

Online Safety Filtering for Autonomous Control

Athindran Ramesh Kumar

Advisor: Prof. Peter Ramadge

ECE, Princeton University

June 25, 2026

- 1 Motivation
- 2 Preliminaries
- 3 Fast, smooth, and safe: Implicit CBFs using Reach-Avoid Differential Dynamic Programming
- 4 Implicit safety filters: from Theory to Practice
- 5 Thinking Ahead

Motivation

Bipedal robotics



Drone delivery

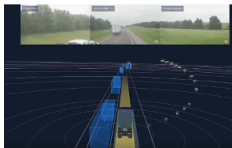


Laundry Robot



Safety crucial for autonomous systems!

Major bottleneck preventing commercialization of transformative technologies



Autonomous trucking



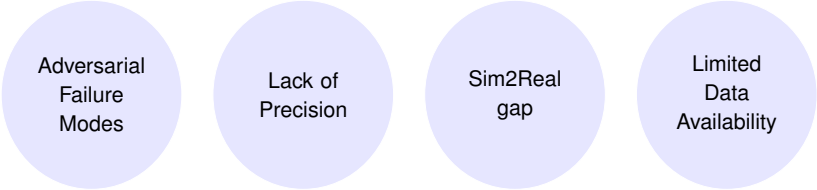
Ride-Hailing



Last-mile delivery

What makes safety hard to achieve?

Unstructured Deep Learning Policies that are limited to statistical validation



Adversarial
Failure
Modes

Lack of
Precision

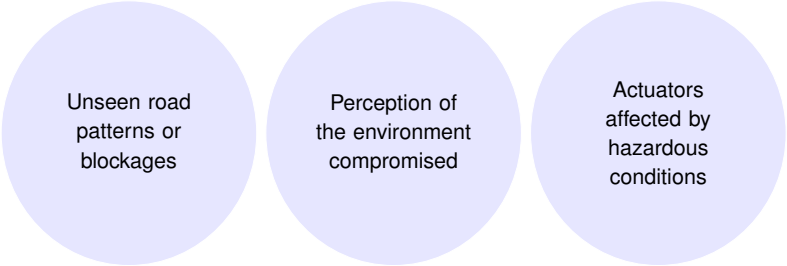
Sim2Real
gap

Limited
Data
Availability

Neural network policy has been validated, but it may trigger a failure mode for a small percentage of time

What makes safety hard to achieve?

Unstructured Environments that are not controlled beforehand



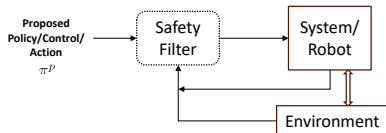
Unseen road
patterns or
blockages

Perception of
the environment
compromised

Actuators
affected by
hazardous
conditions

My robot is designed accurately, but may not function as expected for a small percentage of time

Supervise to prevent failure



What is a safety filter?

Functionally a module that modifies the proposed control to ensure safe operation for a time-horizon of interest

Desiderata of a safety filter

- ▶ **Guaranteed safety** - never fails?
- ▶ **Robust operation** with imperfect knowledge
- ▶ **Smooth fallback** to safety

What information does it have?

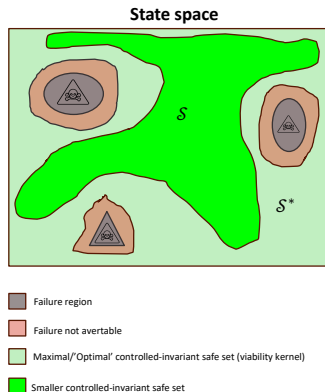
- ✓ Full knowledge of safety specification
- ✗ Full knowledge of system
- ✗ All necessary knowledge of environment

Preliminaries

Refer to $\mathcal{S}$ as controlled invariant if $\forall x_0 \in \mathcal{S}, \exists u(t)$
such that $x(t) \in \mathcal{S}, \forall t$

Controlled invariant safe set - $\mathcal{S} \cap \mathcal{F} = \emptyset$

System is rendered safe if it is possible to avoid the failure region with some control signal for all future times



CBF¹

$$\dot{x} = f(x) + g(x)u$$

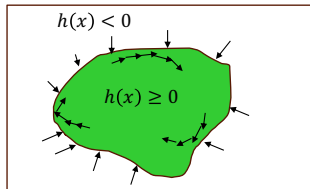
- $\mathcal{S} \subset \mathcal{X}$ - 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \rightarrow \mathbb{R}$
- $\frac{\partial h}{\partial x} \neq 0, \forall x \in \partial \mathcal{S}$.
- h is a valid CBF on $\mathcal{S}$ if $\exists \alpha \in \mathcal{K}_{\infty, e}$ such that for all $x \in \mathcal{S}$,

$$\begin{aligned} \max_{u \in \mathcal{U}} \dot{h}(x, u) &= \max_{u \in \mathcal{U}} [L_f h(x) + L_g h(x)u] \\ &\geq -\alpha(h(x)) \quad \forall x \in \mathcal{X} \end{aligned}$$

- Existence of a CBF $h \implies$

$$\begin{aligned} K_{\text{cbf}}(x) = \\ \{u \in \mathcal{U} \mid L_f h(x) + L_g h(x)u \geq -\alpha(h(x))\} \end{aligned}$$

renders $\mathcal{S}$ controlled invariant



$$\begin{aligned} L_f h(x) &= \frac{\partial h}{\partial x} \cdot f(x) \\ L_g h(x) &= \frac{\partial h}{\partial x} \cdot g(x) \end{aligned}$$

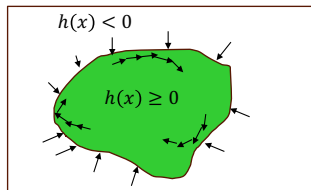
Apply controls as h decreases while nearing the boundary of safe set and pull back inside

¹Aaron D Ames, Xiangru Xu, et al. (2016). "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8, pp. 3861–3876.

CBF-QP Safety filter

$$\begin{aligned}\pi^{\text{sf}}(x) &= \arg \min_{u \in \mathcal{U}} \|u - \pi^p(x)\|_2^2 \\ \text{s.t. } L_f h(x) + L_g h(x)u + \alpha(h(x)) &\geq 0.\end{aligned}$$

If h is continuously differentiable, then $\pi^{\text{sf}}(x)$ is Lipschitz continuous

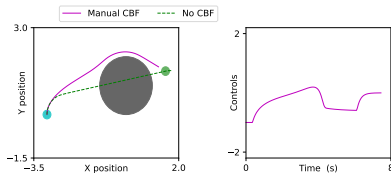


Manually designed CBFs successful for
**Automatic cruise control, Lane maintenance,
Segway stabilization, Bipedal walking robots**

Results in smooth safety
filtering in conjunction
with quadratic program ✓

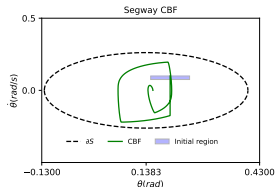
Dubins car

$$\text{CBF: } h(r, \theta) = 0.4 \times (r^2 \times \sin(\theta) - R^2)$$



Segway

$$\text{CBF: } h(\theta, \dot{\theta}) = \theta_m^2 - \dot{\theta}^2 - (\theta - \theta_e)^2$$



