Guaranteeing Safe Online Machine Learning via Reachability Analysis

Jeremy Gillula

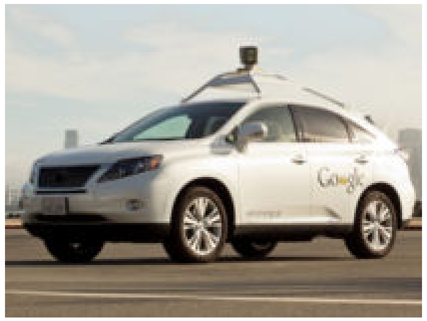
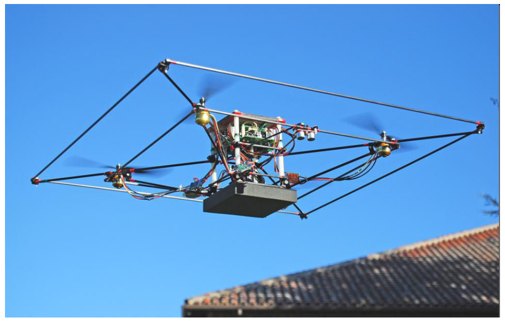
UC Berkeley
Electrical Engineering & Computer Sciences

CPS Games Presentation
Friday, August 9th, 2013

Mea Culpa

- This talk relates to cyber-physical systems…
- …but not games…

The Future of Robotics is Increasingly Complex…

- Robotic systems are expanding their "workspaces" to the everyday world…

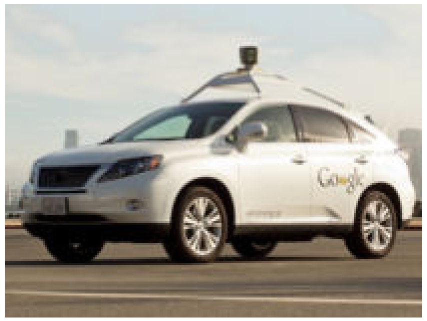
[*Images courtesy BBC, USAF, Google, AutoBlog, Intuitive Surgical, Popular Science, Boston Dynamics; http://tinyurl.com/jhg-defense-(1-8)*]

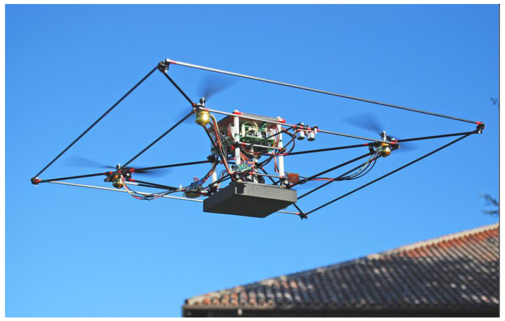
Why safety?

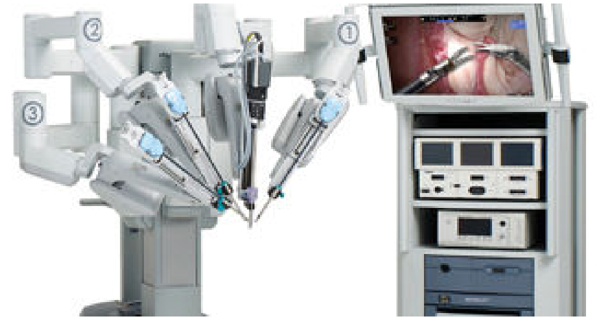
- Safety required to protect robots
  - Expensive robots
  - *"NTSB Cites Lax Safety Controls, Pilot Error in Ariz. Drone Crash"* – The Washington Post, Oct 2007
- Safety required to protect people
  - People in robot workspaces
  - *"Failed robotic surgery focus of Kitsap trial"* – The Seattle Times, May 2013

[*Pictures courtesy Robert Scoble, NTSB (http://tinyurl.com/jhg-defense-10), Cremean et al. 2006*]

The Future of Robotics is Increasingly Complex…







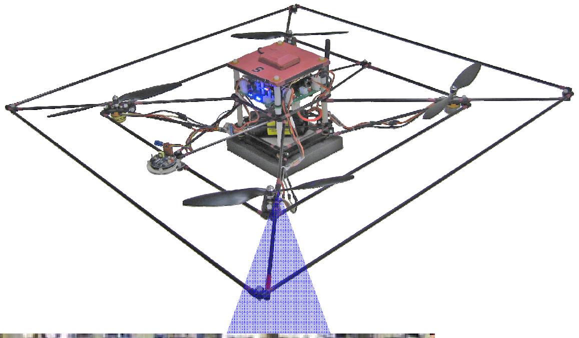
- Robotic systems are expanding their “workspaces” to the everyday world…

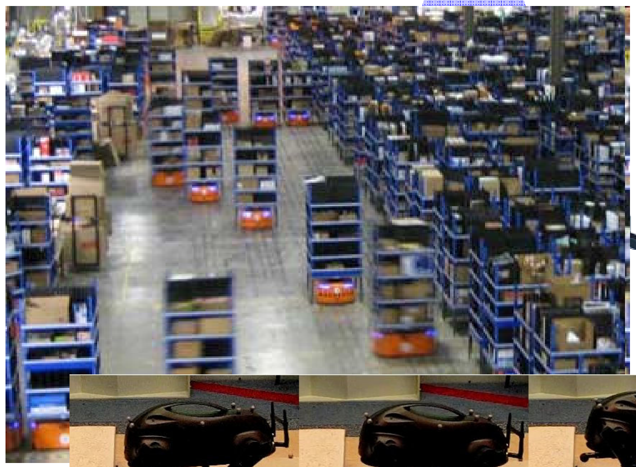
- …and are accomplishing more and more difficult tasks.

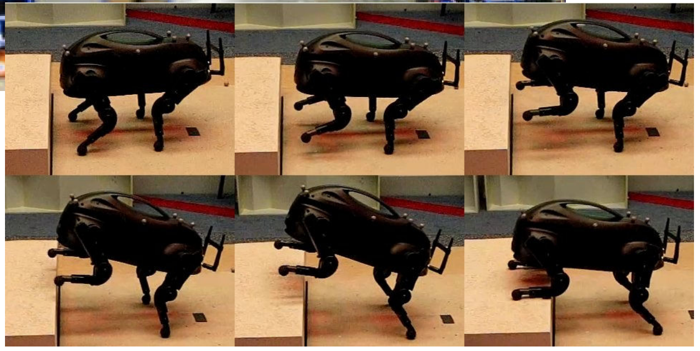
[*Images courtesy BBC, USAF, Google, AutoBlog, Intuitive Surgical, Popular Science, Boston Dynamics; http://tinyurl.com/jhg-defense-(1-8)*]

Why learning?

- Optimal control not sufficient if you don’t know the model well
  - Quadrotor aerodynamics at boundaries of flight envelope
- Situations where goal is to learn
  - Observing behavior
- Cheap robots
- Ease of design/development







[*Pictures courtesy Hoffmann 2008, CNet (http://tinyurl.com/jhg-defense-9)*]

Problem Statement

- **Problem:** Reinforcement learning can't be used in a safety critical environment!
  - Machine learning algorithms converge asymptotically
  - Some natural parameterizations can behave very poorly during

"The E^3-family of algorithms…are the state of the art RL algorithms for autonomous data collection…Unfortunately, such exploration policies do not even try to fly the helicopter well, and thus would invariably lead to crashes."

- *Abbeel et al., NIPS 19, 2006*

- **Answer:** Guaranteed Safe Online Learning via Reachability
  - Framework for wrapping controllers based on reachability analysis around machine learning algorithms to guarantee always-safe control

[1] John W Roberts, Ian R Manchester, and Russ Tedrake. *"Feedback Controller Parameterizations for Reinforcement Learning,"* in IEEE Symposium on AD-PRL, 2011.

Outline

- Introduction
- Reachability Analysis
- Guaranteed Safe Online Learning via Reachability (GSOLR)
- Extensions of GSOLR
- Sampling-Based Reachability Computations
- Conclusions

Outline

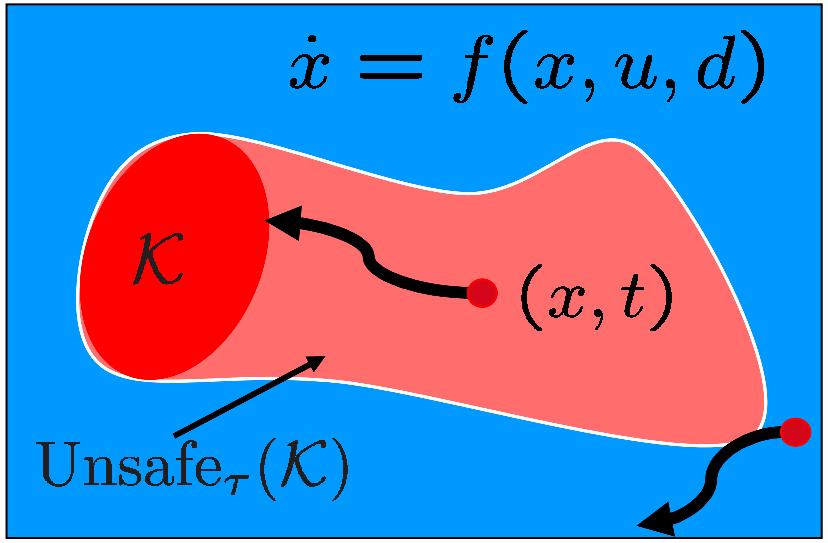
- *Introduction*

➔**Reachability Analysis**

- Guaranteed Safe Online Learning via Reachability (GSOLR)
- Extensions of GSOLR
- Sampling-Based Reachability Computations
- Conclusions

Reachability: Informal Definition

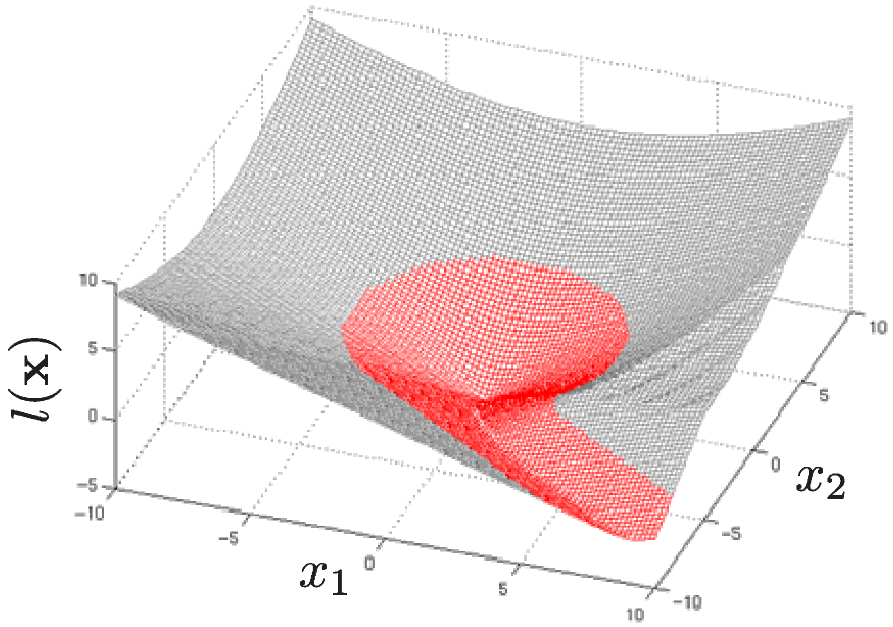
- Given:
  - A system $\dot{x} = f(x, u, d)$
  - A set of control and disturbance inputs $u \in \mathcal{U}, d \in \mathcal{D}$
  - A set of states $\mathcal{K}$
  - Some time horizon τ



- Calculate:
  - The set of states $\mathrm{Unsafe}_\tau(\mathcal{K})$ for which **a disturbance signal exists** that will **drive** the system to $\mathcal{K}$ within time horizon τ, **no matter what control** signal is chosen

Calculating Reachability: Level Sets

- Create a level set function $l(x)$ such that:
  - Boundary of keep-out set $\mathcal{K}$ is defined implicitly by $l(x) = 0$
  - $l(x)$ is negative inside region and positive outside



- Reachability as game:
  - Disturbance attempts to force system into unsafe region, control attempts to stay safe

$$J(x,t) = \max_{u \in \mathcal{U}_{[-\tau,0]}} \min_{d \in \mathcal{D}_{[-\tau,0]}} l(x(0))$$

- Solution can be found via Hamilton-Jacobi-Isaacs PDE:

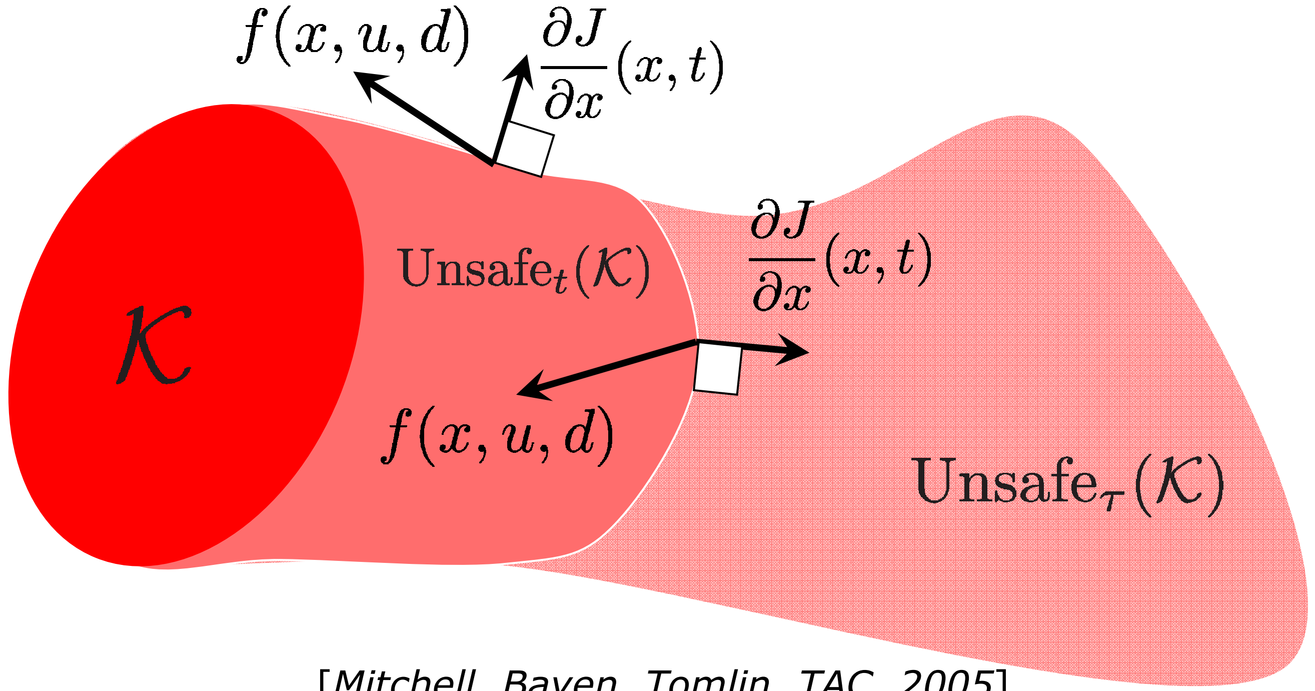
$$J(x,0) = l(x)$$

$$\frac{\partial J(x,t)}{\partial t} = -\min\{0, H^*(x, \frac{\partial J(x,t)}{\partial x})\}$$

Reachable Set Propagation

- Expanding optimal Hamiltonian gives geometric interpretation

$$-\frac{\partial J(x,t)}{\partial t} = \min\{0, \overbrace{\max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \frac{\partial J(x,t)}{\partial x} f(x,u,d)}\}$$

