Autonomous Vehicles

CPS and IoT Security

Alessandro Brighente

Master Degree on Cybersecurity

Università degli Studi di Padova

SPRITZ Security & Privacy Research Group

Autonomous Driving

- An autonomous vehicle is a car capable of sensing the surrounding environment and take sae actions without human intervention
- These sensors include thermographic cameras, radar, lidar, cameras, GPS, and inertial measurement units
- The sensors' output is used as input for a controller which creates a model to identify the most appropriate navigation path

Perception

- Autonomous vehicles use different types of sensors to perceive the environment and monitor their own physical parameters
  - Cameras
  - LiDAR
  - RADAR
  - SONAR
  - IMU
  - GNSS
- Sensor fusion techniques are used to reduce measurement uncertainties due to noise

Planning

- Using the data obtained from the perception system, the ego vehicle performs behavior planning
- The optimal behavior needs to be decided by predicting states of the ego vehicle as well as other objects in the surrounding
- Based on the planned behavior, generate an optimal trajectory
- The entire process is called *motion planning*

SAE Levels of Automation

- The Society of Automotive Engineers (SAE) defined 6 levels of driving automation

- **Level 0:** no automation, manual control
- **Level 1:** driver assistance with monitoring functionalities (e.g., cruise control)
- **Level 2:** partial automation with the car taking autonomous actions such as steering and acceleration, but the human can always take back the control

SAE Levels of Automation

- **Level 3:** conditional automation, where the vehicle can perform most of the driving task, but human override is still required
- **Level 4:** high automation where the vehicle performs all driving tasks under specific circumstances and in geofenced areas. Human override is still an option
- **Level 5:** full automation with the vehicle performing all driving tasks in all conditions. No human attention or interaction is required

Current State

- There are several limitations and barriers that could impede adoption of AVs, including: the need for sufficient consumer demand, assurance of data security, protection against cyberattacks, regulations compatible with driverless operation, resolved liability laws, societal attitude and behavior change regarding distrust and subsequent resistance to AV use, and the development of economically viable AV technologies

- Hacking autonomous cars

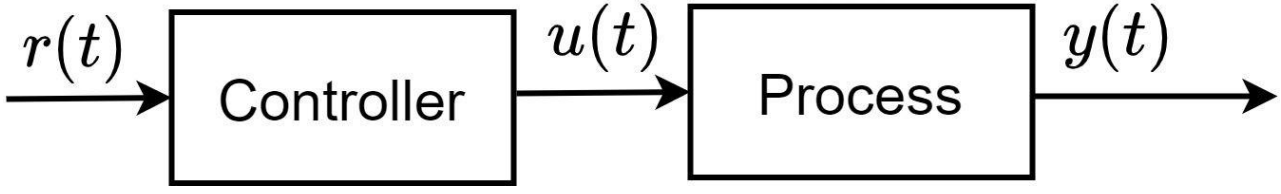
Cruise Control

- Cruise Control (CC) is a system that automatically controls the speed of a motor vehicle
- It is a servomechanism, i.e., a system that automatically uses error-sensing negative feedback to correct the throttle of the car and maintain a constant speed
- CC is a simple implementation of a control system that, in control theory, is called **proportional-integral-derivative (PID) controller**

Basic Stuff on Control Theory

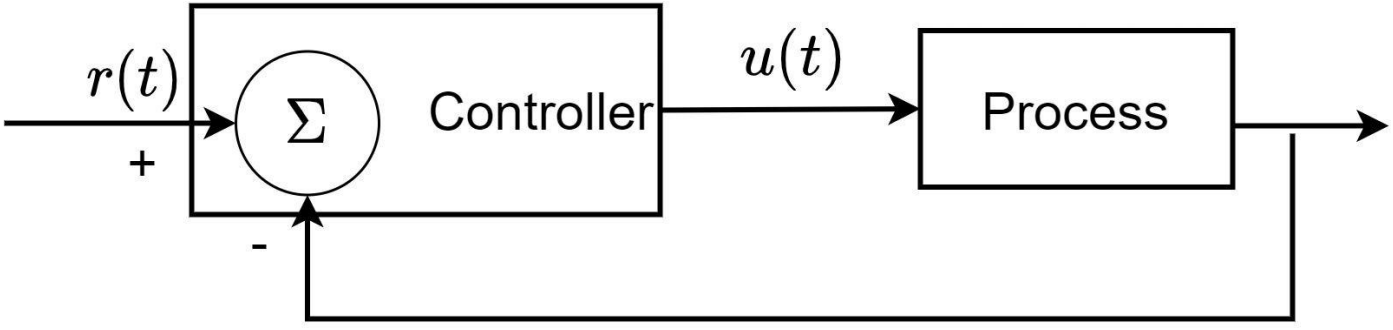
- Field of engineering and applied math. That deals with the control of dynamical systems
- The objective is to develop a *controller*, i.e., a device that acts on the system to achieve a desired state
- The controller monitors the process via a *process variable* and compares it with a *reference* or *set point*
- The difference between these values is called the error signal

Open Loop (Feedforward) Controller



- Compares a value that is not given by the process with the reference input
- For instance, a boiler that needs to heat the water for a given amount of time

Closed Loop (Feedback) Controller



- Compares a value that is given by the process with the reference input
- For instance, a car's cruise control or a thermostat

PID Controller

- It is a control loop mechanism using feedback to apply continuously modulated control
- It continuously computes an error as the difference between a desired setpoint and a measured process variable
- Based on the error value, it applies a correction based on proportional, integral, and derivative terms

Control Strategies

- A control system for an autonomous vehicle is broken down in two fundamental components
  - **Longitudinal control:** control of the longitudinal motion of the vehicle, with variables being the throttle and brake inputs
  - **Lateral control:** control the lateral dynamics of the vehicles, with variables being the steering inputs to govern the steering angle and heading (two different things)

Kinematic Model

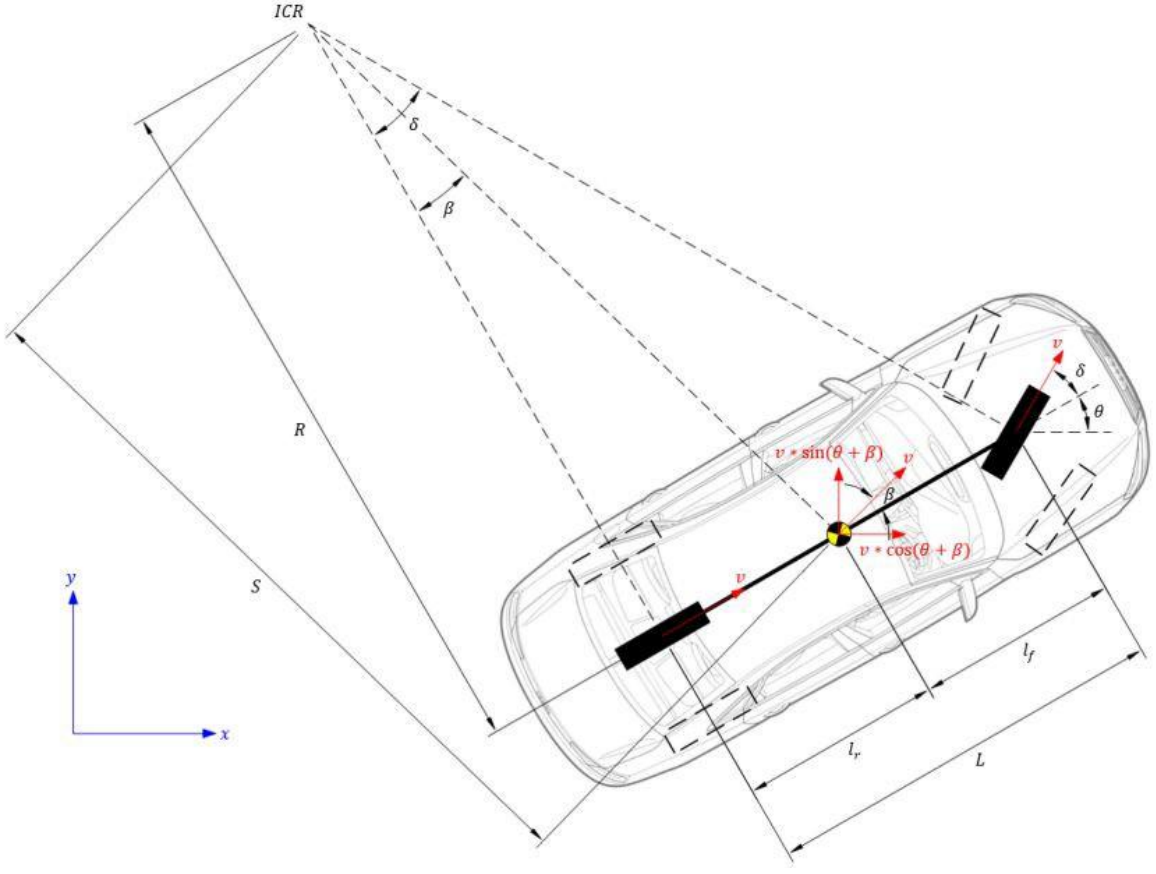
- Kinematics is the study of motion of a system disregarding the forces and torques that govern it
- Can be employed in situations wherein kinematics relations are able to sufficiently approximate the actual system dynamics
- Valid only for systems that do not perform aggressive maneuvers at lower speeds, e.g., driving slowly and making smooth turns
- We however start from them to make things simpler

Kinematic Bicycle Model

- Model to capture vehicle dynamics under normal driving conditions
- Strike a good balance between simplicity and accuracy → widely adopted
- The idea is to define a vehicle state and see how it evolves over time based on the previous state and current control inputs
- Let the vehicle state q comprise the x,y coordinates of location, heading angle theta, and velocity v

$$q = [x, y, \theta, v]^T$$

Kinematic Bicycle Model



Kinematic Bicycle Model

- For controls, we need to consider both longitudinal and lateral steering commands
- Brake and throttle contribute to longitudinal acceleration in range $[-a'_{max}, a_{max}]$
- Steering command alters the steering angle of the vehicle

$$\delta \in [-\delta_{max}, \delta_{max}]$$

- The control vector is hence $u = [a, \delta]^T$

Kinematic Bicycle Model

- Using the distance between the rear wheel axle and the vehicle's center of gravity, we can compute the slip angle beta as

$$tan\left(\beta\right) = \frac{l_r}{S} = \frac{l_r}{\left(\frac{L}{tan(\delta)}\right)} = \frac{l_r}{L} * tan\left(\delta\right)$$

$$\therefore \beta = tan^{-1}\left(\frac{l_r}{L} * tan\left(\delta\right)\right)$$

- Ideally $l_r = L/2 \Rightarrow \beta = tan^{-1}\left(\frac{tan\left(\delta\right)}{2}\right)$

Kinematic Bicycle Model

- Resolving the velocity vector v into x and y we get

$$\dot{x} = v * cos\left(\theta + \beta\right)$$

$$\dot{y} = v * sin\left(\theta + \beta\right)$$

- In order to get theta, we first need to calculate S as

$$S = \frac{L}{tan\left(\delta\right)} \quad \rightarrow \quad R = \frac{S}{cos\left(\beta\right)} = \frac{L}{\left(tan\left(\delta\right) * cos\left(\beta\right)\right)}$$

$$\dot{\theta} = \frac{v}{R} = \frac{v * tan\left(\delta\right) * cos\left(\beta\right)}{L}$$

Kinematic Bicycle Model

- Finally, we can compute $\dot{v} = a$
- We hence get the continuous-time kinematic model

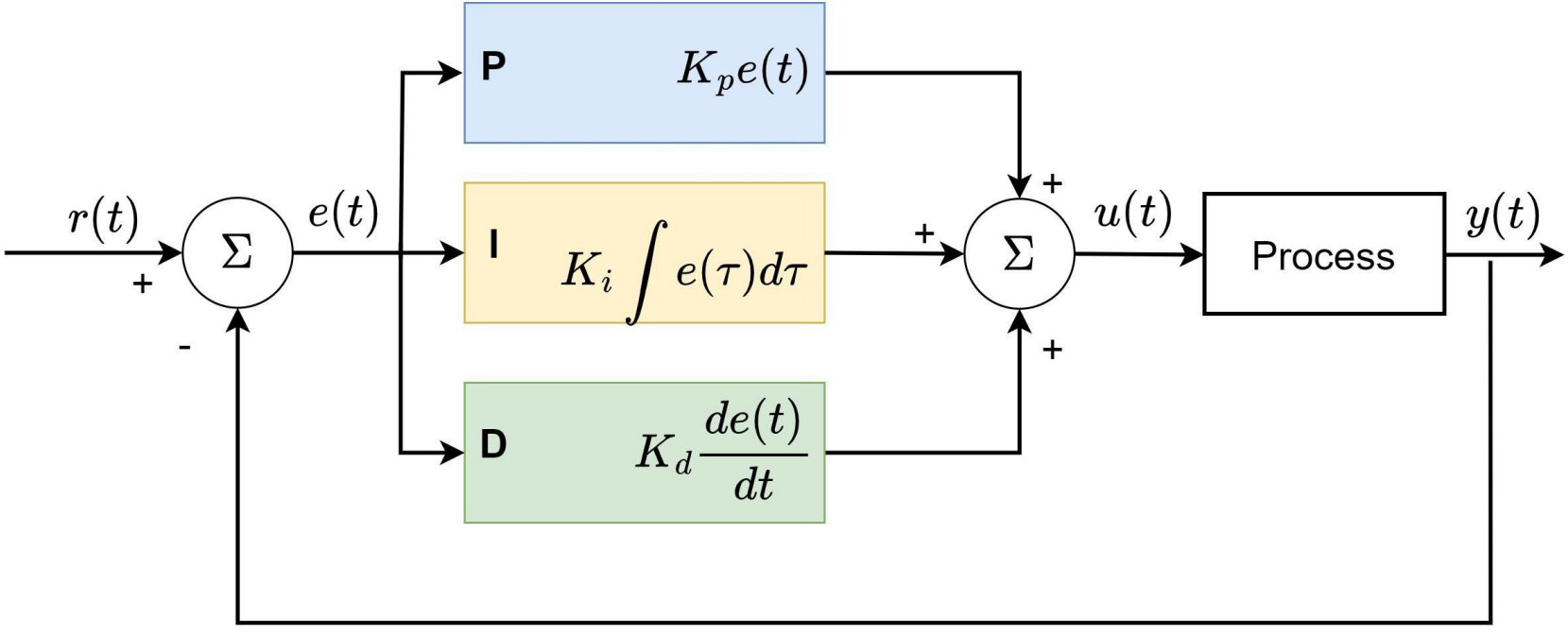
$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v * cos\,(\theta + \beta) \\ v * sin\,(\theta + \beta) \\ \frac{v * tan(\delta) * cos(\beta)}{L} \\ a \end{bmatrix}$$

