Verified Cruise Control on RC Vehicle

Shashank Ojha and Yufei Wang
Objective

- Implement a verified model (static POV system) on real hardware
- Fill the gap between theory & practice
Motivation

- Cruise Control system is useful in practice:
  - A stepping-stone towards self-driving cars
  - Long straight highway trucking
Summary of Deliverables

- Formal model and proof of system in KeYmaera X
- Implementation of model on an RC vehicle
- Video and Live Demos
Formal Model and Proof
Assumptions

- One-dimensional road
- Static Obstacle
- Constant accelerate with rates $acc = \{A, 0, -B\}$
- LIDAR sensor measures the obstacle distance
- ODOM sensor measures the car’s velocity
- Asynchronous read from the sensors & control
Formal Model

**Estimate Obstacle Distance**

\[
\text{if } (\text{in}_\text{sensor}_\text{range}(\text{obstacle})):
\]

\[\text{sensed}_\text{dist} = \text{LIDAR}_\text{reading}\]

\[
\text{else}:
\]

\[\text{sensed}_\text{dist} = \text{sensor}_\text{range}\]

**Estimate Vehicle Velocity**

\[\text{sensed}_\text{vel} = \text{ODOM}_\text{reading}\]

**Control Decision**

\[
\text{if } (\text{safe}(A)):
\]

\[\text{acc} = A\]

\[
\text{elif } (\text{safe}(0.0)):
\]

\[\text{acc} = 0.0\]

\[
\text{else}:
\]

\[\text{acc} = -B\]

**Dynamics**

\[
\{ \text{obstacle}_\text{dist}' = -v, \ v' = \text{acc}, \ t' = 1 \ \& \ v \geq 0 \ \& \ t \leq \text{CTRL}_T \ \}\]
Control must make a decision based on stale data about the velocity and obstacle distance.

\[
\text{ub}_v = \text{sensed}_v + A \times \text{ODOM}_\text{interval}
\]

\[
\text{lb}_\text{obstacle distance} = \\
\text{sensed distance} - (\text{ub}_v \times \text{LIDAR}_\text{interval} + 0.5 \times A \times \text{LIDAR}_\text{interval}^2)
\]
Safety Condition

Safety condition is based on \textit{sensed parameters}, \textbf{NOT} the true values

\begin{equation*}
\text{def safe}(a) : \ lb\_obstacle\_distance \geq \ ub\_v \times CTRL\_T + 0.5 \times a \times \\
\quad CTRL\_T^2 + (ub\_v + a \times CTRL\_T) / (2 \times B) + (BUFFER\_DIST) \\
\end{equation*}

\begin{align*}
ub\_v &= \text{sensed\_vel} + A \times \text{ODOM\_interval} \\
\text{lb\_obstacle\_distance} &= \text{sensed\_distance} - (ub\_v \times \text{LIDAR\_interval} + 0.5 \times A \times \\
&\quad \text{LIDAR\_interval}^2)
\end{align*}

We must have distance left over in order to accelerate safely
Implementation
RC Vehicle

Hardware:
- **LIDAR sensor**: 5.6m range, 10Hz
- **ODOM sensor**: 30Hz
- Max velocity: 6m/s

Software:
- ROS
- Subscribe to get sensor data
- Publish to command velocity
Implementation Challenges

- Cannot command acceleration directly
  - Approximate acceleration control by velocity control

\[ V_{\text{command}} = V_{\text{old\_commanded}} + A \times \varepsilon \]
Implementation Challenges

- Very noisy ODOM sensor: imprecise $V_{odom}$
  - Maintain an analytic velocity $V_{command}$
    - Use $\max(V_{command}, V_{odom})$ to upper bound the real velocity
- Steering linkage was also damaged
  - Manually adjust for bias with software
Live Demo
Challenges Ahead

- Move from static obstacle to dynamic obstacle model (need another car)
  - Need to approximate POV’s velocity
- Update hardware: ODOM sensor, direct acceleration control
- Incorporate feedback from sensors to lower the disparity between commanded controls and actual dynamics
- Model other dynamics such as drag and friction forces
Huge Thanks to

- André Platzer
- Katherine Cordwell
- Aman Khurana