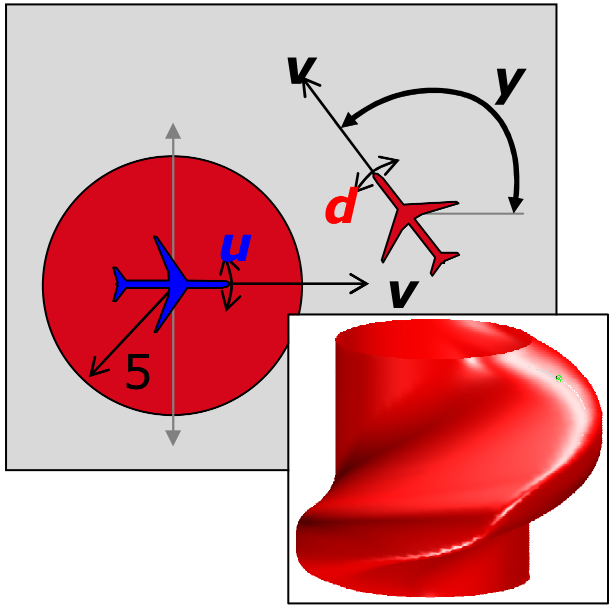


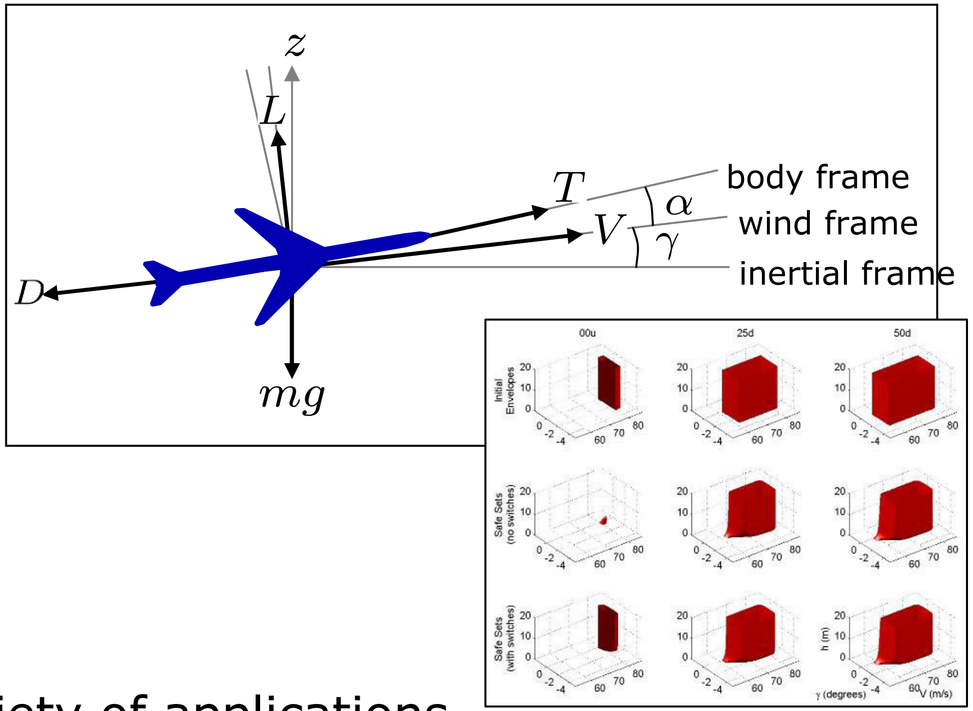
$f(x,u,d)$
$\frac{\partial J}{\partial x}(x,t)$
$\mathcal{K}$
$\text{Unsafe}_t(\mathcal{K})$
$\frac{\partial J}{\partial x}(x,t)$
$f(x,u,d)$
$\text{Unsafe}_\tau(\mathcal{K})$

[*Mitchell, Bayen, Tomlin, TAC, 2005*]

Reachability Computation

- Ian Mitchell's level set toolbox for MATLAB
  - available at: http://www.cs.ubc.ca/~mitchell/ToolboxLS/index.html





- Used for a variety of applications
- Handles 3 dimensions easily, up to 5 tractably
- Library of level set functions

Outline

- *Introduction*
- *Reachability Analysis*

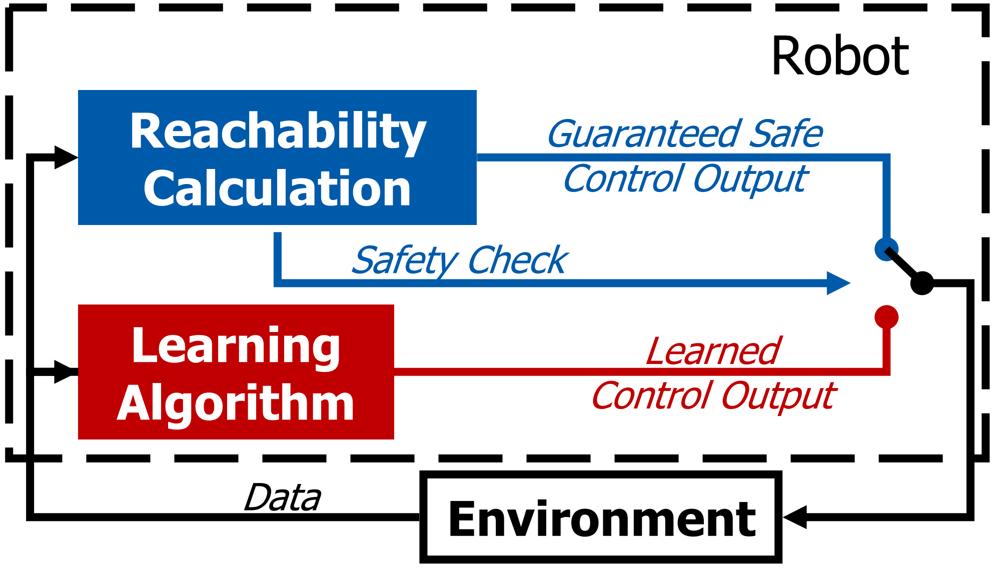
➔ **Guaranteed Safe Online Learning via Reachability (GSOLR)**

- Extensions of GSOLR
- Sampling-Based Reachability Computations
- Conclusions and Future Work

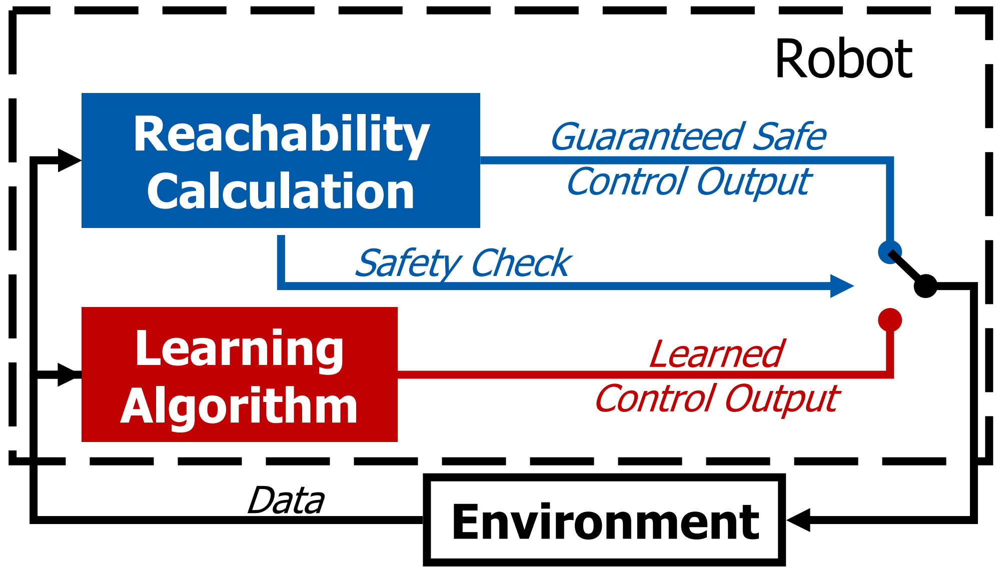
Guaranteed Safe Online Learning via Reachability

- Key insight:
  - Only need to use optimal control on border of unsafe set
  - Can use any other control at all other times

- GSOLR Algorithm:
  - When the system state is near the boundary of the unsafe set $\mathrm{Unsafe}_{\tau}(\mathcal{K})$ perform the optimal control action as determined by the optimal Hamiltonian
  - Otherwise, perform whatever control is dictated by the given machine learning algorithm.



GSOLR Benefits

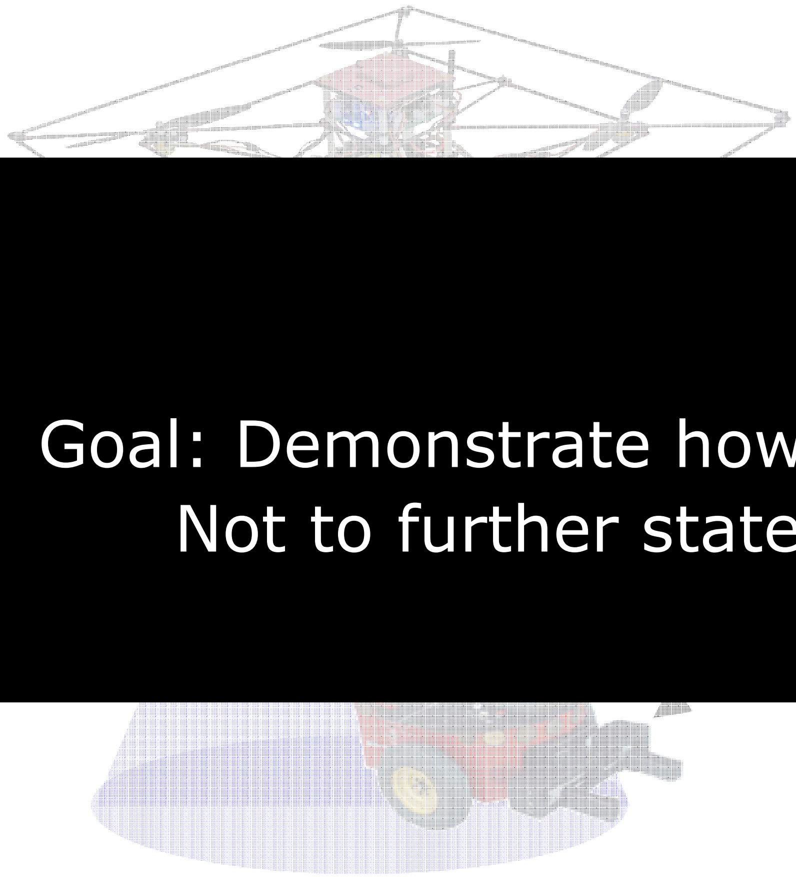


- Completely flexible w.r.t. machine learning algorithm
- Least restrictive control w.r.t. maintaining safety
- Works for nonlinear and hybrid systems

Similar Work

- Aswani et al., 2011: "Provably Safe and Robust Learning-Based Model Predictive Control"
  - Uses a Model Predictive Control (MPC) framework
  - *A priori* system model is used to verify constraints are met (safety)
  - Learned model is used to evaluate the cost function (performance)
- Perkins and Barto, 2002: "Lyapunov Design for Safe Reinforcement Learning"
  - Uses Lyapunov techniques to guarantee safety when switching between a finite number of low-level controllers
- Ng and Kim, 2005: "Stable adaptive control with online learning"
  - Algorithm monitors controllers suggested by a learning algorithm, rejecting unstable ones
  - Restricted to linear systems
- GSOLR
  - More flexible, less restrictive

Example: Safely Learning A Bounded System



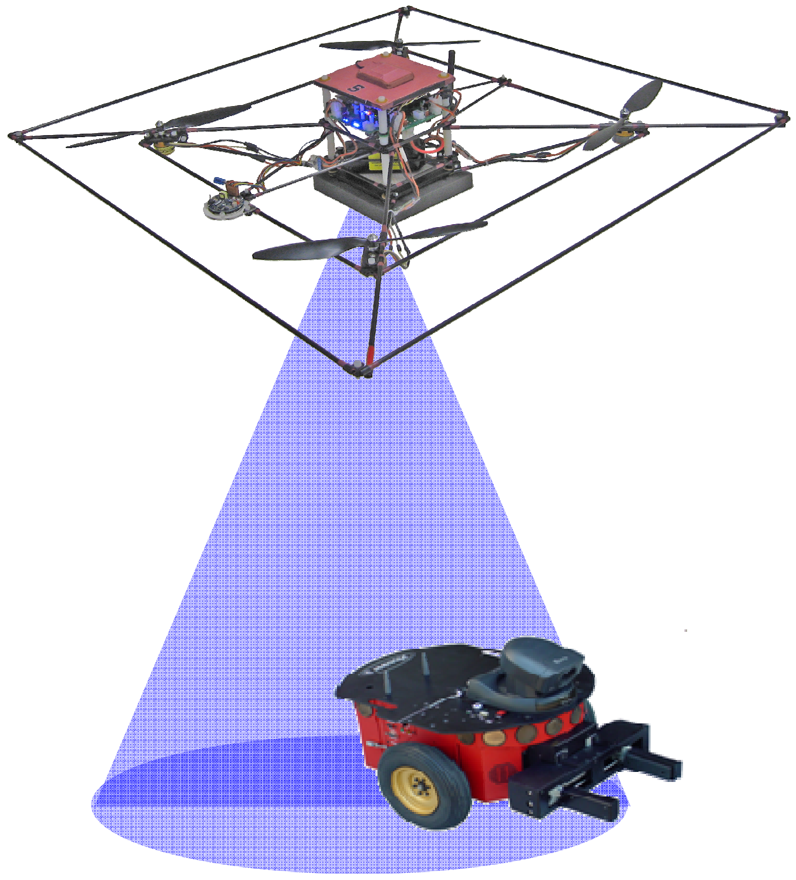
- Learning unknown dynamics of a target vehicle via observation

  – "wins" if it escapes the Observer's field of view

Aside

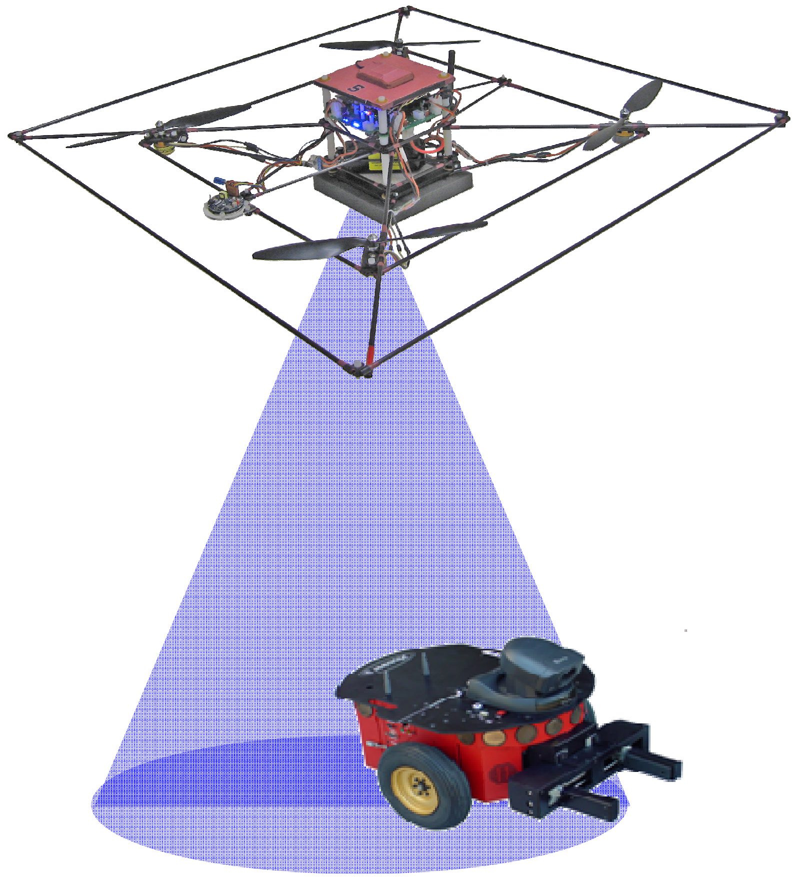
Goal: Demonstrate how GSOLR can be used in practice,
Not to further state of the art in target tracking

Problem Setup



- Observer dynamics:
  - $\ddot{x}_o = u_{o_x},\ \ddot{y}_o = u_{o_y},\ \dot{z}_o = u_{o_z}$
- Target dynamics:
  - $\ddot{x}_t = u_{t_x}(x_t, \dot{x}_t, t) \in \mathcal{U}_{t_x}$
  - $\ddot{y}_t = u_{t_y}(y_t, \dot{y}_t, t) \in \mathcal{U}_{t_y}$
- Observer measurement model
  - Field of view θ
  - $\tilde{\mathbf{x}}_\mathbf{t} = \begin{cases} \mathbf{x}_\mathbf{t} + \eta & \text{if } x_t \in \mathcal{V} \\ \emptyset & \text{otherwise} \end{cases}$
  - Noise: $\eta \sim \mathcal{N}(0, \sigma), \sigma \propto z_o$
  - $\mathcal{V} = \{\mathbf{x}_\mathbf{t} : |x_o - x_t| \leq z_o \tan\theta$ and $|y_o - y_t| \leq z_o \tan\theta\}$

Problem Statement



- Problem statement
  1) Minimize prediction error:
     $$e = \|\hat{x}_t - x_t\|$$
  2) Maintain target in view: $x_t \in \mathcal{V}$