- And hence the discrete-time model

State transition equation

$$\begin{cases} x_{t+1} = x_t + \dot{x}_t * \Delta t \\ y_{t+1} = y_t + \dot{y}_t * \Delta t \\ \theta_{t+1} = \theta_t + \dot{\theta}_t * \Delta t \\ v_{t+1} = v_t + \dot{v}_t * \Delta t \end{cases}$$

PID Controller



PID Controller

- The error term is computed as the difference between a desired setpoint and the output of the process
- Based on the error value, the controller applies a correction
- P is proportional to the error
- I is the integral of the error in a predefined past time window
- D is the derivative of the error and is the best estimate of the future trend

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

PID Controller for CC

- The controller takes the input on the car's speed value from a speedometer cable, the engine's RPM, or wheel speed sensor
- Based on the error value, the controller pulls the throttle cable to increase or decrease the speed
- Proportional part of the controller
  - Adjust the throttle proportional to the speed difference between the target one and the actual one

PID Controller for CC

- The integral of the speed is the distance
  - Difference with the distance it would have traveled at the target speed
  - Deals with hills

$$d(T) = \int_T v(t)dt$$

- The derivative of the speed is acceleration
  - Helps in responding quickly to changes

$$a(t) = \frac{v(t)}{dt}$$

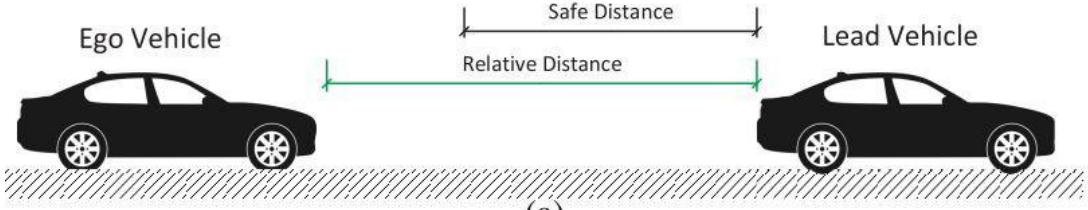
Adaptive Cruise Control

- Adaptive Cruise Control (ACC) is a system that automatically controls the speed of a motor vehicle by integrating sensing capabilities on the vehicle
- The vehicle uses a radar/LiDar/laser to compute the distance from a car in the front
- This distance is reported to the controller, which acts on the speed to maintain a minimum safety distance
- **SAE Level 1 of autonomous cars**

Example of ACC

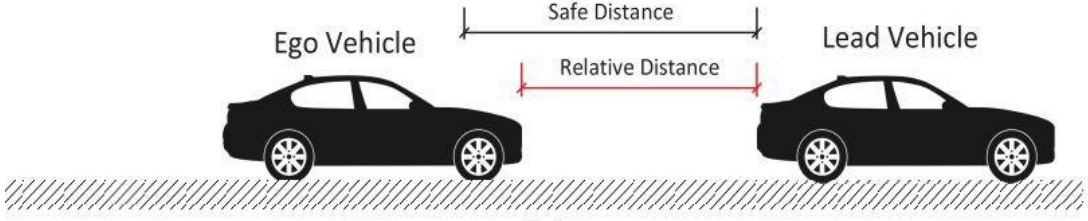
- Assume we have two cars, one in the front (lead) and one in the back
- We assume that the one in the back (ego vehicle) has ACC
- Ego uses a radar to measure the distance to the lead vehicle
- The lead car is supposed to be driving in the same lane
- Ego also uses the radar to measure the relative speed between the two cars

Example of ACC

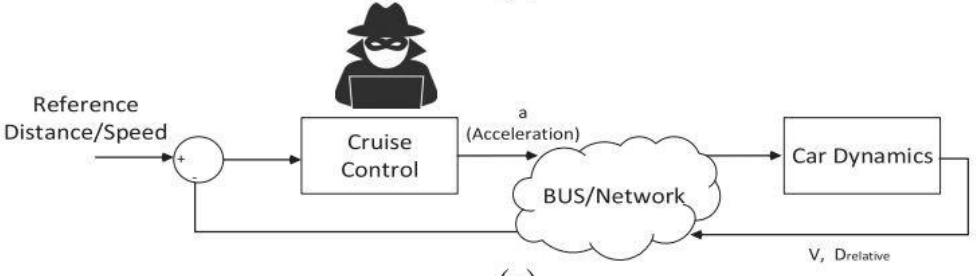


(a)

Control Objective:
$D_{relative}=D_{safe}$



(b)



(c)

Example of ACC

- **First mode:** the ego vehicle should drive at a driver's specified speed as long as it maintains a safe distance with the lead vehicle
- **Second mode:** the space between the two vehicles is controlled (by changing the ego vehicle's speed) so that the two vehicles do not get closer than a safe distance
- According to real-time measurements, either working modes can be enabled by the ACC system

Example of ACC

- To determine the operating mode, the ACC applies the following rules
  - If relative distance is >= safe distance, then use speed control mode to track the driver set velocity
  - If relative distance is < than safe distance, then the space control mode is active and keep track of the distance

ACC Attack Scenarios

- We explore two different attack scenarios
- **Attack 1:** the attacker compromises the ACC unit
- The attacker remains dormant and monitors the measured distance to the front vehicle
- At the times this distance is at its lowest (presumably near the minimum safe distance), it creates a spike in the control signal and makes the vehicle accelerate
- This could similarly happen when the front vehicle suddenly brakes and the compromised ACC refuses to reduce the speed

ACC Attack Scenarios

SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- We explore two different attack scenarios
- **Attack 2:** the attacker compromises the ACC unit
- Unlike the first scenario he/she does not ambush for the attack
- Trivially lowers the ACC's reference distance
- During the times ACC is in mode 2 and tries to maintain the safe distance, it is practically following a false reference
- Since this difference is trivial and not noticeable to the driver, this attack remains covert or stealth

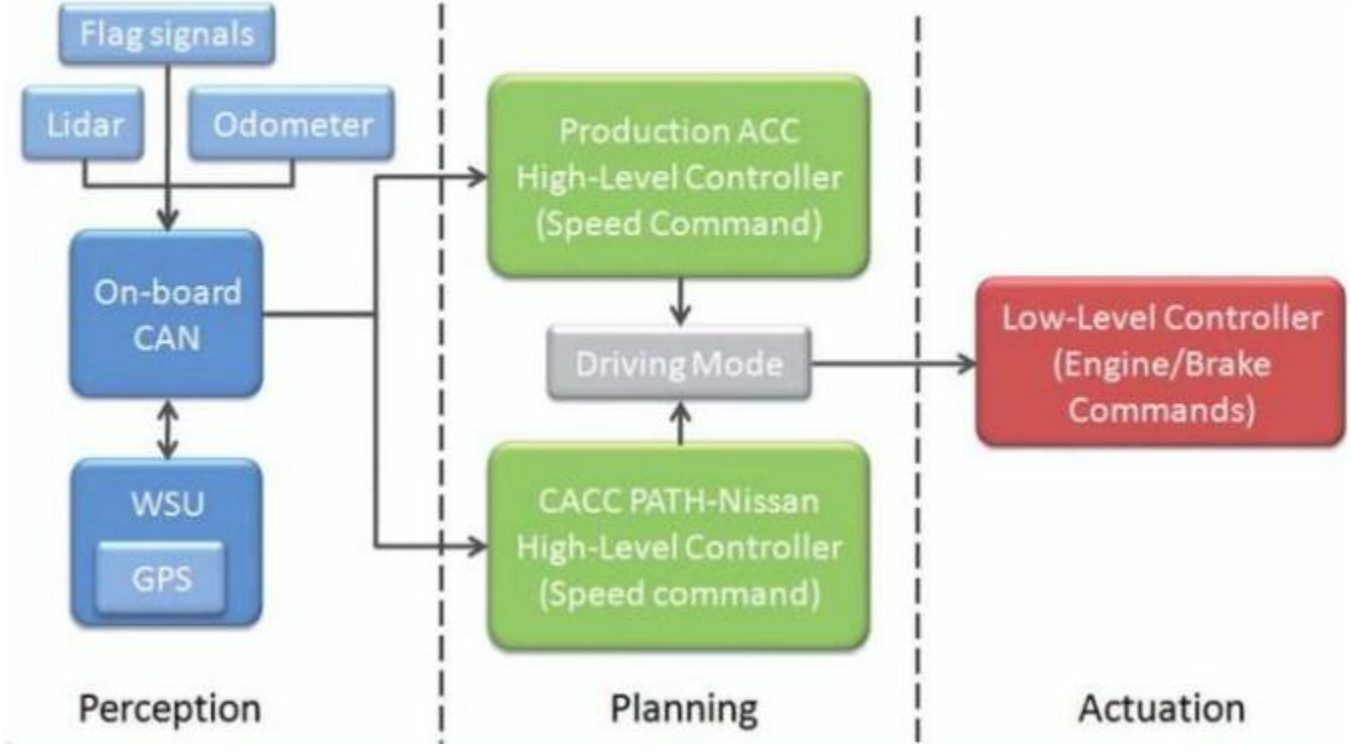
Cooperative Adaptive Cruise Control

- Cooperative Adaptive Cruise Control (CACC) is a system that automatically controls the speed of a motor vehicle by integrating sensing capabilities on the vehicle and communication capabilities with other vehicles

- It uses Vehicle-to-Everything (V2X) communication
- In addition to what ACC does, CACC uses the preceding vehicle's acceleration in a feed-forward loop

- This info passes via the Cooperative Awareness Messages

Cooperative Adaptive Cruise Control



Cooperative Adaptive Cruise Control

- **Perception Phase:**
  - Get information from on-board sensors and include them in CAN data
  - The Wireless Safety Unit (WSU) provides i) data transmitted by other CAVs in the CACC system through V2V communications, ii) data collected by GPS with wider area augmentation system differential corrections (e.g., vehicle position assigned in the CACC system)
  - Information derived from on-board sensors such as Lidar, odometer and flag signals will also be received and included on the CAN bus data structure

Cooperative Adaptive Cruise Control

- **Planning Phase:**
  - Includes the high level controller (longitudinal control algorithms)
  - The controller is usually connected and instructs the vehicle via the CAN bus
  - Very similar to ACC
- **Actuation Phase:**
  - Execute target reference command transmitted from the planning phase
  - Low-level controller converts the target speed commands into throttle and brake actions

Control in CACC

- The vehicle dynamics is calculated by the vehicle controller
- In particular, we often use longitudinal control, i.e., we write the sum of the forces acting on the vehicle and use them to maintain the same longitudinal speed of the other vehicles in a CACC system while keeping a fixed longitudinal inter-vehicle distance
- CACC technology allows CAVs to form vehicle platoons with shorter inter-vehicle distances

A Model of CACC

- Let us consider a platoon of K cars, numbered from 0 to K-1
- We assume that the cars all drive in a single straight lane and that their order can not change
- We denote the spatial position, velocity, acceleration of car i as q_i, v_i, a_i
- We indicate the distance between the front bumper of car i and the rear bumper of car i-1 as $d_i = q_{i-1} - q_i$
- We indicate the desired distance between car i and car i-1 as $d_{r,i}$

A Model of CACC

- Cars desire to follow a constant headway policy
- where the last term is a constant distance offset and h is the desired headway of car i $d_{r,i} = h_{d,i} v_i + L_i$

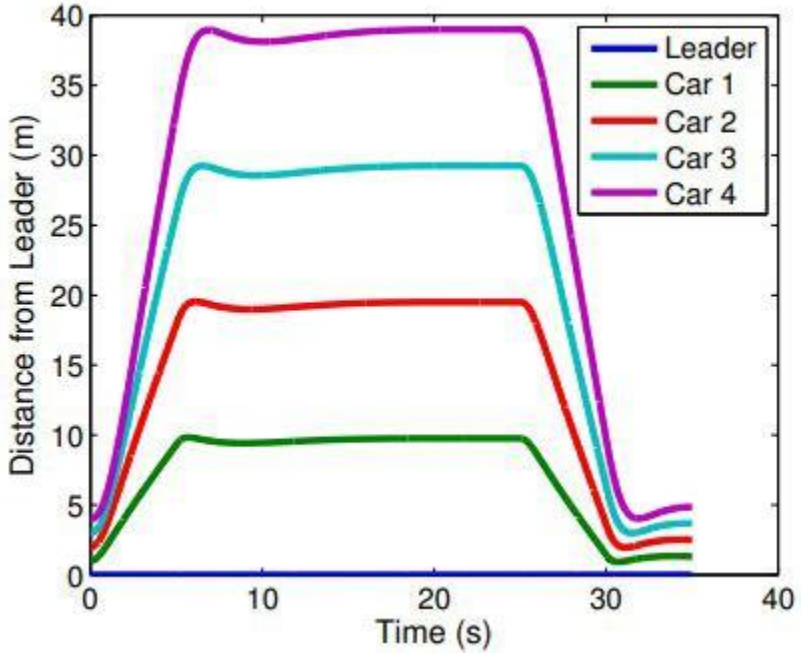
Measured in seconds

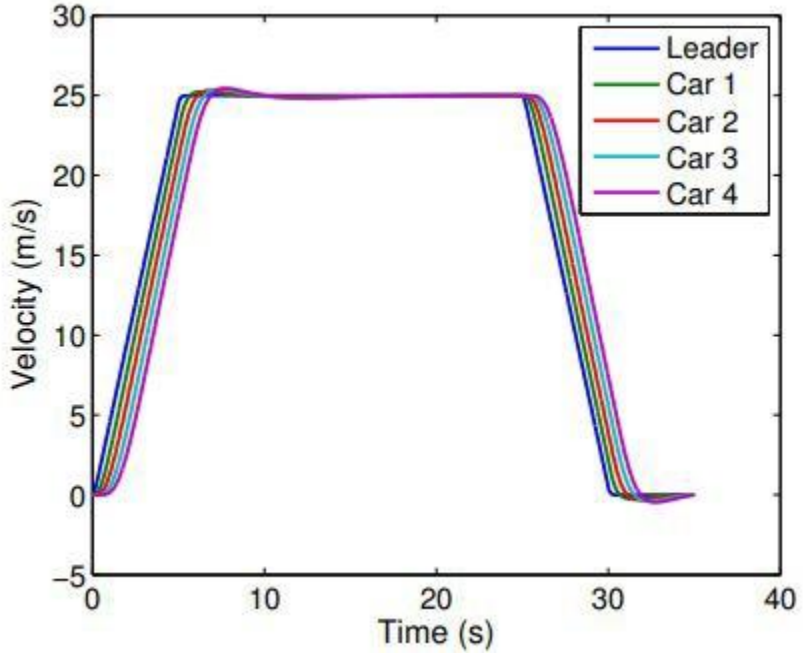
- Given the error at car i $e_i = d_i - d_{r,i}$
- $e_i = q_{i-1} - q_i - h_{d,i} v_i - L_i$
- We can define the control strategy based on the desired acceleration value $u_i = u_{fb,i} + u_{ff,i}$
- Sum of feedback input and communication-received input

