Motion planning and obstacle avoidance with barrier functions for autonomous car racing

Gary Gao, Enrique Mallada

June 17, 2022

Outline

Introduction and Motivation

A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates

- Authors: Stephen Prajna, Ali Jadbabaie, and George J. Pappas
  - Verify safety by introducing the existence of the barrier certificate

Optimization-Based Autonomous Racing of 1:43 Scale RC Cars

- Authors: Alexander Liniger, Alexander Domahidi and Manfred Morari
  - Motion planning with optimization for autonomous car

Some Applications of Polynomial Optimization in Operations Research and Real-Time Decision Making

- Authors: Amir Ali Ahmadi and Anirudha Majumdar
  - generate barrier certificate with Sum-of-Squares programming

Introduction and Motivation

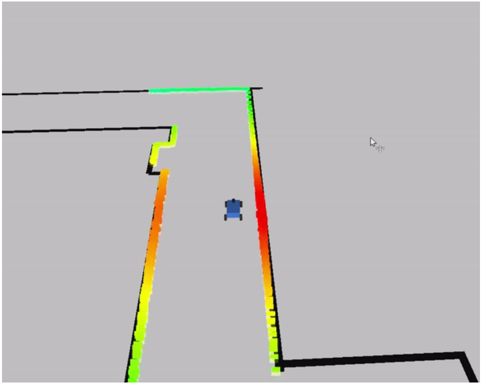


Figure: F1tenth simulation environment developed from UPenn

Introduction and Motivation

Motivation

- Simultaneously guarantee safety and maximize progress along the track through path planning for autonomous car
  - Ensure safety by generating barrier certificate
  - Maximize track progress with optimization

Main challenge

- Computation efficiency for generating barrier certificate and performing motion planning simultaneously
  - About 30Hz for higher level part but less than 10Hz for lower level part without parallel computing

Outline

Preliminaries and Problem Statement

Barrier certificate

- Consider a general differential equation:

$$\dot{x} = f(x) \tag{1}$$

- A barrier certificate $V(x)$ is defined as:

$$V(x) \geq 0 \quad \forall x \in S_0 \tag{2}$$

$$V(x) < 0 \quad \forall x \in S_{unsafe} \tag{3}$$

$$\frac{\partial V(x)}{\partial x} f(x) \geq 0 \quad \forall x \in S \tag{4}$$

- S_0 denotes the set contains all the initial states of x and S_{unsafe} denotes unsafe set of states x. $S_0, S_{unsafe} \subset S$

Preliminaries and Problem Statement

Equivalently, the definition of barrier certificates can be modified as the following,

$$V(x) < 1 \quad \forall x \in S_{safe} \tag{5}$$

$$V(x) > 1 \quad \forall x \in S_{unsafe} \tag{6}$$

$$\frac{\partial V(x)}{\partial x} f(x) \leq 0 \quad \forall x \in S \tag{7}$$

- S_{safe} and S_{unsafe} are two semi-algebraic sets
- Trajectory starts in S_{safe} never ends up in S_{unsafe}

Stability and RoA

- Suppose $f(0) = 0$ and locally asymptotically stable
- RoA($\mathcal{O}$) of the origin is defined as the set of all initial states starting from x_0 which finally converges to the origin when time goes to infinity.

$$\mathcal{O} = \{x_0 \in S | \lim_{t \to \infty} \psi(t; x_0) = 0\} \tag{8}$$

$$V(x) > 0 \quad \forall x \neq 0 \tag{9}$$

$$\dot{V}(x) < 0 \quad \forall x \in \{x \, | V(x) \leq \beta, x \neq 0\} \tag{10}$$

- β: sublevel of Lyapunov function $V(x)$
- $\{x \, | V(x) \leq \beta, x \neq 0\}$ is one part of RoA

Sum-of-Squares(SOS)