Maintain distance to vehicle in front

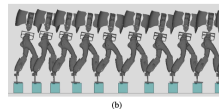
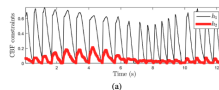
$$h(x) = D - \tau v_f$$

D - distance to vehicle in front

τ - Time headway

v_f - Velocity

Bipedal walker over stepping stones



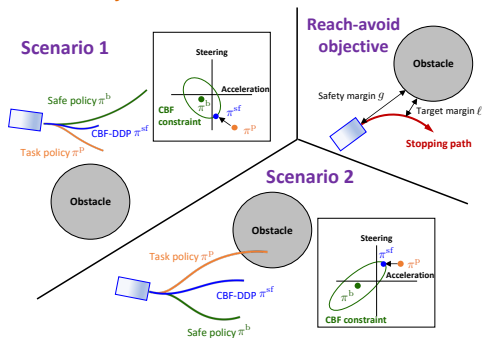
Refer^a for a detailed study of these examples

^aAaron D Ames, Samuel Coogan, et al. (2019). "Control barrier functions: Theory and applications". In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 3420–3431.

- 1 Motivation
- 2 Preliminaries
- 3 Fast, smooth, and safe: Implicit CBFs using Reach-Avoid Differential Dynamic Programming**
- 4 Implicit safety filters: from Theory to Practice
- 5 Thinking Ahead

Fast, smooth, and safe: Implicit CBFs using Reach-Avoid Differential Dynamic Programming

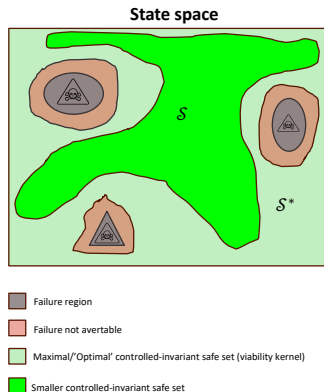
Construct a smooth safety filter at runtime that never fails? $x_{t+1} = f(x_t, u_t)$



Athindran Ramesh Kumar et al. (2023). "Fast, smooth, and safe: Implicit control barrier functions through reach-avoid differential dynamic programming". In: *IEEE Control Systems Letters*

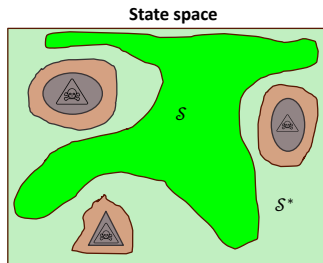
How do we synthesize CBFs?

- ▶ Handcrafted CBFs require prior verification
- ▶ CBF-QP infeasible with additional control or state constraints
- ▶ Handcrafted CBFs conservative and will not find the maximal $\mathcal{S}^*$
- ▶ Possible solutions: Neural CBF, Backup CBF



How do we synthesize CBFs?

- ▶ Handcrafted CBFs require prior verification
- ▶ CBF-QP infeasible with additional control or state constraints
- ▶ Handcrafted CBFs conservative and will not find the maximal S^*
- ▶ Possible solutions: Neural CBF, Backup CBF



- Failure region
- Failure not avertable
- Maximal/"Optimal" controlled-invariant safe set (viability kernel)
- Smaller controlled-invariant safe set

Can we find a better way?

Hamilton-Jacobi Reachability

$$\mathcal{R}(t) := \{x \in \mathcal{X} \mid \exists \pi^b, \forall \tau \in [t, H], \mathbf{x}(\tau; x, \pi^b) \notin \mathcal{F}\}$$

Define constraint margin: $x \notin \mathcal{F} \leftrightarrow c(x) > 0$

Reachability value function

Consider the reachability objective J ,

$$J(x_t, \pi^b, t) := \min_{\tau \in [t, H]} c(x_\tau)$$

$$V(x_t, t) := \max_{\pi^b} J(x_t, \pi^b, t)$$

$\mathcal{R}(t)$ is the zero-superlevel set of V

Least restrictive (LR) filtering

$$\pi^{sf}(x) := \begin{cases} \pi^p(x), & V(f(x, \pi^p(x)), t+1) > 0 \\ \pi^b(x), & \text{otherwise.} \end{cases}$$

De facto use of V - causes chattering

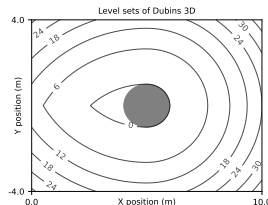


Figure 1: Level sets of Dubins car 3D as captured by Stanford ASL tool

Reach-Avoid Set

$$\begin{aligned} \mathcal{RA}(t) := \{ & x \in \mathcal{X} \mid \exists \pi^b, \forall \tau \in [t, H], \\ & \mathbf{x}(\tau; x, \pi^b) \in \mathcal{T} \\ & \wedge \forall s \in [t, \tau], \mathbf{x}(s; x, \pi^b) \notin \mathcal{F} \} \end{aligned}$$

Define target margin: $x \in \mathcal{T} \leftrightarrow \ell(x) \geq 0$

Reach-avoid value function

Consider the reach-avoid objective J ,

$$J(x_t, \pi^b, t) := \max_{\tau \in [t, H]} \min \left\{ \ell(x_\tau), \min_{s \in [t, \tau]} c(x_s) \right\}$$

$$V(x_t, t) := \max_{\pi^b} J(x_t, \pi^b, t)$$

$\mathcal{RA}(t)$ is the zero-superlevel set of V

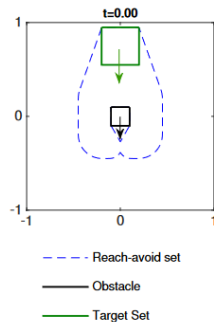


Figure 2: Reach-avoid set from^a

^aJaime F. Fisac et al. 2015.

Discrete-time Dynamic Programming

$$V(x_t, t) := \min \left\{ c(x_t), \max \left\{ \ell(x_t), \max_{u_t \in \mathcal{U}} V(f(x_t, u_t), t + 1) \right\} \right\},$$

with terminal condition $V(x_H, H) = \min \{c(x_H), \ell(x_H)\}$.

Key Takeaways

- ▶ Exact DP suffers from curse of dimensionality
- ▶ Does not scale for more than 6 dimensions
- ▶ Computed V often used as LR filter
- ▶ Amenable to recursive feasibility guarantees

Motivation

Idea: Use HJ value functions with CBF-style constraint to enable smooth safety interventions

How? Reach-avoid formulation is used with a receding horizon of length H



New notation:

$\mathcal{RA}(\mathcal{F}, \mathcal{T}, H)$ - Finite-horizon reach-avoid set from t to $t + H$

$V(x, \tau)$ - Finite-horizon value function from time $\tau \geq t$ until H

H - Length of horizon

c, ℓ - constraint margin and target margin

Related Work

²Offline construction of a valid CBF/CLF based on reachability analysis

Establish relationship between HJ value functions and CBFs

No runtime construction

Does not solve the curse of dimensionality without neural function approximation

³Model-predictive safety certification

Runtime CBF using model-predictive safe set approximation

Ours more general than assumption of terminal safe set

Ours places a lot more emphasis on smooth safety filtering

⁴Learned CBFs

Enforce CBF constraints as a soft loss function with NN

Not always "optimal"

Approximation limited by data

²Choi et al. 2021.