A Model of CACC

- We measure inter-vehicle distance with radar and use PD feedback for in-vehicle info $u_{fb,i} = k_p e_i + k_d \dot{e}_i$
- We account for a values received via DSRC stating the control strategy of vehicle i-1 $\dot{u}_{ff,i} = -h_{d,i}^{-1} u_{ff,i} + h_{d,i}^{-1} \hat{u}_{i-1}$
- This controller has been shown to work in real life
- J. Ploeg, B. T. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer. Design and experimental evaluation of cooperative adaptive cruise control. In Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on, pages 260–265. IEEE, 2011

Effects of Control in CACC





Attack Strategies in CACC

- An attacker can cause a series of abnormal behaviors in CACC system, in particular referring to the platooning case
- The attacker may either be an external actor or an inner vehicle/byzantine node
- **Reduced headway attack:** a car ignores the recommended headway speed that guarantees string stability and follows closer $h_{d,a} < h_{d,min}$
- This attack would likely be implemented by a driver who wants to increase fuel savings by decreasing draft or a driver who manually drives with extremely small headways

Attack Strategies in CACC

- **Joining without radar:** a car attempts to become part of a platoon without having the necessary radar, or other distancing equipment
- A driver who does not want to buy a new vehicle but retrofits a car with DSRC which, unlike radar, does not require per vehicle tuning
- The reaction of the car is uniquely based on the feedforward information which is dangerous in case of communication problems (e.g., congestion)
- The attacker control strategy is hence $u_a = u_{ff,a}$

Attack Strategies in CACC

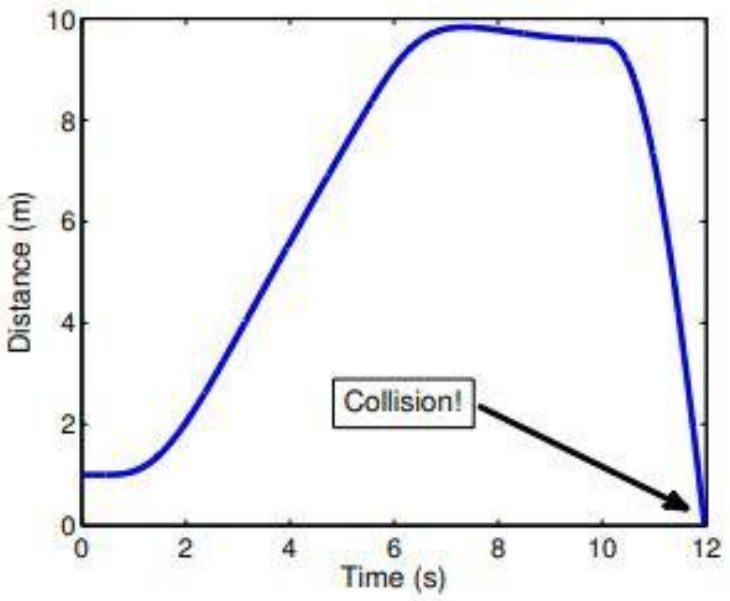
- **Mis-report Attack:** The attacker misinforms the vehicle that is following to increase the following car's headway or to cause a change in the following car's behavior
- The attacker mounting this attack could either follow the prescribed control law or choose an alternative control law

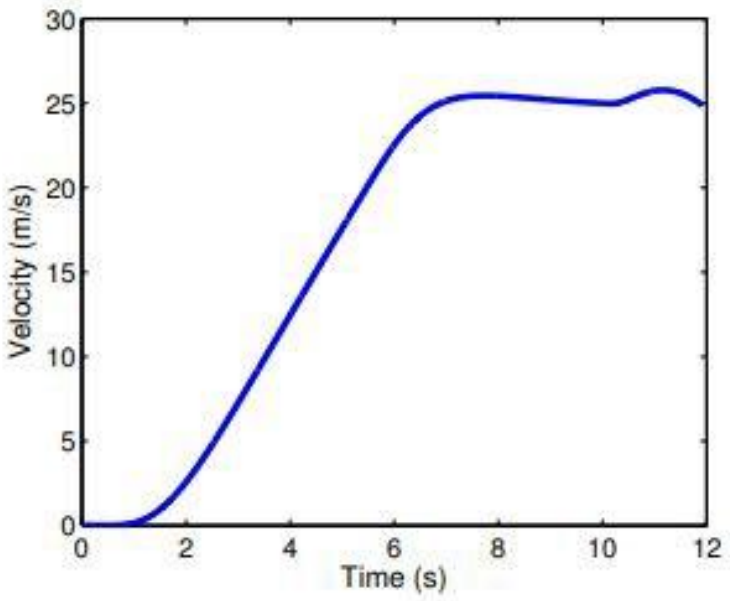
- Assuming the attacker misreporting only its behavior, then $u_a = u_i$
- This is motivated by wanting to increase the following distance of the preceding car

Attack Strategies in CACC

- **Collision Induction Attack:** the attacker broadcasts an acceleration profile indicating that they are speeding up which causes the following vehicle to accelerate
- The attacker starts to aggressively brake which causes the error between the attacker and following car to quickly increase
- Very similar to attacks that could be mounted in the current highway system
- If a driver was to jam on their breaks during rush hour while being tailgated, the vehicle would likely be rear ended.

Collision Induction Attack





Summary

Attack	Impact	Motivation	Method
Reduced Headway Attack	Decreased String Stability	Decreased fuel consumption Increased density	Misbehavior
Joining Without Radar	Decreased String Stability Danger in wireless congestion	Decreased cost over radar equipped car	Misbehavior
Mis-report Attack	Decreased Performance	Mistrust of the system	Misinformation
Collision Induction Attack	Collision Loss of Life Property Damage	Maliciousness Terror	Misbehavior & Misinformation
Non-Attack Abnormalities	Decreased Performance Decreased String Stability	Improper Maintenance	Misbehavior

How do we detect and defend against these attacks?

System Description

- We can describe our CACC system with a double integrator model with a lag constant nu for each car

$$\begin{aligned}\dot{a}_i &= -\eta_i^{-1} a_i + \eta_i^{-1} u_i \\ \dot{v}_i &= a_i \\ \dot{q} &= v_i \\ \dot{e}_i &= v_{i-1} - v_i - h_{d,i} a_i .\end{aligned}$$

System Description

- Let us define a vector with the state of the car $x_i^T = [e_i, v_i, a_i, u_{ff,i}]$
- The state update equation of the car can be written as a linear system

$$\dot{x_i} = A_{i,i}x_i + A_{i,i-1}x_{i-1} + B_{s,i}u_i + B_{c,i}\hat{u}_{i-1}, \ \forall\, i > 0$$
$$\dot{x_0} = A_0 x_0 + B_{s,i}u_r$$

where

$$A_{i,i} = \begin{pmatrix} 0 & -1 & -h_{d,i} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_i^{-1} & 0 \\ 0 & 0 & 0 & -h_{d,i}^{-1} \end{pmatrix}, \qquad A_{i,i-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

System Description (2)

$$B_{s,i}^T = \begin{pmatrix} 0 & 0 & \eta_i^{-1} & 0 \end{pmatrix},$$

$$B_{c,i}^T = \begin{pmatrix} 0 & 0 & 0 & h_{d,i}^{-1} \end{pmatrix},$$

$$A_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_0^{-1} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

- We alsod define a variable X representing the state of the whole system $X^T = [x_0^T, x_1^T, \ldots x_{K-1}^T]$
- We define the inputs to the system as $U^T = [u_0, \hat{u}_0, u_1, \hat{u}_1, \ldots, u_{K-1}]$
- This allows us to write the equation for the whole system

$$\dot{X} = AX + BU$$

Discretization

- We assume that the controller implements digital control instead than analog
- The controller sampling time will depend also on the sampling time of the radar and of the communication system

- Thus, we can rewrite the update equations as $X[k+1] = A_d X[k] + B_d U[k]$
- We also rewrite the control strategy as $u_i = k_1 x_i[k] + k_2 x_i[k-1]$ (1)
- Assuming that radar update is 1ms and DSRC update is 100 ms

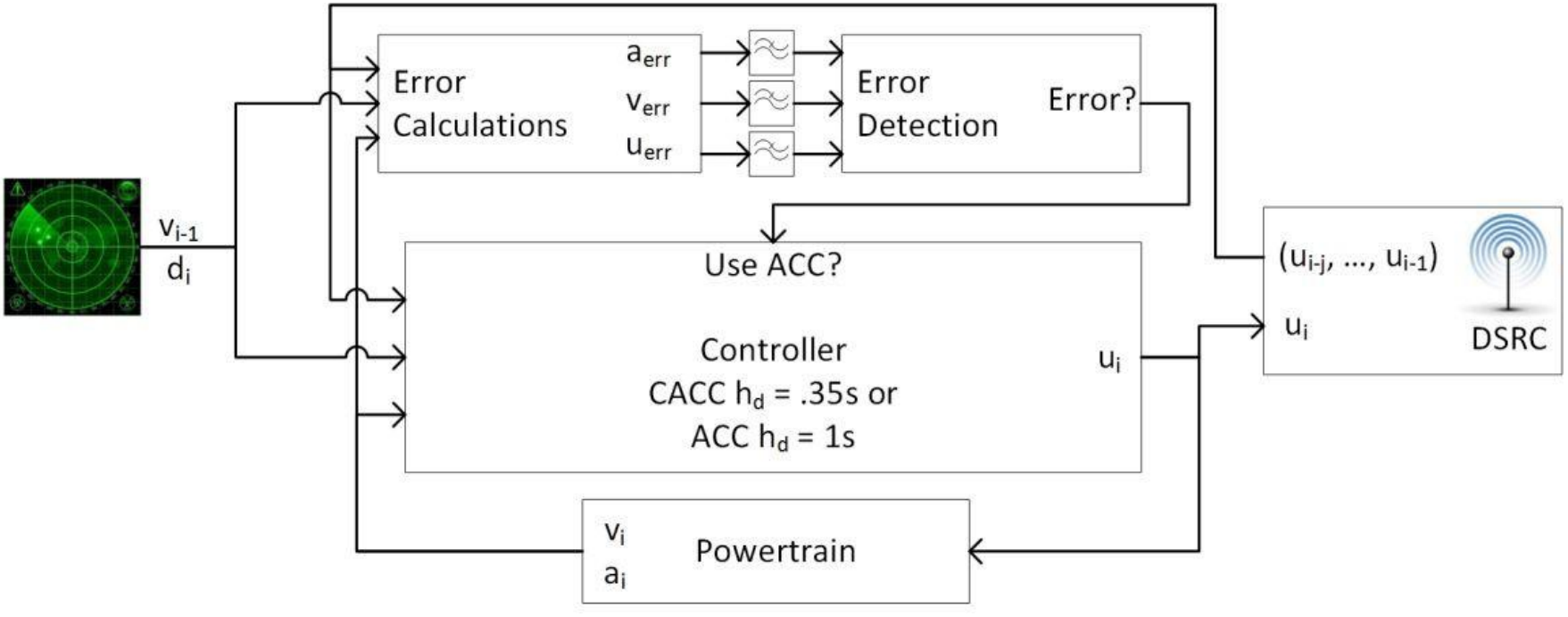
$$k_1 = \begin{pmatrix} k_p + \frac{k_d}{.001} & 0 & 0 & 1 \end{pmatrix} \qquad k_2 = \begin{pmatrix} -\frac{k_d}{.001} & 0 & 0 & 0 \end{pmatrix}$$

Model-Based Detection

- Every car model the expected behavior of the vehicle directly in front of them
- Vehicles then compare the calculated expected behavior with the observed behavior

- The car is then able to detect both malicious and benign abnormalities
- Once abnormal behavior is detected, the car switches from operating in a cooperative platoon framework to a radar only based adaptive cruise control framework where it is safe even if the preceding car is mounting an attack

Model-Based Detection



Model-Based Detection

- Car i wants to model the behavior of car i-1 given the data packets from car i-j
- We define the modeled state of car i-1 as $x_{m,i-1}$ (notice: pedix m stands for “modeled”)
- We can define the state of all cars in the model as

$$X_m = [x_{m,i-j}, x_{m,i-j+1}, \ldots, x_{m,i-1}]$$

- We can write the system update equation for the model as

$$X_m[k+1] = A_m X_m[k] + B_m U_m[k]$$

Model-Based Detection

- We assume all cars behave according to the control law in red
- During an update period we can hence use it to define (remember ❶)

$$U_m[k] = \phi_1 X_m[k] + \phi_2 X_m[k-1] + \phi_3 \hat{u}_{i-j}[k]$$

$$\phi_1 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & k_1 \end{pmatrix}, \phi_2 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & k_2 \end{pmatrix} \quad \phi_3 = \begin{pmatrix} 1, 1, 0, \dots, 0 \end{pmatrix}^T$$

- During a non-update period we can hence use it to define (again ❶)

$$U_m[k] = \phi_4 X_m[k] + \phi_5 X_m[k-1] + \phi_6 U_m[k-1]$$

$$\phi_4 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, \phi_5 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, \quad \phi_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}.$$

Model-Based Detection

- The modeling technique is based on double integrator and can be made complicate at wish
- Considering tradeoffs in accuracy, calculation cost, and time for calculation
- Improvements that could be considered in the modeling include capturing non-linear behavior of the vehicles drivetrain and using terrain mapping to predict variation

Thresholding

- Once we have the prediction model, we can predict whether the error is acceptable or not
- We indicate the measured values as $\hat{x}_{i-1,m}$
- We also assume we can compute acceleration and velocity
- However, we cannot measure the error with vehicles further than i-1 since we do not have a line of sight with cars further than a single hop