- Consider a polynomial function,

$$f(x) = x^2 + 8x^4 \tag{11}$$

  and a given polynomial basis $b(x) = \left[x, x^2\right]^T$ and define a coefficient matrix $Q = \begin{bmatrix} q1 & q2 \\ q3 & q4 \end{bmatrix}$

- Let $f(x) = b^T Q b$, if $q1 = 1$, $q2 = q3 = 0$ and $q4 = 8$, Q is positive definite and $f(x) = b^T Q b = \left\|(x, \sqrt{8}x^2)\right\|^2 \in SOS$

- Sufficient condition for $f(x) \in SOS$ if Q is PSD and symmetric

Converting SOS to SDP

- Consider a semi-definite program,

$$\min_{X \in S^n} \mathbf{Tr}(CX), \tag{12}$$

$$\textbf{s.t. } X \succeq 0, \tag{13}$$

$$\mathbf{Tr}(A_i X) = b_i, \ \forall i = 1, ..., N \tag{14}$$

- we can form a SDP problem to find a Q

$$\min_{Q \in S^n} 1, \tag{15}$$

$$\textbf{s.t. } Q \succeq 0, \tag{16}$$

$$q1 = 1, \tag{17}$$

$$q2 = q3 = 0, \tag{18}$$

$$q4 = 8 \tag{19}$$

Outline

High level path planning

Consider a general car model,

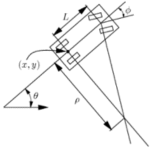


Figure: dynamic model of a simple car

$$\dot{x} = u_s \cos\theta, \tag{20}$$
$$\dot{y} = u_s \sin\theta, \tag{21}$$
$$\dot{\theta} = (u_s/L) \tan u_\phi \tag{22}$$

- $u = (u_s, u_\phi)$ where u_s is speed and u_ϕ is steering angle of front wheel

High level path planning

- Find the nominal trajectory maximizing the progress along the track can be converted into the following integer program

$$\max_{j \in 1,\ldots,M_0} P^*(X_N^j, Y_N^j), \tag{23}$$

$$\textbf{s.t. } x_0^j = x, \tag{24}$$

$$x_{k+1}^j = f_{km}(x_k^j), \tag{25}$$

$$x_k^j \in \mathcal{X}_{track} \tag{26}$$

- $x_k = (X_k, Y_k, \theta_k)$ represents discrete version of position and vehicle heading
- M_0: total number of nominal trajectory
- N: total number of discrete points on a single nominal trajectory
- $f_{km}(x_k^j)$ is discrete version of dynamics

High level path planning

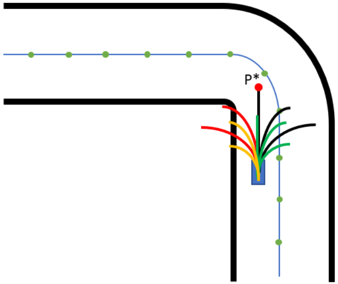


Figure: Nominal trajectories and progress

- $P^* : \mathbb{R}^2 \rightarrow [0, L]$
- L: total length of the centerline
- nominal trajectories on different u_s

Choose a trajectory

- Need to define a starting point for aggregating progress
- Green points: equally spaced points have data structure (x,y,p)
- Vehicle is travelling counter-clockwise

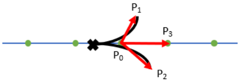


Figure: Two nominal trajectories progress comparison

$$Pr = \left| \frac{\langle P_1 - P_0, P_3 - P_0 \rangle}{\|P_3 - P_0\|} \right| \quad (27)$$

- Pr: projection value for end point P_1

SOS on polynomial optimization

Global optimization

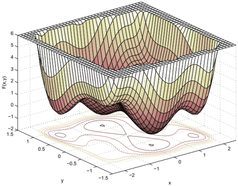
Consider $\min_{x,y} F(x,y)$, with
$F(x,y) := 4x^2 - \frac{21}{10}x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4.$

Not convex. Many local minima. NP-hard. How to find good lower bounds?

