Ensuring Safety in the Real World

Cherie Ho, Mohammad Mousaei
Air Lab Summer School 2020

air lab

Carnegie Mellon University
The Robotics Institute

Session Purpose

1. Outline key challenges and current methods to ensure safety

2. Build intuition behind several formal safety proofs and where guarantees can be violated

3. Hands-on experience with guaranteed-safe control (Interactive Session)

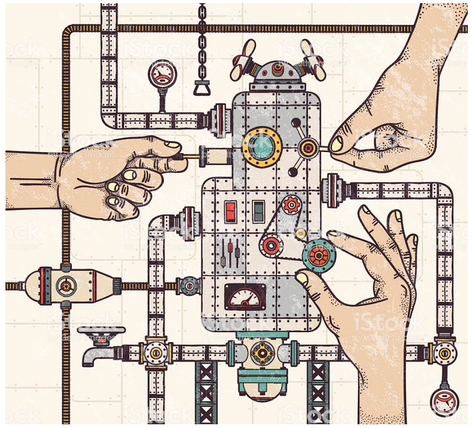
Outline

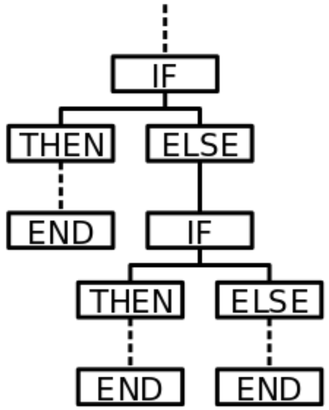
- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

Now: Specialized Wilderness

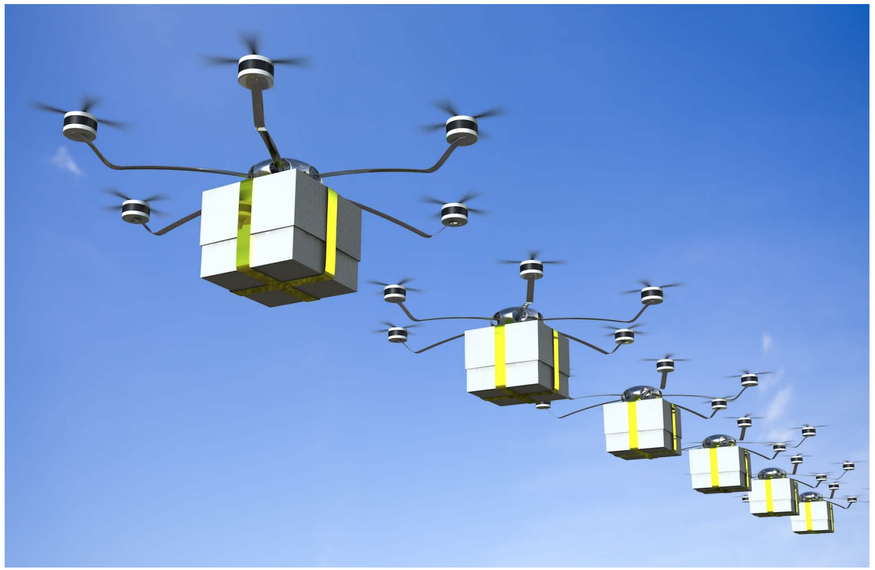
Behind the scenes

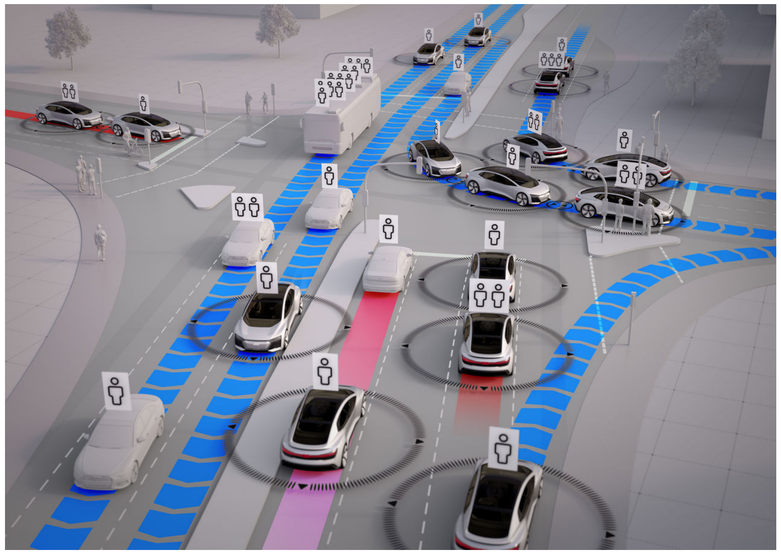
- Specialized missions (heavy handtuning)
- Multiple Safety Pilots
- Handcrafted safety mechanisms