Thresholding

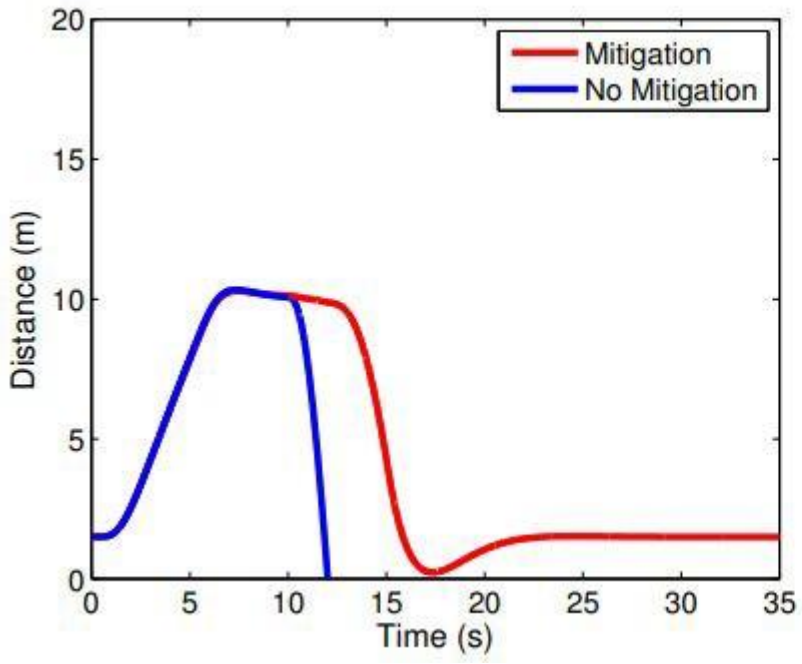
- We use an model error normalized to acceleration for vehicles for which we only have the received information and no measuring possibility

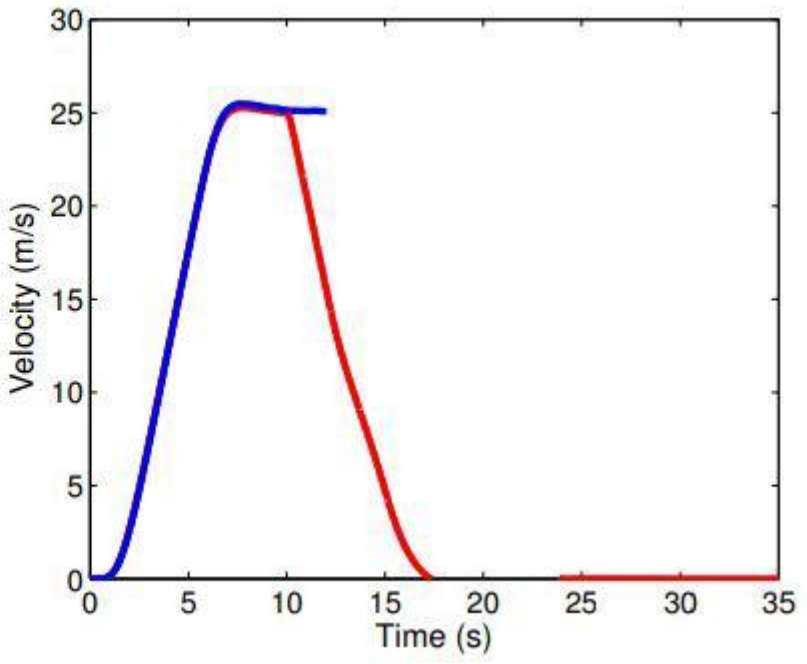
$$(a_{err}|\hat{u}_{i-j}) = \left(\frac{(a_{m,i-1}|\hat{u}_{i-j}) - a_{i-1,}}{a_{i-1}}\right)^2$$

$$(v_{err}|\hat{u}_{i-j}) = \left(\frac{(v_{m,i-1}|\hat{u}_{i-j}) - v_{i-1}}{a_{i-1}}\right)^2$$

$$(\hat{u}_{err}|\hat{u}_{i-j}) = \left(\frac{(\hat{u}_{m,i-1}|\hat{u}_{i-j}) - \hat{u}_{i-1}}{a_{i-1}}\right)^2$$

Results





Collision induction attack detection and mitigation

A Different Methodology

- Model predictive control can be implemented in multiple ways
- In the previous example, each vehicle implements the model of the preceding vehicle

- However, we can model more complex systems
- For instance, it could be possible for vehicle i to simulate the entire platoon up to vehicle i-1

Replay Attacks in CACC

- The platoon of vehicles is ideally going to reach a steady state condition where all the relative distances and the velocities are constant and where the accelerations are equal to zero

- We suppose the attacker can:
  - Record measurements during a steady state situation without influencing in any way the platoon behavior
  - Replays the recorded data while acting to either degrade the platoon performance or damage the vehicles involved in the platoon

Replay Attacks in CACC

SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- We assume the attacker has already got full access to the cryptography keys and to the in-vehicle network and it is acting as an intelligent insider
- The controller of a legitimate vehicle starts relying on malicious information
- We however assume that the leader is immune to hacking, and is hence secure

Detection of Replay Attacks

- The leader vehicle broadcasts its velocity, acceleration, and a noisy signal Δa_k^0
- The leader vehicle generates the noisy control signal as

$$\tilde{a}_k^0 = a_k^0 + \Delta a_k^0$$

Actual control signal

Original control signal

$\Delta \bar{a}_k^0 \sim \bar{N}(0, \vartheta)$

Detection of Replay Attacks

- The following vehicles broadcast their speeds and their accelerations and read from the network the speed, the acceleration of the vehicle in front of them and the noisy signal generated by the leader
- Δa_k^0 is used to update the control law in the vehicle 1
- The noisy signal acts as a timestamped authentication signal which is propagated along the vehicles in the platoon formation through the control laws exploited in each vehicle

Platoon Virtual Model

- We assume the leader vehicle is in a steady state and immune to malicious attacks
- We can model a virtual platoon up to vehicle i to be fed with the noise signal to estimate the acceleration of vehicle i-1 in absence of replay attacks
- We can then compare the estimate with the measure value and detect attacks

Platoon Virtual Model

- The system evolution can be modeled as

Noisy signal

Process noise

$$x_{k+1} = Ax_k + B\Delta a_k^0 + w_k$$

$$y_k = Cx_k + v_k = a_k^{i-1} + v_k$$

Measurement noise

$$x_k = \left[v_k^0, d_k^1, v_k^1, a_k^1, \cdots, d_k^{i-1}, v_k^{i-1}, a_k^{i-1}, d_k^i, v_k^i, a_k^i\right]^T$$

$$x_0 \sim N(\bar{x}_0, \Sigma),\ w_k \sim N(0, Q),\ v_k \sim N(0, R)$$

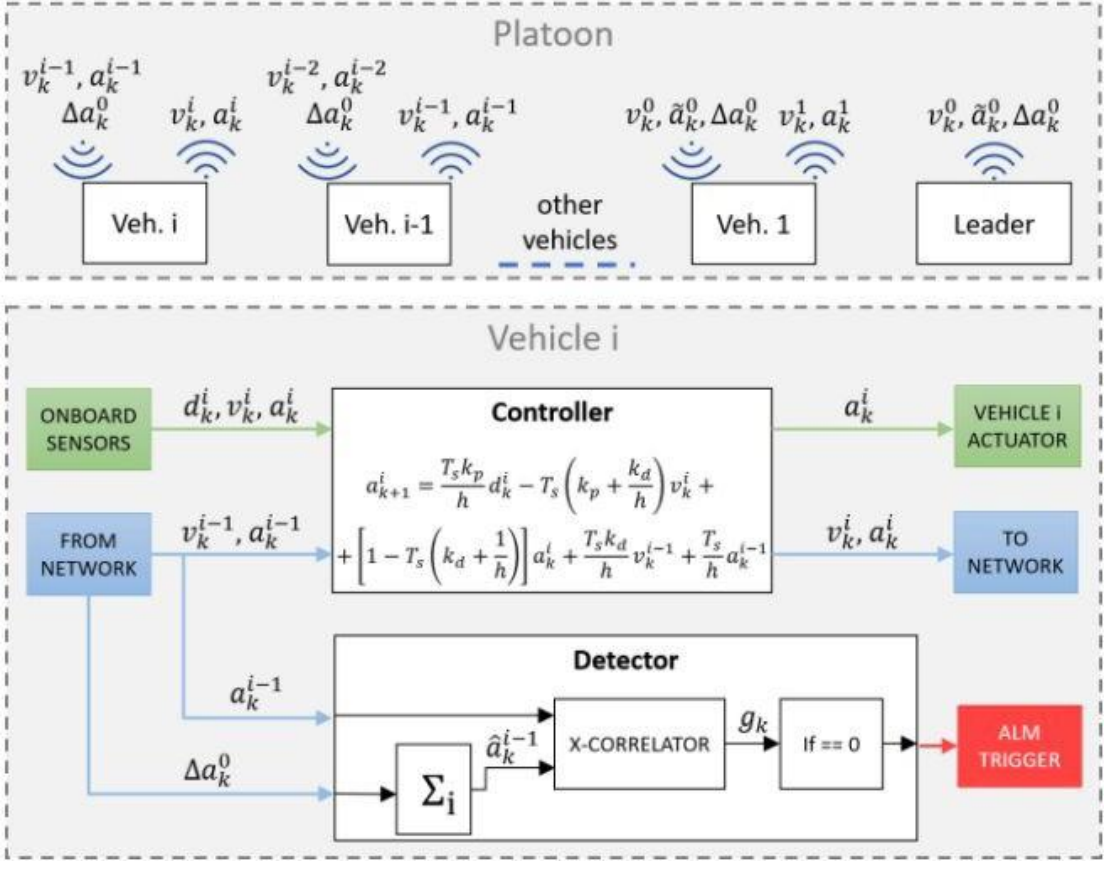
Initial state

Objective: estimate the acceleration $\hat{a}_k^{i-1}$ of vehicle i-1 given input the noisy signal and assuming no replay

Detection of Replay Attacks

- The noisy behavior is propagated by each vehicle to the following one by updating the control action and by exchanging speed and acceleration signals through the network with the next vehicle
- Each vehicle has a control algorithm and a detection algorithm
- The *control algorithm* uses the information from the sensors and DSRC as previously described
- The *detector* instead uses the acceleration of the previous vehicle and the noisy signal

Detection of Replay Attacks



Cross Correlator

- The i-th detector runs a virtual model that simulates the platoon up to vehicle i (in its steady state) with given input the authentication signal Δa_k^0
- The purpose of the model is to estimate the acceleration signal a_k^{i-1} in absence of attack starting from the noisy signal
- We use a cross-correlator to measure the correlation between signal a_k^{i-1} coming from the front vehicle and the estimate output of the mode $\hat{a}_k^{i-1}$
- If there is an attack, the output of the correlator should be zero

Cross Correlator

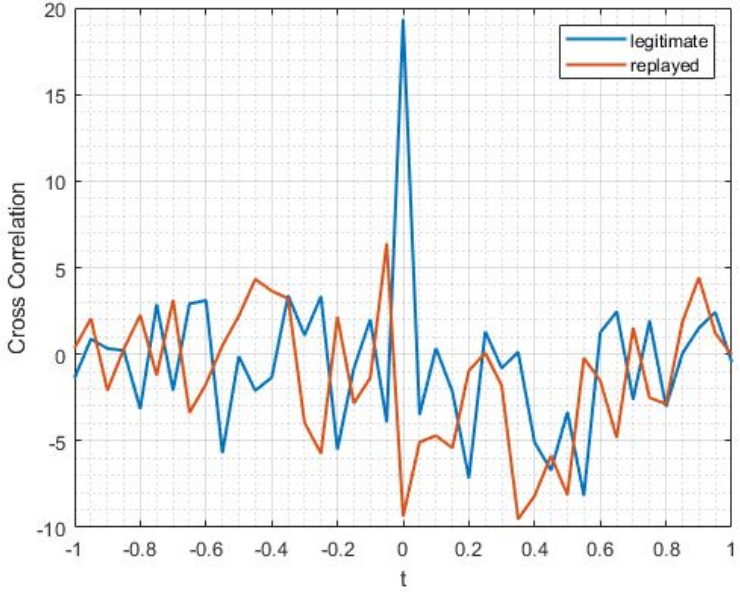
- The cross correlation is computed over a time window with a given length
- Given two functions f and g, the cross correlation can be computed as

$$(f \star g)(\tau) \triangleq \int_{t_0}^{t_0+T} \overline{f(t)} g(t+\tau)\, dt$$

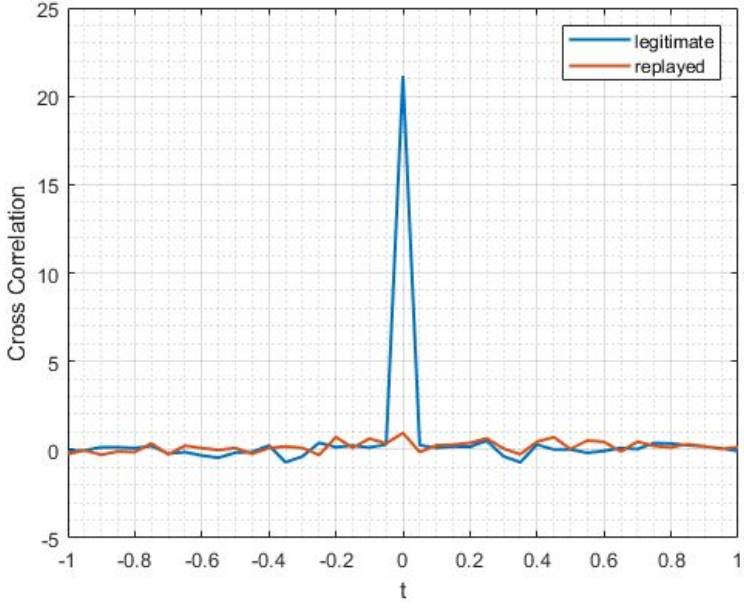
- A tradeoff between false detection and time of the detection is needed to choose the right window size with the purpose to properly detect the replay attack

Cross Correlator

- Effect of the cross correlator



Single realization



Average over 100

Sensors in Autonomous Driving

- Autonomous vehicle act based on i) Sense, ii)Understand, and iii) Act
- The sensing layer of vehicles is comprised of vehicular sensors that measure the physical properties of a vehicle's state and surroundings
- By sensing information from different sensors, the car construct a representation of its environment
- We've already seen that distance measurement sensors allow for ACC and CACC
- Sensors can be divided based on their functions in safety, diagnostic, convenience, and environment monitoring