- Find the largest γ s.t.

$$F(x,y) - \gamma \text{ is SOS.}$$

- If exact, can recover optimal solution.
- Surprisingly effective.



Solving, the maximum γ is -1.0316. Exact bound.
Details in (P. & Sturmfels, 2001).

Direct extensions to constrained case.

Massachusetts Institute of Technology

ACC 2006 - Sum of squares optimization – p. 17/39

Figure: Slide from Parrilo

SOS on polynomial optimization

- Gradient based methods do not work, stuck at local minimum
- Define γ to be lower bound of $F(x, y)$
- Check if $F(x, y) - \gamma > 0$

$$\min_{Q \in S^n} \quad \gamma \tag{28}$$

$$\textbf{s.t. } F(x, y) - \gamma = b^T Q b \in SOS \tag{29}$$

- b is a polynomial basis and $b = [1, x, x^2, x^3, y, y^2, y^3, xy, x^2y, xy^2]$

Obstacle avoidance with Sum-of-squares programming

On lower level of algorithm, our goal is to make autonomous car avoid obstacles by calculating barrier certificate for each control primitives $u_i = (u_s, u_{\phi_{des},i})$, here we let $u_s = 1.0m/s$

$$V(\mathbf{x}_0) = 0, \tag{30}$$

$$V(\mathbf{x}) > 1, \ \ \forall (x, y) \in X_{obs} \setminus (x_0, y_0), \tag{31}$$

$$\dot{V}(\mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}, u_i) < 0, \ \ \forall \mathbf{x} \in X \tag{32}$$

- $u_{\phi_{des},1} = 0$ rad , $u_{\phi_{des},2} = -20\pi/180$ rad, $u_{\phi_{des},3} = 20\pi/180$ rad, $u_{\phi_{des},4} = -45\pi/180$ rad and $u_{\phi_{des},5} = 45\pi/180$ rad
- Search for polynomial function $V(x)$ of degree 4
- $X_{obs} \subset \mathbb{R}^2$

Obstacle avoidance with Sum-of-squares programming

Conditions in (29)-(31) can be transferred into the following SOS programming,

$$V(\mathbf{x}_0) = b_0^T Q b_0 = 0, \tag{33}$$

$$(V(\mathbf{x}) - (1+\epsilon)) - J g_{obs} \in SOS, \tag{34}$$

$$-\dot{V}(\mathbf{x}, u_i) \in SOS, \tag{35}$$

$$J \in SOS \tag{36}$$

- $\epsilon > 0$
- J is a multiplier with a specific degree
- $g_{obs} = \{(x, y) : g_{obs}(x, y) \geq 0\}$ denotes a semi-algebraic set (e.g $g_{obs} = \{(x, y) \in X | 1.0 - (x - 4.0)^2 - (y - 2.0)^2 \geq 0\}$)

Convert to SDP via coefficient matching

We parameterize $V(\mathbf{x}) = b_1^T Q b_1$ where vector $b_1 = \left[1, x, y, \theta, x^2, xy, y^2, x\theta, y\theta, \theta^2\right]^T$

$$\min_{N,M,G,Q \in S^n} \mathbf{Tr}(CQ), \tag{37}$$

$$\textbf{s.t. } V(\mathbf{x}) = b_1^T Q b_1, \tag{38}$$

$$J = b_2^T N b_2, \tag{39}$$

$$(V(\mathbf{x}) - (1+\epsilon)) - J g_{obs} = b_1^T M b_1, \tag{40}$$

$$-\dot{V}(\mathbf{x}, u_i) = b_1^T G b_1, \tag{41}$$

$$N, M, G \succeq 0 \tag{42}$$

- $b_2 = [1, x, y, \theta]^T$, C for choosing particularly elegant solution(e.g minimize sum of diagonal elements)
- SCS solver in CVXPY package can be used to solve this SDP problem
- objective function is set to 1 in simulation