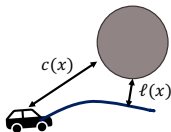
³Wabersich, Hewing, et al. 2021; Wabersich and Zeilinger 2018.

⁴Dawson et al. 2022; Robey et al. 2020.

Motivation

Is reach-avoid value function a valid CBF ?

Yes, with correct design of $\ell(x)$ and $c(x)$



Example: robot moving through sequence of obstacles

$c(x)$ - Amount of constraint margin
e.g. distance to obstacles, distance to road boundary, distance to maximum yaw limits
Minimum of all constraints

$\ell(x)$ - Amount of constraint margin along a counterfactual stopping path

$$x \in \mathcal{T} \leftrightarrow \ell(x) \geq 0$$

Possible to come to a safe stop if target margin is non-negative

Reach-avoid value function provably satisfies the CBF constraint

$$\exists u \in \mathcal{U}, \text{ s.t. } V(f(x, u)) \geq \gamma V(x) \quad \forall x \in \mathcal{S}, \gamma \in (0, 1]$$

Lemma (Recursive Feasibility)

The V satisfies CBF constraint if $x_t \in \mathcal{RA}(\mathcal{T}, \mathcal{F}, H)$

Theorem (Recursive Safety)

Suppose that $\mathcal{T} \cap \mathcal{F} = \emptyset$ and $\mathcal{T}$ is controlled invariant. If $x_t \in \mathcal{RA}(\mathcal{T}, \mathcal{F}, H)$, then $\exists u_t$ such that $x_{t+1} = f(x_t, u_t) \in \mathcal{RA}(\mathcal{T}, \mathcal{F}, H)$. Furthermore, u^* is a feasible solution for $x_{t+1} \in \mathcal{RA}(\mathcal{T}, \mathcal{F}, H)$.

Theorem (Infinite-Horizon Safety)

Given $x_0 \in \mathcal{T}$, satisfying the reach-avoid constraint ensures that $\forall t, x_t \in \mathcal{RA}(\mathcal{T}, \mathcal{F}, H)$.

$$\mathcal{S} = \mathcal{RA}$$

If currently in reach-avoid set, can enforce CBF constraint to remain in reach-avoid set forever.

Guaranteed safety for entire time!

How do we construct V ?

Run-time construction using Differential Dynamic Programming (DDP/ILQR)⁵

ILQR Workflow

Initialize a trajectory of states and controls

- ▶ Form a quadratic approximation of costs along trajectory
- ▶ Form linear/quadratic approximation of dynamics
- ▶ Use the quadratic approximation to form a local LQR formulation
- ▶ Find the closed-loop and open-loop optimal gains
- ▶ Rollout trajectory with new solution
- ▶ Go back to 1

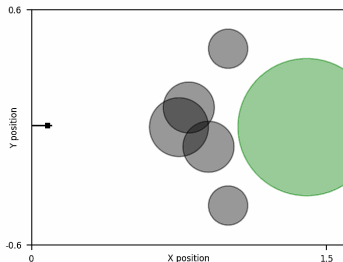


Figure 3: Reach-avoid ILQR optimization

Benefit: Get first and second-order derivatives of V as by-product

⁵Dennis R Anthony et al. (2022). "Back to the future: efficient, time-consistent solutions in reach-avoid games". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6830–6836.

Can we use reach-avoid value function like a CBF?

How do we construct V ?

- ▶ **Run-time construction using Differential Dynamic Programming (DDP/ILQR)**
- ▶ Scalable solution rather than exact DP
- ▶ Allows online construction

Is V continuously differentiable?

- ▶ V is only Lipschitz continuous $\implies V$ is continuously differentiable almost everywhere
- ▶ What does it mean for π^{sf} by using V as CBF? Refer^{6*} for theorems
- ▶ u is continuous a.e., given f , V is Lipschitz a.e.

Does $\frac{\partial V}{\partial x} \neq 0$ for all $x \in \partial S$?

Expectation: Control is Lipschitz continuous a.e.

⁶Ramesh Kumar et al. 2023.

Key Idea of our formulation

Obtain V using Reach-avoid ILQR

$$V(x_{1|t}, 0) \approx V(\bar{x}_{1|t}, 0) + \delta x_{1|t}^\top z + \frac{1}{2} \delta x_{1|t}^\top Z \delta x_{1|t}$$

2nd order approx
around coun-
terfactual state

$$\min_{u \in \mathcal{U}} \frac{1}{2} \|u - u^p\|^2$$

$$V(f(x, u), 0) \geq \gamma V(x, 0) \quad \longleftrightarrow$$

$$\min_{\delta u: \delta u + u^p \in \mathcal{U}} \|\delta u\|_2^2$$

$$\text{s.t. } \delta u^\top P \delta u + p^\top \delta u + c \geq 0.$$

$$P = \frac{1}{2} f_u^\top Z f_u, \quad p = f_u^\top z$$

$$c = V(\bar{x}_{1|t}, 0) - \gamma V(x_t, 0)$$

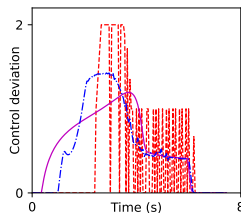
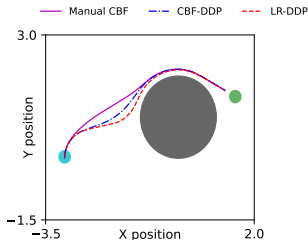
Repeat and resolve until constraint satisfied

Dubins Car

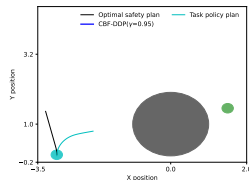
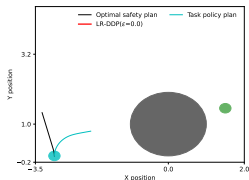
Specs: 3.6 GHz CPU with 64 GB RAM using JAX

$$\dot{x} = v \cos(\theta), \dot{y} = v \sin(\theta), \dot{\theta} = u \quad (\Delta t = 0.05s, H = 40)$$

CBFDDP
smoothly
crosses the
obstacle



LRDDP prone
to chattering

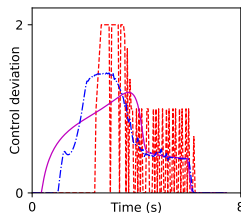
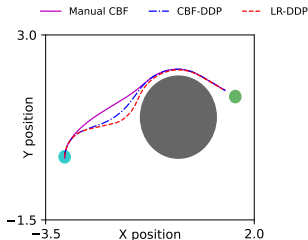