Sensors in Autonomous Driving

- **Safety sensors:**
  - provide night vision, detect impending crashes, and tailor airbag deployment to each passenger's weight and position
- **Diagnostic sensors:**
  - detect vehicle malfunctions and offer malfunction alerts to drivers
- **Convenience sensors:**
  - maintain high air quality within the vehicle, control automatic mirror dimming, perform automatic braking and acceleration
- **Environment monitoring sensors:**
  - track the vehicle's surroundings, including traffic, street signage, and road conditions

Attack Vectors

- The sensing layer is vulnerable to malicious interference conducted both physically and remotely
- Tampering requires the attacker to physically access the car to and attach something like electromagnetic actuators or sound-absorbent foam
- Remote attack can be either
    - Roadside: the attacker places stationary attack equipment on one or more locations along the roadside
    - Front/rear/side: the attack equipment is mounted on the attacker's vehicle which follows the victim

The LiDAR

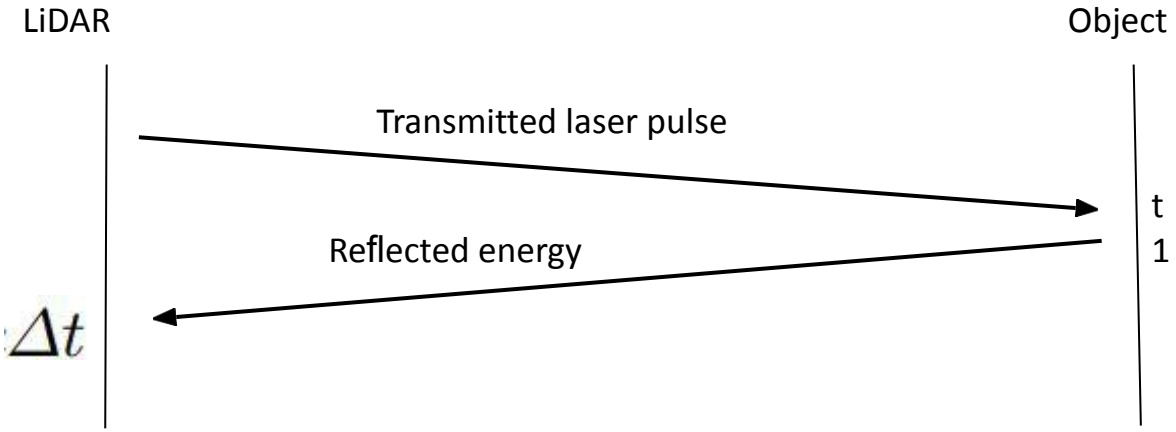
- Light Detection and Ranging (LiDAR) is an active remote sensing method or a sensor using this method to measure distance from nearby objects
- Active sensing is a way of analyzing the target of interest by exposing it to the energy (or signal) intentionally transmitted by the sensor itself

The LiDAR

- Two types of LiDAR
- **Scanning:** mainly composed of laser transceiver(s) and a moving rotary system for scanning, which allows acquiring a round view

- **Solid state:** do not require moving parts to acquire a round view
- Currently however we use scanning LiDARs, as the latter generally are more expensive to obtain the same reliability

How do LiDARs Work



Distance: $l = c\Delta t/2$

- The LiDAR rotates and measure such distances to create a cloud of points
- Multiple-layer LiDARs can also scan the slant angle

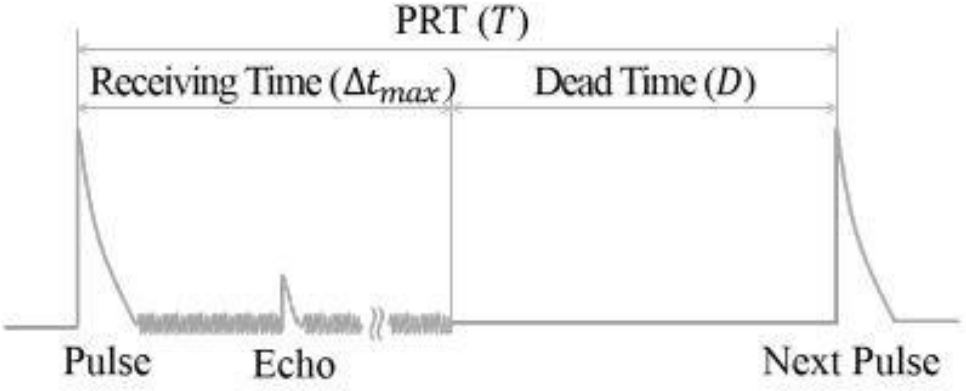
Ambiguities

- Since the LiDAR sends pulses with a given periodicity, there might be ambiguities in the reception of a signal
- To limit uncertainties, LiDARs define the receiving time Δt_{max} and the dead time D
- After sending a pulse, a LiDAR waits for its echoes for the duration of the receiving time. The received echoes are considered as that of the last transmitted pulse
- The the LiDAR ignores all the echoes received from the end of the receiving time up to rec. Time + dead time, and then transmits again

Ambiguities

- The LiDAR can hence detect objects up to a maximum distance, given by

$$l_{max} = c\Delta t_{max}/2$$



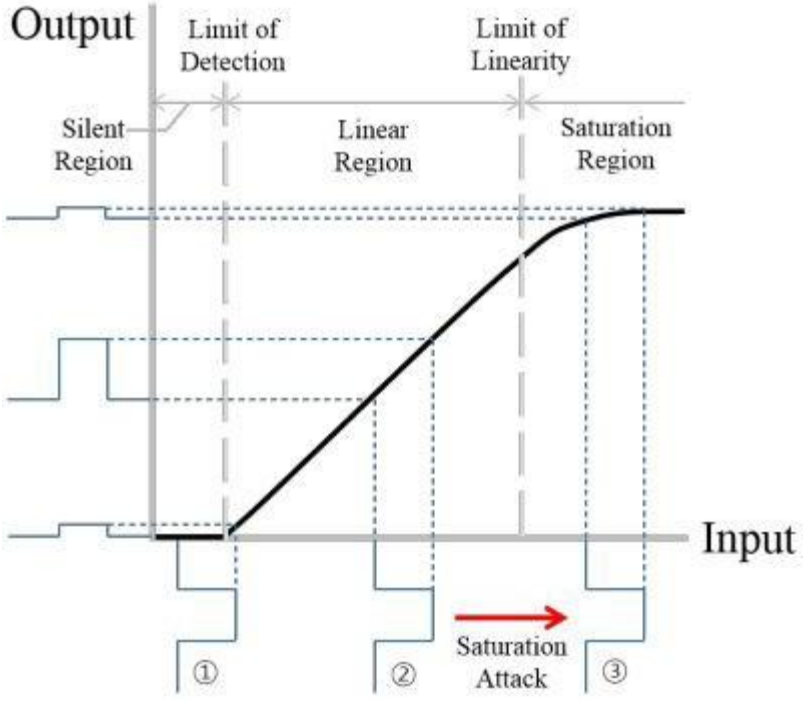
Aperture

- The LiDAR does not require a wide receiving angle if well calibrated
- Only echoes falling within the receiving angle can effectively affect the sensing result
- Te receiving transmission needs to cover the direction of the pulse transmission only during the maximum round trip time
- We can hence derive the receiving angle from the rotating speed and the maximum distance as

$$\Theta_R = \Delta t_{max} \cdot \omega = \frac{2 l_{max}}{c} \cdot \omega \ [^\circ]$$

What can an Attacker Do?

- Sensors transition curve comprise three regions
- The first line separates the silent region from the linear region where we actually have sensing
- The second threshold separate the linear region from the saturation region



What can an Attacker Do? (2)

- Saturation is a DoS attack
- The attacker can also spoof a sensor, deceiving the victim sensor by exposing it to the attacking signal which simulate particular circumstances
- Exploit a *semantic gap* between what the situation really is and what the sensor perceives it
- Example: earthquake and child shaking a seismometer have the same effect on the sensor

What can an Attacker Do? (3)

- Sometimes, active sensors can take a particular waveform (ping waveform) to differentiate its echoes from the other inbound signals
- The attacker should first acquire the ping waveform and then relay it after an intentionally inserted delay
- This is *spoofing by relaying*
- Spoofing is generally very difficult to detect also due to the semantic gap

Blinding Attack

- We can saturate LiDARs using light sources
- In particular, the attacker should point against the LiDAR a light source of the same wavelength as that used by the LiDAR
- Depending on the intensity of the light, the attacker can saturate a bunch of dots or an entire direction
- This type of saturation attack against LiDAR is called *blinding attack*
- Saturation attacks against LiDARs have some common characteristics

Common Features

- **Stealthiness against drivers and pedestrians**
- In order to not hinder human driving and for eye safety, lidars use infrared (IR) lasers

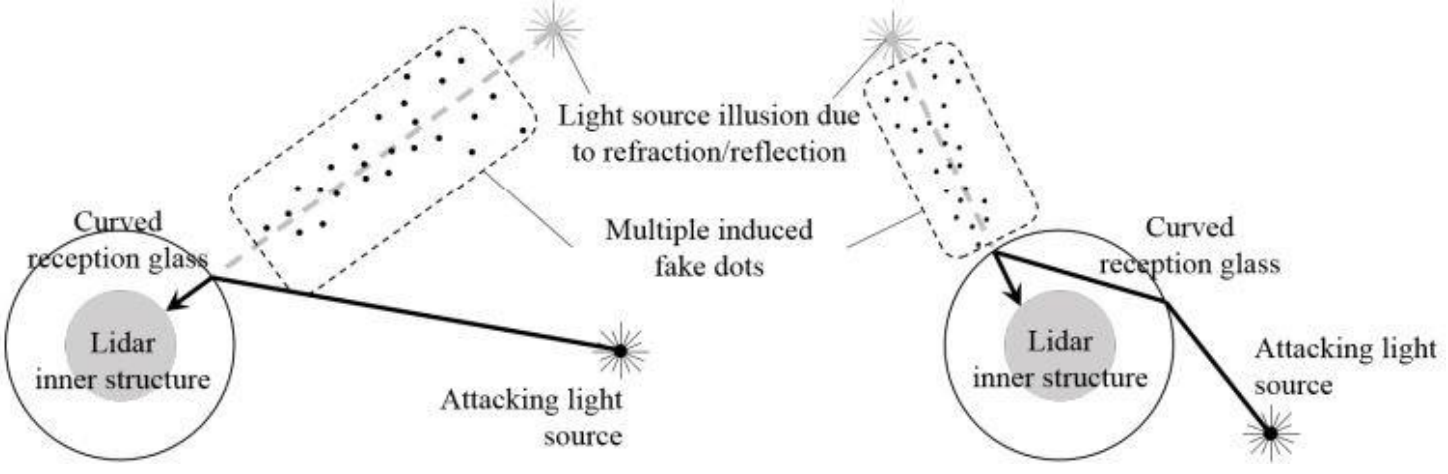
- Invisibility of the medium also assists stealthiness in saturating
- Irrespectively of the intensity, human drivers and pedestrians would be unaware, rendering the attack effective.

Common Features

- **Receiving angle**
- A wide receiving angle is not essential for lidars to sense objects in the field of view
- Therefore, LiDAR receivers typically have much smaller receiving angles compared to the angle of view
- This can limit the effect of saturating, because the attacking light comes from a certain direction when the LiDAR rotates
- In reality, however, the receiving angles of lidars are much larger than required

Common Features

- **Curved reception glass**
- An oblique incidence of strong light onto the curved reception glass can cause the appearance of fake dots in directions other than that of the attacking source

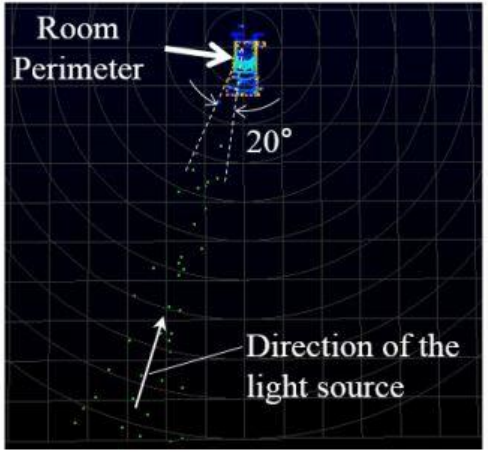


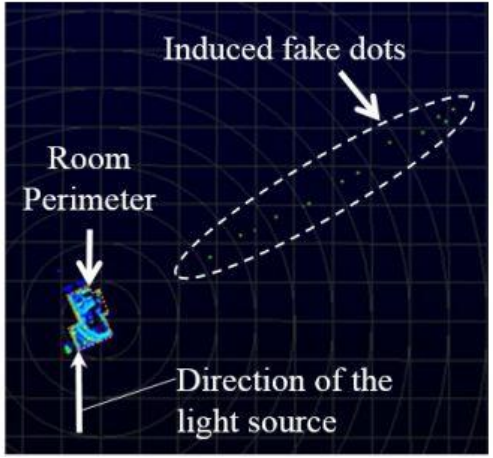
Effects of Saturation

- Experiments conducted on a LiDAR commonly used for cars, UAVs, and robots
- The LiDAR used offers a tool for real time visualization of the collected info
- For saturation, only a light source is needed: a 30mW, 905nm laser module (40 USD) and a 800mW, 905nm laser module (350 USD)

Effects of Saturation

Weak source

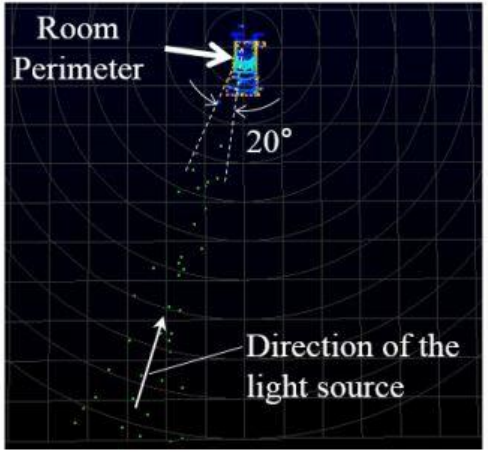


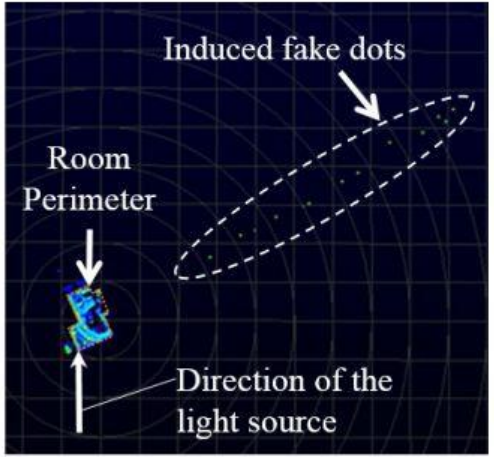


Strong source

- With weak light source, the LiDAR observes numerous randomly located fake dots
- Fake dots only in the direction of the source

