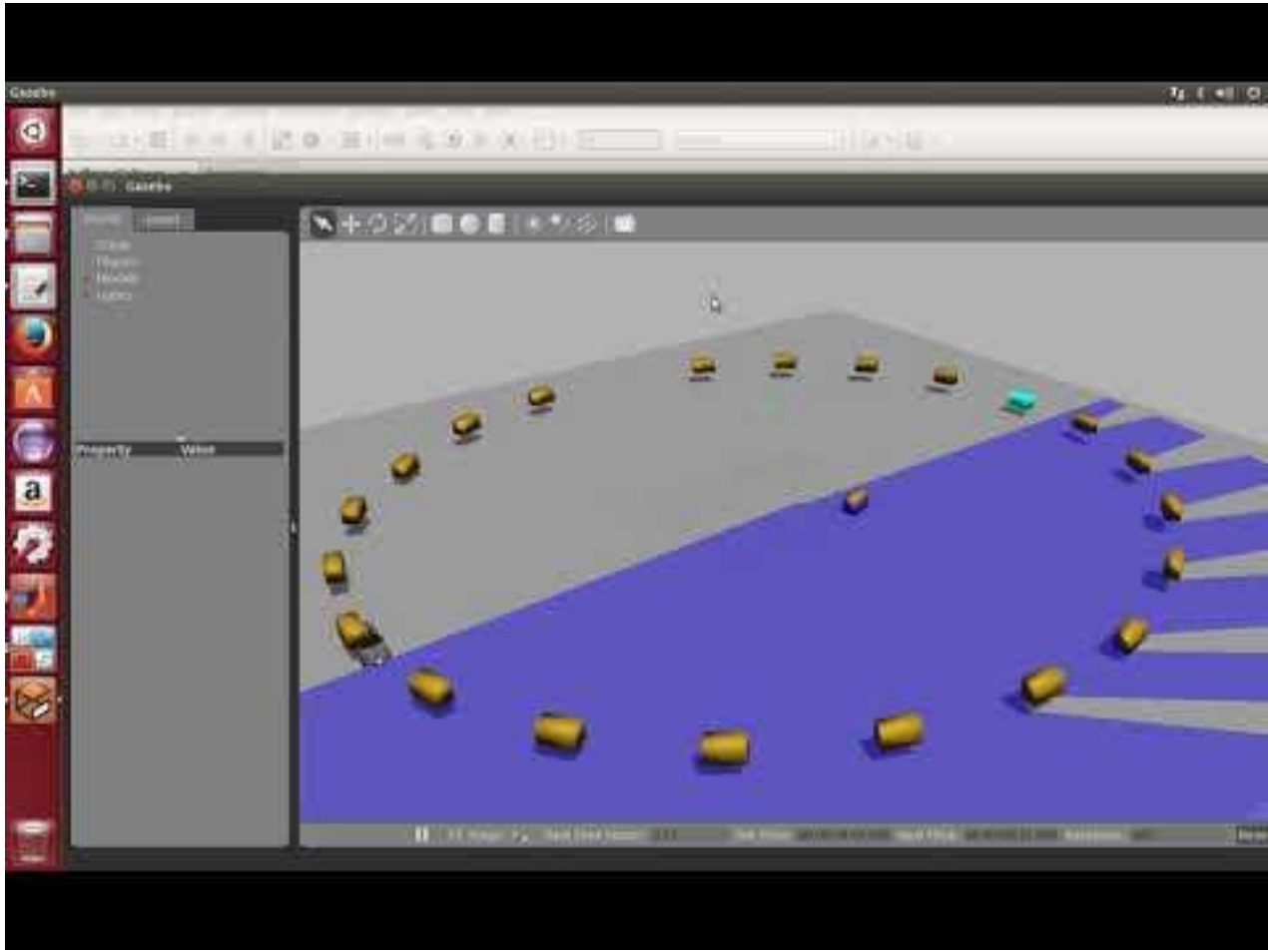
Electrical & Computer Engineering, The University of Arizona

Imagine the future…

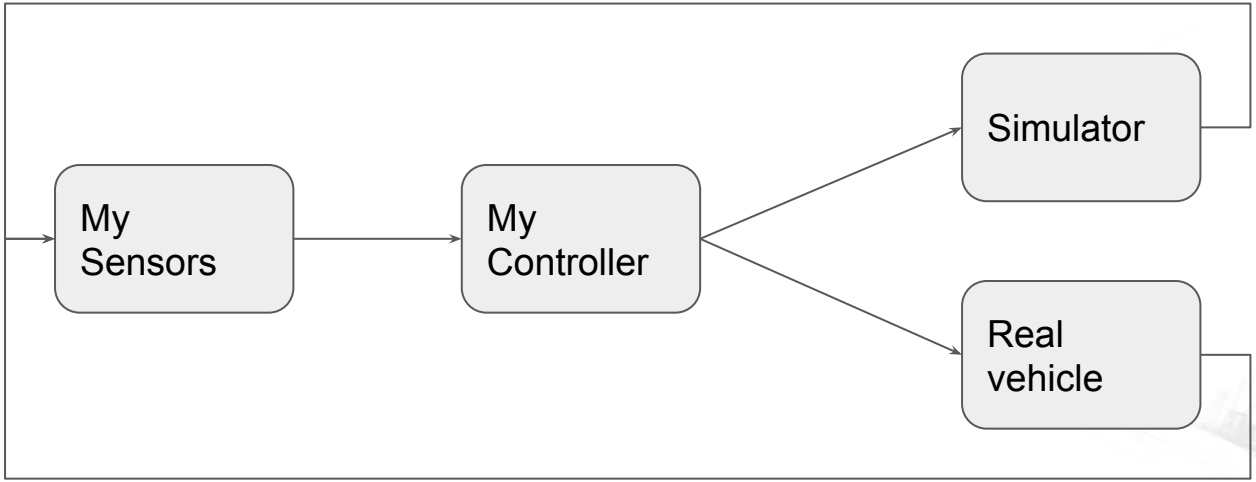
https://www.youtube.com/watch?v=2mBjYZTeaTc&feature=emb_logo

That depends on advanced testing...

https://www.youtube.com/watch?v=XBQ95pj5FCA&feature=emb_logo



What do I need?

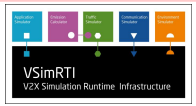


And…it should produce the same answer every time (within reason)!

Application development for connected and autonomous vehicles (CAV)

Simulation

- Unlimited use cases ✓
- Approximation of physical processes ✓
- Safety ✓
- ✗ Sensor simulation support
- Sensor Simulation Support ✓



veins
Vehicles in Network Simulation

aimsun.

SUMO
SIMULATION OF URBAN MOBILITY

ROS

GAZEBO

UNREAL ENGINE

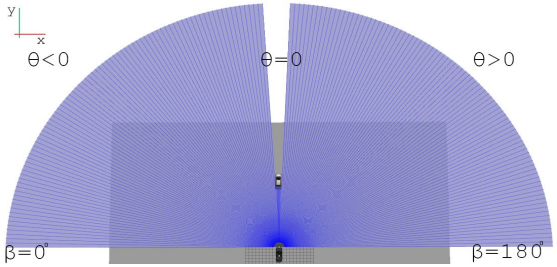
unity

Field Operation Tests

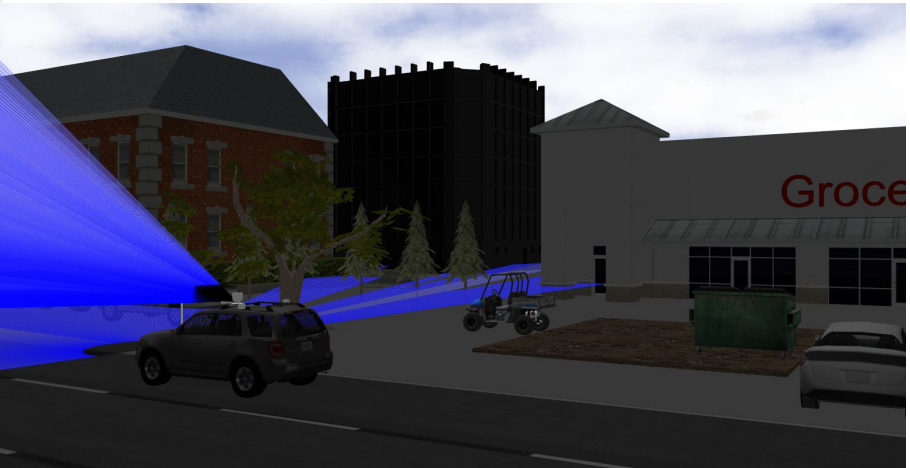
- Real Vehicle/Hardware ✓
- Negligible approximation of physical processes ✓
- ✗ Significant Costs
- ✗ Scalability issue
- ✗ Reproducibility issue

Physics-enabled simulation for CAV

- Real-life 3D scenario
- Vehicle interaction in real-time
- Emulate inter-vehicle communications for CAV



Schematic of LiDAR scanning



Sensor-enabled environment sensing

Real-time message exchange with vehicles and other simulated infrastructure

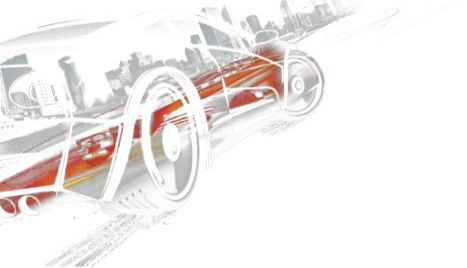
Device-driver assisted hardware-in-the-loop testing

How is it done currently?

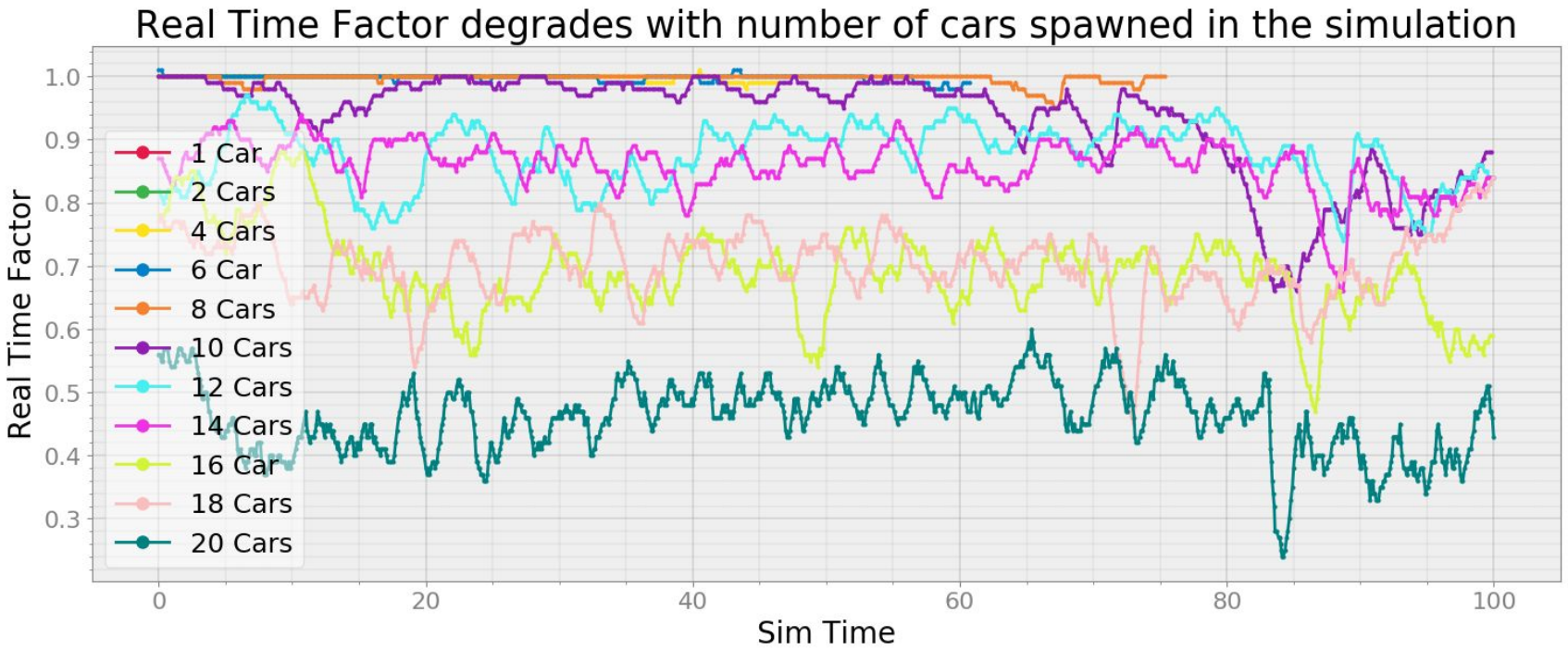
- Autonomous driving simulation in 3D world with simulated sensor using ROS/Gazebo is most popular.
- One such simulator with ROS/Gazebo: CAT Vehicle Testbed < https://github.com/jmscslgroup/catvehicle >
- Other popular simulators with ROS integration: Carla, Airsim, LG Simulator

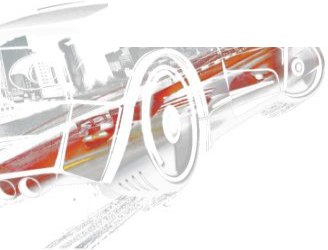
Scaling the simulation: how large we can go?