- For (1) use machine learning

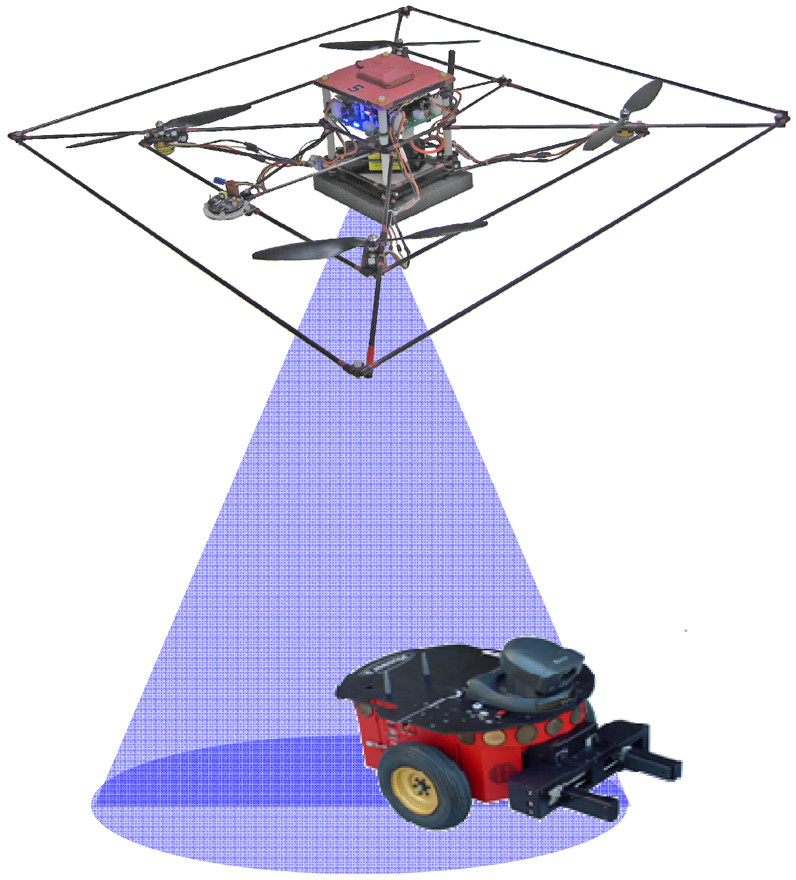
Machine Learning Formulation

- Assuming target motion can be described by linear combination of *N* basis functions $\phi_i(\mathbf{x_t}, t)$
  - $\ddot{x}_t = \sum_{i=0}^{N} W_i \phi_i(\mathbf{x_t}, t) = [W_1 \ldots W_N] \begin{bmatrix} \phi_1 \\ \ldots \\ \phi_N \end{bmatrix}$
- Learn W_i via linear regression
- Optimal information-theoretic control: reduce z_o
  - Conditional entropy:

$$H(\mathbf{x_t}|\mathbf{x_o}) = \frac{1}{2} \log \left(2\pi e \left| \Sigma^{-1} + (\sigma^{-2} z_o^{-2}) \right|^{-1} \right)$$

  - with $p(\mathbf{x_t})$ approximated as Gaussian with covariance Σ

Problem Statement



- Problem statement
  1) Minimize prediction error:
     $e = \|\hat{x}_t - x_t\|$
  2) Maintain target in view:
     $x_t \in \mathcal{V}$

- For (1) use machine learning

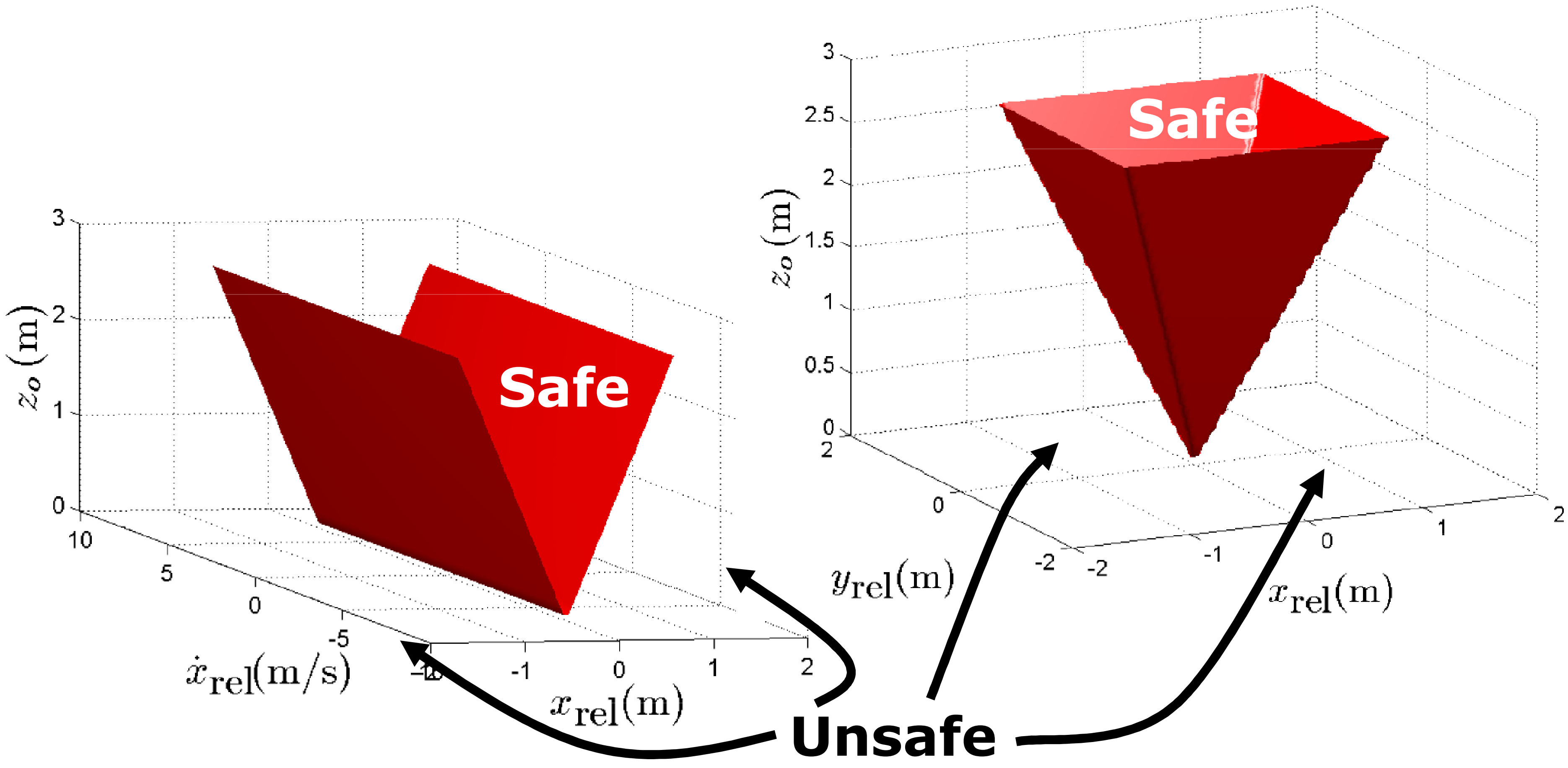
- For (2) use reachability analysis

Reachability Analysis Formulation

- Define augmented system
  - $\mathbf{x} = \left[\begin{array}{ccccc} x_o - x_t & y_o - y_t & \dot{x}_o - \dot{x}_t & \dot{y}_o - \dot{y}_t & z_o \end{array} \right]^\top$
  - $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{d}) = \begin{bmatrix} \dot{x}_o - \dot{x}_t \\ \dot{y}_o - \dot{y}_t \\ u_{o_x} - u_{t_x} \\ u_{o_y} - u_{t_y} \\ u_{o_z} \end{bmatrix} = \begin{bmatrix} \dot{x}_{\text{rel}} \\ \dot{y}_{\text{rel}} \\ u_{o_x} - u_{t_x} \\ u_{o_y} - u_{t_y} \\ u_{o_z} \end{bmatrix}$
  - $\mathbf{u} = \left[\begin{array}{ccc} u_{o_x} & u_{o_y} & u_{o_z} \end{array} \right]^\top \in \mathcal{U}_o$
  - $\mathbf{d} = \left[\begin{array}{cc} u_{t_x} & u_{t_y} \end{array} \right]^\top \in \mathcal{U}_t$

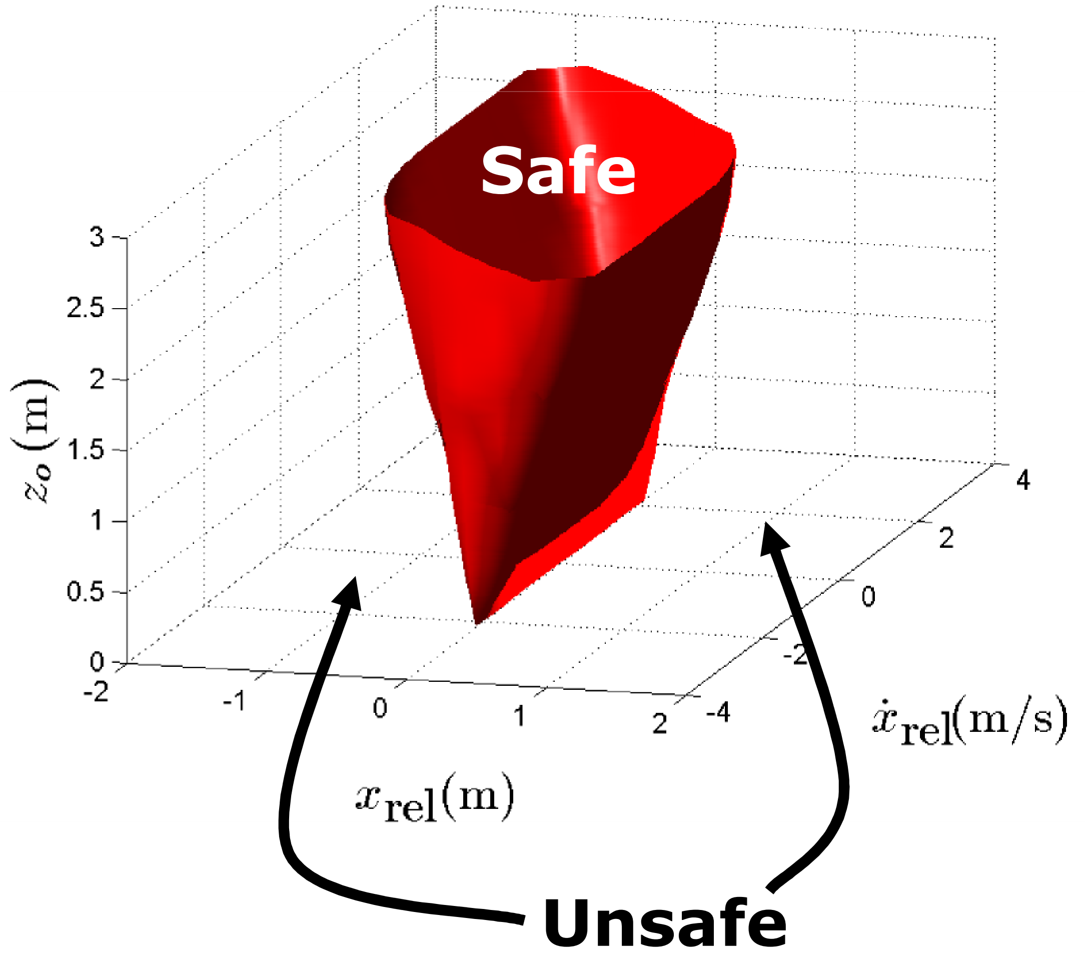
Reachability Analysis Formulation (cont'd)

- Initial keepout set: $\mathcal{K} = \{\mathbf{x} : \mathbf{x_t} \notin \mathcal{V}\}$

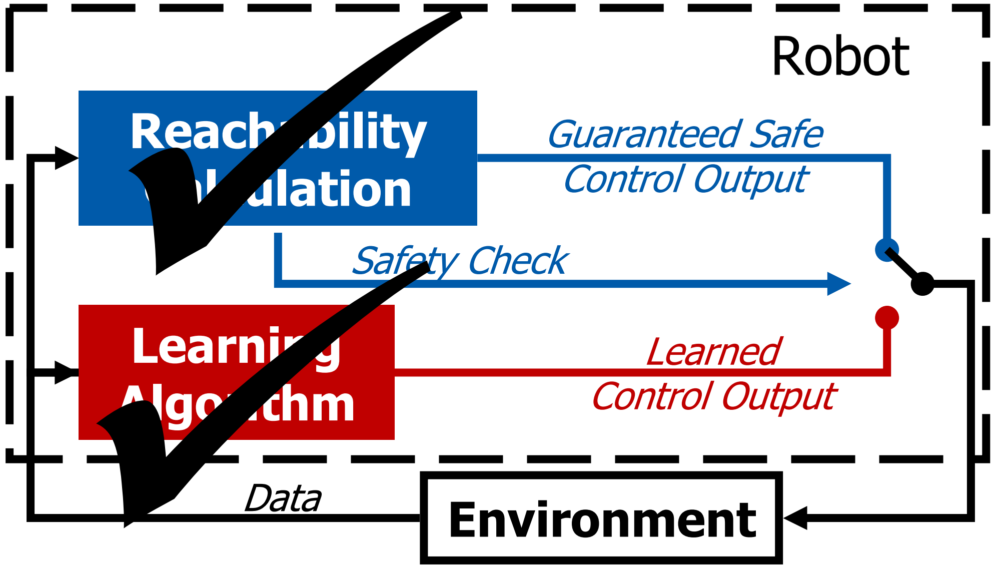


Reachability Analysis Formulation (cont'd)

- $\tau = 1$ unsafe set: $\mathrm{Unsafe}_\tau(\mathcal{K})$



Apply GSOLR!



- **Philosophical Interlude**
  - Safety only guaranteed with respect to system model, bounds
  - Disturbance term can capture uncertainties in system model

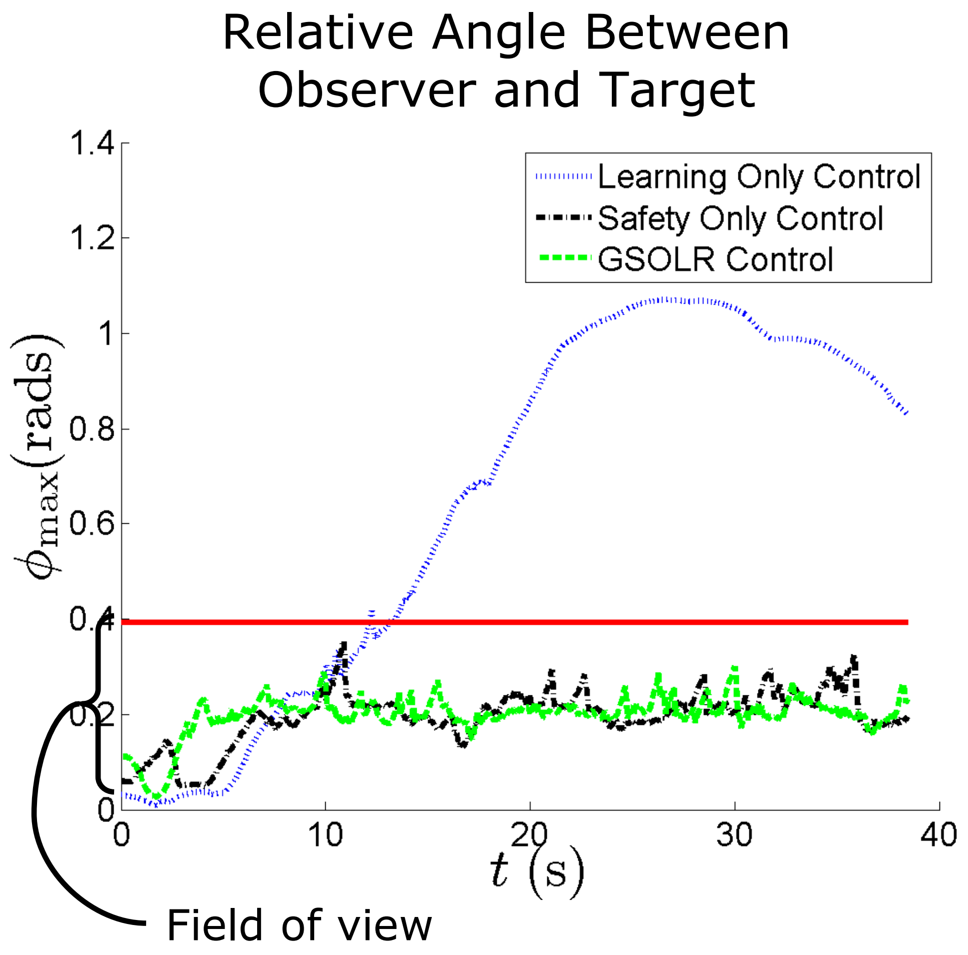
Experimental Setup: Three Control Schemes

1. Learning only
   - After each measurement, perform machine learning
   - Horizontal control: use learned model to predict next location, use PD to track to it
   - Vertical control: use information-theoretic control