Effects of Saturation

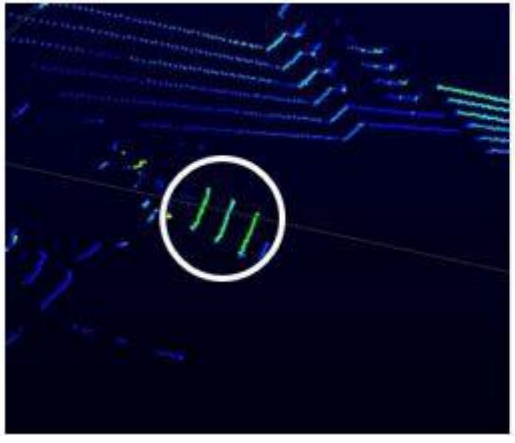


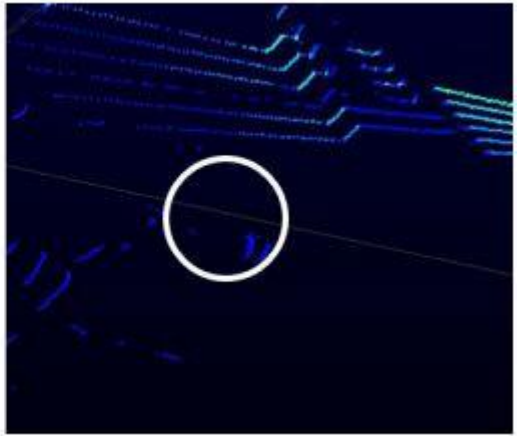


- With strong oblique light source the LiDAR sees fake dots also in direction other than that of the attacker's source
- This confirms that curved glasses can change the incoming direction of light sources

Effects of Saturation

Before





After

- With strong direct light source the LiDAR becomes completely blind in a sector

Spoofing by Relaying: Ideal Spoofing

- LiDARs measure distance as the round trip time of light
- The light source bounces on the first object it meets and then gets back to the transmitter
- The ideal spoofing procedure mimics this process
  1) Prepare an attack tool composed of a receiver, an adjustable delay component, and a transmitter of the same wavelength as that used by the lidar

  2) Aim at the target lidar with the attack equipment

Ideal Spoofing

3) Receive the target lidar pulse signal using the receiver

4) Add the required delay using the delay component

5) Fire a laser pulse back to the target lidar using the transmitter

- Ideally this process creates a unique dot
- The delay components should be carefully computed for the attack to be effective
- In particular, we can compute the delay required to create an object at distance l from the LiDAR

Ideal Spoofing

- Let the distance between the spoofer and the victim LiDAR be $l_s \leq l$
- We want to generate a delay that makes an echo appear $l - l_s$ further than the spoofer, i.e., round-trip time for the distance $l - l_s$
- This distance is derived as $d_i = 2 (l-l_s) / c$
- Although the procedure is correct, we have limitations
- The LiDAR receiving angle must be facing the attacker for the attack to be effective
- The LiDAR ignores echoes received after a certain time → need to fire back within the receiving time

Actual Spoofing

- The laser pulse from the lidar diverges. Accordingly, the attacker receiver obtains multiple adjacent laser pulses
- Only a part of these pulses exactly head in the direction of the receiver
- Next, irrespective of how close the receiver and transmitter are placed in the attack tool, they are apart by a certain distance
- Consequently, there is a time difference (S) between the detection of a laser pulse by the receiver and the firing of a pulse toward the transmitter

Actual Spoofing

- Due to this space, we need to change the aforementioned equation for generating delays $d_a = d_i + nT + S + d_p$

  where the last term is processing/propagation delay
- The time differences (T and S) are compensated by adding them to the ideal delay, because the delay component is triggered by the first received pulse
- The processing delay can be compensated likewise

Spoofing Characteristics

- **Stealthiness:** as for the saturation attack, the source is invisible to humans
- **Inducing multiple fake dots:** If the lidar rotates at a constant speed, an attacker can generate multiple fake dots with one attack tool
- This can be done by periodically firing back the attacking pulses, immediately after the first attacking pulse, with the same period as the Pulse Repetition Time

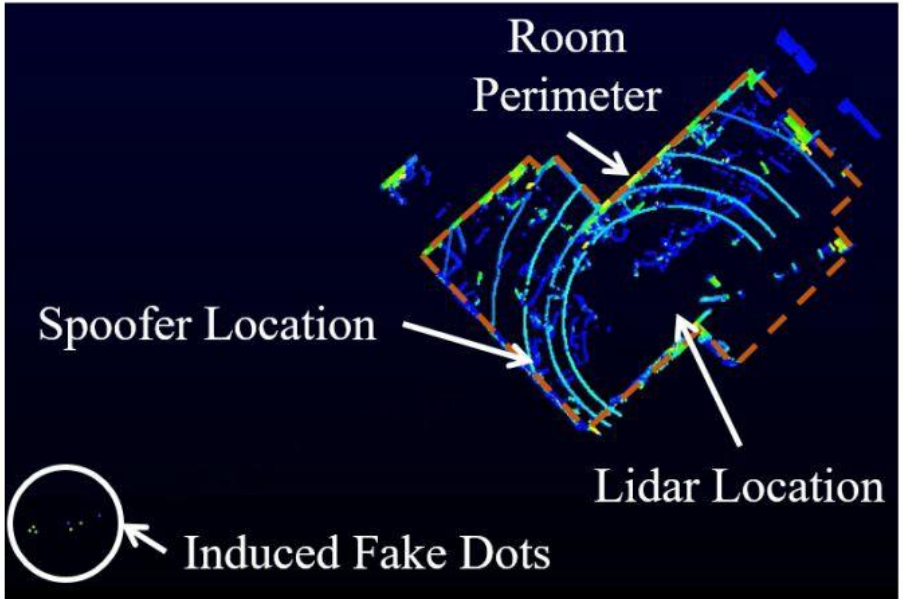
Spoofing Characteristics

- **Receiving angle:** a small receiving angle limits the maximum number of fake dots inducible by a fixed spoofer → use multiple transmitters
- **Curved Reception Glass:** the oblique incidence of a strong laser pulse to readily induce fake dots in sectors, other than the direction of the attacker

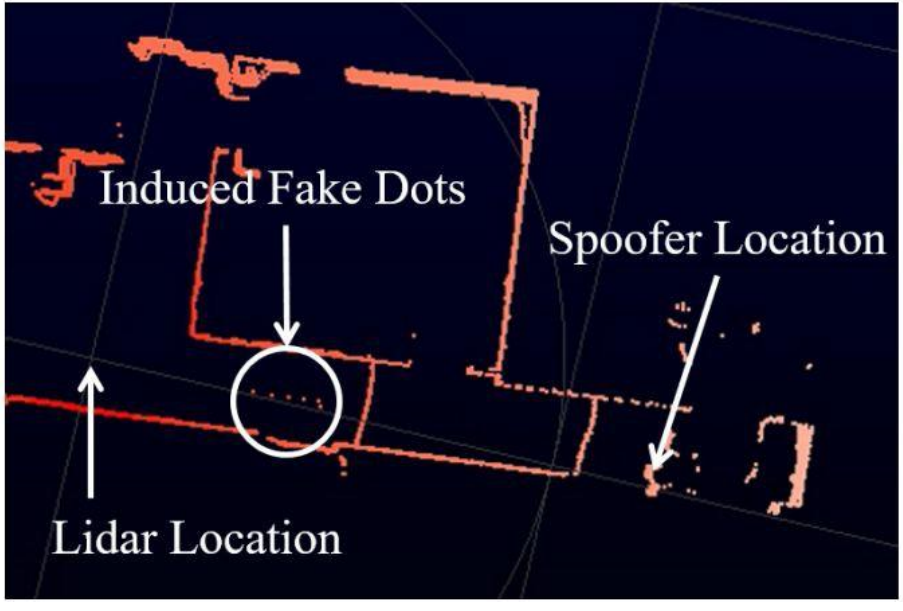
Spoofing Experiments

- The attacking tool first receives LiDAR's generated pulses
- When the incoming pulses are captured by the photo detector, a comparator converts them into a series of 5V pulses
- Then, the pulses are fed to a function generator triggered by the first received pulses
- The function generator waits for a predefined delay and transmits a predefined number of copies of the output pulse to the PLD driver
- Finally, the PLD driver lets the PLD fire laser pulses as signaled

Spoofing Results



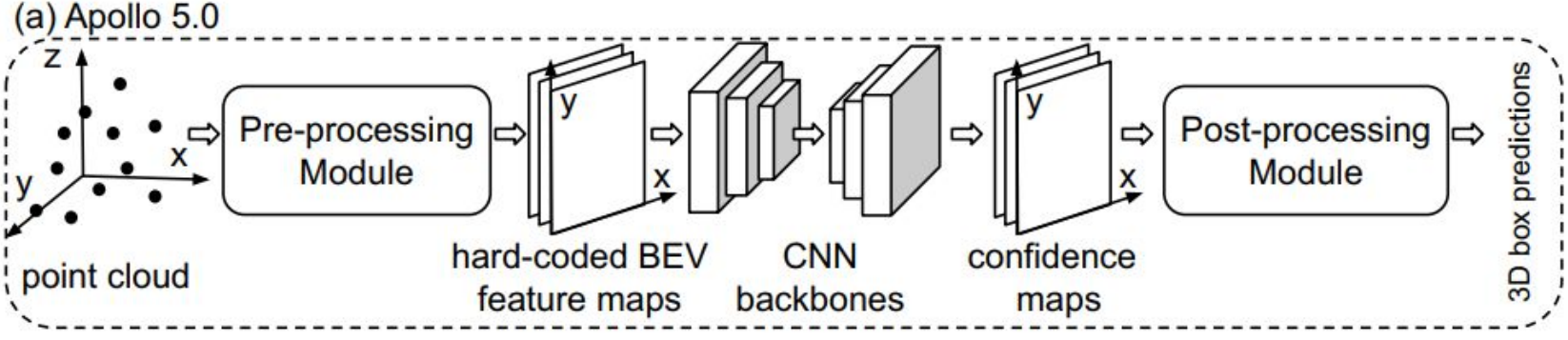
Results of multiple induced fake dots



Induced dots closer than spoofer

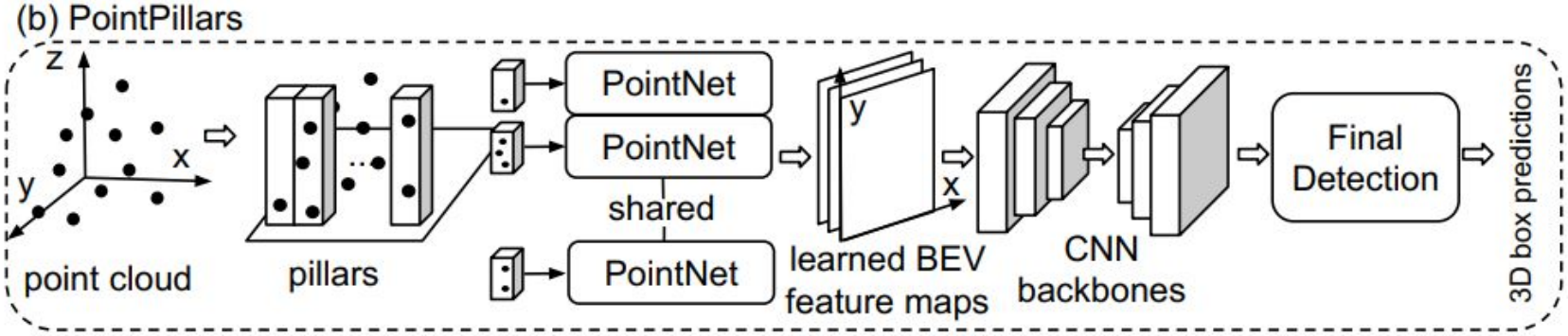
LiDAR 3D Object Detection

- LiDAR is better if estimated 3D objects
- Achieved via 3D object detection (deep learning) models, which output 3D bounding boxes
- We can group these models into three categories:
- **Byrd's-eye view (BEV)-based 3D object Detection:** project point clouds into the top-down view and use CNN to perform the final decision



LiDAR 3D Object Detection

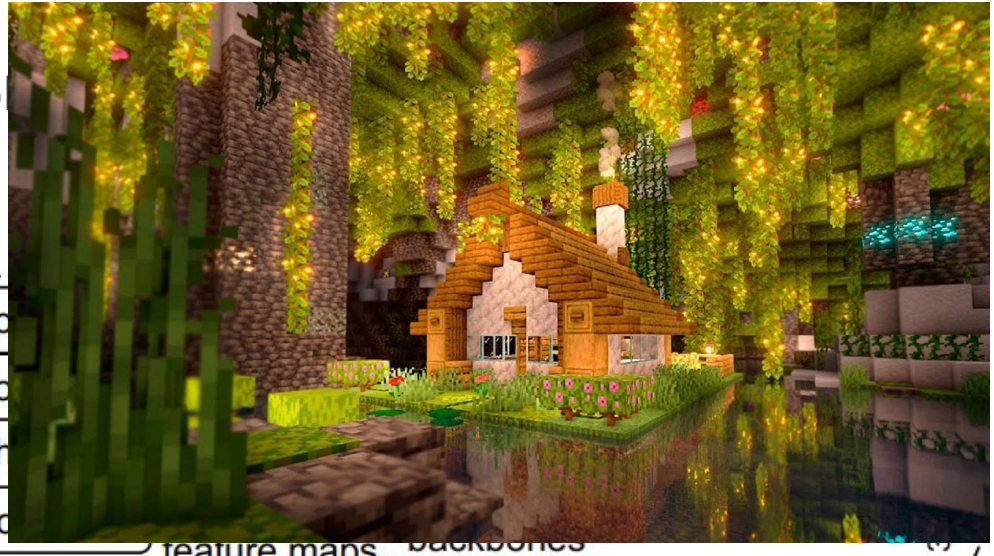
- **Voxel-based 3D Object detection:** VoxelNet slices the point clouds into voxels and extract learnable features by applying a PointNet to each voxel
- A 2D convolution layer is applied in the final stage



LiDAR 3D Object Detection

- **Voxel-based 3D Object detection:** VoxelNet slices the point clouds into voxels and extract learnable features by applying a PointNet to each