Dubins Car

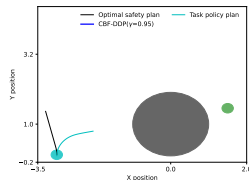
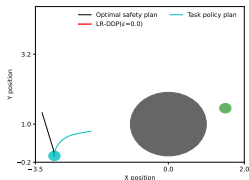
Specs: 3.6 GHz CPU with 64 GB RAM using JAX

$$\dot{x} = v \cos(\theta), \dot{y} = v \sin(\theta), \dot{\theta} = u \quad (\Delta t = 0.05s, H = 40)$$

CBFDDP
smoothly
crosses the
obstacle

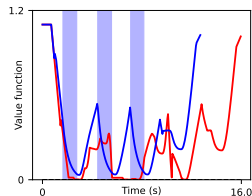
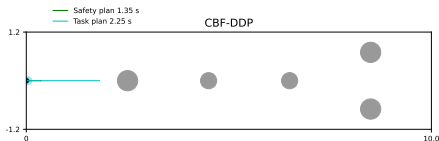
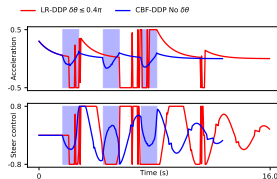
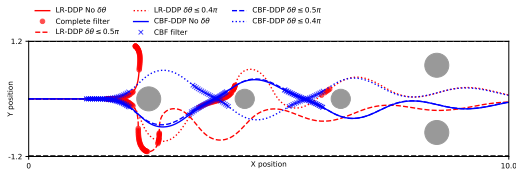


LRDDP prone
to chattering



Bicycle 5D dynamics - avoid all road boundaries and obstacles

$$\dot{x} = v \cos(\theta), \dot{y} = v \sin(\theta), \dot{v} = u_1, \dot{\theta} = v \frac{\tan(\delta)}{L}, \dot{\delta} = u_2 \quad (\Delta t = 0.05s, H = 45)$$

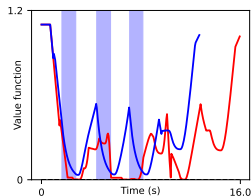
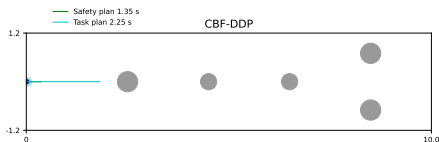
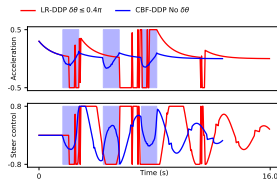
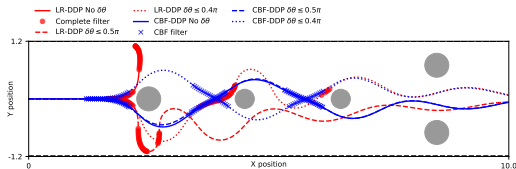


LR provides jerk reactions and takes panic U-turns without constraints on yaw

Experimental Validation

Bicycle 5D dynamics - avoid all road boundaries and obstacles

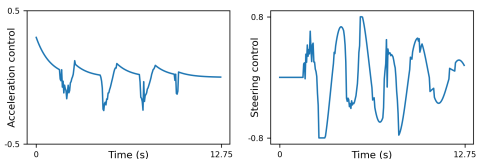
$$\dot{x} = v \cos(\theta), \dot{y} = v \sin(\theta), \dot{v} = u_1, \dot{\theta} = v \frac{\tan(\delta)}{L}, \dot{\delta} = u_2 \quad (\Delta t = 0.05s, H = 45)$$



LR provides jerk reactions and takes panic U-turns without constraints on yaw

Problem: Interacting constraints may cause discontinuities in controls

Introduction of interacting constraints such as yaw causes more discontinuous behavior



CBFDDP fails to give smooth controls even with loose yaw constraints

Safety preserved

Problem: Reach-avoid DDP is rate-limited by stopping path computation and corresponding backward pass

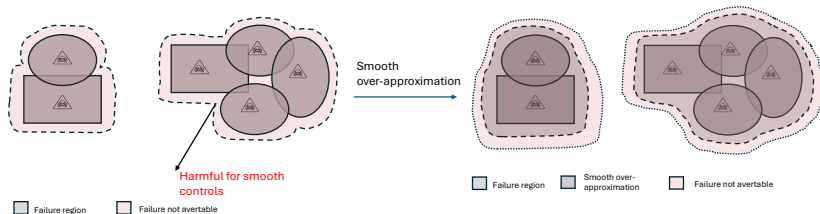
Method	Control cycle time (s) (Mean + 1-Standard deviation)
CBF-DDP with minimal constraints	0.10 +- 0.10
CBF-DDP with additional yaw constraint	0.27 +- 0.39
LR-DDP with additional yaw constraint	0.21 +- 0.22

Practical Remedies Discussed in Next Chapter

- 1 Motivation
- 2 Preliminaries
- 3 Fast, smooth, and safe: Implicit CBFs using Reach-Avoid Differential Dynamic Programming
- 4 Implicit safety filters: from Theory to Practice**
- 5 Thinking Ahead

Implicit safety filters: from Theory to Practice

Practical Remedies towards a Realizable Solution



Is our value function smooth?

Is V continuously differentiable?

Recall we expect c and ℓ to have smooth derivatives as much as possible

Composition of constraints

Provided a set of constraints, $\{m \in M : c_m(x)\}$, how do we compose them into a single $\hat{c}(x) = H(\{c_m(x)\})$?

Require $\hat{\mathcal{F}} \supseteq \cup \{\mathcal{F}_m : m \in M\}$

Possible solutions

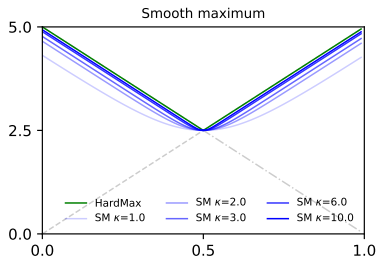
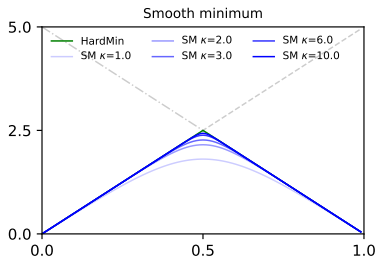
- ▶ Solution: $c(x) = \min_{m \in M} c_m(x)$ -> Causes severe discontinuities
- ▶ Solution: Use multiple CBFs for each constraint -> intractable
- ▶ **Idea: Use a principled smoothing operation that maintains the set-invariance property**

Is our value function smooth?

