Bridging the Gap Between Safety and Real-Time Performance for Autonomous Vehicle Control

Ram Vasudevan

Mechanical Engineering
Robotics Institute
University of Michigan
Improving Vehicle Safety

NHTSA, 2014
Improving Vehicle Safety

- Traffic Lights
- Modern Speed Limit (MUTCD)
- Seat Belts
- Air Bags
- ABS
- Electronic Stability Control

Fatalities per 100 Million Vehicle Miles Traveled

Years

NHTSA, 2014
How Much AV Driving Has Happened?

How far can we get in verified planning/control of autonomous systems without resorting to simulation?
Planning and Control Hierarchy

- High-level Planner
- Trajectory Planner
- Tracking Controller
Why Apply A Planning Hierarchy?
Planning and Control Hierarchy – How to Do It?

- High-level Planner
- Trajectory Planner
- Tracking Controller
Path Generation + MPC
When Planning Goes Wrong
Bringing in Safety

- High-level Planner
- Trajectory Planner
- Tracking Controller
Overview of Validation + Path Planning Methods

Check - perform collision checking for precomputed trajectories
Zonotope Methods (Althoff, TRO 2014)
Verification via Check Methods - Zonotopes

precomputed trajectory

tracking error

state space
Overview of Validation + Path Planning Methods

**Check** - perform collision checking for precomputed trajectories
- Zonotope Methods (Althoff, TRO 2014)

**Correct** – modify generated control inputs to ensure safety
- Control Barrier Functions (Ames et al., ECC 2019)
- FaSTrack (Herbert, CDC 2017)
Verification via Correct Methods - CBFs

control inputs

state space

$U$

$X$
Verification via Correct Methods - CBFs
Verification via Correct Methods - FaSTrack

tracking error

state space
Verification via Correct Methods - FaSTrack

- Tracking error
- State space

$E$
Bringing in Safety

High-level Planner

Trajectory Planner

Tracking Controller
Overview of Validation + Path Planning Methods

**Check** - perform collision checking for precomputed trajectories
   Zonotope Methods (Althoff, TRO 2014)

**Correct** – modify generated control inputs to ensure safety
   Control Barrier Functions (Ames et al., ECC 2019)
   FaSTrack (Herbert, CDC 2017)

**Select** – select a not-at-fault trajectory at run-time
   Funnel Library (Majumdar, IJRR 2017)
   Reachability-based Trajectory Design (IJRR, in press)
Verification via Select Methods - Funnel Library

- Parameter space $K$ and state space $X$
- Reference trajectories connecting the parameter space to the state space
Verification via Select Methods - Funnel Library

$K$

trajectories parameters

state space

tracking error
Verification via Select Methods - Funnel Library

reachable set

$K$

traj. parameters

state space

$X$
Verification via Select Methods - Funnel Library

$K$  

traj. parameters  

state space  

$X$
Collision check is not verified to operate safely

Have to pick finite library a priori
Reachability-based Trajectory Design

Reachability with traj-dependent error (offline)

Obstacle discretization for real-time validation (online)
Outline

1. Reachability-based Trajectory Design for Provably Safe, Real-Time Planning for Autonomous Vehicles – STATIC

2. Reachability-based Trajectory Design for Provably Not-at-Fault, Real-Time Planning for Autonomous Vehicles – DYNAMIC

3. Conclusion
Outline

1. Reachability-based Trajectory Design for Provably Safe, Real-Time Planning for Autonomous Vehicles – STATIC

2. Reachability-based Trajectory Design for Provably Not-at-Fault, Real-Time Planning for Autonomous Vehicles – DYNAMIC

3. Conclusion
An Example of Trajectory Parameterization

\[ \dot{x}(t) = f(t, x(t), k^*) \]

Assumption

Each trajectory is defined over a finite time interval, \( T = [0, t_f] \), and ends with a fail-safe maneuver.
An Example of Trajectory Parameterization
Bounding Tracking Error

Assumption (DSCC 2019, Smith et al. CDC 2019)

For each $j \in \{1, 2\}$, there exists $g_j$ such that

$$|x_{hi,j}(t; k) - x_j(t; k)| \leq$$

Vehicle Trajectory

Reference Trajectory
The Forward Reachable Set (FRS)

\[ \dot{x}(t) = f(t, x(t), k) + g(t, k)d(t) \]

\[ d(t) \in [-1, 1] \]
SOS Reachability (offline)

\[
\inf_{w,v,q} \int_{X \times K} w(x, k) d\lambda_{X \times K}
\]

\( w : X \times K \rightarrow \mathbb{R} \) converges to indicator function on FRS
SOS Reachability (offline)

\[
\inf_{w,v,q} \int_{X \times K} w(x, k) d\lambda_{X \times K}
\]

s.t. \[
- \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} f \right) \geq 0 \\
- v(0, x, k) \geq 0
\]