- Physics-based simulation is a compromise between speed and accuracy
- A physics 3D simulation consists of
  - Rigid body dynamics
  - Discrete collision detection
  - Collision response and friction
  - Joint simulation
  - Raycasting
  - Other aspects such as log, message passing, etc
- As number of objects are increased, more computation is needed, as a compromise, real-time simulation is decreased - meaning - simulation slows down



- Adding more cars to simulation
  - Decreases real-time factor of simulation
- Adding more cars to simulation depends on
  - Limited by RAM availability on how many car-like rigid body can be added
  - How many sensors, such as LiDAR, are being simulated





RTF = 1.0 means 1 sec simulation time = 1 sec wall clock time

A graph showing how real-time factor degrades as we spawn more number of cars in Gazebo simulation. Each curve represent a new simulation executing from scratch. Max Update Rate provided: 100, Time-Step = 0.01 for Gazebo Physics Engine
System Configuration: Intel® Xeon(R) CPU E3-1505M v6 @ 3.00GHz × 8 ; 64 GB RAM

Pitfalls due stochasticity of real-time factor

- How reliable is simulation when real-time factor changes?
- What implications does it have on feedback control?
- Is the result repeatable in such conditions?
- Can we trust a controller developed and tested in such settings for transfer to field experiments?

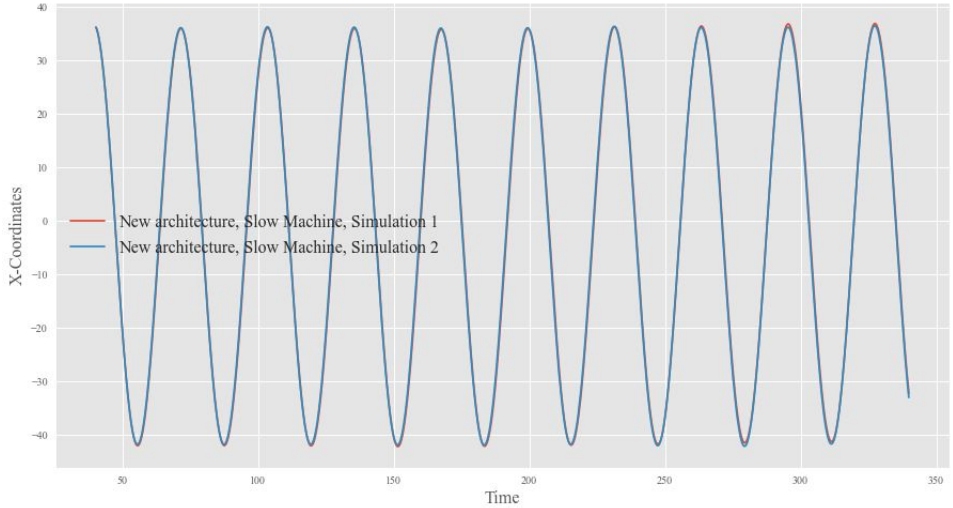
We see some examples of fragility in the simulation

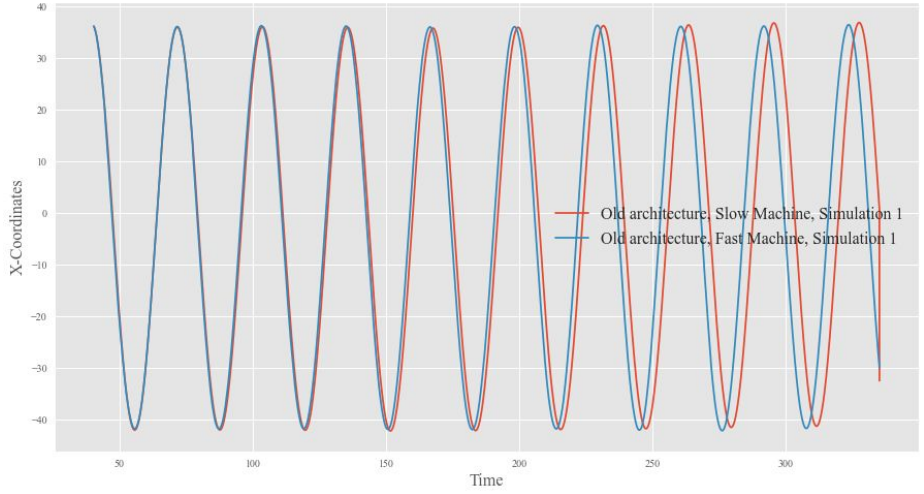
Slow computer: A VM with 8 GB RAM
Fast computer:: Intel® Xeon(R) CPU E3-1505M v6 @ 3.00GHz × 8 ; 64 GB RAM

RMS Error for Traditional Approach		Slow Machine		Fast Machine	
		Simulation 1	Simulation 2	Simulation 1	Simulation 2
Slow Machine	Simulation 1		0.5645	11.6666	10.2738
	Simulation 2	0.5645		11.3878	9.9420
Fast Machine	Simulation 1	11.6666	11.3878		2.4205
	Simulation 2	10.2738	9.9420	2.4205	

TABLE I: RMS error of X-coordinates of the trajectory for different simulation with the traditional approach. RMS error is the squared root of the mean squared of difference of X-coordinates of the trajectory between two simulation runs.

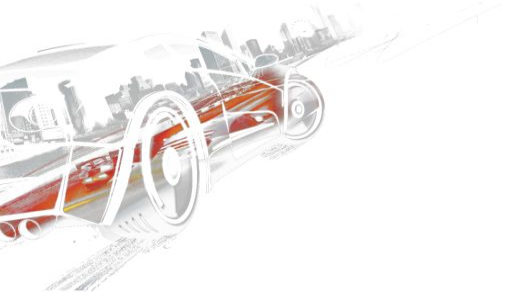
Run two simulations on slow and fast computer each with the older architecture, where we know there is issues with repeatability



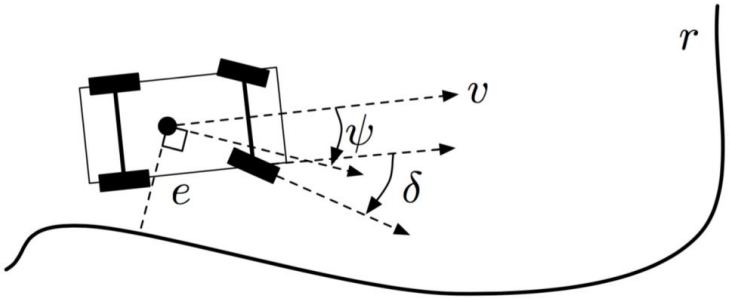


How much x-coordinates on circle different between simulations

Examples of fragility in simulations



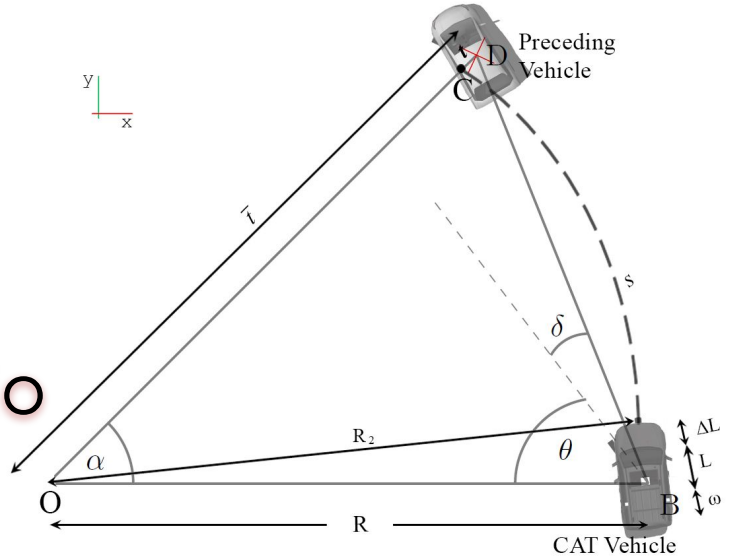
Car model for the control



Input: Longitudinal Velocity and Steering angle

For a circular trajectory, steering angle remain constant

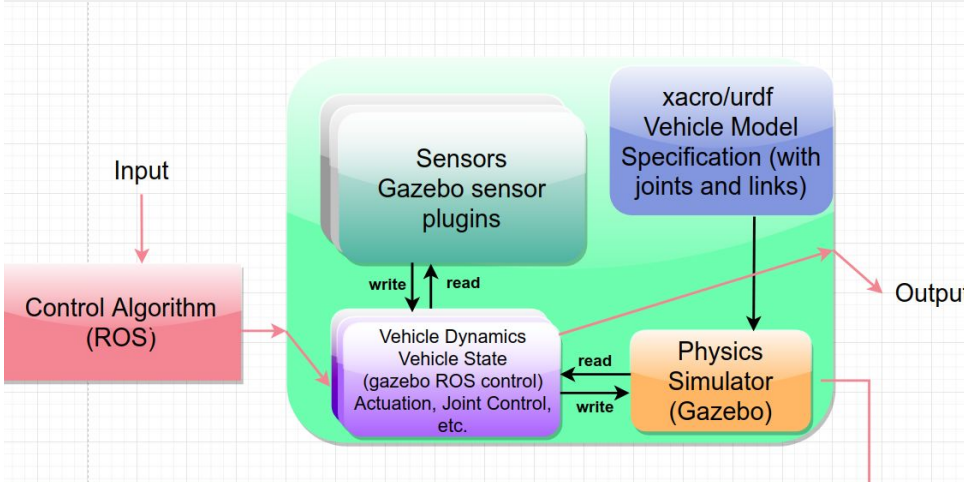
δ = steering angle
ψ = heading angle

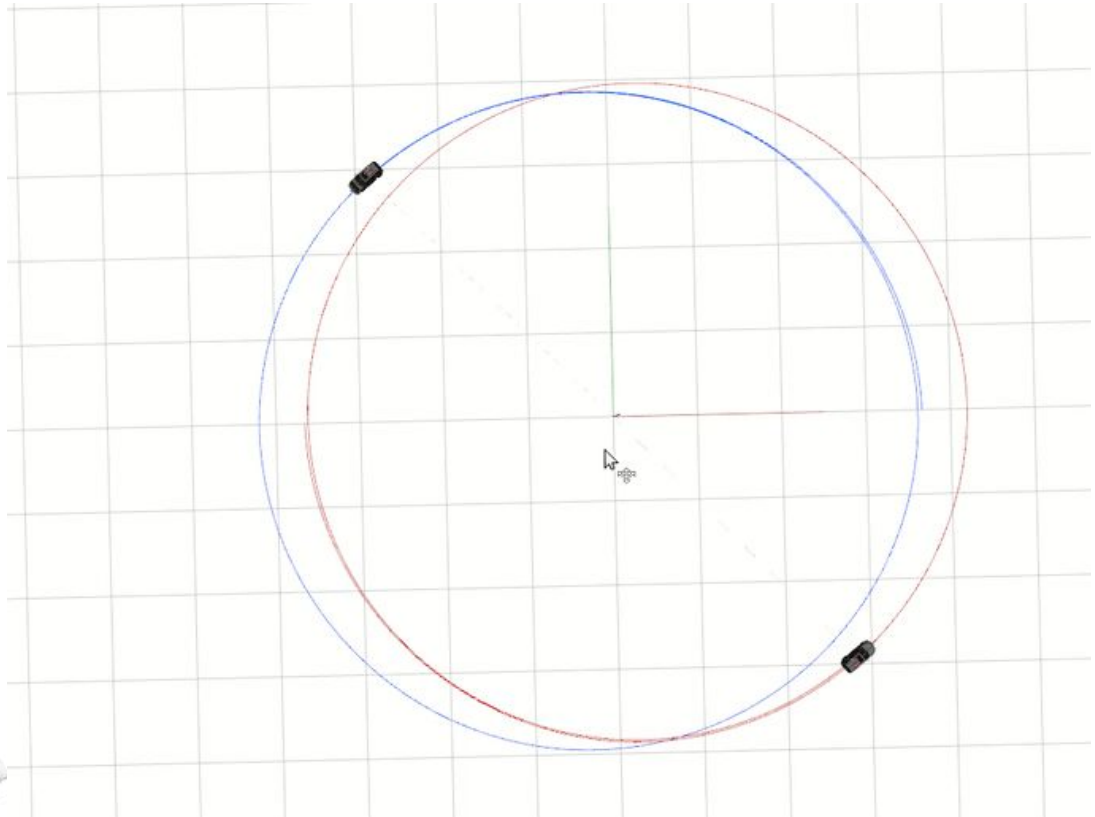


Leader-follower scenario

Open-loop control, two vehicles

Example 1

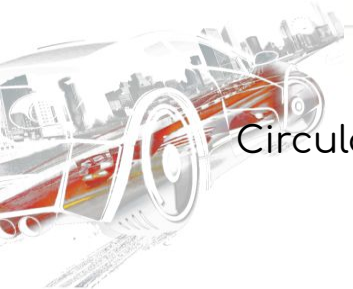




Open loop controller with constant velocity and constant steering angle

Root cause of trajectory deviation is jitter in the initial conditions of the vehicle as well as imperfect inertia tensor for the rigid body. It is really difficult to calculate accurate inertial tensorf for an arbitrary 3D Model. When inertial tensor for the model is not accurate enough, Gazebo behaves in unexpected way.