Algorithm: Hierarchical obstacle avoidance with BC

Algorithm 2: Hierarchical obstacle avoidance with BC2

Data: $f(\mathbf{x}), speed, \mathbf{x}_0, \phi_{des,i}$
Result: return control input $u = (u_s, u_\phi)$
while *True* **do**
  $u_s \leftarrow speed$;
  $\psi_{candidates} \leftarrow$ empty list;
  Solve integer program and get ϕ_{best};
  Store candidate trajectories into $\psi_{candidates}$;
  Check if vehicle is close to any obstacle $(x, y) \in X_{obs}$;
  if *not close* **then**
    $u \leftarrow (u_s, \phi_{best})$
  end
  else
    Solve SOS program on ϕ_{best};
    if *available* **then**
      $u \leftarrow (u_s, \phi_{best})$
    end
    else
      Solve first available ϕ in $\psi_{candidates}$ with SOS;
      $u \leftarrow (u_s, \phi)$
    end
  end
  execute control u
end

- SDP program in CVXPY package includes both setting up time and solving time
- Recommend parallel computing on SOS program

Algorithm implemented in simulation

Algorithm 1: Hierarchical obstacle avoidance with BC1

```
Data: f(x), speed, x_0, φ_{des,i}, threshold
Result: return control input u = (u_s, u_φ)
while True do
    u_s ← speed;
    ψ_candidates ← empty list;
    Solve integer program and get φ_best;
    Store candidate trajectories into ψ_candidates;
    Check if vehicle is close to any obstacle (x, y) ∈ X_obs;
    if not close then
        u ← (u_s, φ_best)
    end
    else
        Solve SOS program on φ_best and get solving time t;
        if t < threshold then
            u ← (u_s, φ_best)
        end
        else
            Solve first available φ in ψ_candidates with SOS;
            u ← (u_s, φ)
        end
    end
    execute control u
end
```

- Through experiment results, solving time increases for colliding direction when vehicle approaches the obstacle

Outline

High level path planning simulation

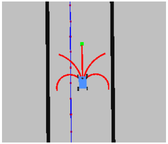


Figure: Going straight maximizes the track progress

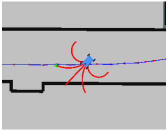


Figure: Slight right turn maximizes the track progress

- Green points indicate best nominal trajectory in both vehicle state, red lines denote five nominal trajectories and blue line is "centerline"

Lower level barrier certificate generation

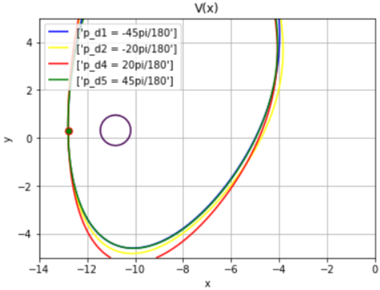


Figure: 1 sublevel of barrier function

- 2-D circle obstacle with radius $r = \sqrt{0.4}$ and locate at (-10.82,0.32)
- Green dot indicates origin of the vehicle coordinate location in the global coordinate system

Numerical error and solving method

Numerical error

- SCS doen't show numerical error on desired angle p_d3 but SeDuMi solver in MATLAB indicate numerical error
- There does not exist such barrier certificate on p_d3

Solving method

- Similar to primal-dual interior point method
  - Setup dual problem for SDP
  - Compute primal dual search direction by solving linear system of equations from KKT condition
  - Perform line search to ensure PSD for matrix variables (F Alizadeh 1998 SIAM Journal)

  Optimality
  - Maximum iteration of the solver is 10000
  - Primal,dual residual and duality gap r_p, r_d, r_g are less than ϵ (user defined)