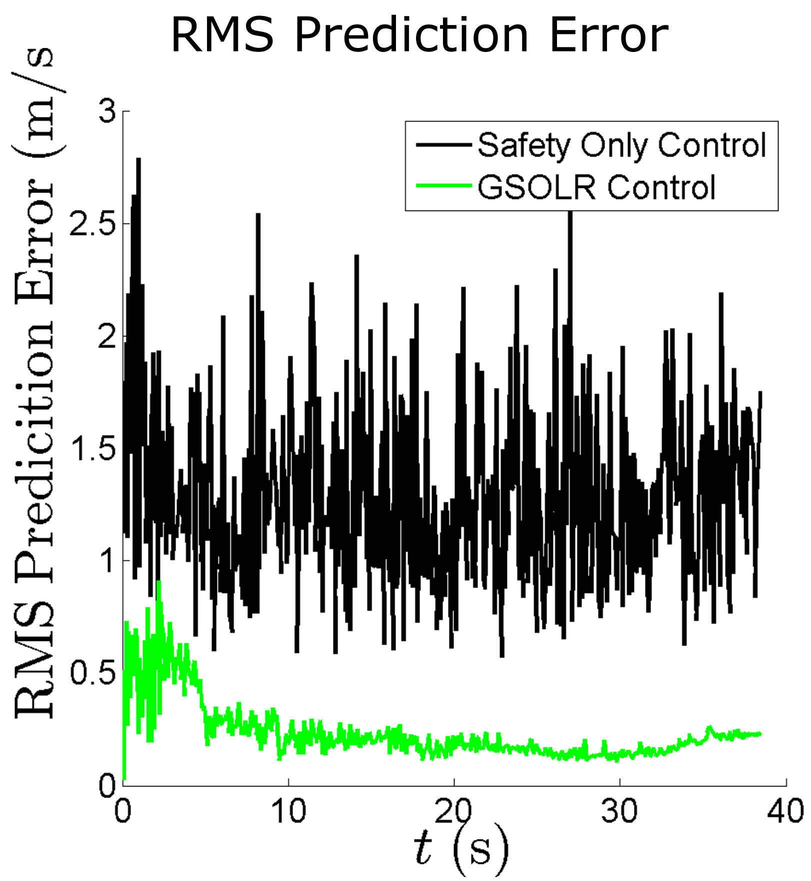
2. Safety only
   - After each measurement, check for safety
     - ☐ Use optimal control if unsafe, otherwise:
     - ☐ Horizontal control: use PD to track to last known position
     - ☐ Vertical control: use information-theoretic control

3. GSOLR

Results



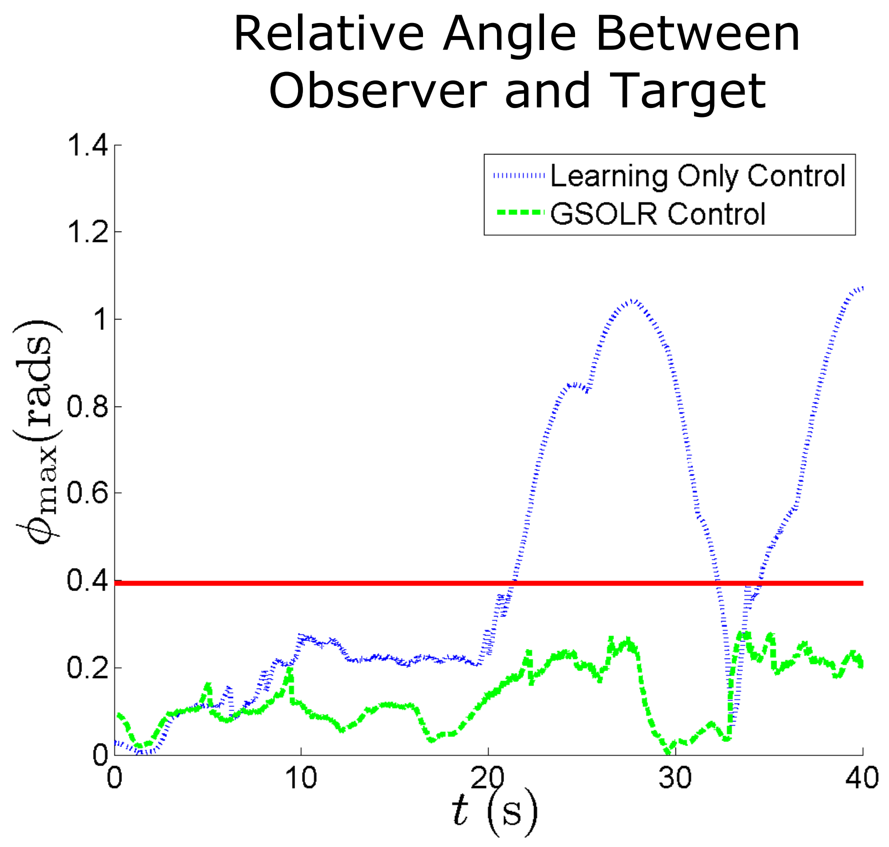
Relative Angle Between Observer and Target
Learning Only Control
Safety Only Control
GSOLR Control
$\phi_{\max}$(rads)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
t (s)
Field of view

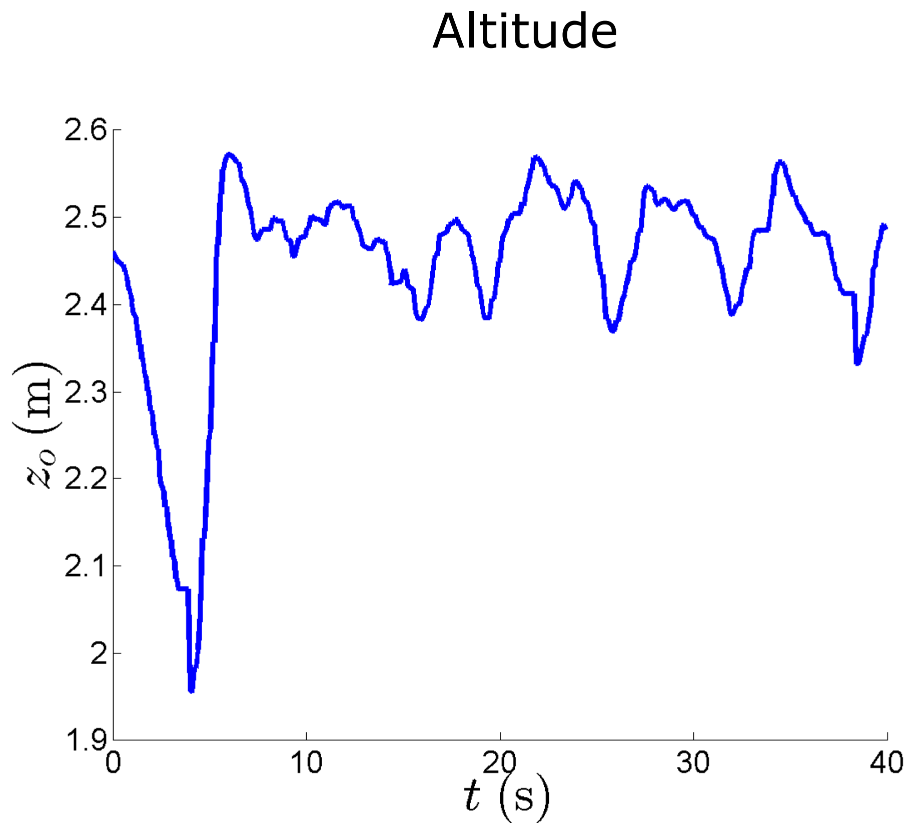


RMS Prediction Error
Safety Only Control
GSOLR Control
RMS Predicition Error (m/s)
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
t (s)

Results: Improper Learning

Results: Improper Learning



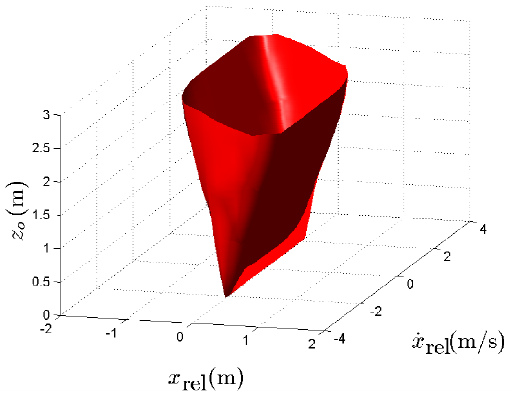


Limitations of GSOLR

- Fixed worst-case over state space

- No incorporation of new information in safety

- HJI level-set reachability limited to only five dimensions



[*Image credit:http://tinyurl.com/7v322er, 6puafjd, 7vs8lh6, 83jp67z, 7wzmsrg, bncvfz8, 6p5a94r*]

Outline

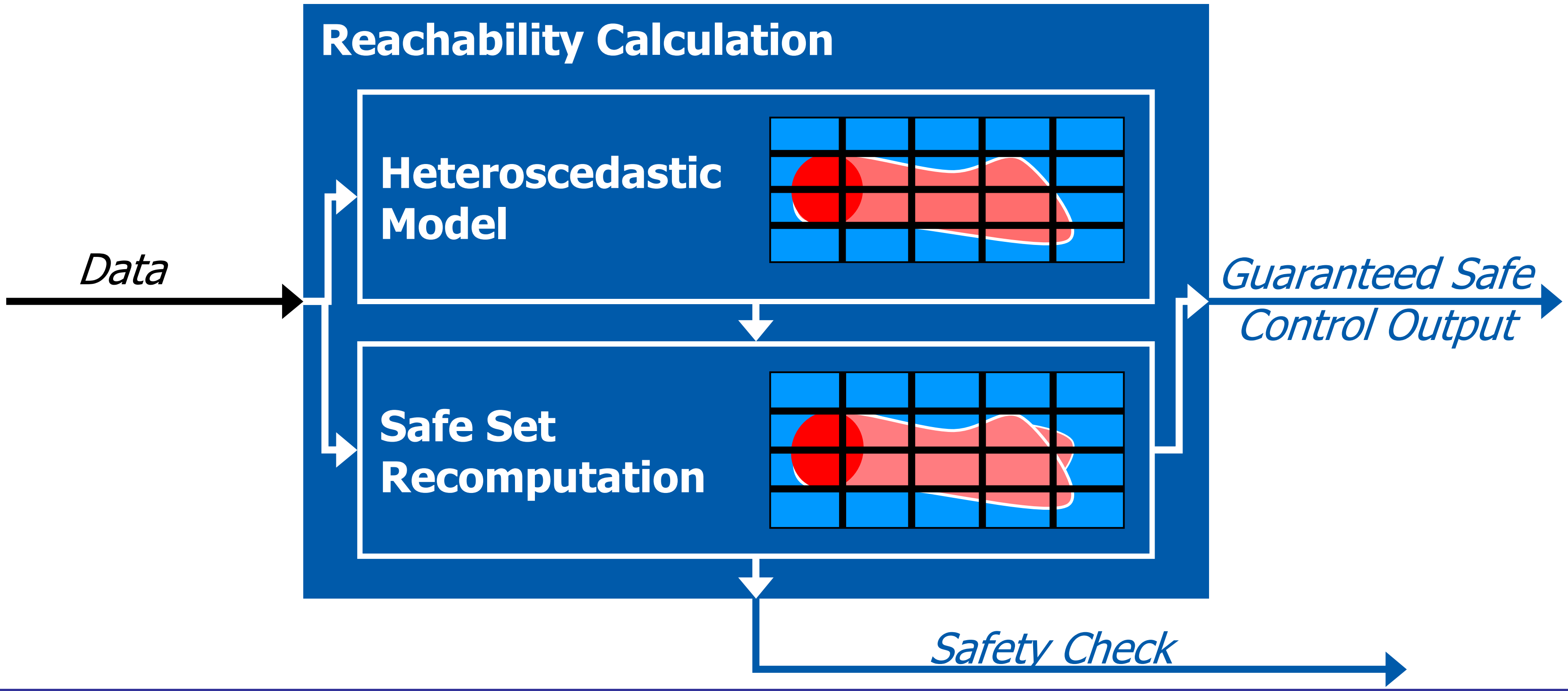
- *Introduction*
- *Reachability Analysis*
- *Guaranteed Safe Online Learning via Reachability (GSOLR)*

➔**Extensions of GSOLR**

- Sampling-Based Reachability Computations
- Conclusions and Future Work

Iterated Guaranteed Safe Online Learning via Reachability (IGSOLR)

- Model disturbances as heteroscedastic
- Start with conservative estimate, and learn disturbances online



Philosophical Interlude, Part II

- How can you guarantee safety if you're learning the disturbances?
  - Learning algorithm converges asymptotically, so update the disturbance model only when a large amount of data has been collected
- How is it worst-case if you're recomputing the safe sets?!
  - It's worst-case based on the known data: recomputation simply reduces conservativeness
- ***Goal of IGSOLR***
  - Base safety guarantees on actual data as the system runs (as opposed to *a priori* estimates) so as to maintain safety but not be *too* conservative during runtime

Example Application: Quadrotor Altitude Control

- Problem: Classical linear control insufficient
  - Solution: Reinforcement learning!
- Problem: Reinforcement learning dangerous!
  - Solution: GSOLR!
- Problem: Disturbance varies with altitude and velocity!
  - Solution: IGSOLR!

Problem Setup

- Continuous altitude dynamics:

$$\ddot{x} = g + \gamma u + d$$

Thrust Multiplier

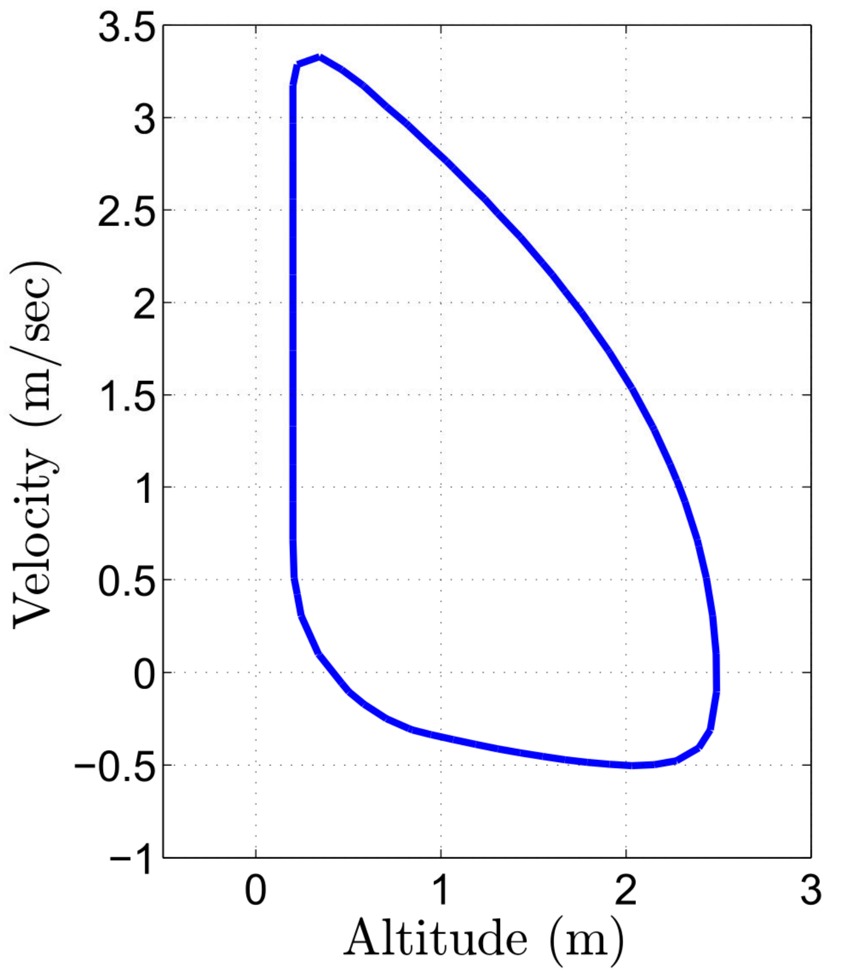
Gravity

Altitude

- Discretized dynamics:

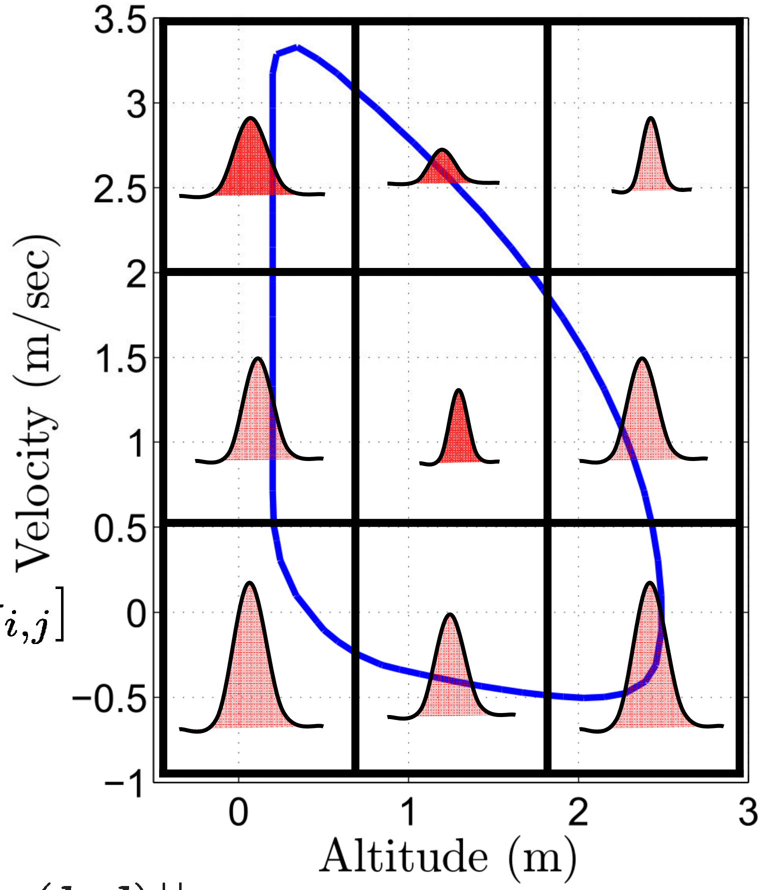
$$(\dot{x}_{i+1} - \dot{x}_i)/\Delta t = g_{\Delta t} + \gamma_{\Delta t} u_i + d$$