v = 8.0m/s
δ = 0.065 radian



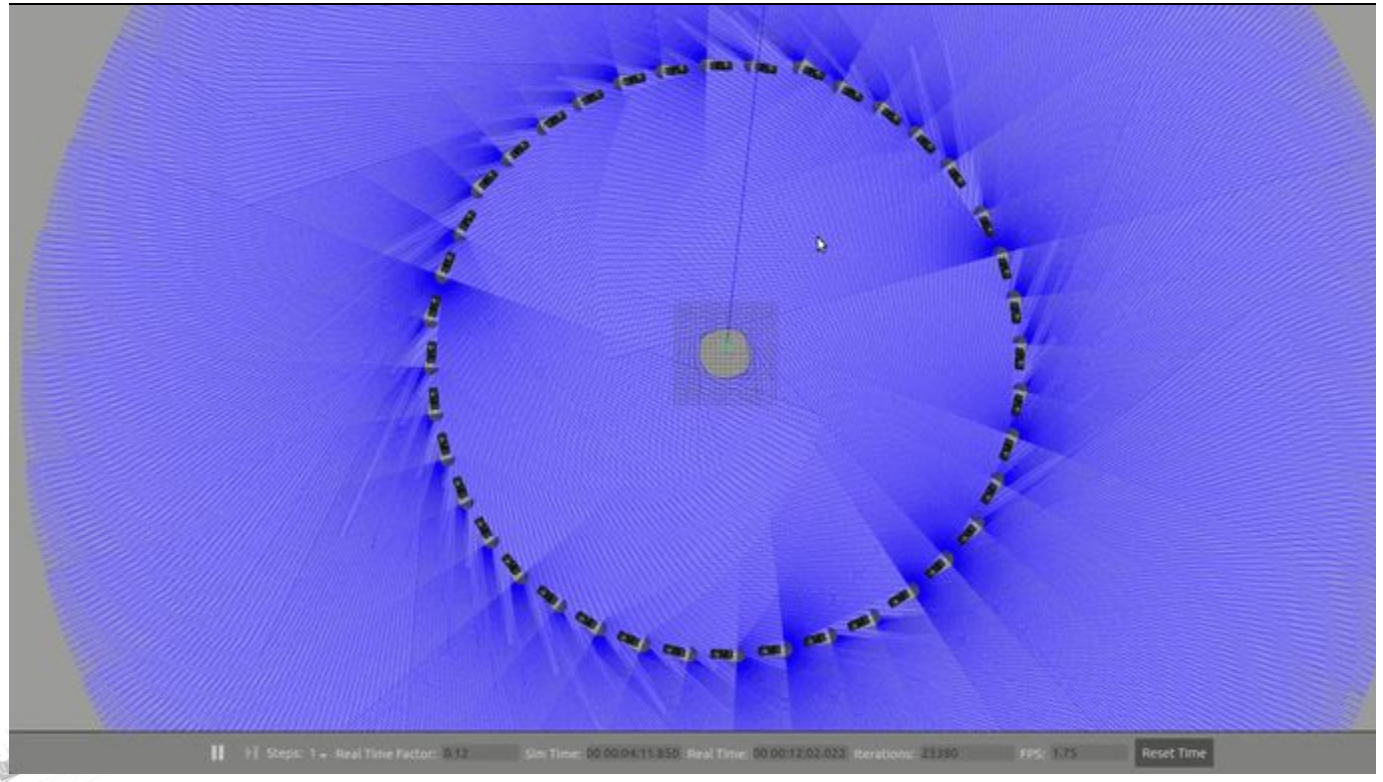
Circular Trajectory Simulation with small deviation that grows over the time
https://www.youtube.com/watch?v=1Mf6_zwBUV0

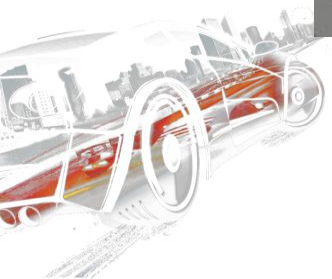
System Configuration: Intel® Xeon(R) CPU E3-1505M v6 @ 3.00GHz × 8 ; 64 GB RAM

Desired Real-time factor = 1.0

Open-loop control, scaling to many vehicles

Example 2

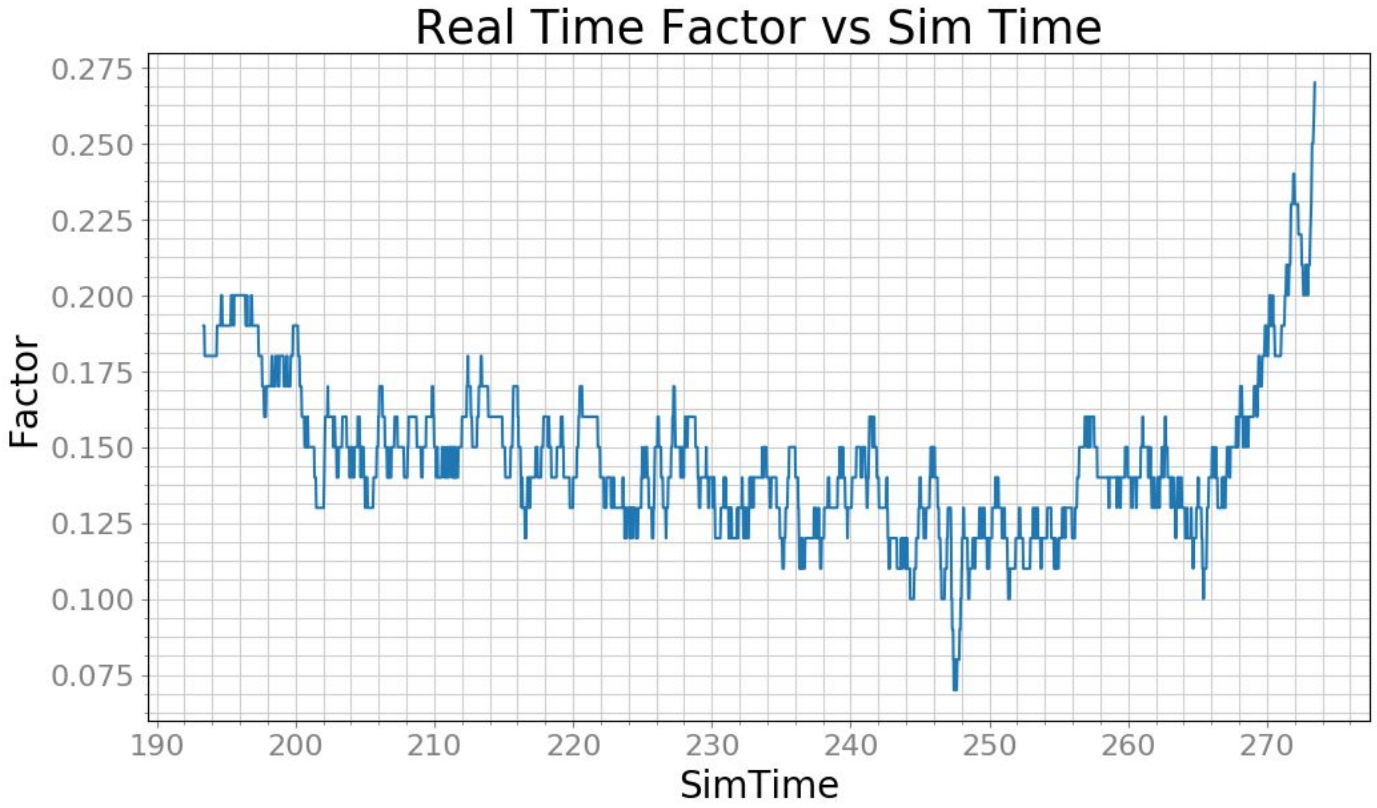




Open loop velocity control with 40 vehicles.
Few cars ultimately lose their circular tracks 4x speed
https://www.youtube.com/watch?v=vo0V2ziqhGs

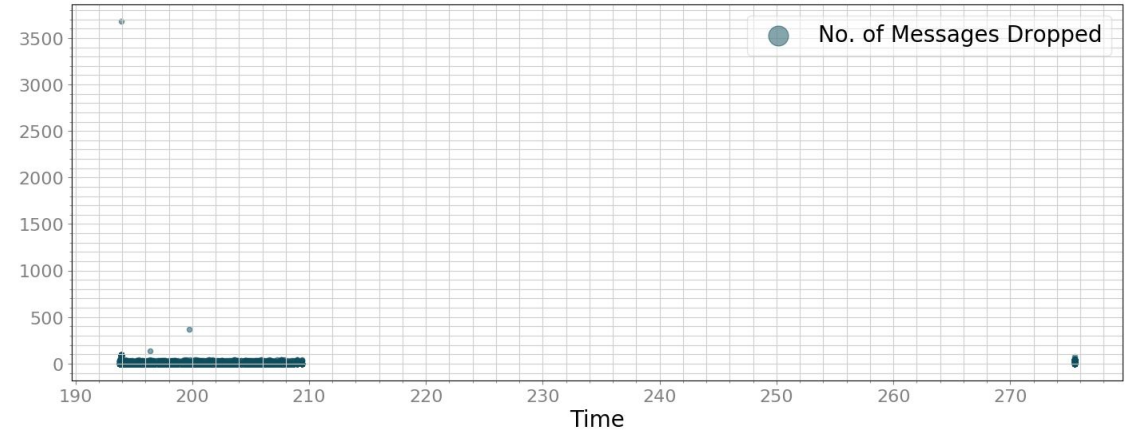
System Configuration: Intel® Xeon(R) CPU E3-1505M v6 @ 3.00GHz × 8 ; 64 GB RAM