Algorithm simulation result

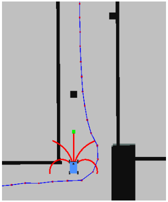


Figure: Vehicle's optimal direction pointing toward the obstacle

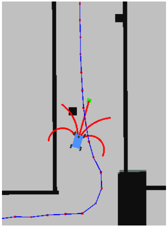


Figure: Vehicle steers to the right to avoid obstacles

Algorithm simulation result

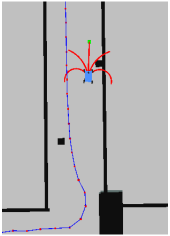


Figure: Vehicle steers to the left to avoid second obstacles

- Initialize pixel obstacles located at (-10.82,0.32) and (-8.504,-0.840) globally
- Treat pixel obstacle as a circle with a certain radius
- Append corresponding g_{obs} constraint to optimization problem when close to obstacle

Algorithm simulation result

For multiple obstacles detected, we need to specify new multiplier for each obstacle $J_i = b_2^T N_i b_2$ and let $N_i \succeq 0$

$$(V(\mathbf{x}) - (1 + \epsilon)) - J_i g_{obs_i} \in SOS \quad \forall (x, y) \in X_{obs_i} \tag{43}$$

- Where g_{obs_i} corresponds different obstacle
- Other constraints in the SDP problem stay same
- When travelling toward obstacle, it takes more steps to enforce zero duality gap

Outline

Obstacles with Lidar detection

Recall a polyhedron is the intersection of finite number of half spaces,

$$P_{polyhedron} = \{\mathbf{x} \in \mathbb{R}^2 \mid \alpha_i^T \mathbf{x} \leq b_i,\ i = 1, ..., n\} \tag{44}$$

- n is the total number of half space constraint to form the polyhedron

Then we can form the matrix inequality $A\mathbf{x} \leq \mathbf{b}$

- Each row in $A \in \mathbb{R}^{n\times 2}$ is α_i^T
- $\mathbf{b} \in \mathbb{R}^n$ where each entry in $\mathbf{b}$ is b_i

Obstacles with Lidar detection

Let vector $\mathbf{G}(\mathbf{x}) = \mathbf{b} - A\mathbf{x}$ that each entry in vector $\mathbf{G}(\mathbf{x})$ is non-negative, then we have the semi-algebraic set $g_{polyhedron} = \{\mathbf{x} \in X \mid \mathbf{G}(\mathbf{x}) \geq \mathbf{0}\}$. Conditions for searching for a valid $V(\mathbf{x})$ can be modified as the following to handle polyhedron obstacle,

$$V(\mathbf{x}_0) = b_0^T Q b_0 = 0, \tag{45}$$

$$(V(\mathbf{x}) - (1+\epsilon)) - \mathbf{J}^T \mathbf{G}(\mathbf{x}) \in SOS, \tag{46}$$

$$-\dot{V}(\mathbf{x}, u_i) \in SOS, \tag{47}$$

$$J_i \in SOS, \ \ \forall i = 1, ..., n \tag{48}$$

- $\mathbf{J}$ is vector of multipliers containing J_i
- Generate convex hull with laser points

Dealing with noise term

Assume there is an uncertain "cross-wind" term w bounded between $[-0.05, 0.05]$ in the dynamics,

$$\dot{x} = u_s \cos\theta + \mathbf{w}, \tag{49}$$

$$\dot{y} = u_s \sin\theta, \tag{50}$$

$$\dot{\theta} = (u_s/L)\tan u_\phi \tag{51}$$

Previous condition can be modified as the following,

$$\dot{V}(\mathbf{x},\mathbf{w}) = \frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}, u_i, \mathbf{w}) < 0, \;\; \forall \mathbf{x} \in X, \; \forall \mathbf{w} \in [-0.05, 0.05] \tag{52}$$

Outline

Conclusion

Possible future improvement

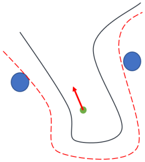


Figure: Maximize region of barrier certificate

- Maximizing safe region with permissive barrier certificate (L Wang 2018)
- Introduce parallel computing on SOS for efficiency
- Take noise term into account on dynamics to increase robustness of the algorithm
- Design control algorithm for higher level part

Thank You