- Initial conservative reachable set:



Disturbance Learning Formulation

- Grid up state space
- Take measurements, calculate residuals
- Model as truncated Gaussian per grid cell
  - $\mathcal{D}(\mathbf{x} \in c(i,j)) = [-3\sigma_{i,j} + \mu_{i,j}, \mu_{i,j} + 3\sigma_{i,j}]$
- Propagate based on distance to "empty" cells
  - $\mathcal{D}(\mathbf{x} \in c(k,l)) = \delta\mathcal{D}(c(i,j))||c(i,j) - c(k,l)||_2$
- Essentially "discrete" Gaussian process model



Reinforcement Learning Formulation (PGSD)

- Use control law linear in state features
  - $u_t = \theta^\top \phi(\mathbf{x}_t)$

Design Choices

- Cost function over horizon l
  - $J(\mathbf{x}_0, \theta) = \sum_{t=1}^{l} C(\mathbf{x}_t, u_t)$
  - $C(\mathbf{x}_t, u_t) = (\mathbf{x}_t - \mathbf{x}_t^*)^\top Q(\mathbf{x}_t - \mathbf{x}_t^*) + u_t^\top R u_t$

- Standard policy gradient:
  - $\theta \leftarrow \theta - \alpha \nabla_\theta J(\mathbf{x}_0, \theta)$

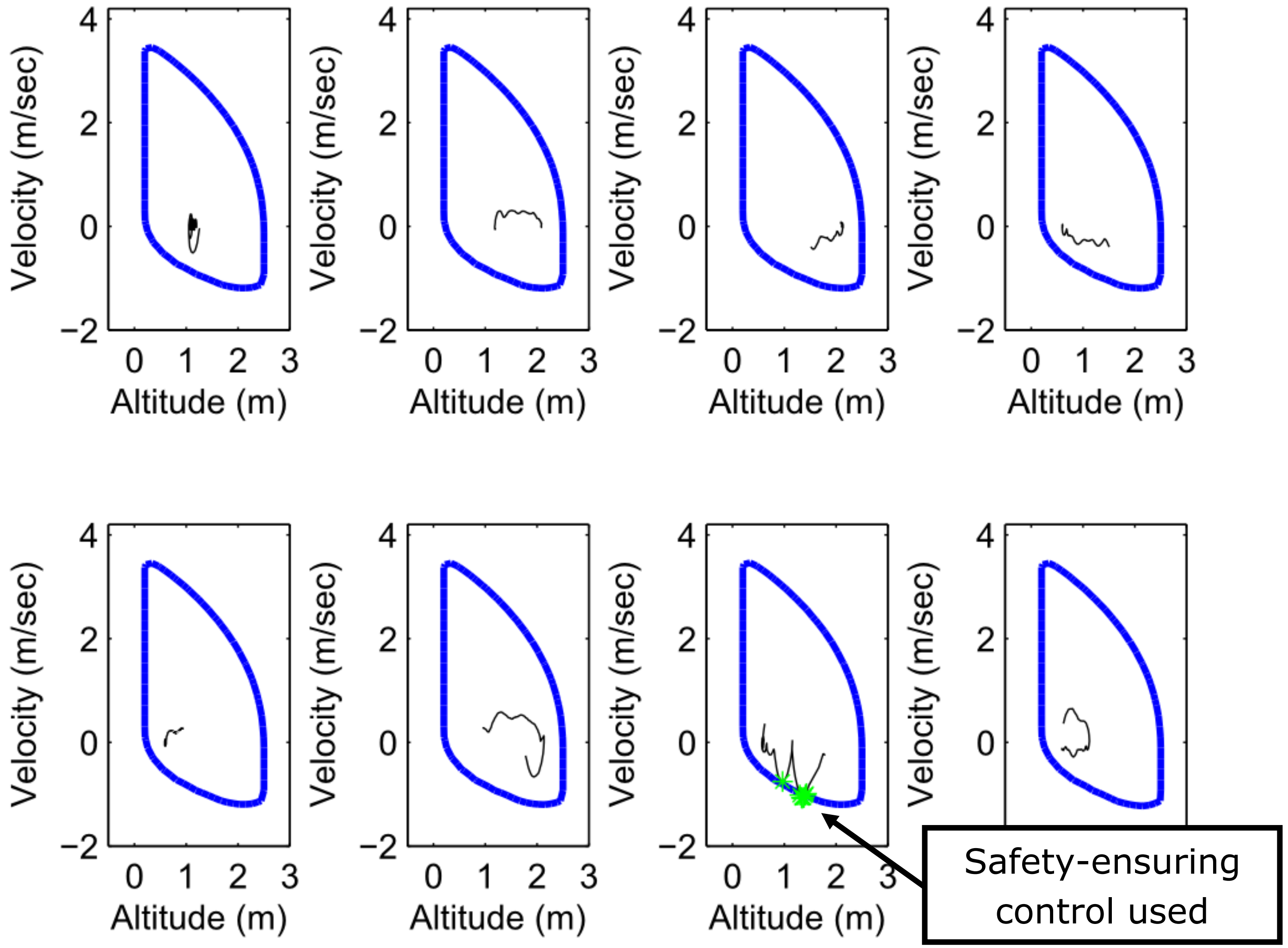
- PGSD approximates $\nabla_\theta J(\mathbf{x}_0, \theta)$ with a signed derivative matrix *S*
  - All entries ±1 or 0
  - Only one non-zero entry in each row

Example Application: Quadrotor Altitude Control

- Implemented on quadrotor
- Ran IGSOLR on training trajectory
- Compared to test trajectory

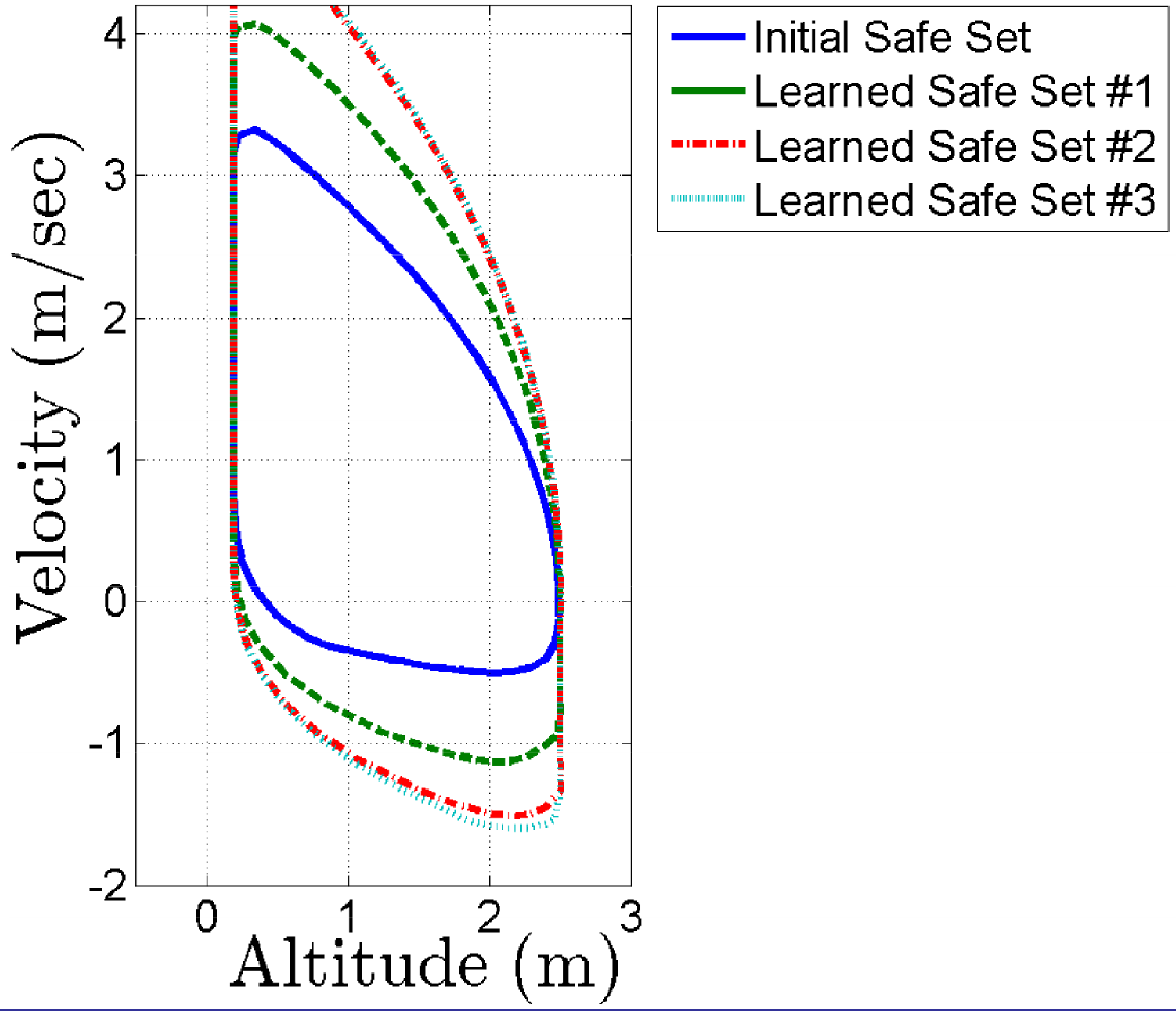
Results: Safety

State Traces and Reachable Sets



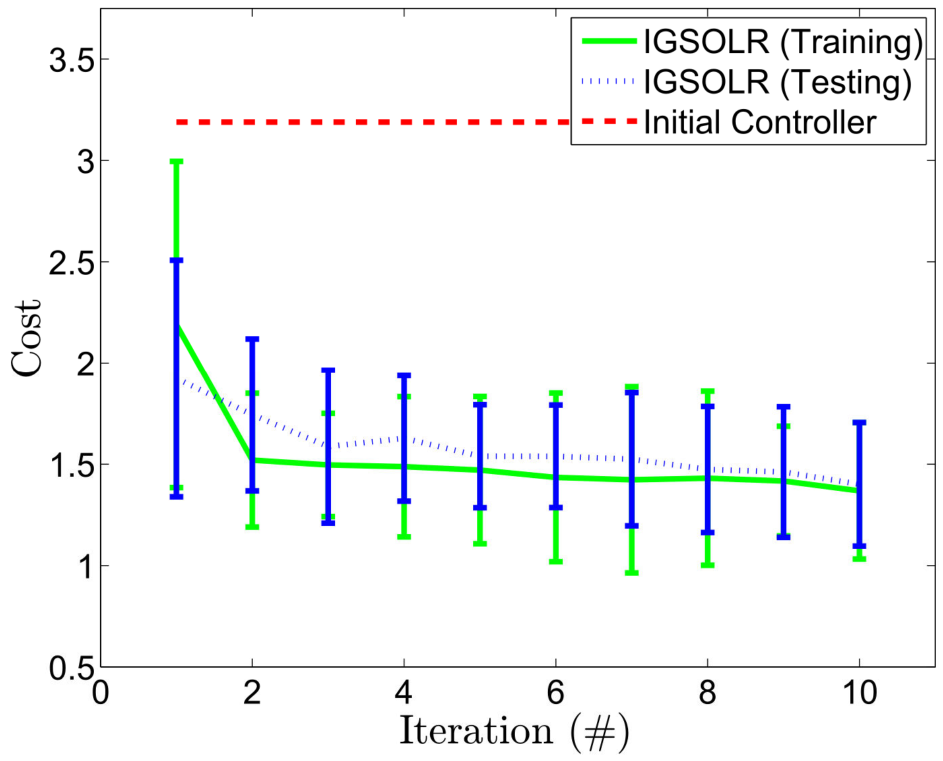
Safety-ensuring control used

Results: Learning Disturbances Online

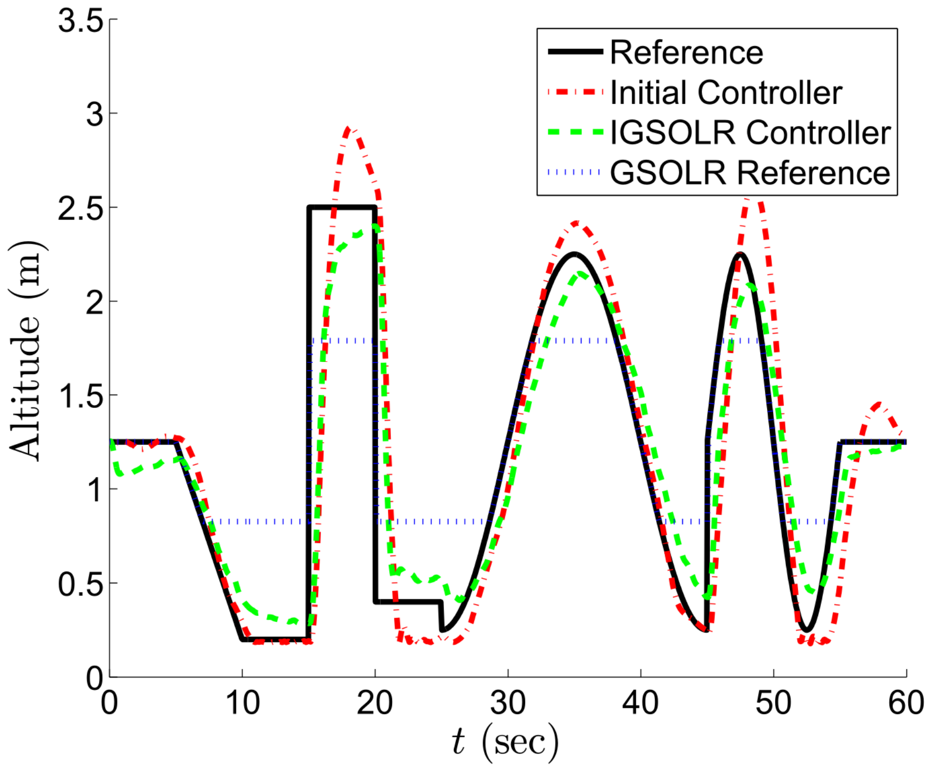


Results: Performance

Average Cost Over Testing Trajectory (w/95% confidence intervals)



Sample State Trace Over Testing Trajectory



Outline

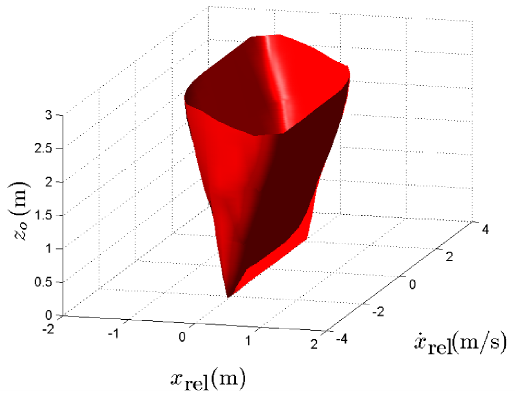
- *Introduction*
- *Reachability Analysis*
- *Guaranteed Safe Online Learning via Reachability (GSOLR)*
- *Extensions of GSOLR*

→**Sampling-Based Reachability Computations**

- Conclusions and Future Work

Limitations of GSOLR

- *Fixed worst-case over state space*
- *No incorporation of new information in safety*
- **HJI level-set reachability limited to only five dimensions**



[*Image credit:http://tinyurl.com/7v322er, 6puafjd, 7vs8lh6, 83jp67z, 7wzmsrg, bncvfz8, 6p5a94r*]

Limitations in Computing Reachable Sets

- Level-Set (Eulerian) Methods; Recursive Methods
  - Computations are grid-based; memory use exponential in number of state dimensions
- Decomposition Techniques
  - Increases conservatism
- Ellipsoidal Approximations
  - Too conservative
- **Our Contribution: Sampling-based algorithm for computing the viability kernel for linear sampled-data systems**

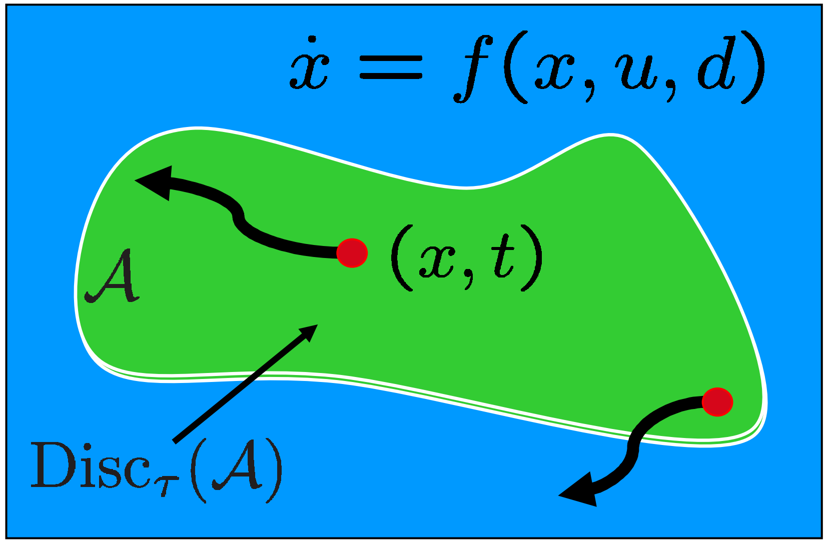
What is a…?