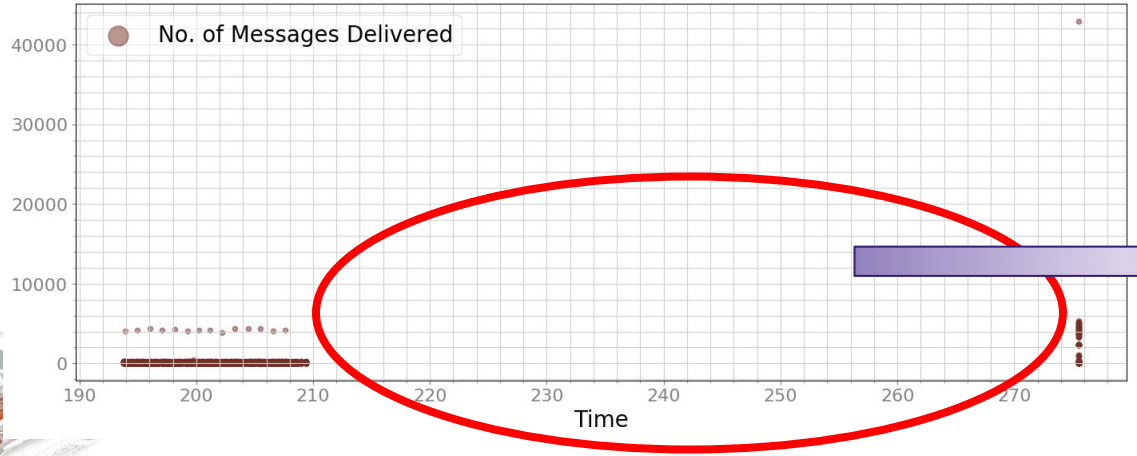
Desired Real-time factor = 1.0



Desired Real-time factor = 1.0

Real-time factor degrades drastically



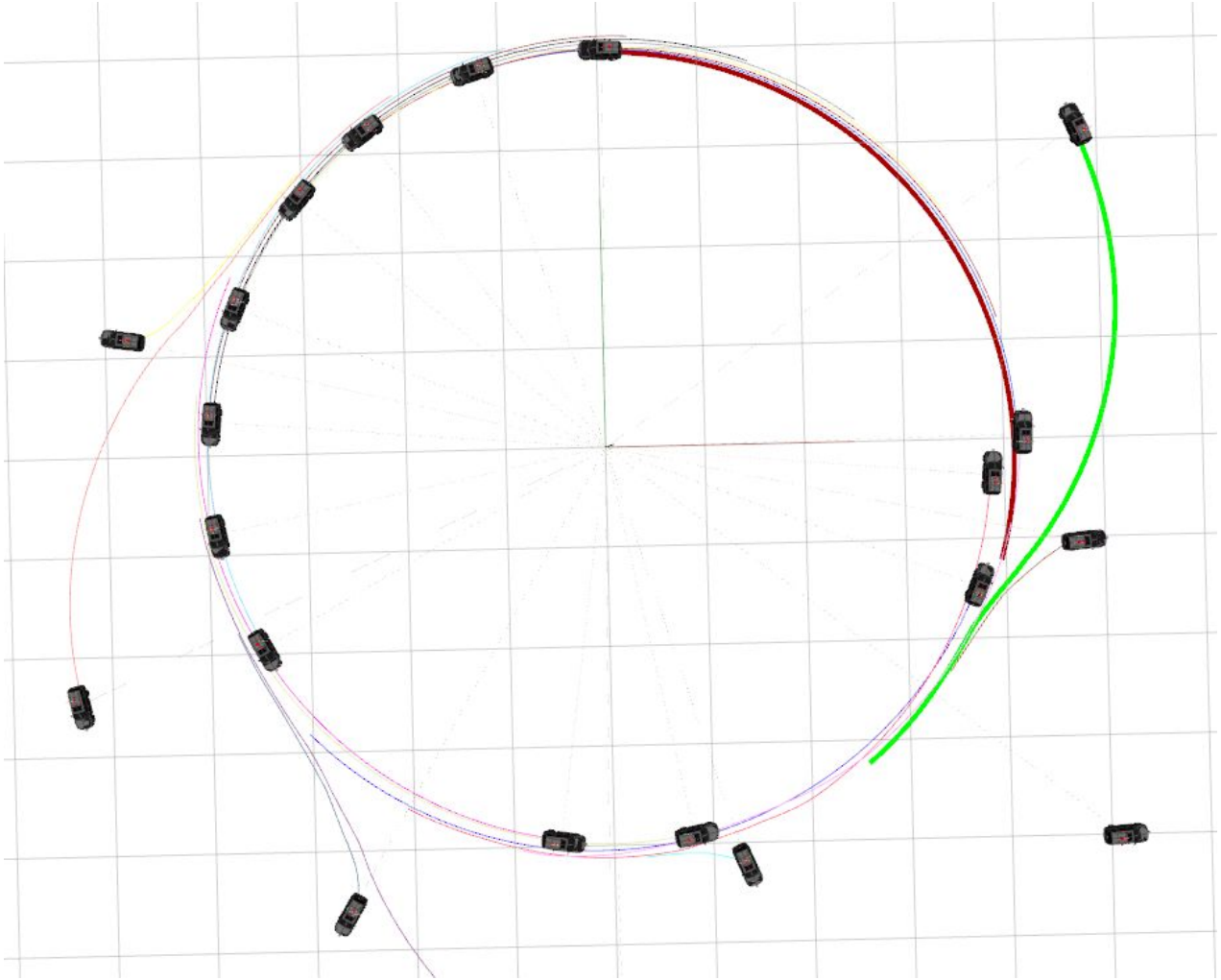


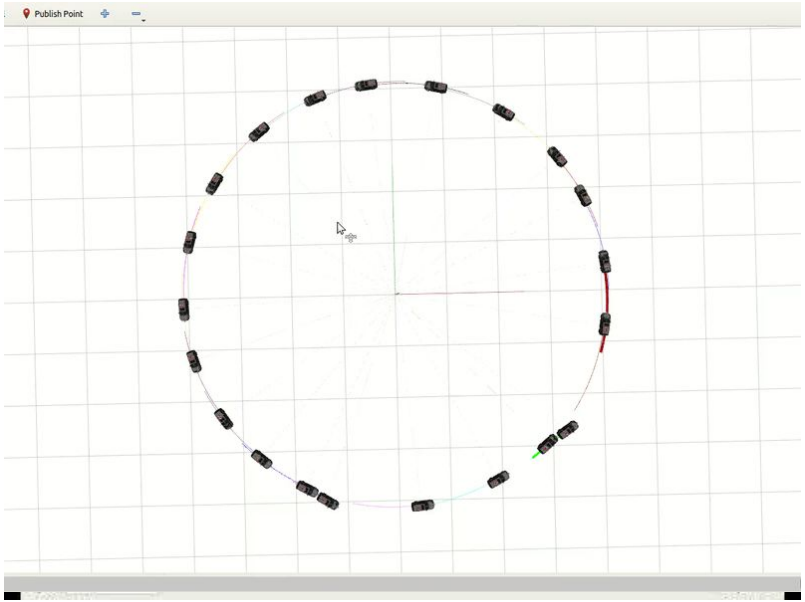
No message delivered during this period

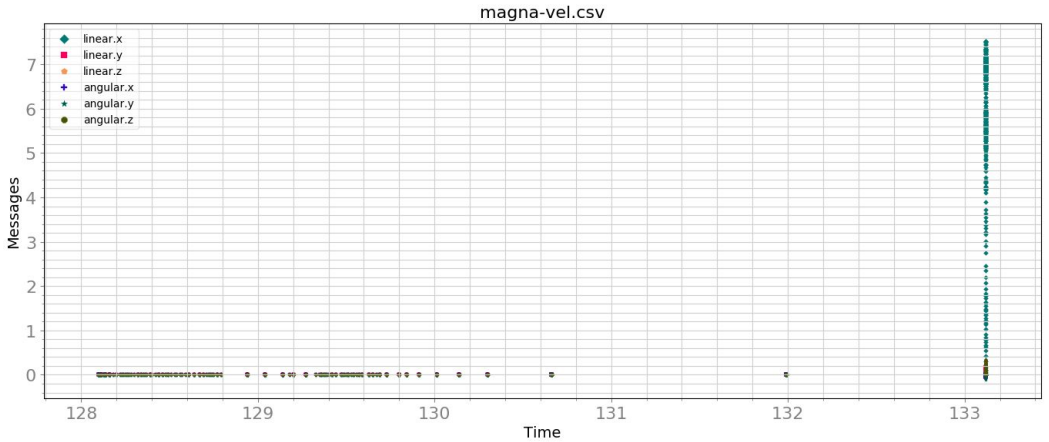
Replicating Arizona Ring-road trajectory

Example 3

Inject speed data for 21 vehicles that was collected from the ring-road experiment where Followerstopper controller was employed on a single vehicle to dissipate phantom traffic jam







4x speed
https://youtu.be/RpfWOu1yako

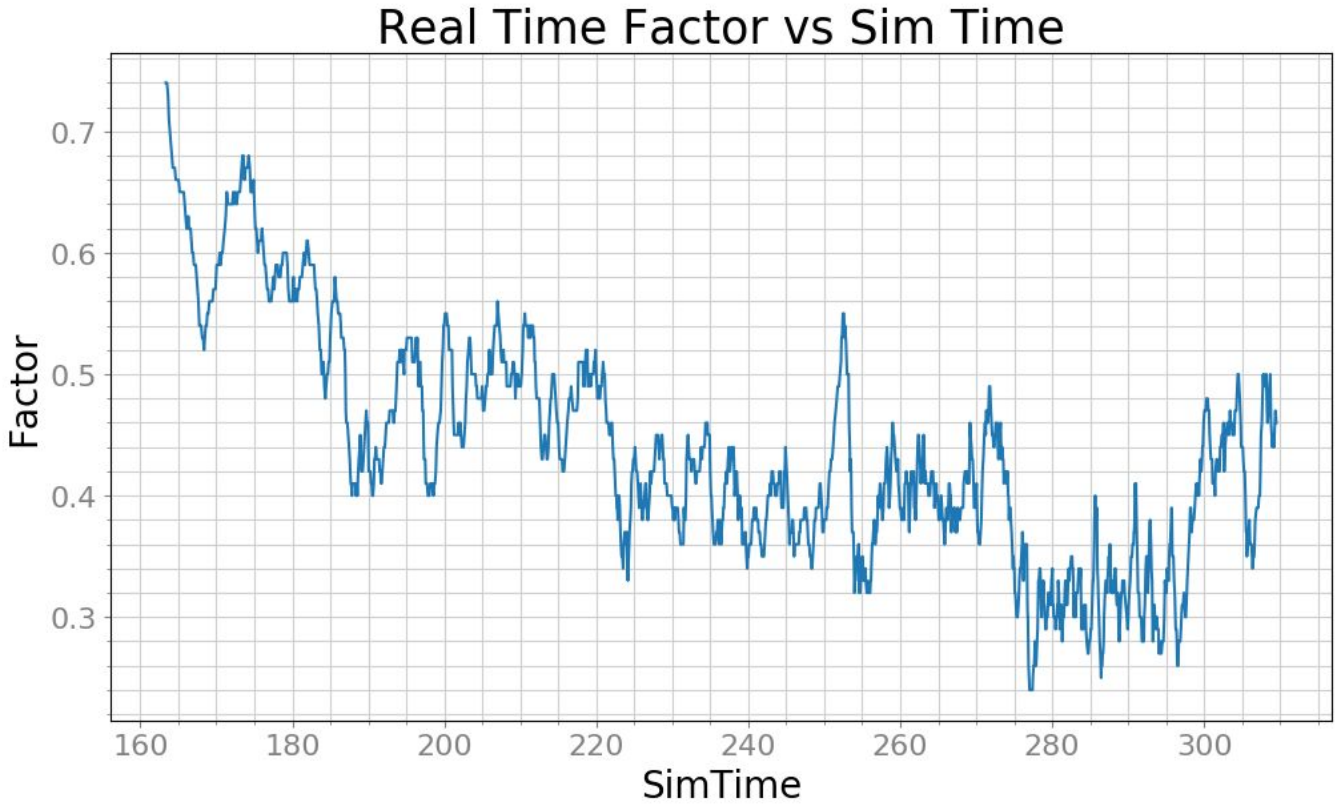
We also tried to replicate the trajectory of all 21 vehicles obtained from Arizona Ring Road Experiment where we demonstrated wave-dissipation effect of the Followerstopper. We injected velocity of each car in the simulation with an exact time-interval obtained with a fixed steering angle from the recorded data

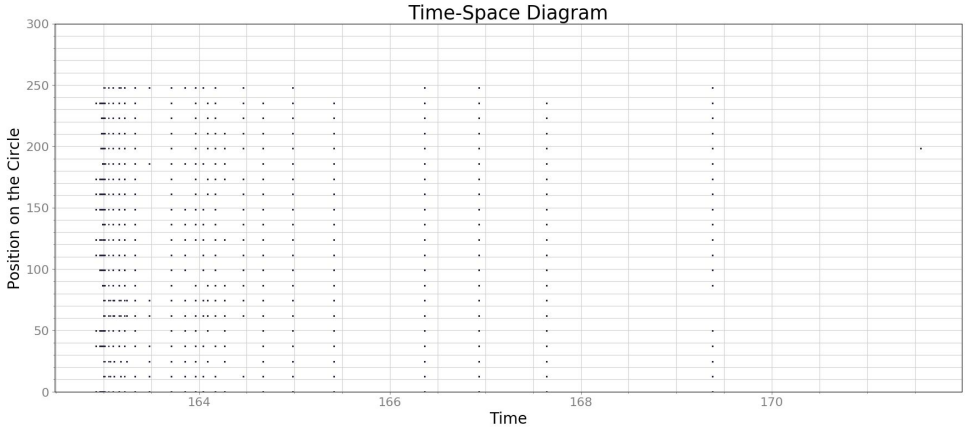
System Configuration: Intel® Xeon(R) CPU E3-1505M v6 @ 3.00GHz × 8 ; 64 GB RAM

Outcome:

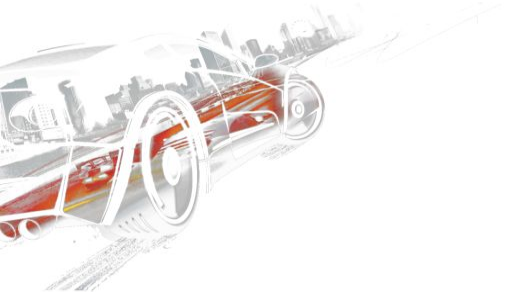
Simulation to replicate the trajectory too slow, plenty of packets dropped.
The recording of the experiment demonstrates that data was not received in the right order as shown in the plot above.