\( v : T \times X \times K \rightarrow \mathbb{R} \) is like a Lyapunov function

IJRR, in press
SOS Reachability (offline)

\[
\inf_{w,v,q} \int_{X \times K} w(x, k) d\lambda_{X \times K}
\]

s.t. \[
- \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} f \right) - q(t, x, k) \geq 0
\]
\[
- v(0, x, k) \geq 0
\]
\[
q(t, x, k) \pm \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} g \right) \geq 0
\]
\[
q(t, x, k) \geq 0
\]

\( q : T \times X \times K \rightarrow \mathbb{R} \)
incorporates uncertainty
SOS Reachability (offline)

\[
\inf_{w,v,q} \int_{X \times K} w(x, k) d\lambda_{X \times K}
\]

s.t. \[- \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} f \right) - q(t, x, k) \geq 0 \]

\[- v(0, x, k) \geq 0 \]

\[ q(t, x, k) \pm \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} g \right) \geq 0 \]

\[ q(t, x, k) \geq 0 \]

\[ w(x, k) + v(t, x, k) - 1 \geq 0 \]

\[ w(x, k) \geq 0 \]

Theorem

\[(x, k) \in \text{FRS} \implies w(x, k) \geq 1 \]
Projection from $K$ to $X$

\[ w(\cdot, k^*) \geq 1 \]
Projection from $K$ to $X$

$w(\cdot, k^*) \geq 1$
Projection from $K$ to $X$
Goal for Online Computation

\[
\min_{k \in K} J(k)
\]

s.t. \( w(x, k) < 1, \ \forall \ x \in \text{obstacle} \)
Goal for Online Computation

- Optimize an arbitrary cost function
- Obstacles produce semidefinite constraints

<table>
<thead>
<tr>
<th>Obstacle Shape</th>
<th>Method</th>
<th>Mean Time [ms]</th>
<th>Std. Dev [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box</td>
<td>Set Intersection</td>
<td>17,800</td>
<td>1010</td>
</tr>
<tr>
<td>Line</td>
<td>Set Intersection</td>
<td>1050</td>
<td>73</td>
</tr>
</tbody>
</table>
Obstacle Representations
Suppose the vehicle is convex and the obstacles are buffered by a distance $b$, then there exists a point spacing $r > 0$, which can be pre-computed offline, that generates an outerapproximation to the set of safe trajectory parameters.
Theorem

If $w(\bullet, k^*) < 1$, then AV avoids obstacle.

Point Spacing $\leq 2b$

Buffer $b \in (0, \frac{1}{2}W)$

Provably Safe, Obstacle Representations for AVs
RTD Results – Rover

robot’s planning view

waypoint (black)
obstacles (red)
robot (blue)
reachable set (green)

mock road +
random obstacles

camera on robot

IJRR, in press

https://youtu.be/bgDEAi_Ewfw
RTD Results – Rover

Comparison to Other Planners – Real-Time

Evaluate real-time ability of planners to get to a goal and avoid randomly generated obstacles over 10k scenarios

<table>
<thead>
<tr>
<th>$\tau_{\text{plan}}$ [s]</th>
<th>$D_{\text{sense}}$ [m]</th>
<th>Planner</th>
<th>Goals [%]</th>
<th>Crashes [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>4.0</td>
<td>RTD</td>
<td>87.4</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RRT</td>
<td>77.2</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMPC</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Comparison of FaSTrack to RTD
Comparison of FaSTrack to RTD
CarSim Full Powertrain Experiment

https://youtu.be/lmtki6elFlw?t=20

ACC 2019
Comparison: 10 trials, $\tau_{\text{plan}} = 0.5s$

<table>
<thead>
<tr>
<th>Planner</th>
<th>Avg Planning Time</th>
<th>% of Track Complete</th>
<th>Crashes</th>
<th>Safe Stops</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>RRT [1]</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NMPC [2]</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTD (ours)</td>
<td>0.09</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Outline

1. Reachability-based Trajectory Design for Provably Safe, Real-Time Planning for Autonomous Vehicles – STATIC

2. Reachability-based Trajectory Design for Provably Not-at-Fault, Real-Time Planning for Autonomous Vehicles – DYNAMIC

3. Conclusion
Moving Obstacles
Moving Obstacles
Moving Obstacles

Assumption

The predictions are conservative and contain the true behavior of the obstacles.
Moving Obstacles
Moving Obstacles

Theorem

Assumption

Definition

The AV is not at fault at time $t$ if it is stationary or not intersecting any obstacles.
Real-Time, Provably Not-at-Fault Planning
SOS Reachability (offline)

\[
\inf_{w,v,q} \int_{X \times K} w(x, k) d\lambda_{X \times K}
\]

s.t. \[-\left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} f \right) - q(t, x, k) \geq 0 \]

\[- v(0, x, k) \geq 0 \]

\[q(t, x, k) \pm \left( \frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} g \right) \geq 0 \]

\[q(t, x, k) \geq 0 \]

\[w(x, k) + v(t, x, k) - 1 \geq 0 \]

\[w(x, k) \geq 0 \]

\[ (t, x, k) \in \text{FRS} \implies v(t, x, k) \leq 0 \]
Real-Time, Provably Not-at-Fault Planning