- …sampled-data system?
  - Continuous system with piecewise constant controls and measurements
  - $\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t))$
  - $\mathbf{x}_k = \mathbf{x}(t_k = k\delta)$
  - $\mathscr{U}_{\mathbb{T}}^{\mathrm{pw}} := \{\mathbf{u} \colon \mathbb{T} \to \mathbb{R}^m \mid \mathbf{u}(t_k) \in \mathcal{U}\ \forall k,\ \mathbf{u}(t) = \mathbf{u}(t_k)\ \forall t \in [t_k, t_{k+1})\}$
  - $\mathbb{T} := [0, \tau]$

- …discriminating kernel?

Discriminating Kernel: Formal Definition

- Given:
  - A system $\dot{x} = f(x, u, d)$
  - A set of control and disturbance inputs $u \in \mathcal{U}, d \in \mathcal{D}$
  - A set of states $\mathcal{A} = \mathcal{K}^c$
  - Some time horizon τ



- Calculate:
  - The set of states $\mathrm{Disc}_\tau(\mathcal{A})$ for which **a control signal exists** that will **keep** the system in $\mathcal{A}$ for time horizon τ, **no matter what disturbance** signal is chosen
  - If $\mathcal{A} = \mathcal{K}^c$ then $(\mathrm{Disc}_\tau(\mathcal{A}))^c = \mathrm{Unsafe}_\tau(\mathcal{K})$

What is a…?

- …sampled-data system?
  - Continuous system with piecewise constant controls and measurements
  - $\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t))$
  - $\mathbf{x}_k = \mathbf{x}(t_k = k\delta)$
  - $\mathscr{U}_{\mathbb{T}}^{\text{pw}} := \{\mathbf{u} \colon \mathbb{T} \to \mathbb{R}^m \mid \mathbf{u}(t_k) \in \mathcal{U}\ \forall k,\ \mathbf{u}(t) = \mathbf{u}(t_k)\ \forall t \in [t_k, t_{k+1})\}$
  - $\mathbb{T} := [0, \tau]$

- …~~discriminating~~ viability kernel?
  - Discriminating kernel, but no disturbances
  - States for which there exists a control input that will keep the trajectory in $\mathcal{A}$ for the entire time horizon
  - $\text{Viab}_{\mathbb{T}}^{\text{sd}}(\mathcal{A}) := \left\{\mathbf{x}_0 \in \mathcal{A} \mid \exists \mathbf{u}(\cdot) \in \mathscr{U}_{\mathbb{T}}^{\text{pw}}, \forall t \in \mathbb{T},\ \mathbf{x}_{\mathbf{x}_0}^{\mathbf{u}}(t) \in \mathcal{A}\right\}$

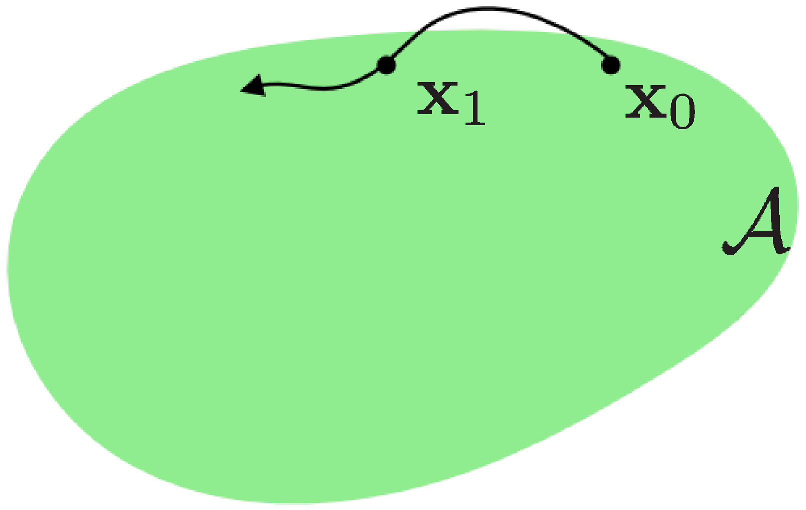
Assumptions

- Compact, convex constraints $\mathcal{A}, \mathcal{U}$
- LTI system: $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$

Algorithm Overview: Point Feasibility

- How to determine if $v_i \in \mathrm{Viab}^{\mathrm{sd}}_{\mathbb{T}}(\mathcal{A})$?
- Let $\mathbf{x}_0 := v_i$
- Use forward simulation to verify $\exists \mathbf{u}(\cdot) \in \mathscr{U}^{\mathrm{pw}}_{\mathbb{T}}$ such that $\mathbf{x}^{\mathbf{u}}_{\mathbf{x}_0}(t) \in \mathcal{A}\ \forall t \in \mathbb{T}$

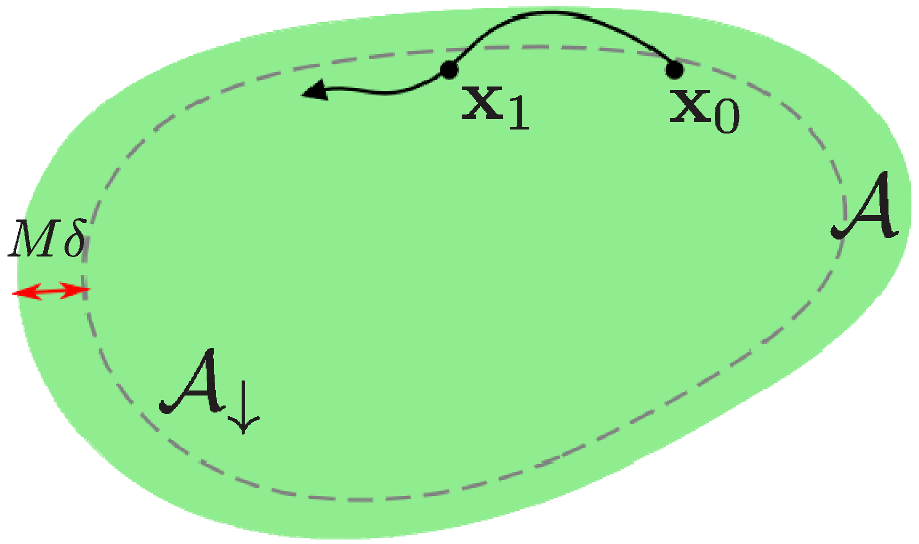
- Instead verify $\mathbf{x}^{\mathbf{u}}_{\mathbf{x}_0}(t_k) \in \mathcal{A}\ \forall k \in [0, \ldots, N_\delta]$



Turning Continuous Verification into Pointwise Verification

- Let $M = \sup_{\mathbf{x}\in\mathcal{A},\mathbf{u}\in\mathcal{U}} \|f(\mathbf{x},\mathbf{u})\|_\infty$
- Then let $\mathcal{A}_\downarrow(M,\delta) := \{x \in \mathcal{A} \mid \mathrm{dist}_{p_1}(x,\mathcal{A}^c) \geq M\delta\} = \mathcal{A} \ominus \mathcal{B}^n_\infty(0, M\delta)$
- Then if $\exists \mathbf{u}(\cdot) \in \mathscr{U}^{\mathrm{pw}}_{\mathbb{T}}$ such that $\mathbf{x}^{\mathbf{u}}_{\mathbf{x}_0}(t_k) \in \mathcal{A}_\downarrow$ $\forall k \in [0,\ldots,N_\delta]$

➔Then $\mathbf{x}^{\mathbf{u}}_{\mathbf{x}_0}(t) \in \mathcal{A}$ $\forall t \in (t_k, t_{k+1})$



Turning Continuous Pointwise Verification into Discrete Pointwise Verification

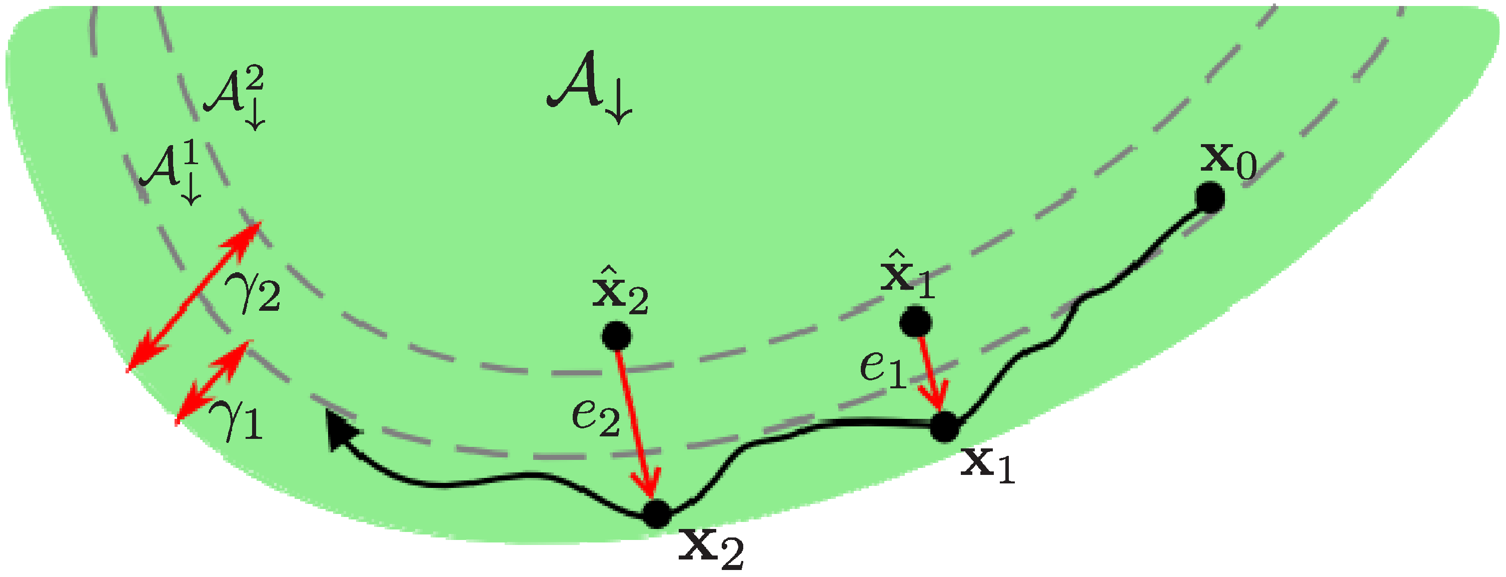
- Problem: can't verify $\mathbf{x}^{\mathbf{u}}_{\mathbf{x}_0}(t_k) \in \mathcal{A}_{\downarrow}$ exactly for

$$\mathbf{x}_{k+1} = e^{A\delta}\mathbf{x}_k + \left(\int_0^{\delta} e^{A\lambda} d\lambda \cdot B \right) \mathbf{u}_k$$

- Need to account for discretization error:

$$e^{A\delta} = \sum_{i=0}^{\infty} \frac{(A\delta)^i}{i!} = \underbrace{\sum_{i=0}^{\zeta} \frac{(A\delta)^i}{i!}}_{A_\delta} + \underbrace{\sum_{i=\zeta+1}^{\infty} \frac{(A\delta)^i}{i!}}_{E_\delta}$$

Turning Continuous Pointwise Verification into Discrete Pointwise Verification (cont'd)



- $\mathbf{x}_k = \underbrace{A_\delta \mathbf{x}_{k-1} + \left(\int_0^\delta A_\lambda d\lambda B\right)\mathbf{u}_{k-1}}_{\hat{\mathbf{x}}_k} + \underbrace{E_\delta \mathbf{x}_{k-1} + \left(\int_0^\delta E_\lambda d\lambda \cdot B\right)\mathbf{u}_{k}}_{e_k}$

- $\exists \gamma_k \geq 0 : \|e_k\|_p \leq \gamma_k$

$\mathcal{A}_\downarrow^k(A,\delta,\gamma_k) := \mathcal{A}_\downarrow(A,\delta) \ominus \mathcal{B}_p^n(0,\gamma_k)$

➔ Then $\hat{\mathbf{x}}_k \in \mathcal{A}_\downarrow^k(A,\delta,\gamma_k) \Longrightarrow \mathbf{x}_k \in \mathcal{A}_\downarrow(A,\delta)$

Discrete Pointwise Verification

- Construct prediction equation:

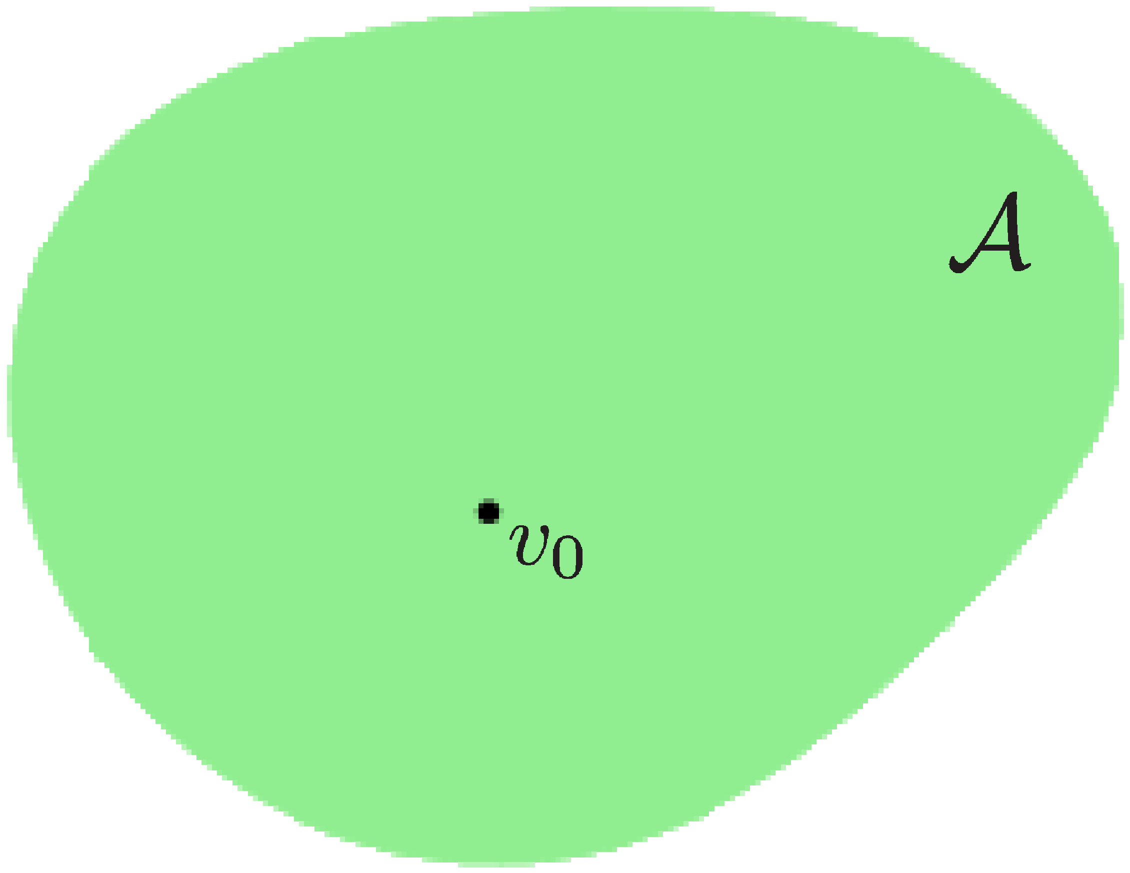
$$\begin{bmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \\ \vdots \\ \hat{\mathbf{x}}_{N_\delta} \end{bmatrix} = \underbrace{\begin{bmatrix} I \\ A_\delta \\ A_\delta^2 \\ \vdots \\ A_\delta^{N_\delta} \end{bmatrix}}_{G} \mathbf{x}_0 + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B_\delta & 0 & \dots & 0 \\ A_\delta B_\delta & B_\delta & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ A_\delta^{N_\delta - 1} B_\delta & A_\delta^{N_\delta - 2} B_\delta & \dots & B_\delta \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_\delta - 1} \end{bmatrix}}_{\mathbf{u}}$$

- Solve convex feasibility program:

$$\begin{aligned} \min_{\mathbf{u}}. \quad & 0 \\ \text{s.t.} \quad & \mathbf{u} \in \mathcal{U}^{N_\delta} \\ & \hat{\mathbf{x}}_k \in \mathcal{A}_{\downarrow}^{k}(M, \delta, \tilde{\gamma}_k), \quad k = 0, \dots, N_\delta \\ & \begin{bmatrix} \hat{\mathbf{x}}_0 & \cdots & \hat{\mathbf{x}}_{N_\delta} \end{bmatrix}^T = G\mathbf{x}_0 + H\mathbf{u} \end{aligned}$$