ODE engine of Gazebo failed to respond in any reasonable way. Real-time factor (RTF) dropped to 0.33 when specified RTF was 1.0


Real Time Factor vs Sim Time
Factor
0.7
0.6
0.5
0.4
0.3
160
180
200
220
240
260
280
300
SimTime

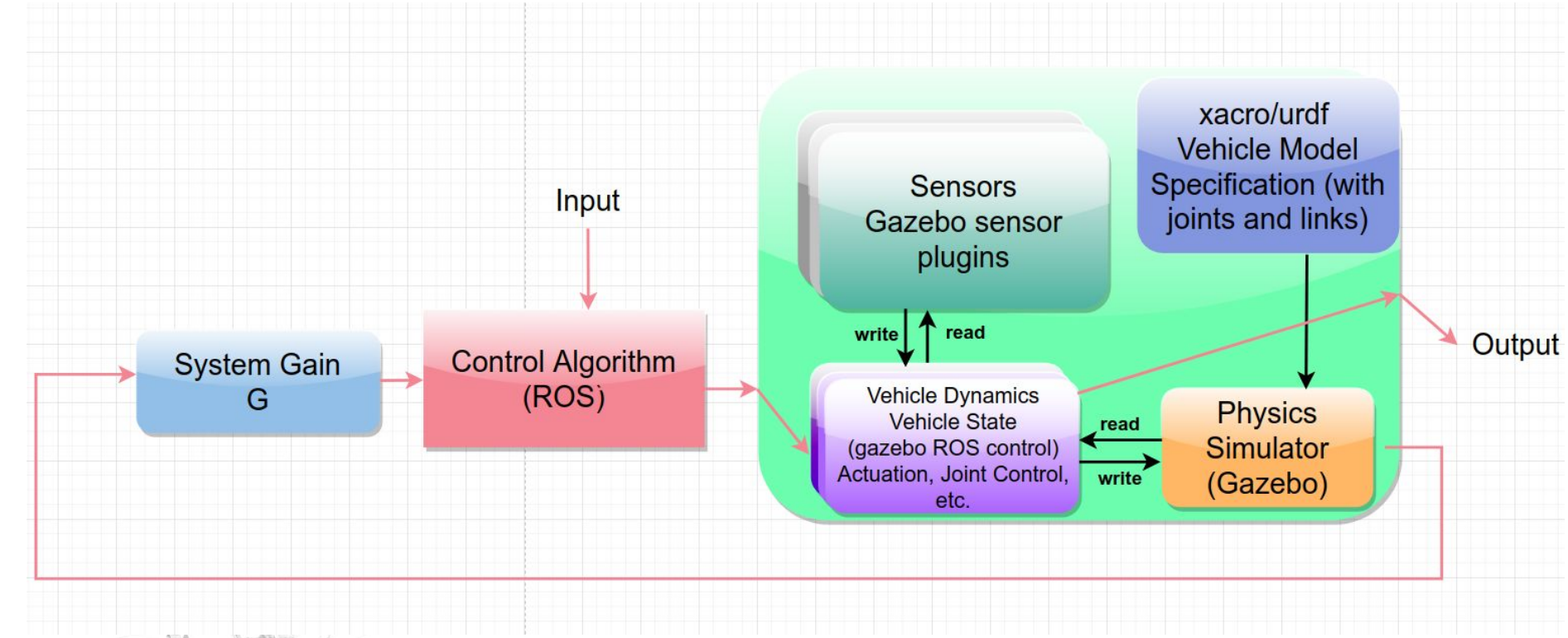


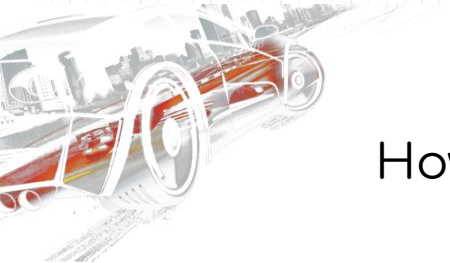
Well, system was so much overloaded that rosbag couldn't record most of the data



Very few packets delivered

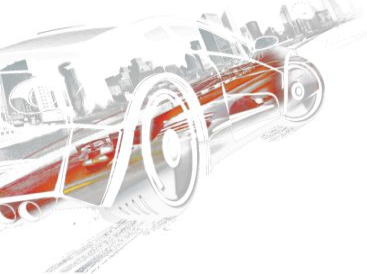
Standard ROS architecture





How simulation is done currently (closed loop)?

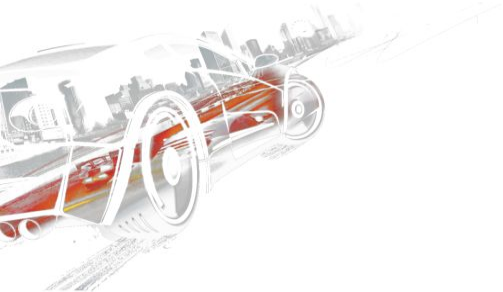
- Pain points
  - Too many messages to exchange for large simulations
  - Physics-engine based agent model too complex to simulate and return state update in real-time
  - Synchronization issue at larger-scale
  - Lack of repeatability as a result of emergent changes in real-time factor, based on current system load

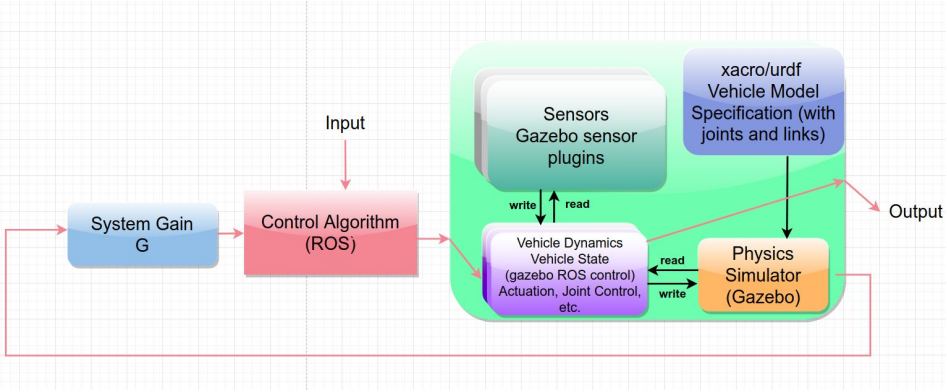


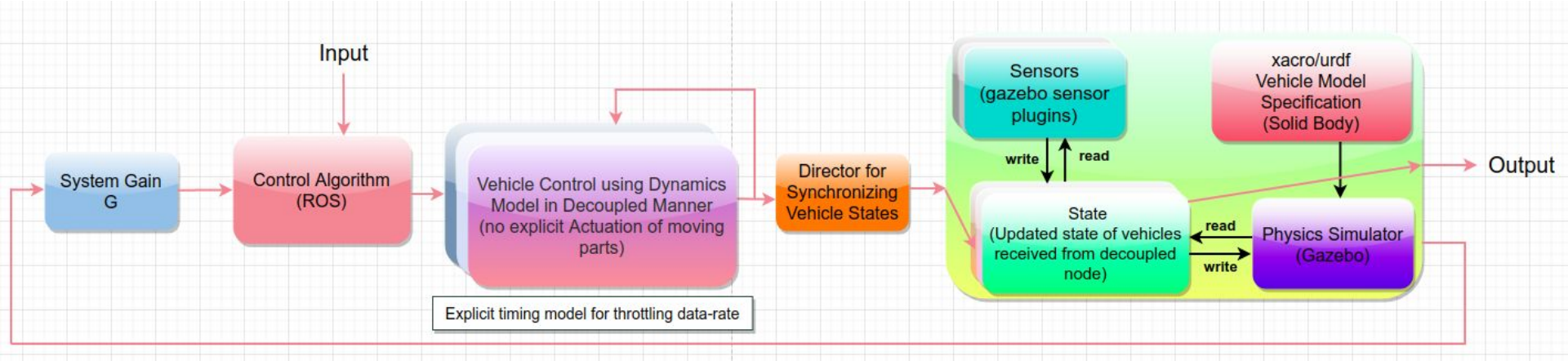
- Proposed remedy
  - Throttle down message rate to reduce system load
  - Explicitly simulate at slower than real-time factor
  - Add a director node to synchronize updated states of various agents
  - Simplify agent model and decouple from physics engine, and propagate updated states to physics engine

Pain points and Remedies

Proposed Architecture for improving simulation

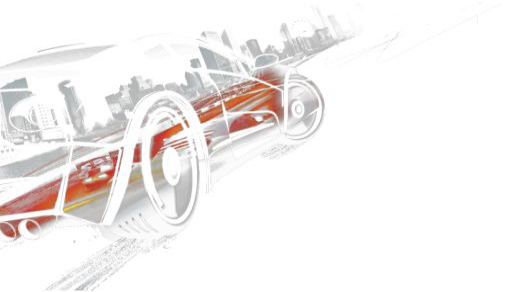






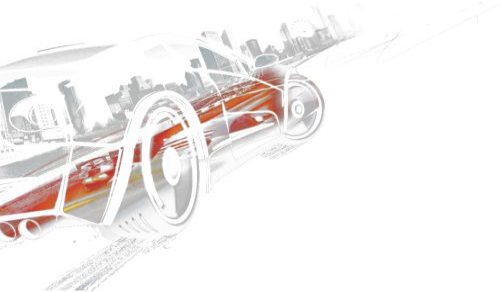
Proposed Architecture for scaling the simulation with repeatability

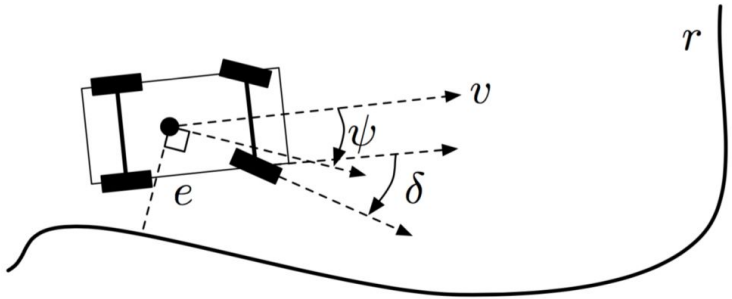
Simulation examples with new architecture