- ▶ Smooth lower and upper bounds of the minimum of multiple constraints are known⁷,

$$\min_j c_j(x) \geq \frac{-\log \left(\sum_j \exp(-\kappa c_j(x)) \right)}{\kappa}$$

- ▶ Similarly, defined, for a maximum of multiple constraints $c(x) = \max\{c_j(x)\}_{j=1}^n$
- ▶ Maintain $\hat{c}(x) \leq \min_j c_j(x)$ by choosing $\hat{c}(x)$ as the lower bound of $c(x)$
- ▶ Easy to see that $\hat{\mathcal{F}} \supseteq \cup\{\mathcal{F}_m : m \in M\}$



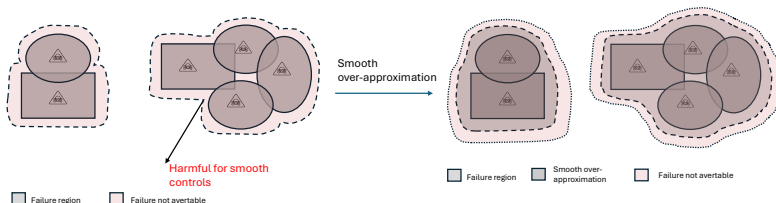
⁷ Princeton ELE lecture notes on optimization n.d.

Is our value function smooth?

- Smooth lower and upper bounds of the minimum of multiple constraints are known,

$$\min_j c_j(x) \geq \frac{-\log \left(\sum_j \exp(-\kappa c_j(x)) \right)}{\kappa}$$

- Similarly, defined, for a maximum of multiple constraints $c(x) = \max\{c_j(x)\}_{j=1}^n$
- Maintain $\hat{c}(x) \leq \min_j c_j(x)$ by choosing $\hat{c}(x)$ as the lower bound of $c(x)$
- Easy to see that $\hat{\mathcal{F}} \supseteq \cup\{\mathcal{F}_m : m \in M\}$



All guarantees from previous chapter applicable

Use a reachability CBF instead of reach-avoid for speed

Now, the CBF guarantee is applicable only if the horizon is sufficiently long^a

^aOh, Duy P Nguyen, Hu, and Jaime Fernández Fisac 2026; Hirsch, Jaime Fernández Fisac, and Herbert 2025.

Introduce a heavy penalty control cost in addition to reachability cost

Common feature for stabilizing reachability DDP^a

^aFridovich-Keil and Tomlin 2021; Anthony et al. 2022.

Use within a neural network training loop to guard against safety violations

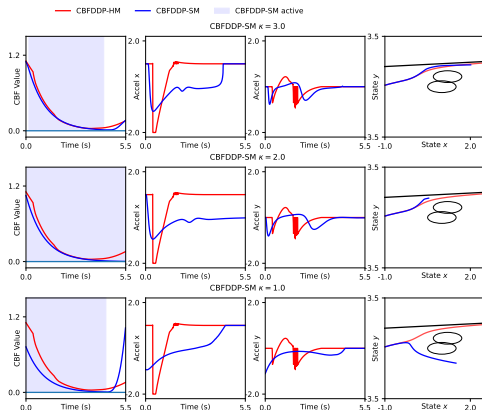
Useful for shielding reinforcement learning while collecting data^a

^aOh, Duy P Nguyen, Hu, and Jaime F Fisac 2025.

Toy examples - Reachability CBFDDP

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM - using only Numpy

$\Delta t = 0.01$, $H = 40$, $\gamma = 0.97$



Story so far

Handcrafted CBF
small safe set and
may violate safety

LRDDP
prone to chattering

CBFDDP-HM
cannot handle mul-
tiple constraints

CBFDDP-SM
handles multiple constraints

Figure 8: 2D point mass with double integrator dynamics in both X and Y directions

Tuning

Only caveat: γ needs to be tuned by mapping from continuous time for Δt

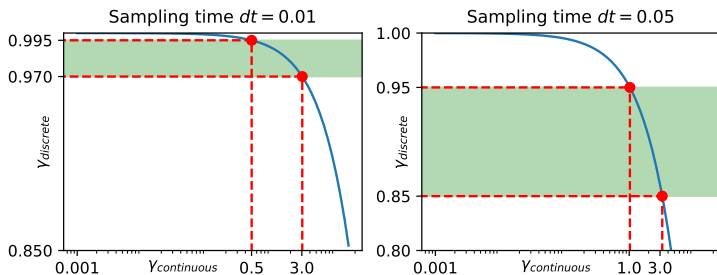


Figure 9: γ needs to be fine-tuned within a thin range

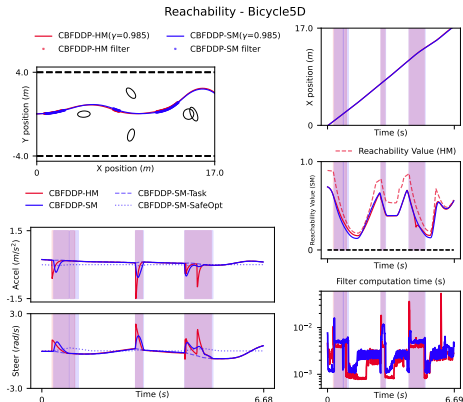
Smaller Δt helpful for our simulator. Will talk about this at the end...

Smoothing - experimental validation

Leveraging JAX for fast computation - Reachability CBFDDP

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM

$\Delta t = 0.01$, $H = 50$, $\gamma = 0.985$, $L = 0.8$, $\kappa = 4.0$



Composition of velocity, steering, road boundary, and obstacle constraints

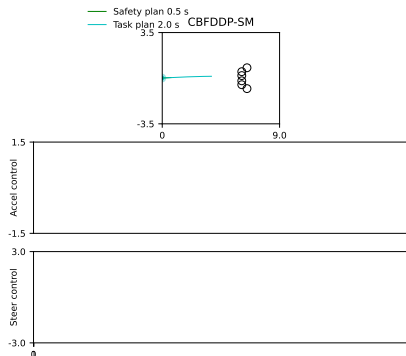
10x faster than reach-avoid CBFDDP
CBFDDP-SM performs better overall

Smoothing - experimental validation

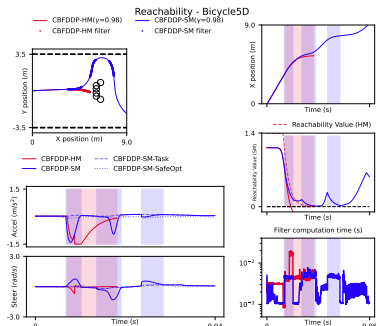
Leveraging JAX for fast computation - Reachability CBFDDP

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM

$$\Delta t = 0.01, H = 50, \gamma = 0.98, L = 0.6, \kappa = 4.0$$

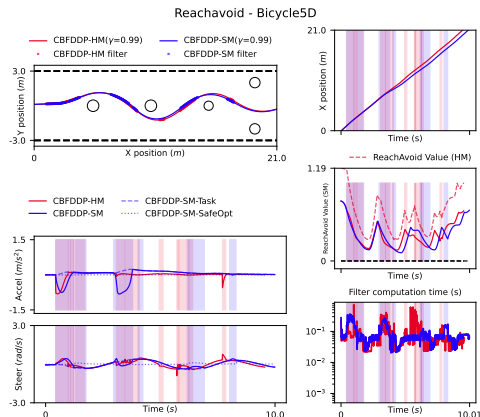


Smooth controls while crossing
a staggered array of obstacles,
with steering and yaw constraints