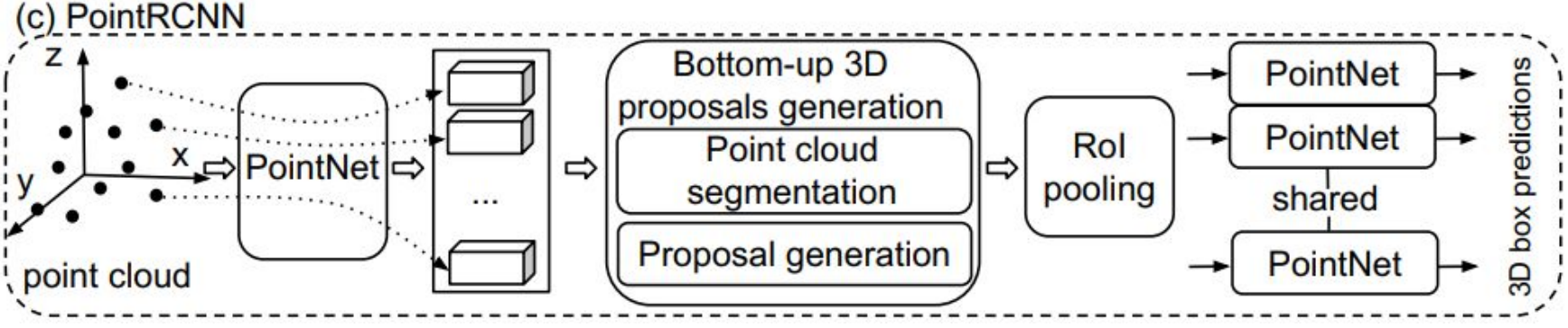
voxel



- A 2D convolution layer is appl

LiDAR 3D Object Detection

- **Point-Wise 3D Object detection:** directly operate on point clouds for 3D object detection
- Two stage architecture: i) generate high-quality region proposals in the 3D space, ii) regress bounding box parameters and classify detected objects



Adversarial Attacks

- Besides spoofing fake points at the sensor levels, attacks to LiDAR include attacks towards its post-processing pipeline
- Let's consider the pristine point cloud X, and the Apollo pipeline (BeV) with hardcoded features x
- Given the function phi that preprocesses feature maps and given T', t' as spoofed point coude and corresponding features maps, the attack can be modeled as

model

$$\min \quad \mathcal{L}(x \oplus t'; \mathcal{M})$$

Sensor attack capabilities

$$\text{s.t.} \quad t' \in \{\Phi(T') \mid T' \in \mathcal{A}\} \;\&\; x = \Phi(X)$$

Adversarial Attacks

- The first adversarial attack to LiDAR has been proposed by Cao et al.

  Cao, Yulong, et al. "Adversarial sensor attack on lidar-based perception in autonomous driving." Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019.

  - They demonstrated a successful attack with 60 cloud points and 8 degree horizontal angle
  - However, a valid e.g., front vehicle contains around 2000 points and occupies about 15 degrees horizontal angle

Adversarial Attacks

- Two situations where a vehicle contains a smaller number of points
- **Occluded Vehicle:** occlusion between objects make the occluded one being partially visible in points clouds (only closer solid objects are reflected)
- **Distant vehicle:** intensity of points decreases with increasing distance from the LiDAR
- In the end, LiDAR works as human eyes

On Occlusions

- Attackers can hence spoof a vehicle by imitating occlusions and sparsity patterns
- We can abstract the neglected physical features as two occlusion patterns inside the LiDAR point clouds
- **Inter occlusion:** the occluder cause the occludee to be partially visible
- Related false positive condition: if an occluded vehicle can be detected in the pristin point cloud by the model, its point set will still be detected as a vehicle when directly moved to a front-near vehicle

On Occlusions

- **Intra occlusion:** the facing surface of a solid object occludes itself in the point cloud which indicates that the LiDAR cannot perceive the interior of the object
- Related false positive condition: if a distant vehicle can be detected in the pristine point cloud by the model, its point set will still be detected as a vehicle when directly moved to a front near location

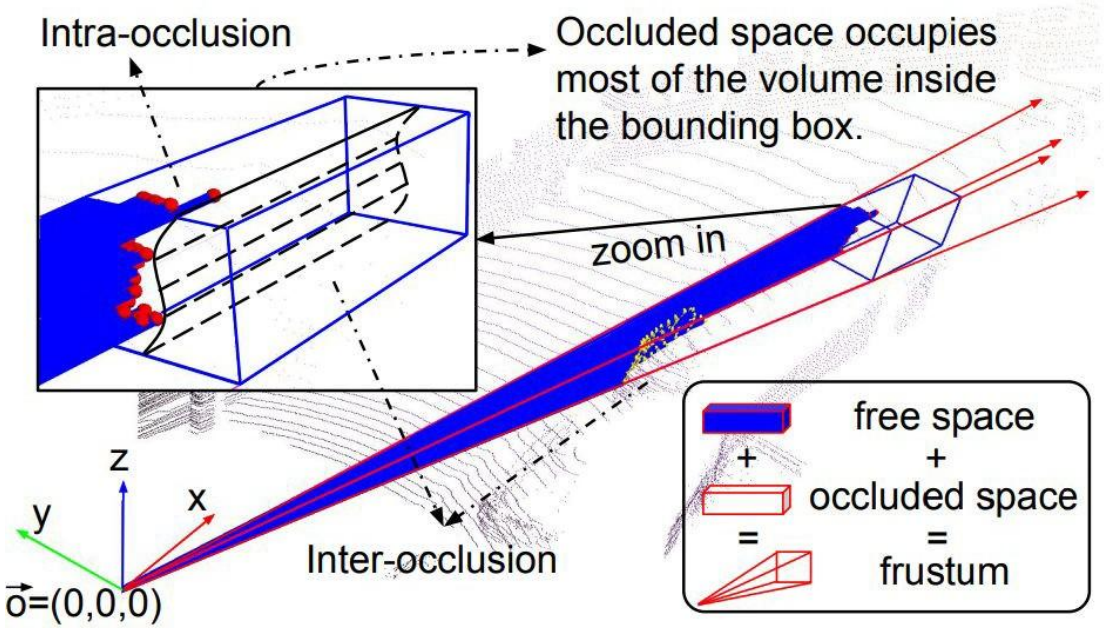
CARLO

- What about countermeasures?
- We may base our anomaly detection on the intuition that some attacks violate laws of physics

- CARLO: oCclusion-Aware hieRarchy anomaLy detectiOn
- Harnesses occlusion patterns as invariant physical features to accurately detect spoofed fake vehicles
- Consists of two building blocks: i) free space detection, and ii) laser penetration detection

Free Space Detection

- Integrate inter and intra occlusion to detect spoofed values
- Recalling that each laser in LiDAR sensor is responsible for perceiving a direction on the spherical coordinates, its resolution limits make it belonging to a thin frustum in the 3D space



Free Space Detection

- The frustum as well as the straight-line path from the LiDAR sensor to any point in the point cloud is considered as free space
- The entire 3D space can be divided into *free space* and *occluded space*
- The former is embedded at the point level, the latter at the object level
- Free space includes information from occluded space
- We hence leverage this information to detect fake vehicles

Free Space Detection

- Due to inter-occlusion and intra-occlusion, we observe that the ratio f of the volume of FS over the volume of a detected bounding box should be subject to some distribution and upper-bounded by b
- This means that f is in (0, b]
- Since fake vehicles do not obey the occlusion pattern, their ratio should be large enough and bounded such that f is in [a,1)
- As long as a>b, we have opportunities to distinguish valid vehicles with the spoofed fake vehicles statistically

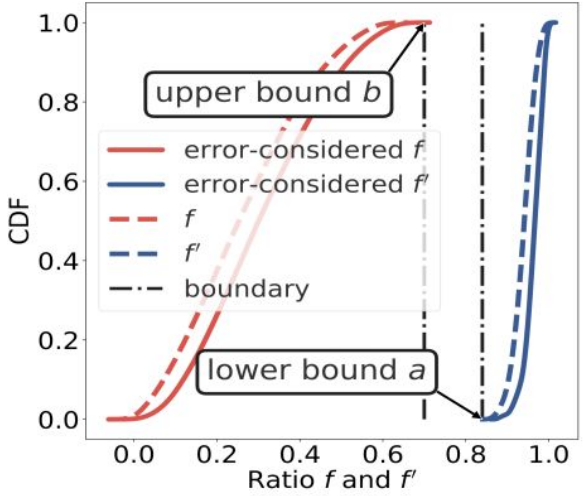
Cell free or not

$$f_B = \frac{\sum_{c \in B} \mathbb{1} \cdot FS(c)}{|B|}$$

Number of cells

Free Space Detection

- Using the KITTI dataset, we can estimate the distribution of valid and fake vehicles
- We use the valid traces and the attack traces generated by the adversarial framework
- Though FSD is statistically significant, it is too time consuming to perform ray casting on all the detected bounding boxes in real time

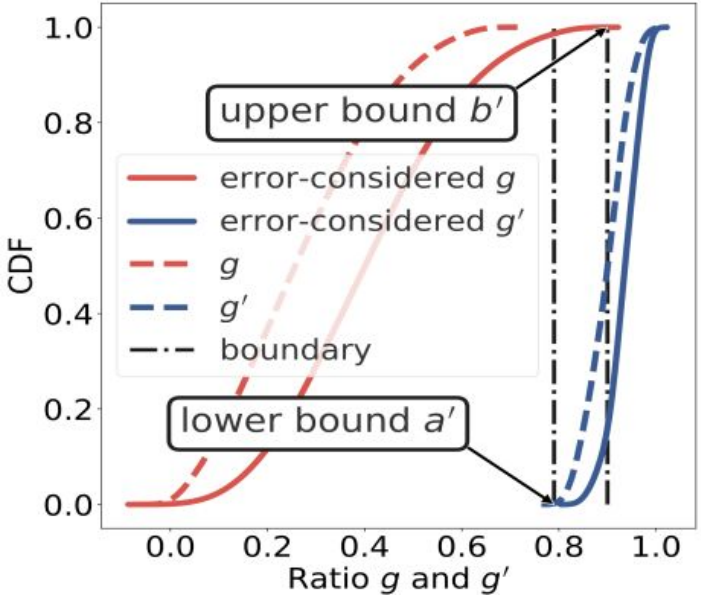


Laser Penetration Detection

- Each point in the point cloud represents one laser ray and the boundary between free space and occluded space
- Given a vehicle's point set, its bounding box B also divides the corresponding frustum into three spaces
  - the space between the LiDAR sensor and the bounding box
  - the space inside the bounding box
  - the space behind the bounding box
- From the perspective of the LiDAR sensor, the ratio g of the number of points located in the space behind the bounding box over the total number of points in the whole frustum should be upper bounded

Laser Penetration Detection

- For the same reason, the ratio g' of the spoofed vehicles is supposed to be large enough and lower bounded in a' in (0,1)



$$g_{\mathrm{B}} = \frac{\sum_{\vec{p} \in \mathrm{B}\downarrow} \mathbb{1}(\vec{p})}{\sum_{\vec{p} \in \mathrm{B} \cup \mathrm{B}\downarrow \cup \mathrm{B}\uparrow} \mathbb{1}(\vec{p})}$$

Inside bounding box

Behind bounding box

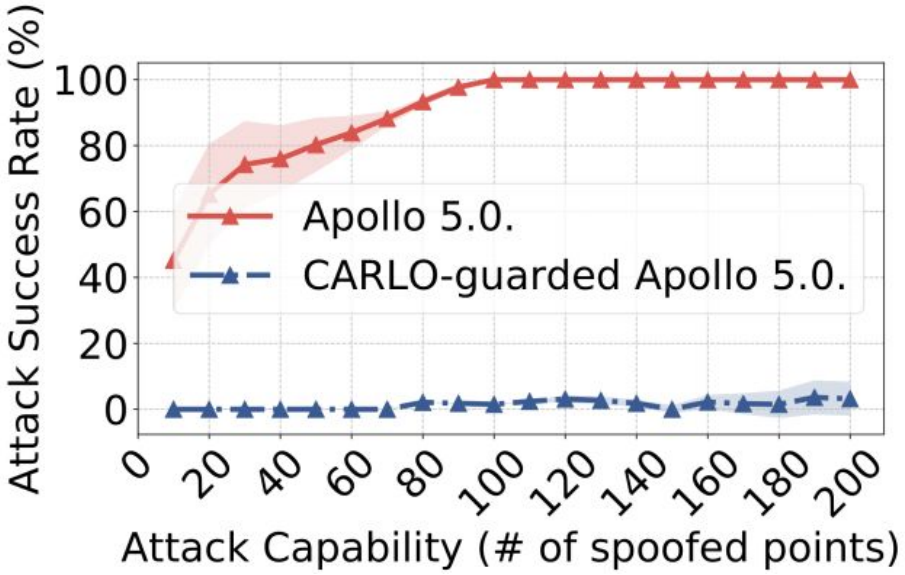
Between sensor and bounding box

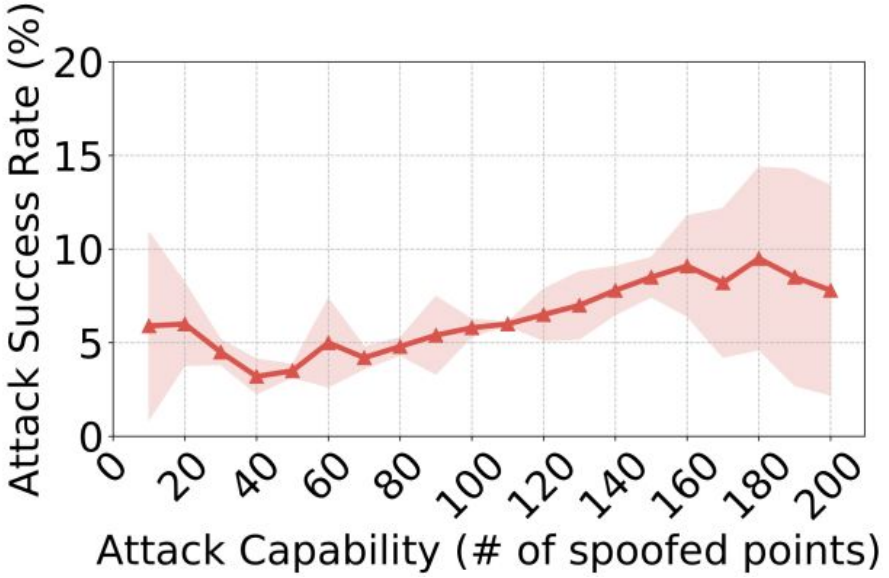
- Results on the KITTI dataset, showing separate but overlapping distributions