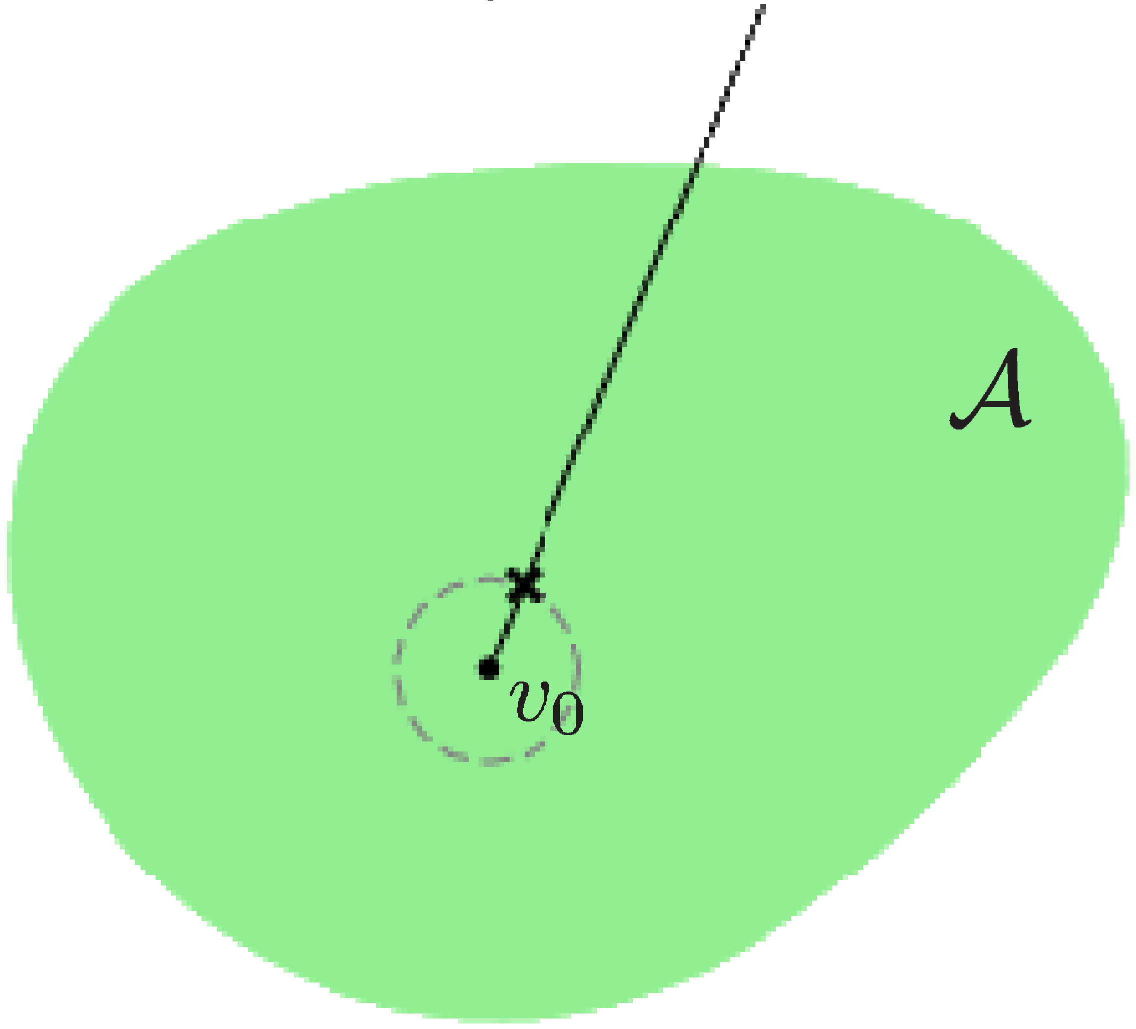
Algorithm Overview

1. Pick a point in $\mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$



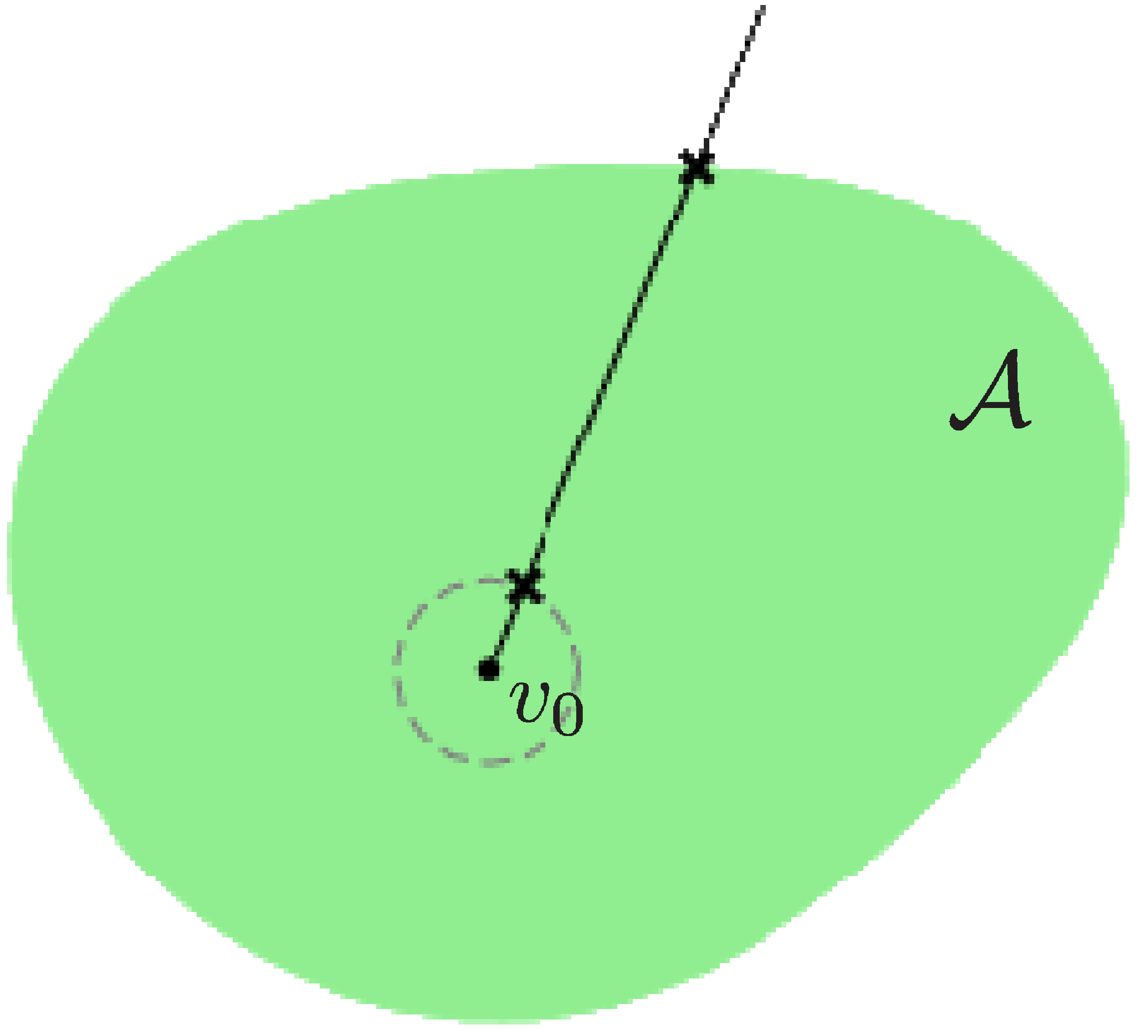
Algorithm Overview

2. Sample a direction uniformly at random



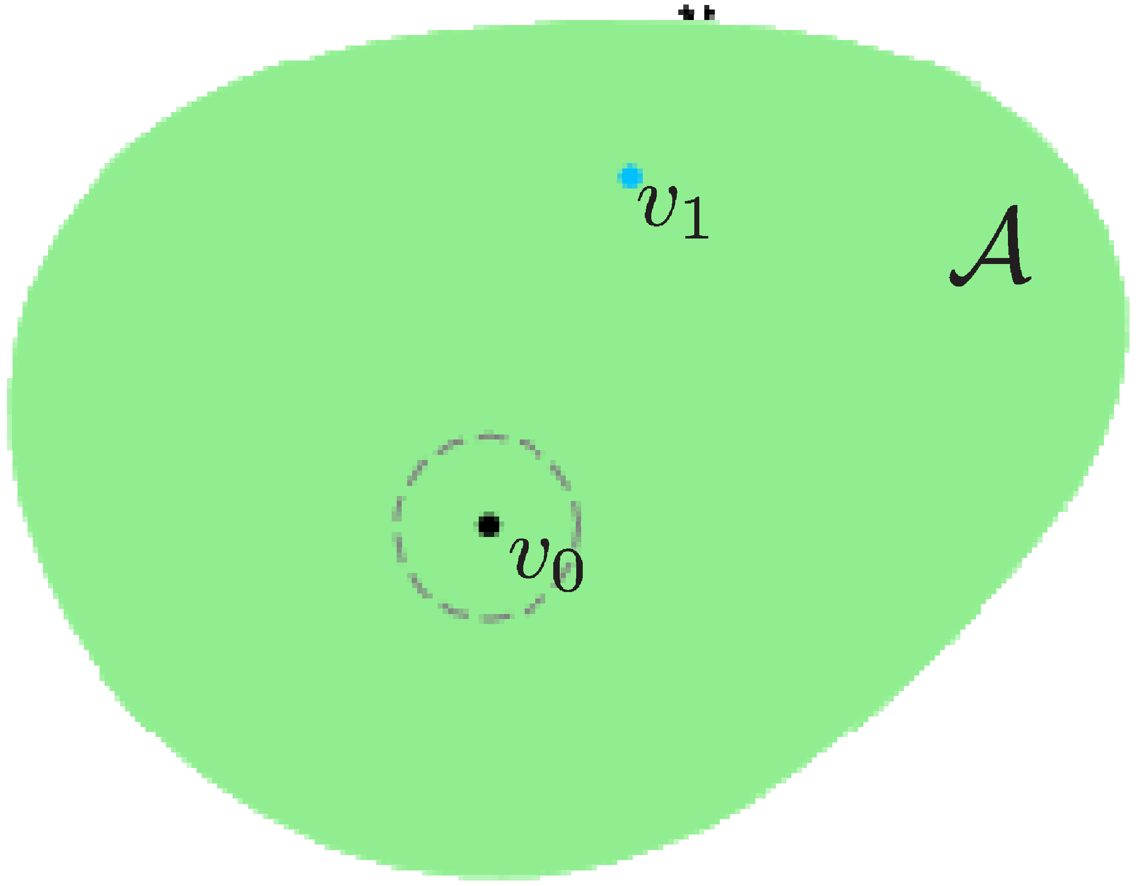
Algorithm Overview

3. Find the intersection with $\partial\mathcal{A}$



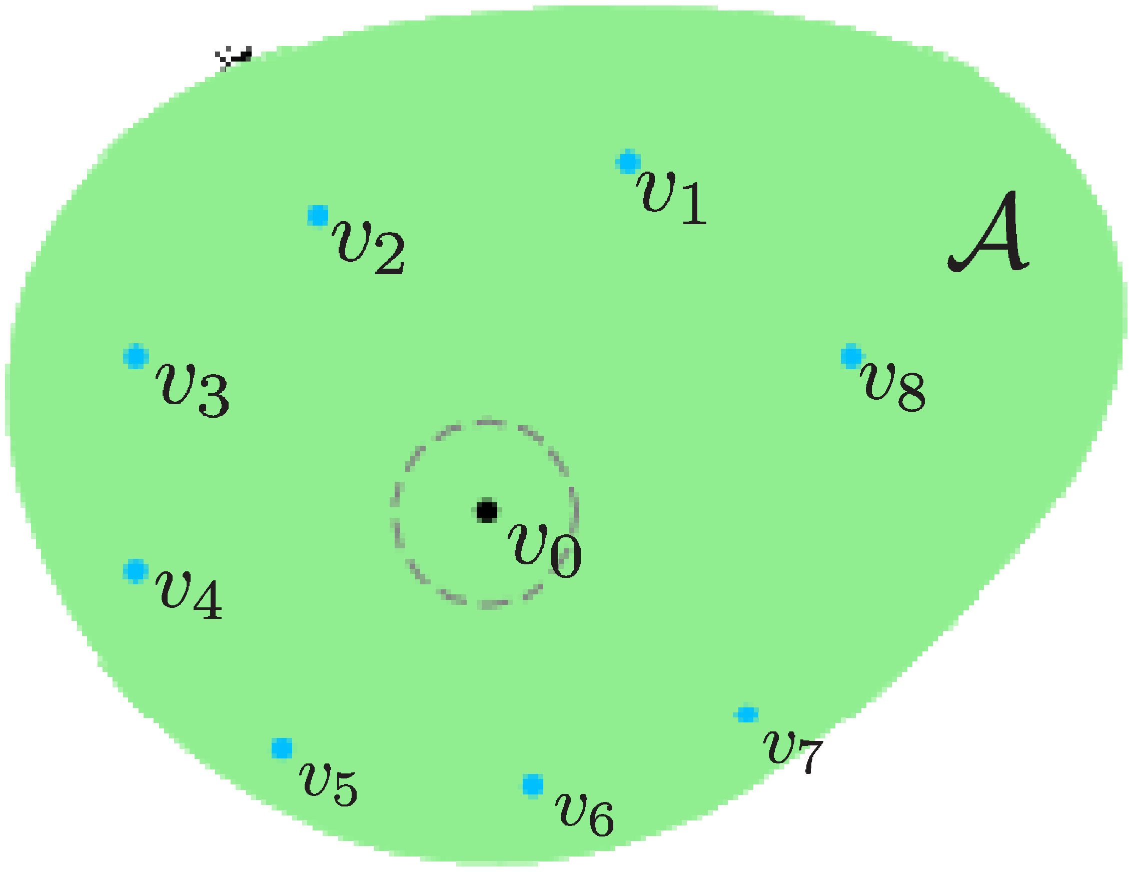
Algorithm Overview

4. Perform a bisection search, checking each point to see if it is in $\mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$ and add it to $\mathcal{V}_N$



Algorithm Overview

5. Repeat



Convexity of the Viability Kernel

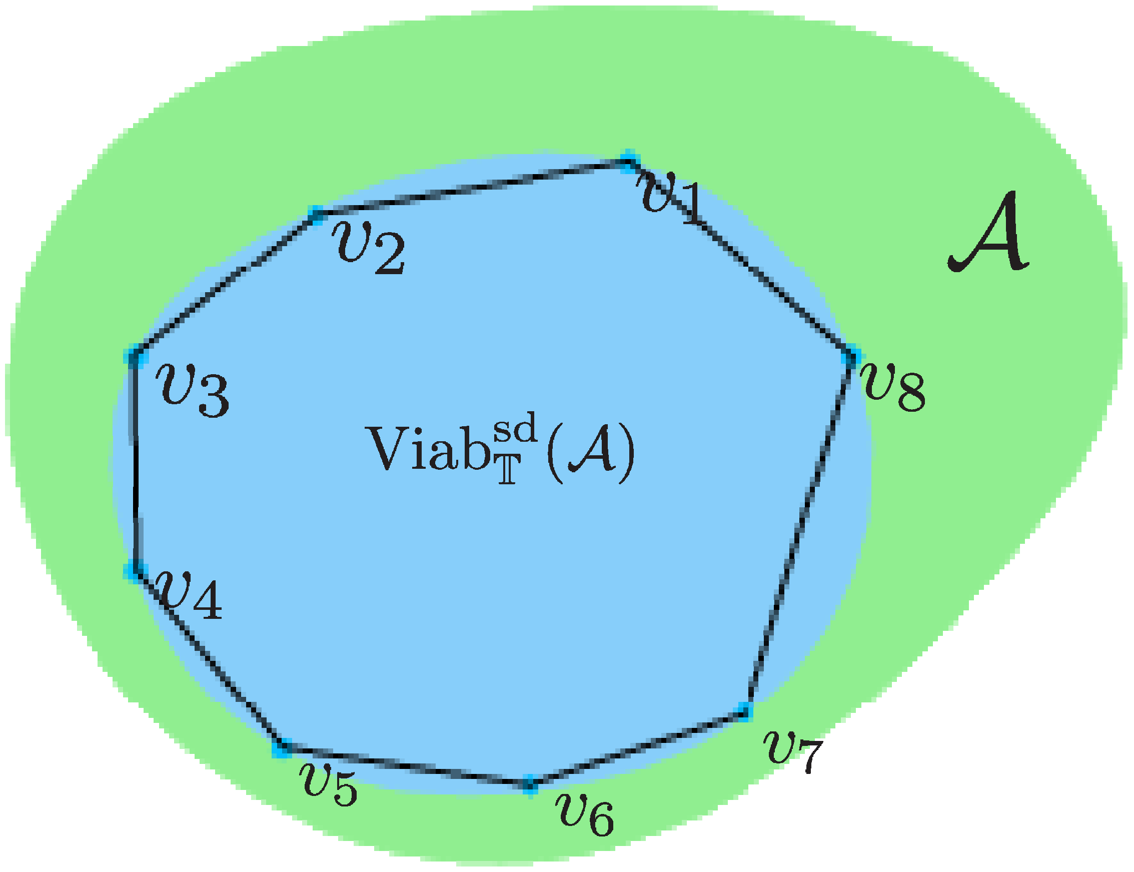
- Compact, convex constraints $\mathcal{A}$, $\mathcal{U}$
- LTI system: $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$

- Suppose $\mathbf{x}_1(0) \in \mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$, $\mathbf{x}_2(0) \in \mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$, then