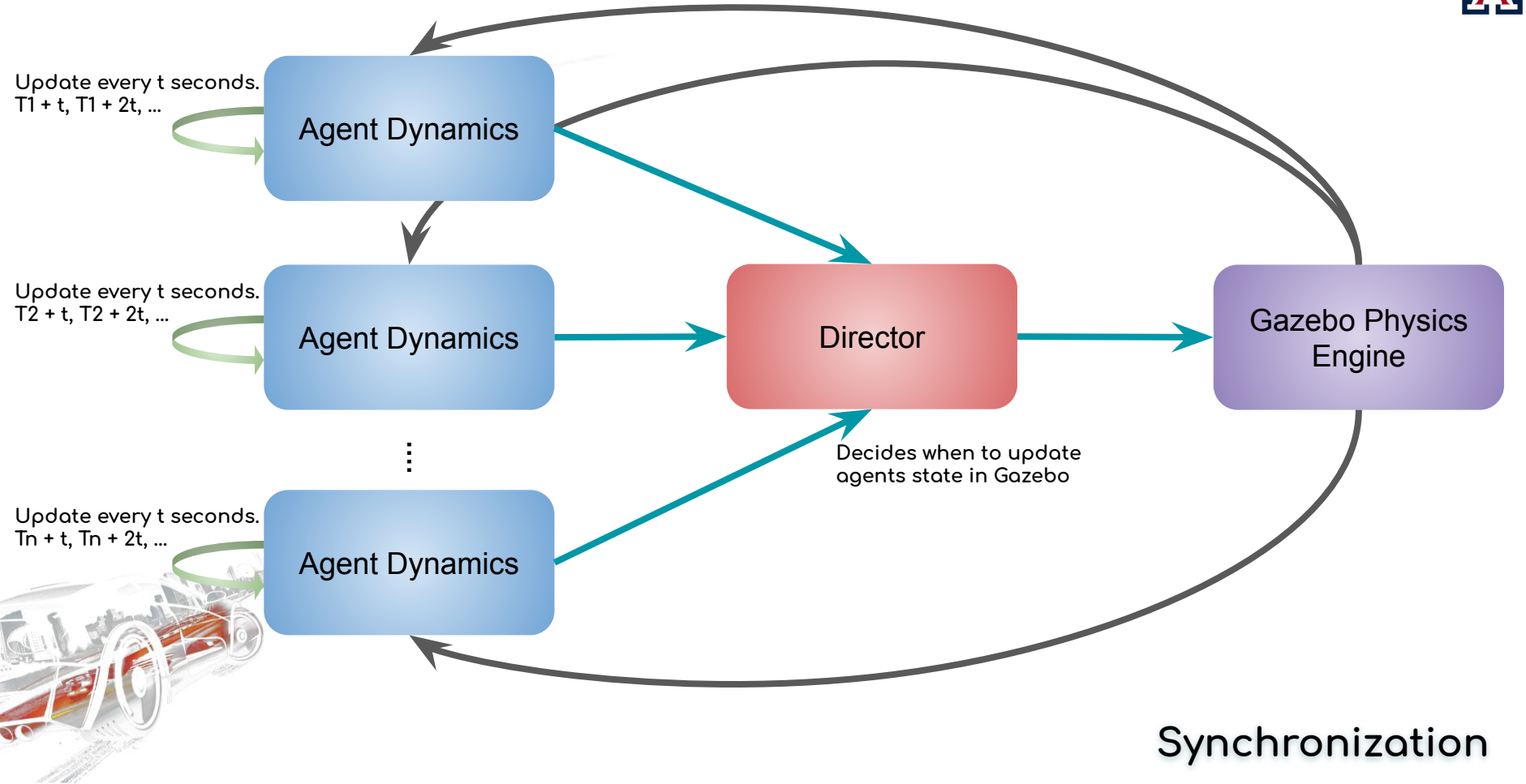


$$
\begin{aligned}
x_1^{(t+\delta t)} &= x_1^{(t)} + \delta t u_1^{(t)} \cos\left(x_3^{(t)}\right) \cos\left(x_4^{(t)}\right) \\
x_2^{(t+\delta t)} &= x_2^{(t)} + \delta t u_1^{(t)} \cos\left(x_3^{(t)}\right) \sin\left(x_4^{(t)}\right) \\
x_3^{(t+\delta t)} &= x_3^{(t)} + \delta t u_2^{(t)} \\
x_4^{(t+\delta t)} &= x_4^{(t)} + \delta t u_1^{(t)} (1/L) \sin\left(x_3^{(t)}\right)
\end{aligned}
$$

Bicycle model for the longitudinal dynamics,
update the state every-time step and send to Gazebo



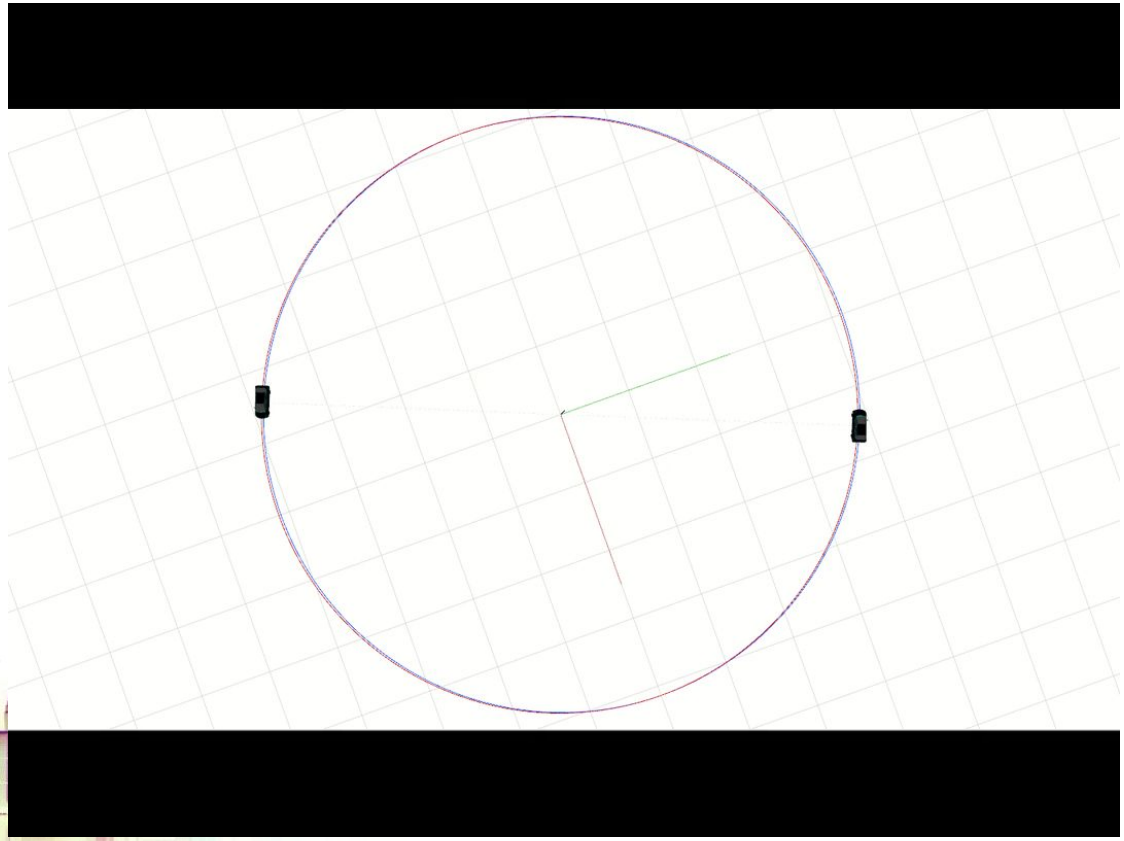




Synchronization

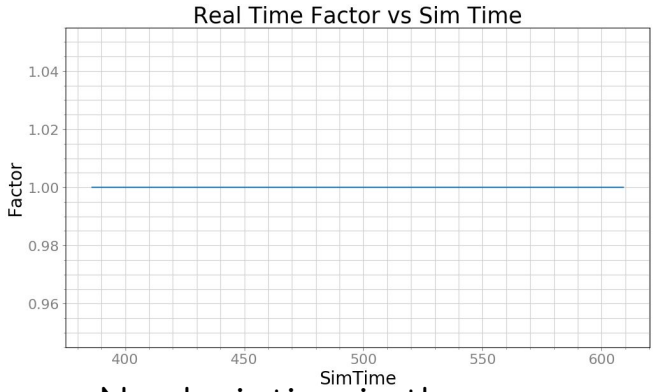
The two car trajectory with decoupled architecture

https://www.youtube.com/watch?v=QQslG9ICiSA&feature=emb_logo



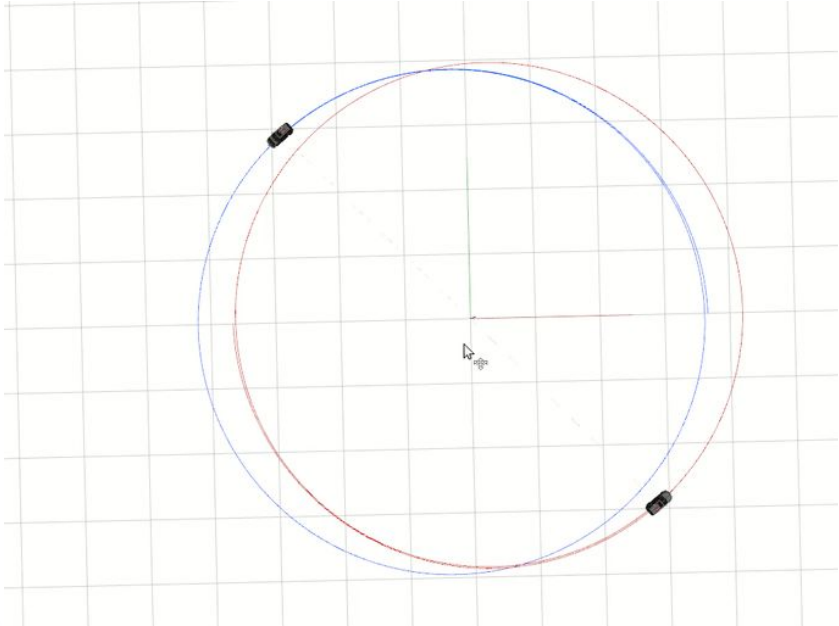
When decoupled the dynamics, the trajectory is repeatable. In every rotation, we see both, blue trajectory for the first car and the red trajectory for the second car are overlapping, unobservable deviation observed.