SM performs better over
specified range of γ

Similar results for Reach-Avoid



Bottleneck: Stopping path computation

Sources of acceleration:

- ▶ Analytical expression for stopping path
- ▶ Increased sampling time only for stopping path rollout
- ▶ Terminal set invariance

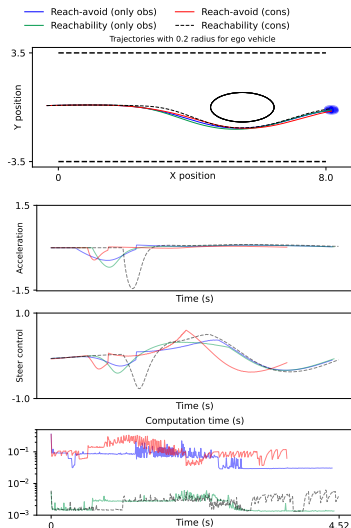
Tested with a variety of configurations - Code available
Validated with both reachability and reach-avoid, ILQR and DDP
Validated on PVTOL (6D), Ackermann (6D) and 4-Wheel Bicycle (7D) models

Tight computational budget

- ▶ Reachability ILQR pass - consumes 0.001s
- ▶ ReachAvoid ILQR pass - consumes 0.01s
- ▶ Difficult configs require > 10 iterations
- ▶ Faster runtime possible by using QP instead of QCQP

Highly parallelized GPU implementation of primal-dual ILQR
Faster by orders of magnitude^a

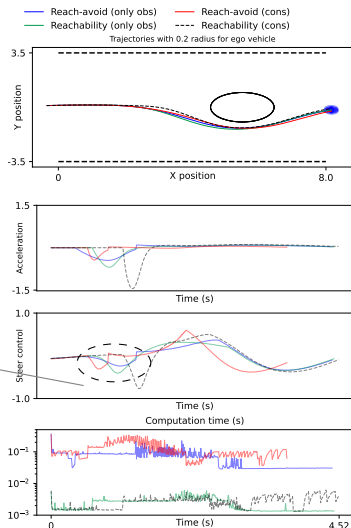
^aAmatucci et al. 2025.



Tight computational budget

- ▶ Reachability ILQR pass - consumes 0.001s
- ▶ ReachAvoid ILQR pass - consumes 0.01s
- ▶ Difficult configs require > 10 iterations
- ▶ Faster runtime possible by using QP instead of QCQP

Why is there an isolated but sharp discontinuity in reach-avoid BUT not in reachability???



More on reach-avoid singularities

Illustration on toy example

Problem: Non-differentiable points where the global optima switches between one local optima and another. Known as **singular points**

No panacea - heuristic fixes

- ▶ Multiple initial conditions
- ▶ Make the problem easier

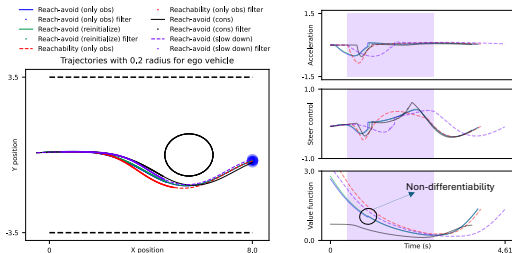


Figure 10: Illustration of local minima near singular points

Verdict: Comparable to many vanilla CBFs

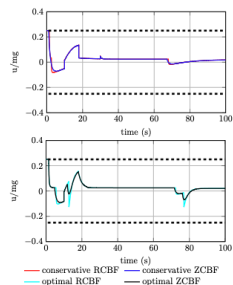


Figure 11: Lane keeping CBF example. Figure borrowed from^a

^aAmes, Xu, et al. 2016.

Off-the-shelf Application to the MJX engine

Demos of Reachability CBFDDP on the Reacher robot

Maximum velocity constraint: Prevent reacher arms from rotating rapidly

$$c(\theta, \dot{\theta}) = \min\{c_0 - \dot{\theta}_0^2, c_1 - \dot{\theta}_1^2\} \quad (H = 20, \gamma = 0.9, \Delta t = 0.02s)$$

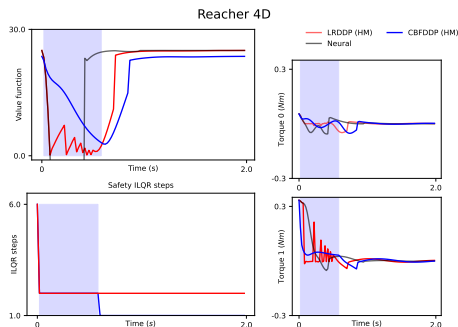


Figure 12: Reachability CBFDDP is smoother than LR counterpart. Neural task policy violates safety

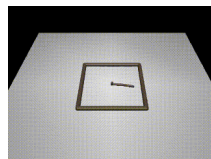


Figure 13: CBFDDP on reacher with maximum velocity constraints

Not amenable to real-time DDP implementation yet - **Bottleneck is software wiring**

Off-the-shelf Application to the MJX engine

Demos of Reachability CBFDDP on the Barkour robot

Obstacle avoidance constraint: Prevent barkour from rapidly approaching obstacles
($H = 25$, $\gamma = 0.99$, $\Delta t = 0.02s$)

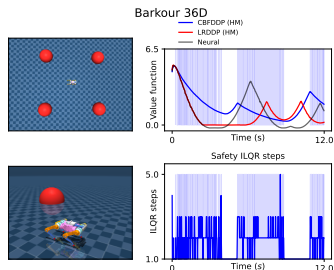


Figure 14: Reachability CBFDDP is smoother than LR counterpart. Neural task policy violates safety

Not amenable to real-time DDP implementation yet - **Bottleneck is software wiring**

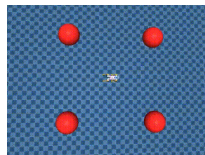


Figure 15: CBFDDP on Barkour with obstacle constraints

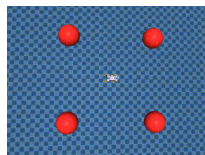
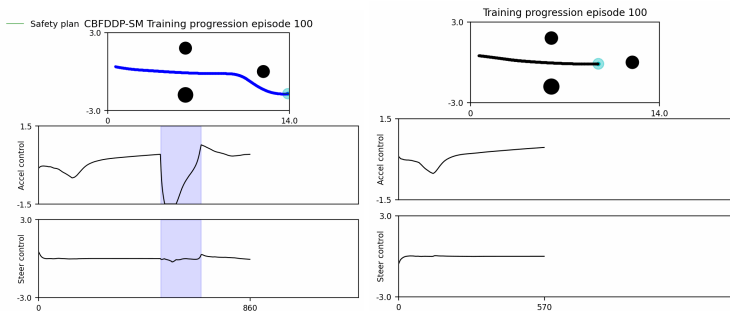