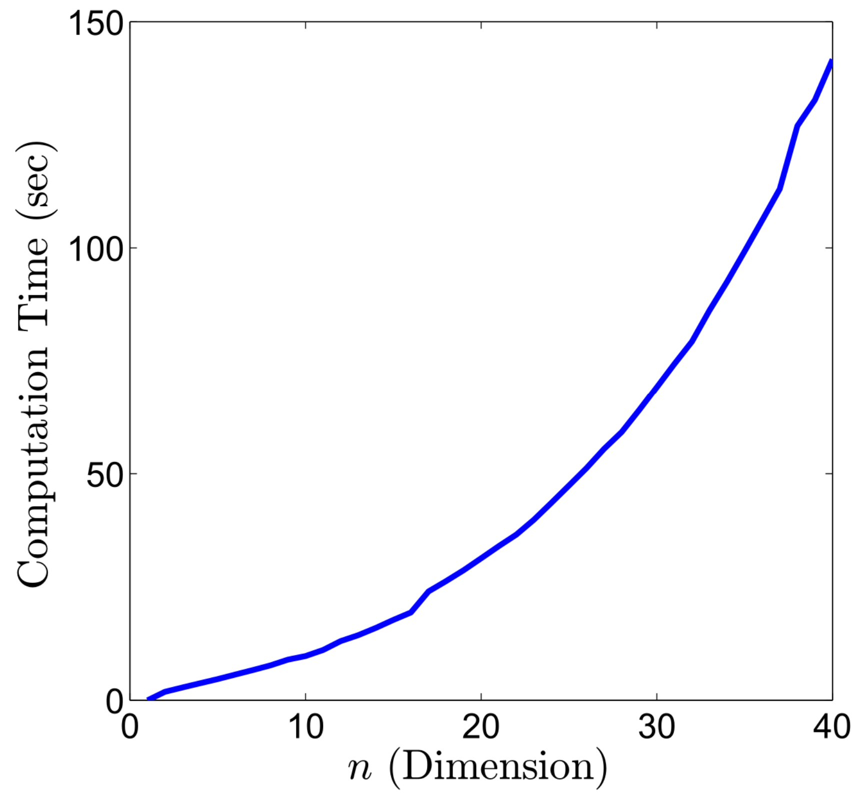
  $\exists \mathbf{u}_{\mathbf{x}_1}(\cdot) \in \mathscr{U}_{\mathbb{T}}^{\mathrm{pw}}$, $\exists \mathbf{u}_{\mathbf{x}_2}(\cdot) \in \mathscr{U}_{\mathbb{T}}^{\mathrm{pw}}$, such that
  - $\mathbf{x}_1(t) = e^{At}\mathbf{x}_1(0) + \int_0^t e^{A(t-r)} B u_{\mathbf{x}_1}(r) dr \in \mathcal{A} \quad \forall t \in \mathbb{T}$
  - $\mathbf{x}_2(t) = e^{At}\mathbf{x}_2(0) + \int_0^t e^{A(t-r)} B u_{\mathbf{x}_2}(r) dr \in \mathcal{A} \quad \forall t \in \mathbb{T}$
- Then if $\mathbf{x}_3(t) = \theta \mathbf{x}_1(t) + (1-\theta)\mathbf{x}_2(t)$ for $\theta \in [0,1]$
- Then $\mathbf{x}_3(t) \in \mathcal{A} \quad \forall t \in \mathbb{T}$ and so $\mathbf{x}_3(0) \in \mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$

Algorithm Overview

6. By the convexity of $\mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$, $\mathrm{conv}(\mathcal{V}_N = \{v_1, \ldots, v_N\}) \subseteq \mathrm{Viab}_{\mathbb{T}}^{\mathrm{sd}}(\mathcal{A})$

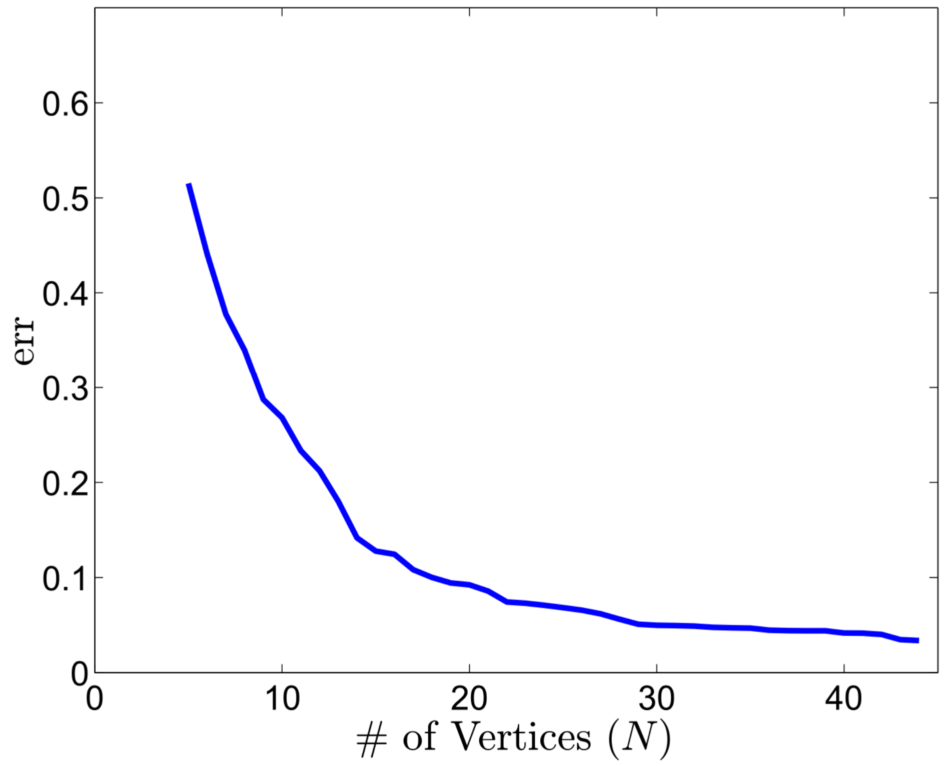


Algorithm Complexity



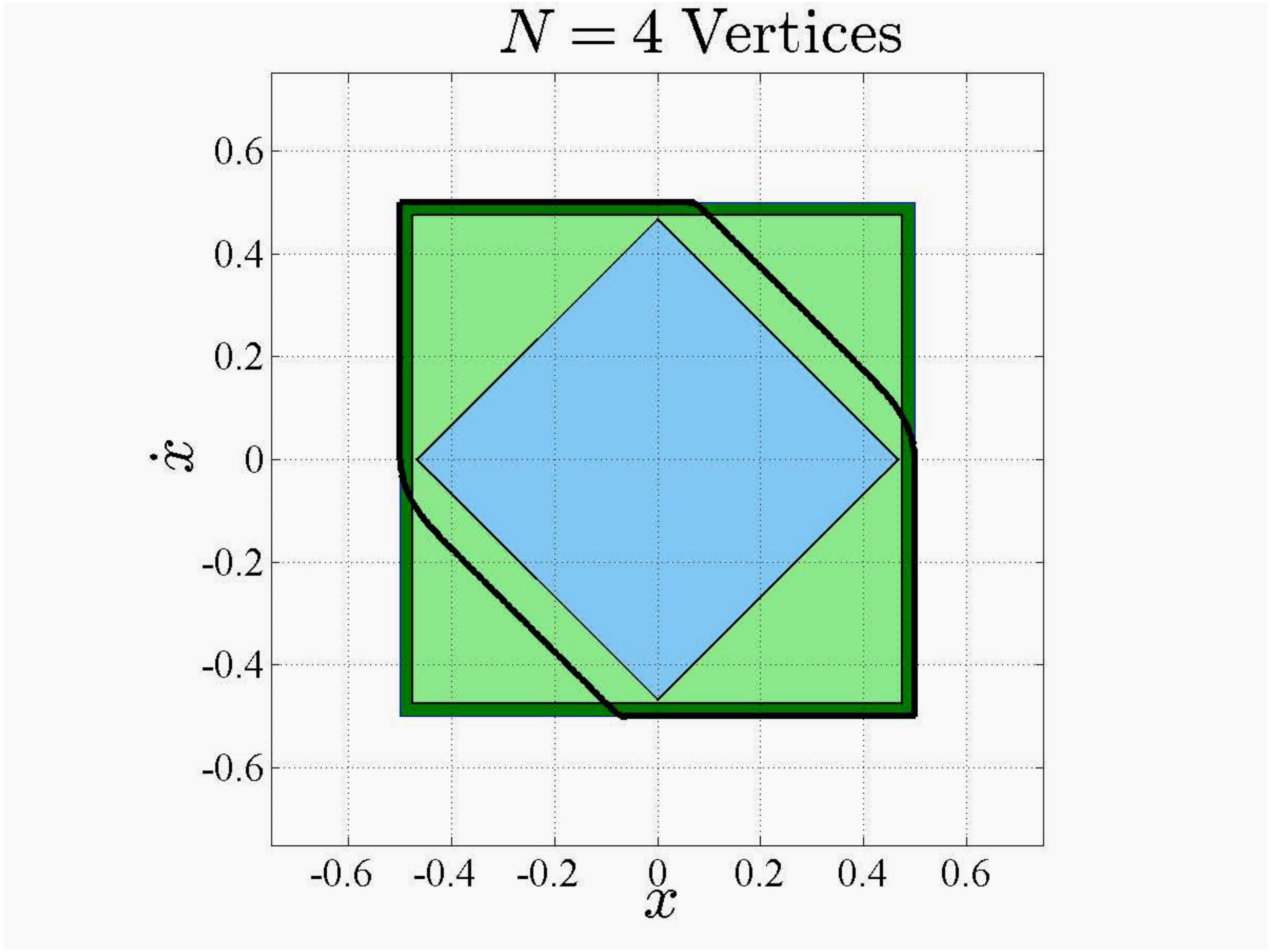
- Algorithm complexity $O(N\log(d)\Phi)$
  - N: number of samples
  - d: diameter of $\mathcal{A}$
  - Φ: runtime complexity of feasibility program, i.e. $O(n^{2.37})$
  - n: dimension

Algorithm Convergence



- The algorithm converges as $\lim\limits_{\substack{N\to\infty \\ \zeta\to\infty \\ \epsilon\to 0}} \epsilon_{\text{vol}}(N,\zeta,\epsilon) N^{\frac{2}{n-1}} = c_n(\text{Viab}_{\mathbb{T}}^{\text{sd}}(\mathcal{A}))$
  - Where $\epsilon_{\text{vol}}(N,\zeta,\epsilon) := \text{vol}(\text{Viab}_{\mathbb{T}}^{\text{sd}}(\mathcal{A})) - \text{vol}(\mathcal{V}_N^{\zeta,\epsilon}) - \epsilon_{\text{cont}}(M\delta)$
- Convergence rate is optimal!

Example: Double Integrator



Example: 12D Linearized Quadrotor

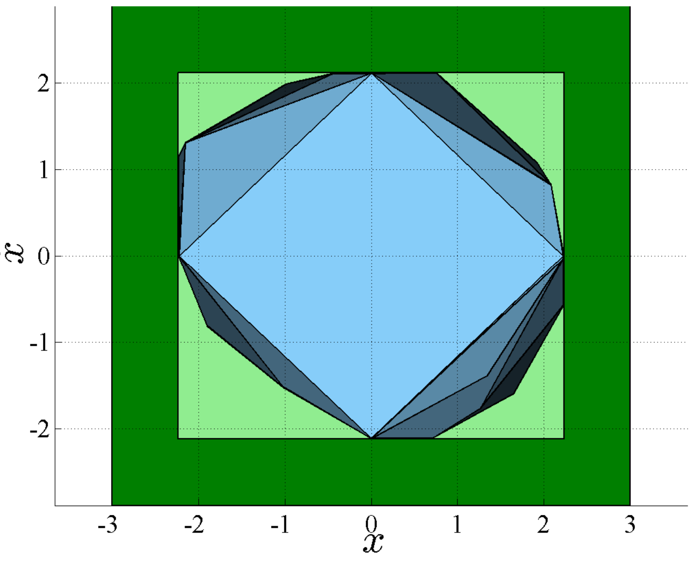
- State: $\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^\top$

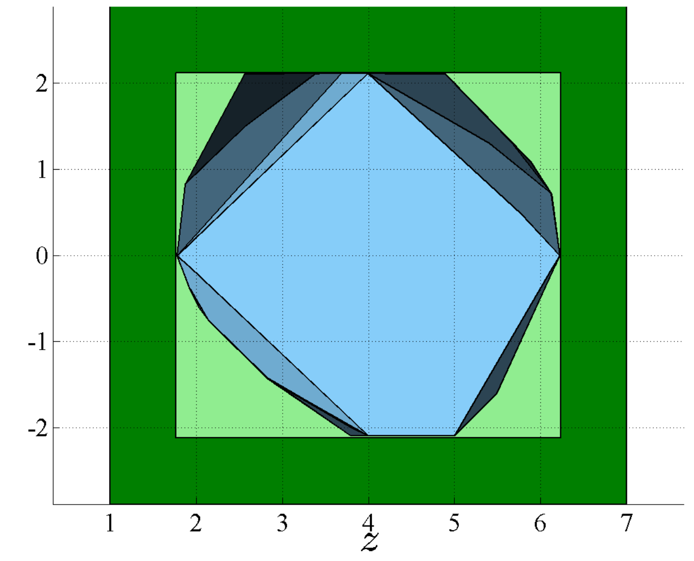
- Linearized Dynamics:
  - $\ddot{x} = -g\theta,\ \ddot{y} = g\phi,\ \ddot{z} = -u_1,\ \ddot{\phi} = u_2,\ \ddot{\theta} = u_3,\ \ddot{\psi} = u_4,$

- Implementation details:
  - 4th order discretization approximation
  - 0.01-accurate bisection search (typically eight levels)
  - 100 to 2000 vertices
  - Typically 7sec/vertex

Example: 12D Linearized Quadrotor

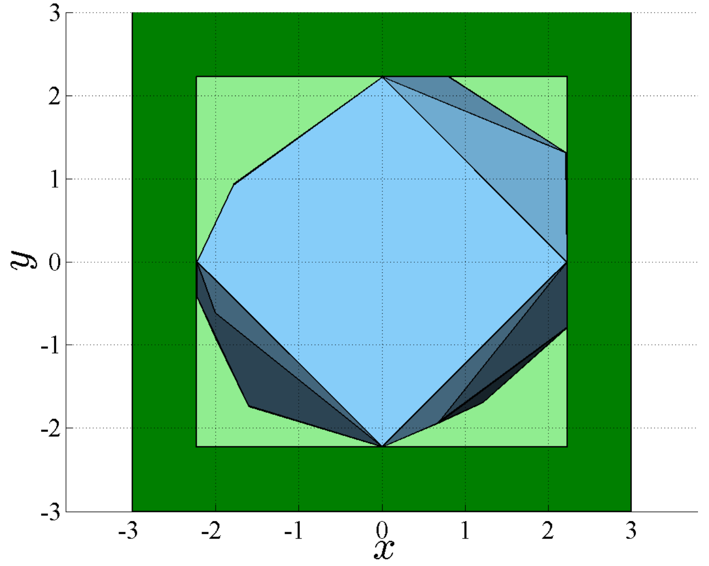
$x, \dot{x}$

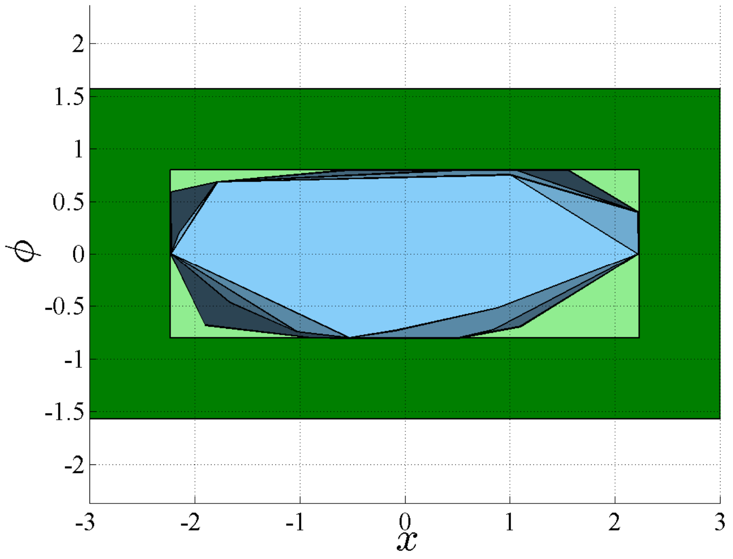




$z, \dot{z}$

x, y





x, ϕ

Outline

- *Introduction*
- *Reachability Analysis*
- *Guaranteed Safe Online Learning via Reachability (GSOLR)*
- *Extensions of GSOLR*
- *Sampling-Based Reachability Computations*

→**Conclusions**

Conclusions

- **Guaranteed Safe Online Learning via Reachability**
  - Novel framework for performing any machine learning algorithm in an online manner while maintaining safety guarantees
  - Completely flexible, least restrictive
- **Extensions of GSOLR**
  - Methods for tuning conservatism in safety guarantees based on a robot’s experience of the world
- **Sampling-Based Reachability Computations**
  - Algorithm for computing the viability kernel in high-dimensions, with proofs on convergence rate and optimality

Questions?