Hierarchy Design

- CARLO hierarchically integrates FSD and LPD
- In the first stage, CARLO accepts the detected bounding boxes and leverages LPD to filter the unquestionably fake and valid vehicles by two thresholds
- The remaining bounding boxes are uncertain and will be further fed into FSD for final checking

CARLO Peroformance

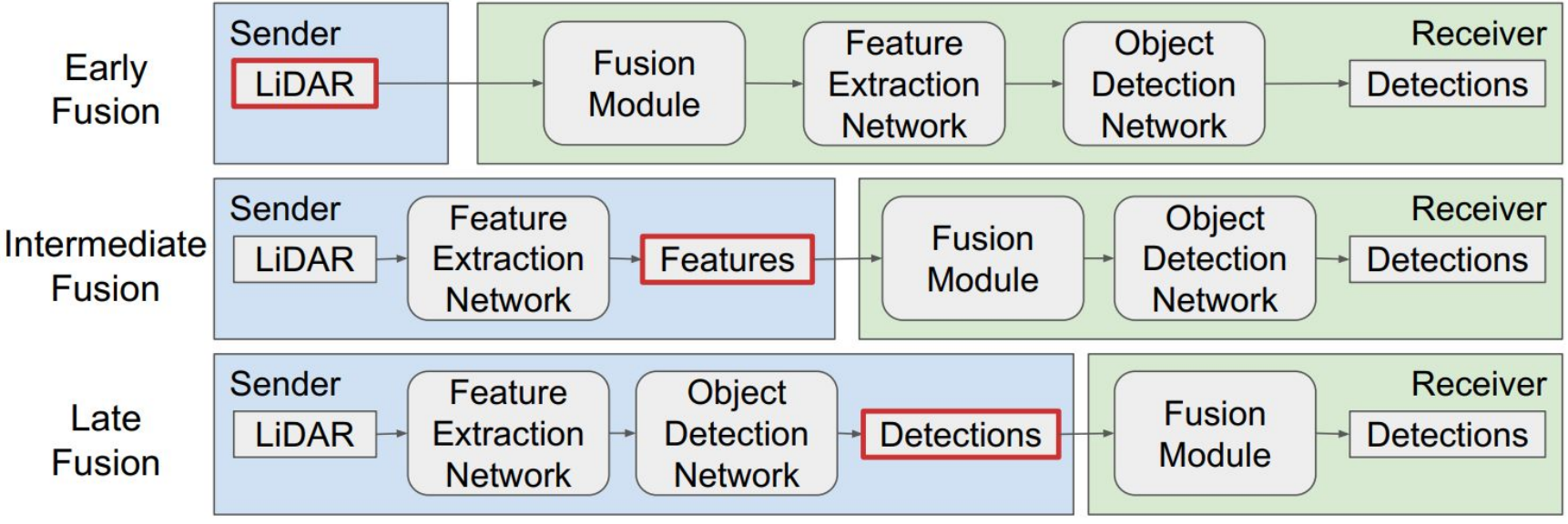




Collaborative Perception

- Collaborative perception has been proposed to enhance Connected Autonomous Vehicles' (CAV) perception by sharing raw or processed sensor data among infrastructures or vehicles
- **Early fusion sharing:** directly exchange raw sensor data whose format is usually universal and can be naively concatenated
- **Intermediate fusion:** transmit feature maps (intermediate product) offering trade-off between network efficiency and perception accuracy
- **Late fusion:** share object bounding boxes

Collaborative Perception



Defenses and Data Fabrication

- Several security standards define how vehicle-to-everything (V2X) communications should be implemented to avoid attacks (e.g., authentication, access control,..)
- However, they cannot defend against **data fabrication attacks**, as attackers can modify the data before wrapping it into valid messages
- Furthermore, attackers can collude to create realistic but malicious data, so physics enabled rules may not solve the problem

Data Fabrication Formulation

- We consider an attacker that aims at spoofing or removing an object in a collaborative perception CAV setting
- We can formulate the attack as an optimization problem
- We denote the LiDAR data at frame i from the attacker, victim, and other benign vehicles as A_i, V_i, j in {0,1,...,N}
- LiDAR data with the same frame index will be merged at the receiver to generate the result
- We denote pre-process data sharing as f and post-process data sharing as g

Data Fabrication Formulation

- A normal collaborative perception for the victim on frame i can be described as $y_i = g(f(V_i), f(A_i), f(X_i^0), f(X_i^1), \ldots, f(X_i^N))$
- The attacker replace $f(A_i)$ with malicious data, appending a minor perturbation to it $y_i' = g(f(V_i), f(A_i) + \delta_i, f(X_i^0), f(X_i^1), \ldots, f(X_i^N))$
- The attack can be hence formulate according to a fitness function I and a constraint set C as

$$\max_{\delta_i} I(y_i') \quad \text{s.t. } C(\delta_i)$$

Threat Model

SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- We assume that the attacker can physically control at least one vehicle participating in collaborative perception
- The attacker can hence directly manipulate the data to share
- We focus on early- and intermediate-fusion collaboration schemes where attacker need to subtly craft complicated structured data
- We assume the presence of benign vehicles that the attacker cannot control

Attack Constraints

- **Sensor physics and definition ranges:** the attacker needs to obey basic rules in terms of data format, otherwise it is trivial to detect anomalies
- For instance, point clouds should have a reasonable distribution, with reasonable occlusion effects, the angle of laser should comply with LiDAR configuration
- **Targeted Attacks:** the attacker should be able to designate a target region for either spoofing or removal attacks

Attack Constraints

- **Real-time temporal constraints:** collaborative perception is an asynchronous multi-agent system where each vehicle produces LiDAR images in cycles but is not synchronized in time
- Hence the attacker, to attack the victim's perception at time i should respect some constraints
- i) the optimization of the perturbation shall be finished before the victim's processed LiDAR data V_i is generated
- This means that the attacker cannot leverage the victim's data on the same frame

Attack Constraints

- ii) the optimization of the perturbation takes time, especially when the attack involves online adversarial machine learning
- To ensure that the perturbation is created and sent before its use, the attacker can either design fast real-time attacks or optimize the perturbation before frame i arrives

Zero-Delay Attack Scheduling

SPRITZ
Security & Privacy
Research Group

Università
degli Studi
di Padova

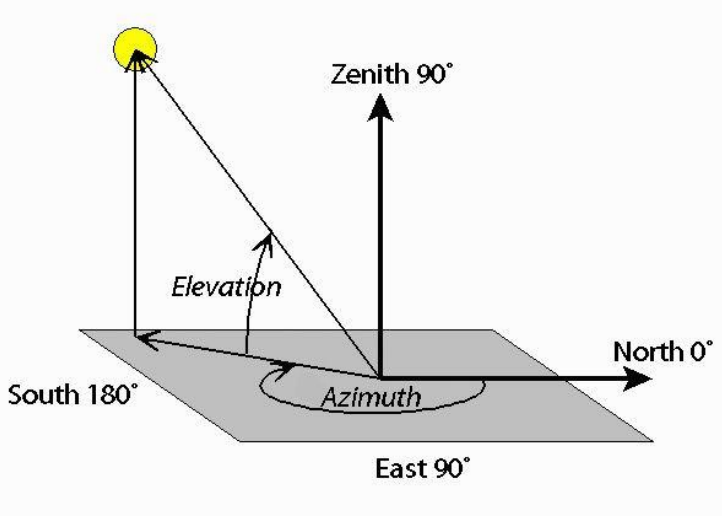
- The key idea is to parallelize attack generation and perception processes
- The attacker can identify the set of vehicles that collaborate with the victim and align frame indices of their shared sensor data based on timestamps
- For each LiDAR cycle: i) for object spoofing the trajectory of the spoofed object is defined, ii) for removal use an object detection algorithm to localize the target

Zero-Delay Attack Scheduling

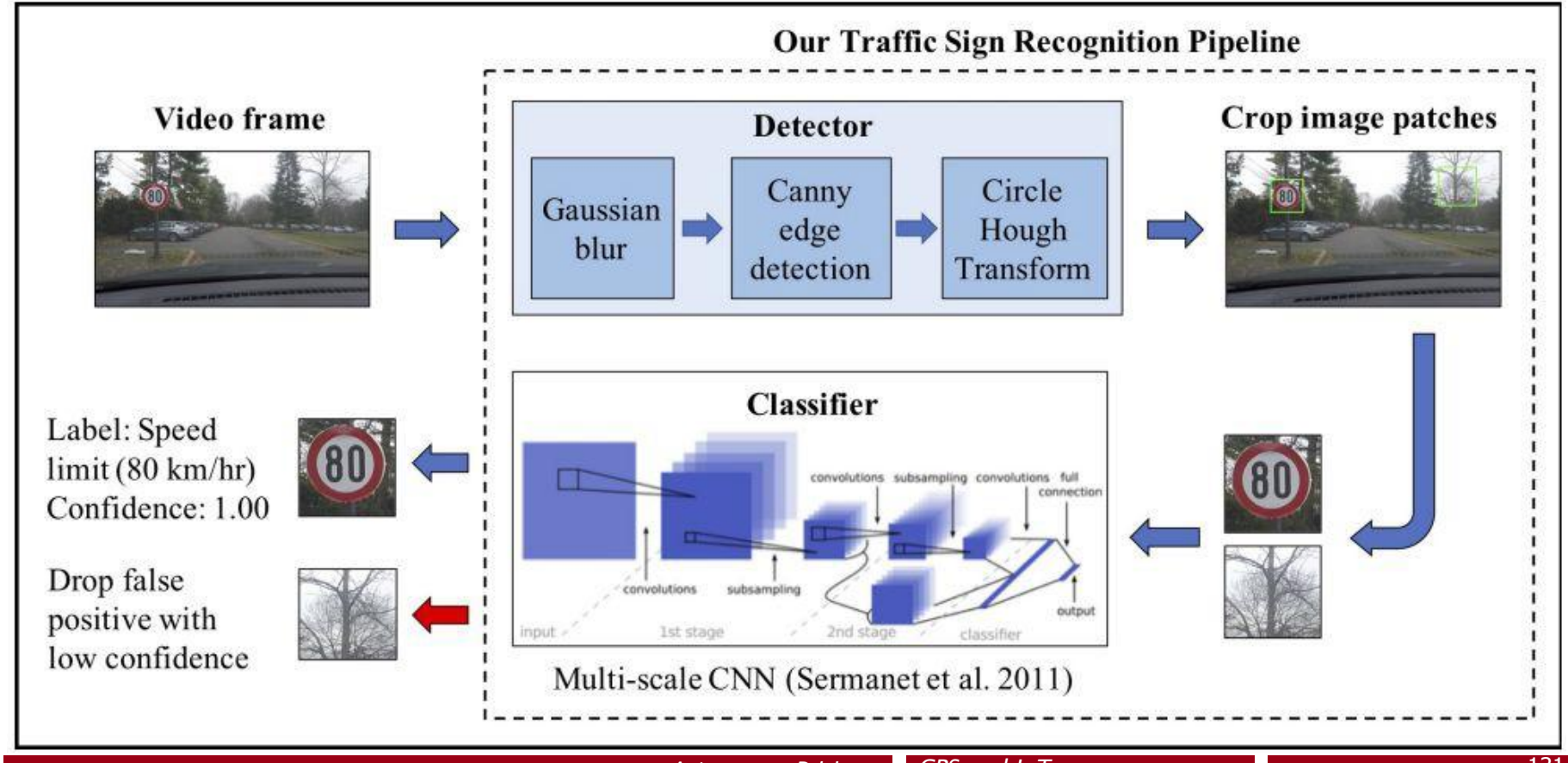
- Then, optimize the malicious perturbation that can be used to attack the victim's perception at the current frame
- Need to transform the perturbation into one that has a similar attack impact on the next frame → prediction
- Thus, the attacker has a single LiDAR cycle time to complete the optimization

The Use of Cameras

- LiDAR and radars are generally more expensive than cameras, therefore it is preferable to use cameras to save money
- Autonomous cars are generally equipped with multiple cameras spaced around the vehicle
- Each camera provides monocular vision and resolve azimuth and elevation angles



Example: Sign Recognition



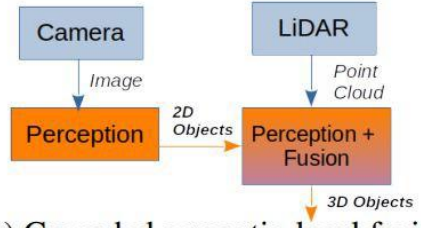
Sensor Fusion

- To improve the performance of the sensing capabilities of cars, we can use sensor fusion techniques
- Sensor fusion refers to the process of merging information coming from different sensors related to the same scenario to reduce uncertainty
- Sensors can be either of the same type or based on different technologies (i.e., camera and LiDAR)

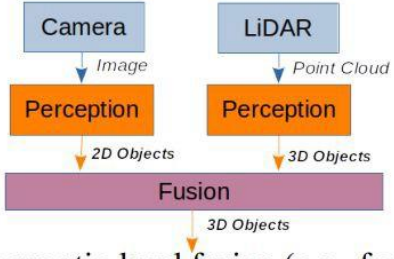
Example

- An example of calculation via sensor fusion is *inverse variance weighting*
- Let us consider two measurements $\mathbf{x}_1$ and $\mathbf{x}_2$ with variance σ_1^2 and σ_2^2 respectively
- Then, the fused measurement is given by $\mathbf{x}_3 = \sigma_3^2(\sigma_1^{-2}\mathbf{x}_1 + \sigma_2^{-2}\mathbf{x}_2)$
- Where the variance of the combined estimate is given by $\sigma_3^2=(\sigma_1^{-2}+\sigma_2^{-2})^{-1}$

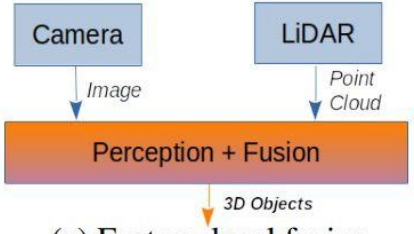
Classes of Fusion in Perception



(a) Cascaded semantic-level fusion



(b) Integrated semantic-level fusion (e.g., fusion at tracking)



(c) Feature-level fusion

- a) using the output of perception on one or more sensors to augment the input of other single-sensor perception
- b) runs isolated perception for each sensor and fuses semantic outputs
- c) combines low-level (machine-learned) features from multiple perception sources to produce a unified output