Figure 16: LRDDP on Barkour with obstacle constraints

Shield RL task policies from crashing while collecting data



(a) Training progress with filter in loop.
Sampling time $\Delta t = 0.01s$

(b) Counterfactual sim with no filter.
Sampling time $\Delta t = 0.01s$

Zero training time safety violations

Shield RL task policies from crashing while collecting data

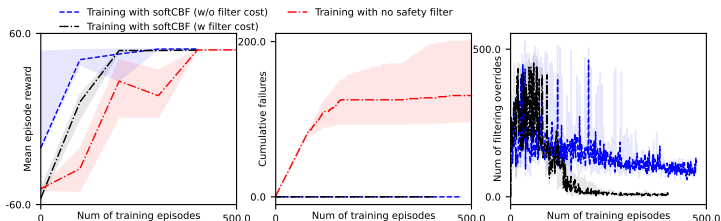


Figure 18: Training an RL agent with bicycle dynamics to complete the racetrack

Zero training time safety violations

More details at: https://github.com/athindran/cbf_ddp_sac_learner

Conclusions

- ▶ Implicit safety filters from CBFDDP result in smooth safety filtering with minimal overrides
- ▶ Practical enhancements needed for real-world use cases
- ▶ SM better for non-convex obstacle shapes and composition of constraints
- ▶ Useful for training task policies in the real-world
- ▶ Real-time implementation is foreseeable in the near future

Desiderata of a safety filter

- ✓ **Guaranteed safety** - never fails?
- ✓ **Smooth fallback**
- ✓ **Maximal safe set**
- ▶ **Robustness to uncertainty** - not covered in this talk

Thinking Ahead

Opportunities on the Horizon

- ▶ **Problem:** Decay factor γ , control costs weights, and smoothness coefficient κ need to be adequately tuned
Solution: Adaptively learn CBF parameters on-the-fly⁸
- ▶ **Problem:** Fully online DDP solutions are too expensive
Solution: Bootstrap with an offline learned policy⁹
- ▶ **Problem:** Adapt to dynamically changing environments with limited sensing
Solution: Dynamically adapt an occupancy grid on the fly and cover it with smooth corners
- ▶ **Problem:** Robust to unknown system dynamics
Solution: Uncertainty estimation with online adaptation

⁸Ehsan Sabouni et al. (2024). “Reinforcement learning-based receding horizon control using adaptive control barrier functions for safety-critical systems”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. IEEE, pp. 401–406.

⁹Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernandez Fisac (2023). “ISAACS: Iterative soft adversarial actor-critic for safety”. In: *Learning for Dynamics and Control Conference*. PMLR, pp. 90–103.

Blend with Deep RL

- ▶ Use safety DDP as an expert for training safe RL policies
- ▶ Shielding data collection and Sim2Real transfer
- ▶ Neural CBFs¹⁰ can use data collected offline from CBFDDP for training

Opportunities outside robotics

- ▶ Guardrails for AI agents
- ▶ Prevent LLM from leaking sensitive information
- ▶ Constrained control in Power Systems
- ▶ Maintaining chemical reactors within setpoints

¹⁰Alexander Robey et al. (2020). "Learning control barrier functions from expert demonstrations". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 3717–3724.

Acknowledgements

Special thanks to Professor Ramadge and his lab!



Abhinandan Ramesh Kumar



Kai-Chieh Hsu



Peter J. Ramadge



Jaime Fernández Fisac

Thanks to my co-authors Professor Fisac,
Kai-Chieh, Ting-Han and Sulin!

Thanks to my family members for
persisting!



Thanks to Aurora colleagues for
their support!

Thank You!

-  Amatucci, Lorenzo et al. (2025). “Primal-dual ilqr for gpu-accelerated learning and control in legged robots”. In: *IEEE Robotics and Automation Letters* 11.1, pp. 1010–1017.
-  Ames, Aaron D, Samuel Coogan, et al. (2019). “Control barrier functions: Theory and applications”. In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 3420–3431.
-  Ames, Aaron D, Xiangru Xu, et al. (2016). “Control barrier function based quadratic programs for safety critical systems”. In: *IEEE Transactions on Automatic Control* 62.8, pp. 3861–3876.
-  Anthony, Dennis R et al. (2022). “Back to the future: efficient, time-consistent solutions in reach-avoid games”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6830–6836.
-  Choi, Jason J et al. (2021). “Robust control barrier-value functions for safety-critical control”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 6814–6821.

-  Dawson, Charles et al. (2022). “Safe nonlinear control using robust neural lyapunov-barrier functions”. In: *Conference on Robot Learning*. PMLR, pp. 1724–1735.
-  Fisac, Jaime F. et al. (2015). “Reach-avoid problems with time-varying dynamics, targets and constraints”. In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control (HSCC)*. Seattle, WA, USA, pp. 11–20. DOI: 10.1145/2728606.2728612.
-  Fridovich-Keil, David and Claire J. Tomlin (2021). “Approximate solutions to a class of reachability games”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12610–12617. DOI: 10.1109/ICRA48506.2021.9561655.
-  Hirsch, Dylan, Jaime Fernández Fisac, and Sylvia Herbert (2025). “Viscosity CBFs: Bridging the Control Barrier Function and Hamilton-Jacobi Reachability Frameworks in Safe Control Theory”. In: *arXiv preprint arXiv:2510.09929*.
-  Hsu, Kai-Chieh, Duy Phuong Nguyen, and Jaime Fernandez Fisac (2023). “ISAACS: Iterative soft adversarial actor-critic for safety”. In: *Learning for Dynamics and Control Conference*. PMLR, pp. 90–103.

-  Oh, Donggeon David, Duy P Nguyen, Haimin Hu, and Jaime F Fisac (2025). “Provably Optimal Reinforcement Learning under Safety Filtering”. In: *arXiv preprint arXiv:2510.18082*.
-  Oh, Donggeon David, Duy P Nguyen, Haimin Hu, and Jaime Fernández Fisac (2026). “Synthesis and Deployment of Maximal Robust Control Barrier Functions through Adversarial Reinforcement Learning”. In: *arXiv preprint arXiv:2604.13192*.
-  *Princeton ELE lecture notes on optimization* (n.d.). URL: https://yuxinchen2020.github.io/large_scale_optimization/lectures/smoothing.pdf.
-  Ramesh Kumar, Athindran et al. (2023). “Fast, smooth, and safe: Implicit control barrier functions through reach-avoid differential dynamic programming”. In: *IEEE Control Systems Letters*.
-  Robey, Alexander et al. (2020). “Learning control barrier functions from expert demonstrations”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 3717–3724.
-  Sabouni, Ehsan et al. (2024). “Reinforcement learning-based receding horizon control using adaptive control barrier functions for safety-critical systems”. In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. IEEE, pp. 401–406.

-  Wabersich, Kim P, Lukas Hewing, et al. (2021). “Probabilistic model predictive safety certification for learning-based control”. In: *IEEE Transactions on Automatic Control* 67.1, pp. 176–188.
-  Wabersich, Kim P and Melanie N Zeilinger (2018). “Linear model predictive safety certification for learning-based control”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, pp. 7130–7135.