State Update Rate at 20 Hz

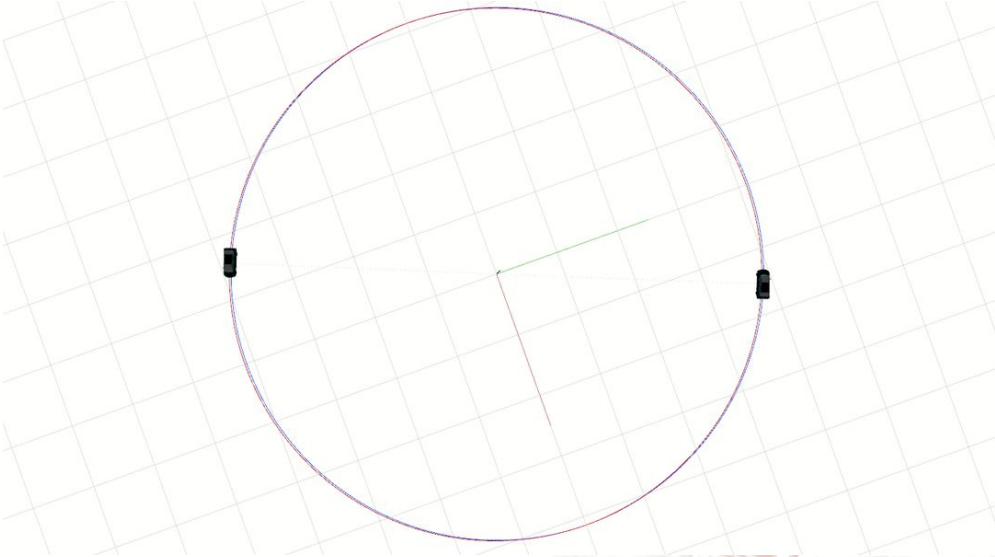


No deviation in the real-time factor

Comparison

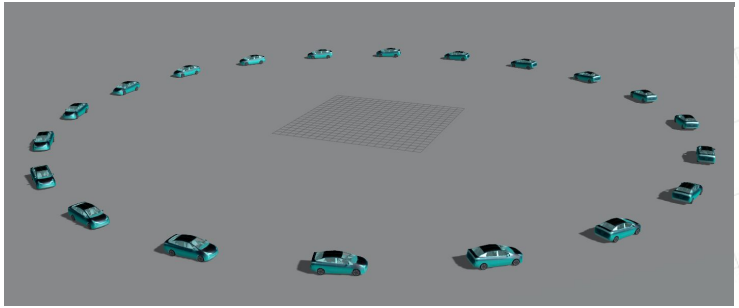


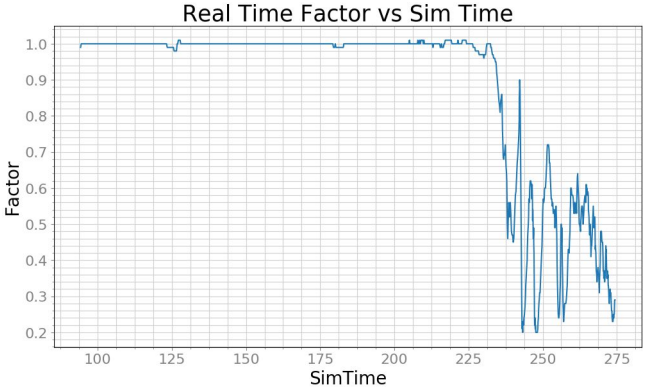
With Standard Architecture



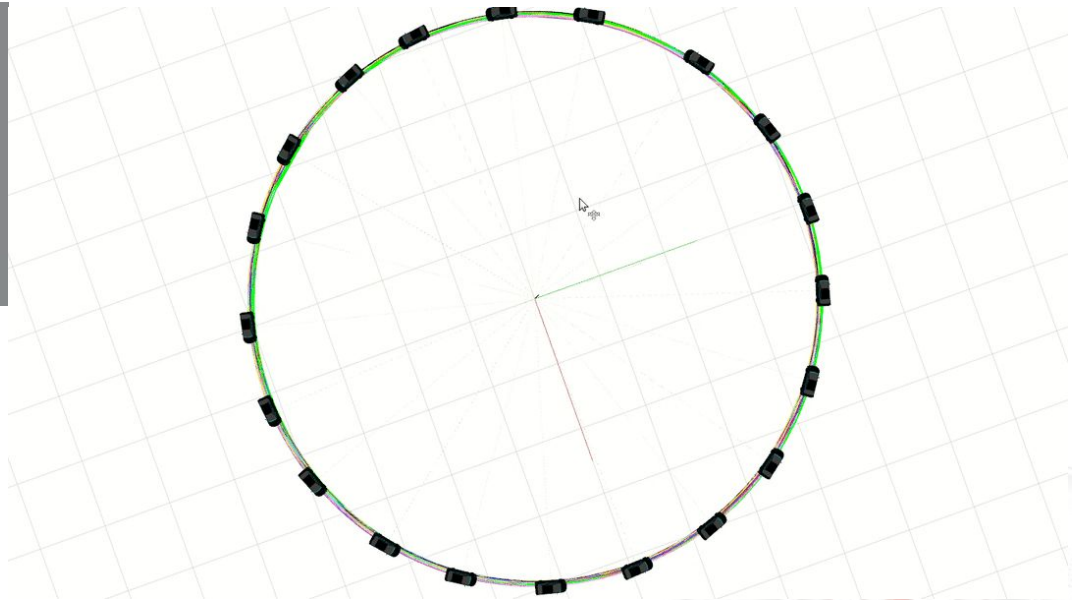
With Decoupled Architecture

What about scalability? With 20 cars?





Desired RTF 1.0, but eventually degrades

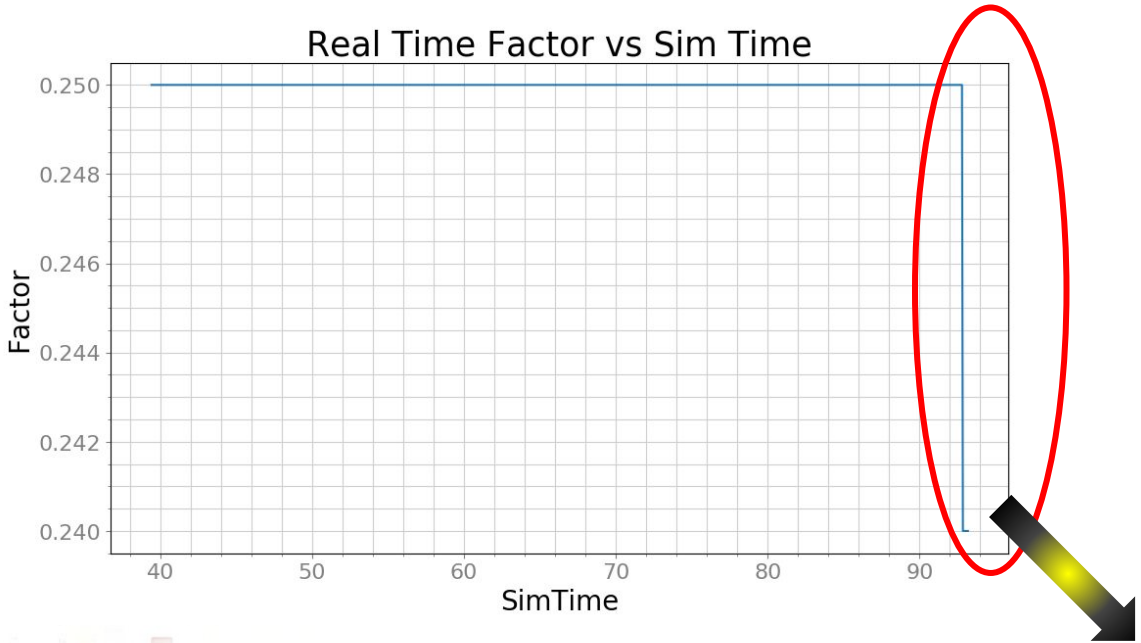


Open loop control.

State update Throttled at 20 Hz for each vehicle separately, but synchronization problem. Repeatability is okay here.

Need director to synchronize state updates

But can be scaled when we explicitly specify RTF < 1.0



Specified real-time factor = 0.25

Synchronization problem still remains

This sudden drop of RTF occurs when we start destroying the simulation

Revisit the RMS statistic

BAD

RMS Error for Traditional Approach		Slow Machine		Fast Machine	
		Simulation 1	Simulation 2	Simulation 1	Simulation 2
Slow Machine	Simulation 1		0.5645	11.6666	10.2738
	Simulation 2	0.5645		11.3878	9.9420
Fast Machine	Simulation 1	11.6666	11.3878		2.4205
	Simulation 2	10.2738	9.9420	2.4205	

TABLE I: RMS error of X-coordinates of the trajectory for different simulation with the traditional approach. RMS error is the squared root of the mean squared of difference of X-coordinates of the trajectory between two simulation runs.

GOOD

RMS Error for Newer Approach		Slow Machine		Fast Machine	
		Simulation 1	Simulation 2	Simulation 1	Simulation 2
Slow Machine	Simulation 1		0.2815	0.2743	0.2525
	Simulation 2	0.2815		0.2605	0.2693
Fast Machine	Simulation 1	0.2743	0.2605		0.2724
	Simulation 2	0.2525	0.2693	0.2724	

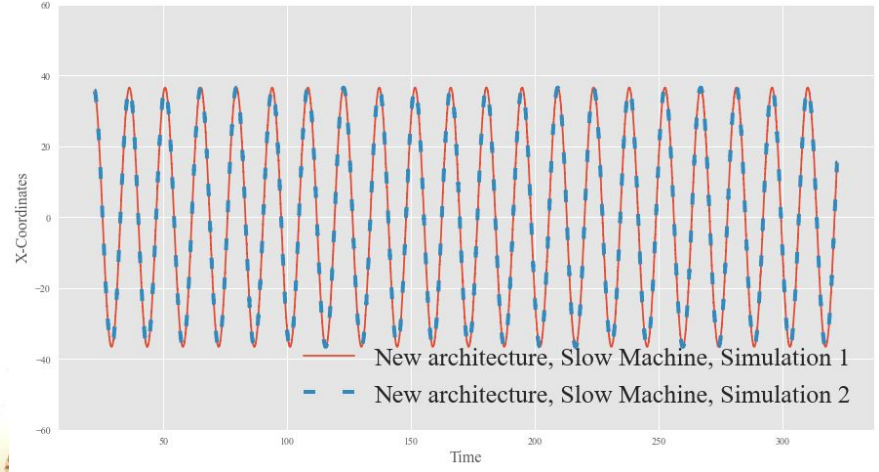
TABLE II: RMS error of X-coordinates of the trajectory for different simulation with the newer approach. RMS error is the squared root of the mean squared of difference of X-coordinates of the trajectory between two simulation runs.

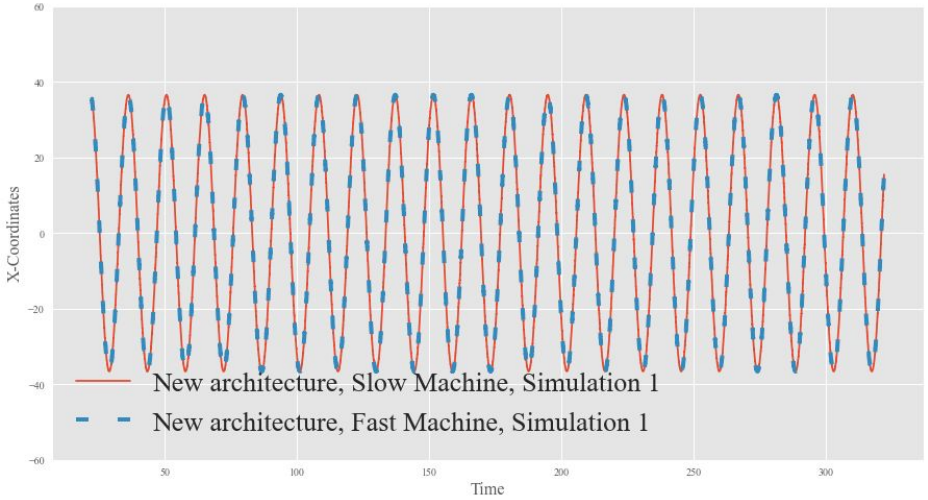
Revisit the RMS statistic

RMS Error for Newer Approach		Slow Machine		Fast Machine	
		Simulation 1	Simulation 2	Simulation 1	Simulation 2
Slow Machine	Simulation 1		0.2815	0.2743	0.2525
	Simulation 2	0.2815		0.2605	0.2693
Fast Machine	Simulation 1	0.2743	0.2605		0.2724
	Simulation 2	0.2525	0.2693	0.2724	

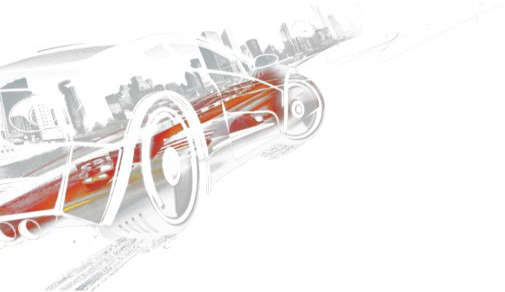
TABLE II: RMS error of X-coordinates of the trajectory for different simulation with the newer approach. RMS error is the squared root of the mean squared of difference of X-coordinates of the trajectory between two simulation runs.

How much x-coordinates on circle different between simulations across machines and within the same machine





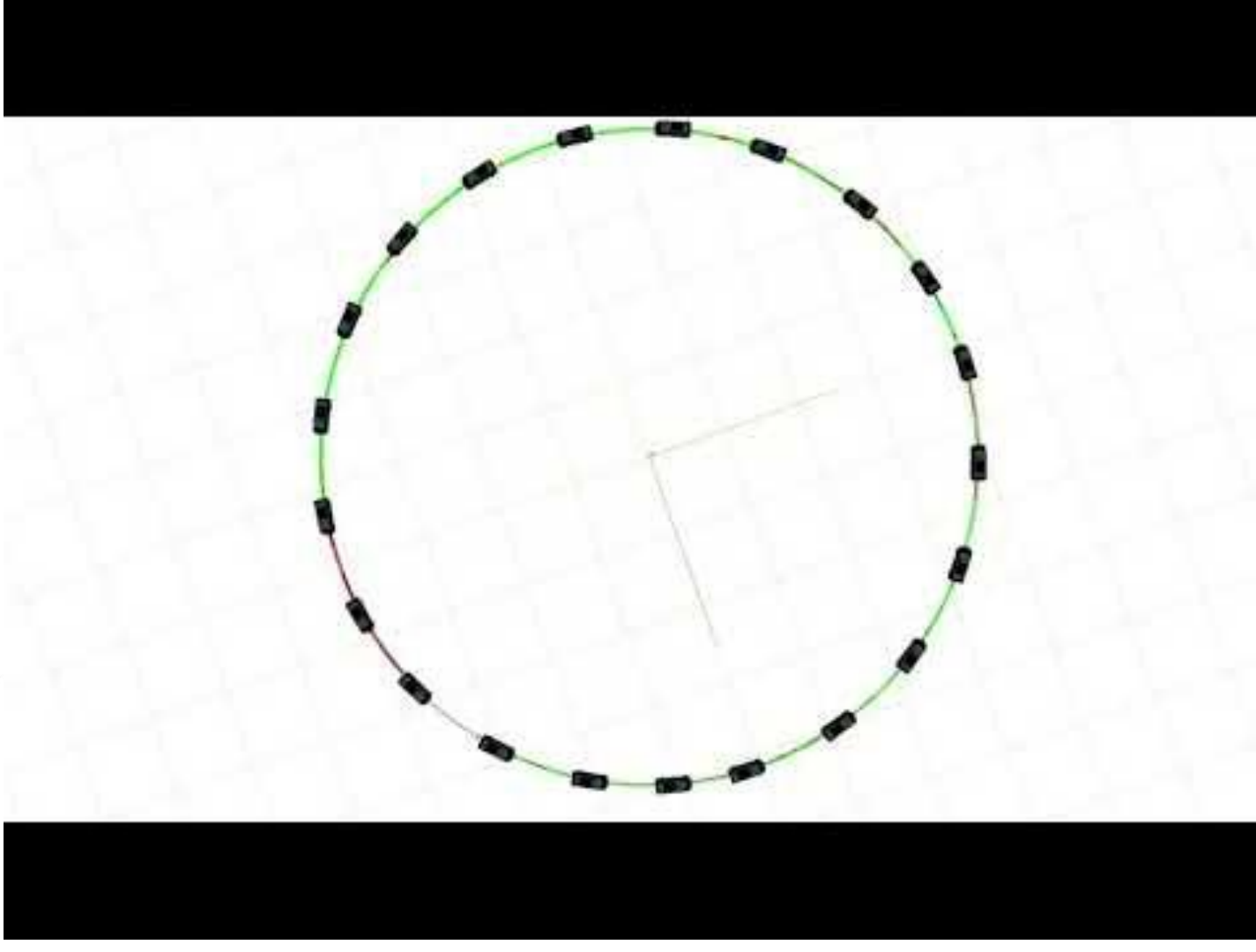
Replicating Arizona ring road experiment trajectory with the proposed method