FRS at $t = t_1$
Real-Time, Provably Not-at-Fault Planning
Real-Time, Provably Not-at-Fault Planning

Suppose the vehicle is convex, the maximum relative speed between any obstacle and the vehicle is bounded, then there exists a time discretization, which can be pre-computed offline, that generates an outerapproximation to the set of safe trajectory parameters.
Real-World, Car-Like Experiment

Comparison: EV Unprotected Left Turns

- 100 trials
- up to 4 cars, 2 pedestrians
- at-fault if stopped in intersection

<table>
<thead>
<tr>
<th>Planner</th>
<th>At-fault Collisions (%)</th>
<th>Goals (%)</th>
<th>Average Time to Goal (s)</th>
<th>Average Speed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear MPC</td>
<td>19.0</td>
<td>80.0</td>
<td>14.9</td>
<td>3.35</td>
</tr>
<tr>
<td>RTD</td>
<td>0.00</td>
<td>99.0</td>
<td>20.9</td>
<td>2.71</td>
</tr>
</tbody>
</table>
### More Comparisons

<table>
<thead>
<tr>
<th>Other Method</th>
<th>Robot</th>
<th>Crash %</th>
<th>Goal %</th>
<th>RTD Goal %</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMPC (GPOPS-II)</td>
<td>Segway</td>
<td>0.0</td>
<td>0.0</td>
<td>87.4</td>
</tr>
<tr>
<td>RRT (Kuwata et al., 2009)</td>
<td>Segway</td>
<td>6.3</td>
<td>77.2</td>
<td>87.4</td>
</tr>
<tr>
<td>NMPC (GPOPS-II)</td>
<td>Rover</td>
<td>0.0</td>
<td>0.0</td>
<td>93.1</td>
</tr>
<tr>
<td>RRT (Kuwata et al., 2009)</td>
<td>Rover</td>
<td>2.1</td>
<td>97.4</td>
<td>93.1</td>
</tr>
<tr>
<td>NMPC (GPOPS-II)</td>
<td>Fusion</td>
<td>0.0</td>
<td>0.0</td>
<td>93.1</td>
</tr>
<tr>
<td>RRT (Kuwata et al., 2009)</td>
<td>Fusion</td>
<td>1.0</td>
<td>38.0</td>
<td>100.0</td>
</tr>
<tr>
<td>State Lattice (McN., 2010)</td>
<td>Segway</td>
<td>7.9</td>
<td>92.4</td>
<td>100.0</td>
</tr>
<tr>
<td>State Lattice (McN., 2010)</td>
<td>EV</td>
<td>17.2</td>
<td>77.3</td>
<td>90.7</td>
</tr>
<tr>
<td>Linear MPC</td>
<td>EV</td>
<td>19.0</td>
<td>80.0</td>
<td>99.0</td>
</tr>
<tr>
<td>FasTrack (Herbert, 2017)</td>
<td>TurtleBot</td>
<td>0.0</td>
<td>85.3</td>
<td>90.0</td>
</tr>
<tr>
<td>CHOMP (Zucker, 2013)</td>
<td>Fetch</td>
<td>18.0</td>
<td>82.0</td>
<td>84.0</td>
</tr>
</tbody>
</table>

IJRR, in press

ACC ’19

RSS ’19

ITSC ’19

RSS ’20
Real-World, Ford Fusion
Outline

1. State of the Art in Verified AV Control

2. Reachability-based Trajectory Design for Provably Safe, Real-Time Planning for Autonomous Vehicles – STATIC

3. Reachability-based Trajectory Design for Provably Not-at-Fault, Real-Time Planning for Autonomous Vehicles – DYNAMIC

4. Conclusion
RTD – Quadrotor

3m x 5m test area

https://youtu.be/lcldHVQK3Yw?t=94
RTD – High DOF Manipulator
RTD – Walking Robot
Takeaways and Challenges

- RTD enables collision-free/not-at-fault planning
- Variety of hardware platforms and 10M+ simulations
- How to verify perception and prediction algorithms?
- How to incorporate models of uncertainty?
The Real Automators and Our Sponsors

Patrick Holmes
Shreyas Kousik
Hannah Larson
Jinsun Liu
Zehui Lu
Sean Vaskov
Bohao Zhang
Pengcheng Zhao

github.com/skousik/RTD_tutorial
github.com/ramvasudevan/arm_planning
github.com/skvaskov/RTD
Takeaways

RTD enables collision-free/not-at-fault planning

Variety of hardware platforms and 10M+ simulations

github.com/skousik/RTD_tutorial

github.com/ramvasudevan/arm_planning

github.com/skvaskov/RTD