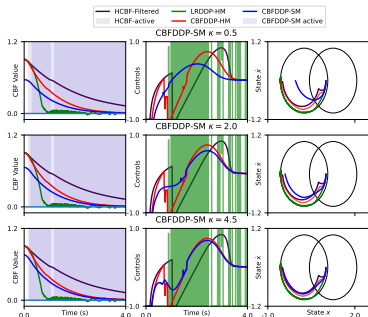
Additional Results

Smoothing - experimental validation

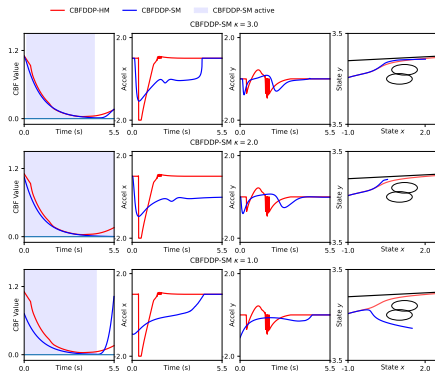
Toy examples

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM - using only Numpy

$\Delta t = 0.01, H = 15$



Double integrator in 1D

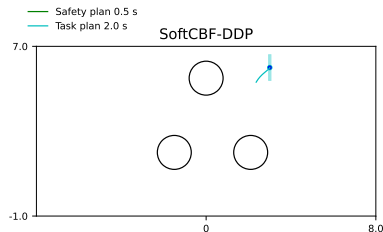


Double integrator in both X and Y directions

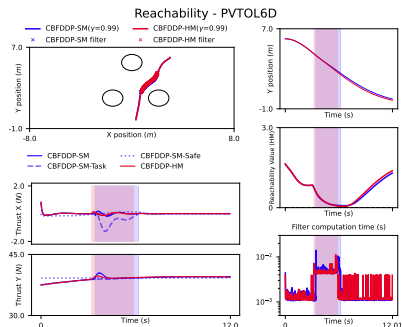
Smoothing - experimental validation

Leveraging JAX for fast computation

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM $\Delta t = 0.01$, $H = 50$



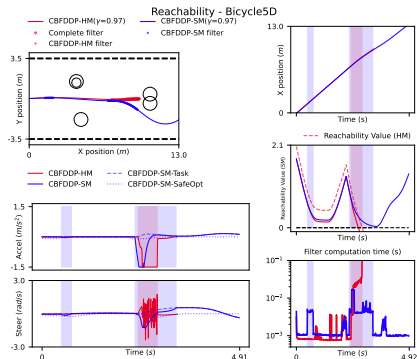
Smooth controls with the composition of constraints



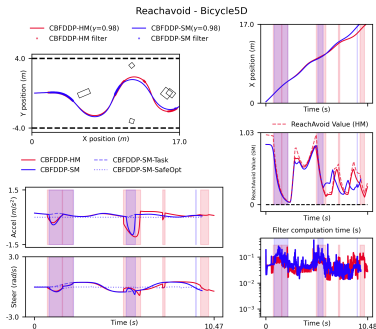
Cubic complexity with state and action dimension

CBFDDP-SM on JAX

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM $\Delta t = 0.01$, $H = 50$



CBFDDP-HM tends to get stuck with non-convex obstacle shapes

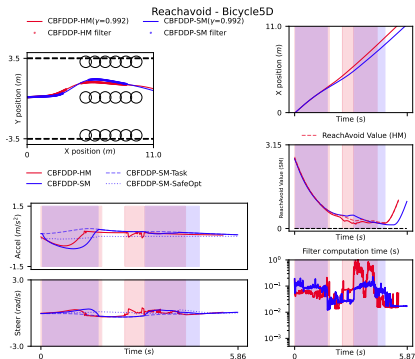


Box obstacles smoothed to give smoother safety filtering

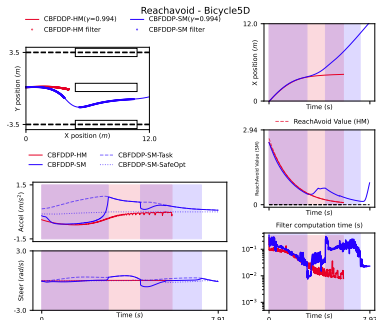
Smoothing - experimental validation

CBFDDP-SM on JAX

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM $\Delta t = 0.01$, $H = 50$



CBFDDP-HM tends to get stuck
with non-convex obstacle shapes



Box obstacles may replace array of circles

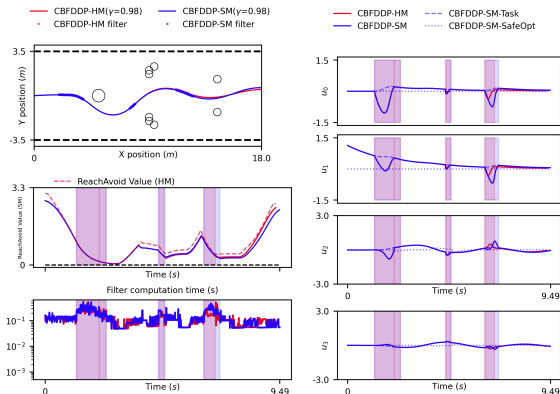
Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM $\Delta t = 0.01$, $H = 50$

65 / 67

CBFDDP-SM on 4-Wheel Bicycle model

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM $\Delta t = 0.01$, $H = 50$

Reachavoid - TwoWheel7D



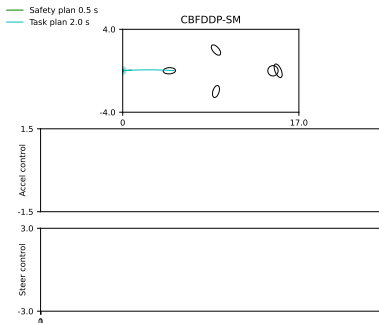
Four-wheel bicycle model with 7D states and 4D controls - Reach-Avoid CBFDDP-SM harder but works

Smoothing - experimental validation

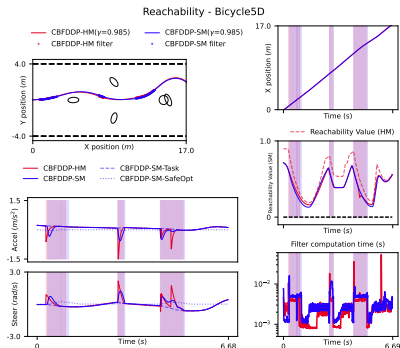
Leveraging JAX for fast computation

Specs: Macbook Pro M3 CPU with 8-core CPU and 8GB of RAM

$$\Delta t = 0.01, H = 50, \gamma = 0.985, L = 0.8, \kappa = 4.0$$



Composition of velocity, steering, road boundary, and obstacle constraints



- ▶ 10x faster than reach-avoid DDP
- ▶ CBFDDP-SM performs better overall