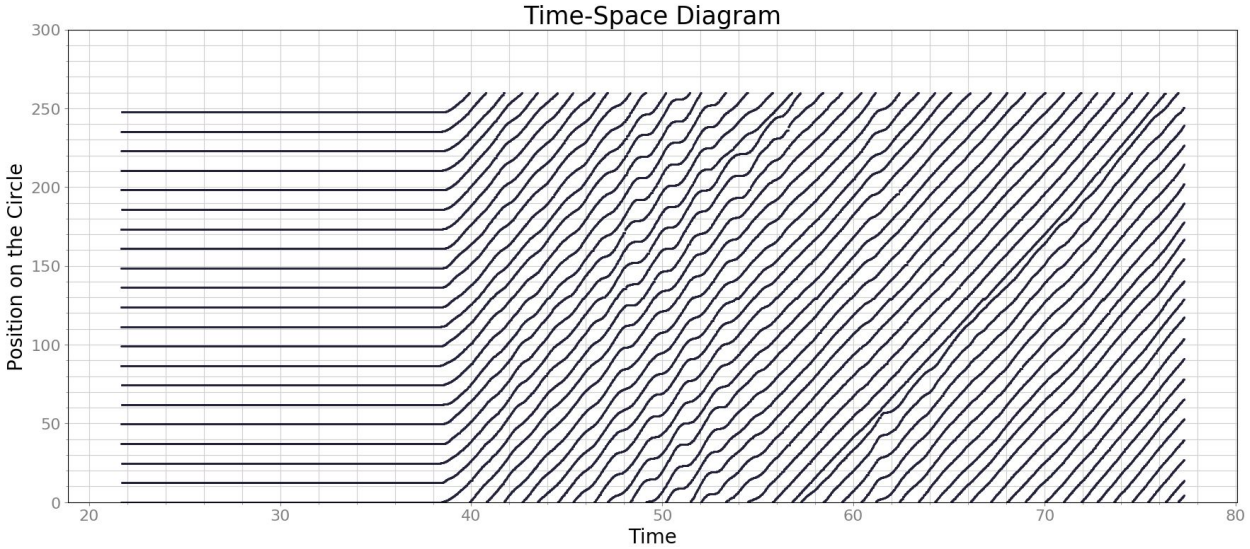
Simulation Result

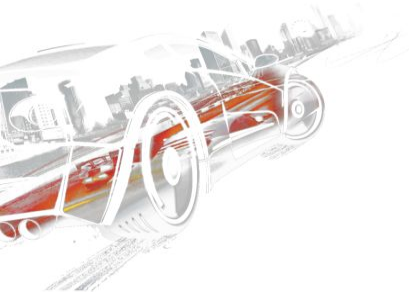
At RTF = 0.1

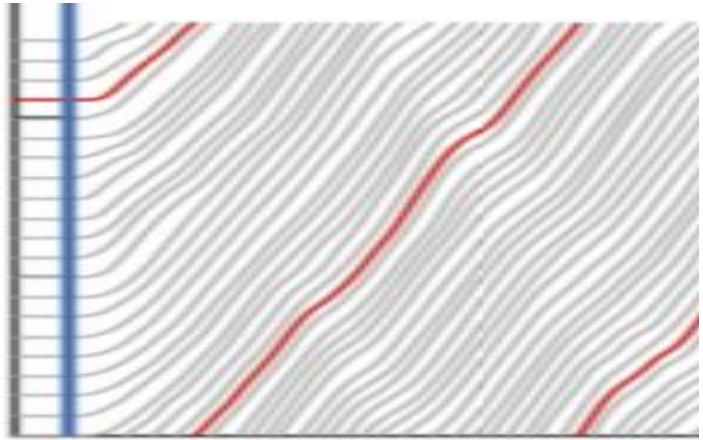
replicate the trajectory of all 21 vehicles obtained from Arizona Ring Road Experiment where we demonstrated wave-dissipation effect of the Followerstopper. We injected velocity of each car in the simulation with an exact time-interval obtained with a fixed steering angle from the recorded data



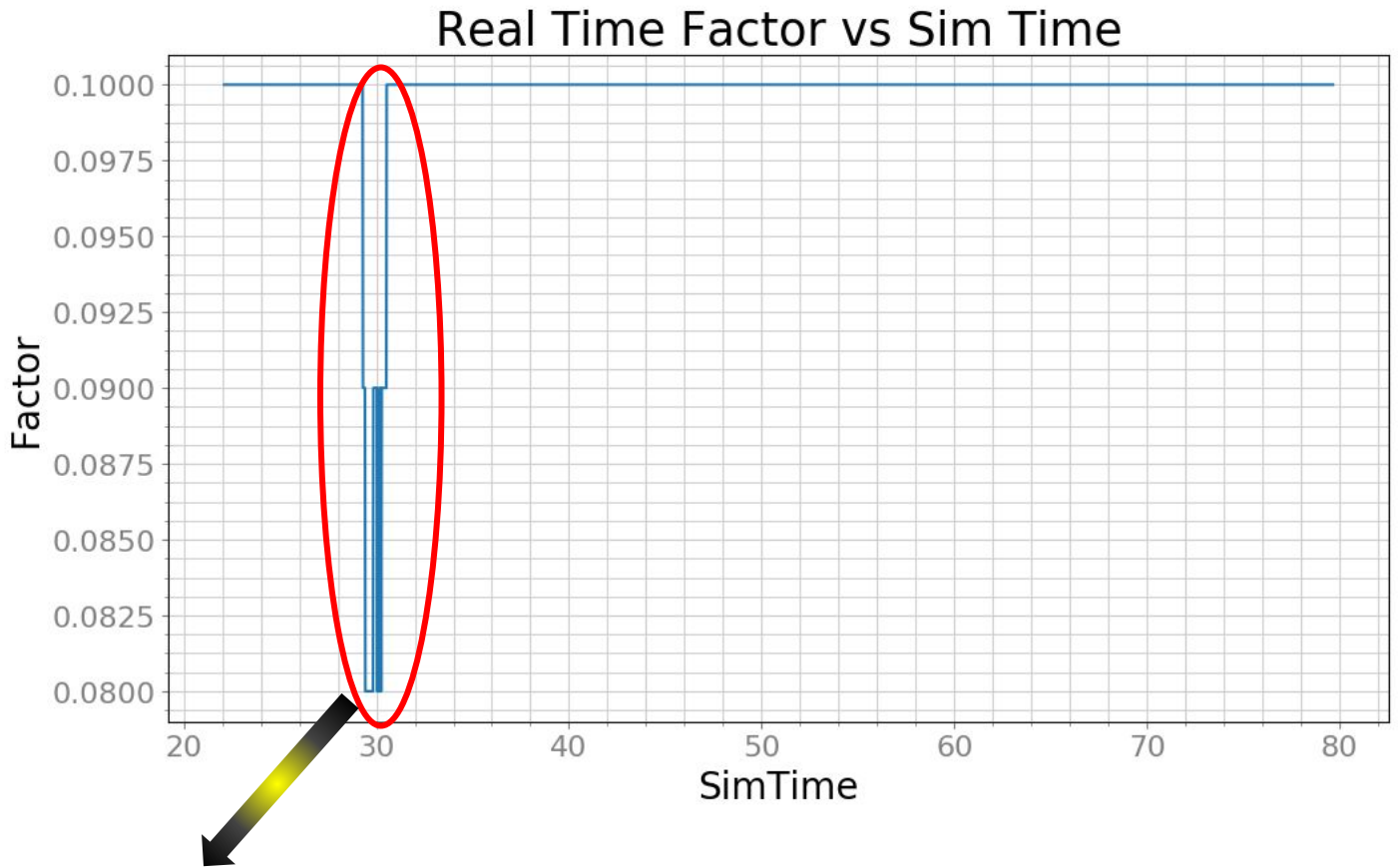
https://www.youtube.com/watch?v=sJIV1l40bLo&feature=emb_logo







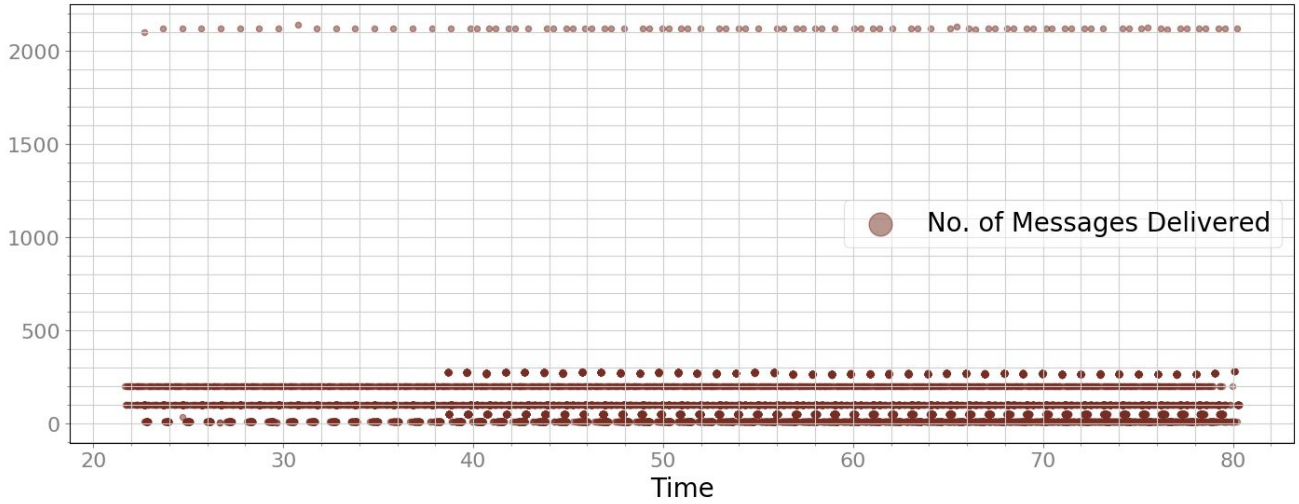
Time space diagram



This sudden drop of RTF may be explained by a sudden system load, but controller really didn't activate until t > 40s as seen time-space diagram in the previous slide

Real-time factor

Compare to
state of the art



Packet Statistics

Conclusions

- Until you can run the same simulation twice and depend on the results, there is no way to depend on that simulation for safety
- Changes in simulation architecture can improve
  - Reproducibility, if you replace simulation engines that are not intended for reproducibility with ones that are
  - Scalability, if you can reproduce simulations at different real-time factors in order to add more vehicles
- Limitations
  - Full reproducibility of the distributed system is likely not obtainable without rewriting all the heterogeneous systems

Support provided by

- “CPS: Synergy: Control of Vehicular Traffic Flow via Low Density Autonomous Vehicles” NSF CNS-1446435, 1446690, 1446702, 1446715 (joint with B. Piccoli, D. Work, B. Seibold)
- “CAREER: Domain-Specific Modeling Techniques for Cyber-Physical Systems” NSF CNS-1253334
- “Computationally Aware Cyber-Physical Systems” NSF CNS-1544395 (joint with R. Sanfelice)