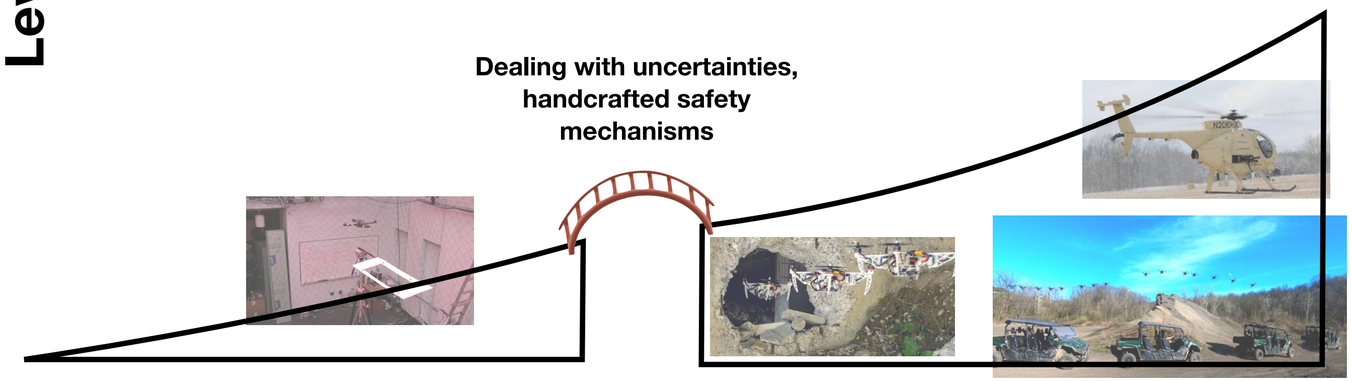
Vision: Ubiquitous Autonomy

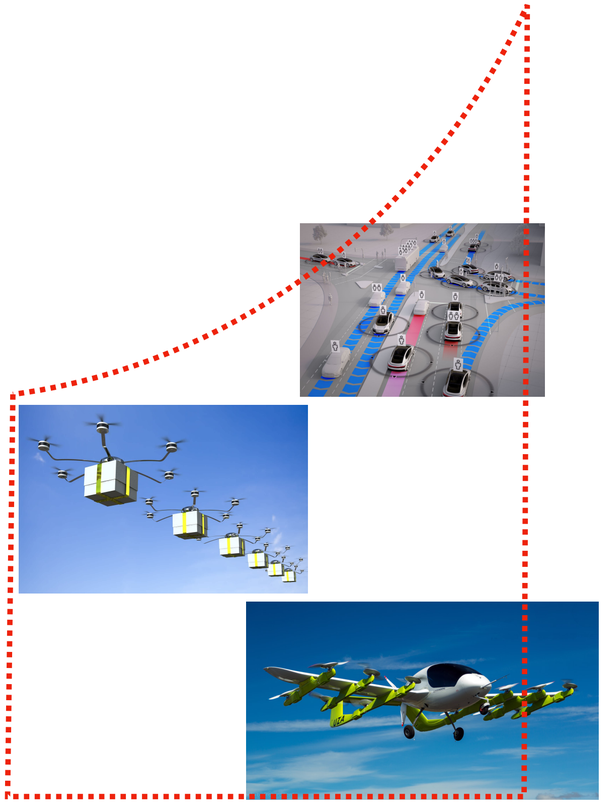




To Solve: Generalizability, Handling Out-of-Norm Cases, Self-accountable

Level of Risk





Before:
Controlled
Environment

Look mom,
No hands!

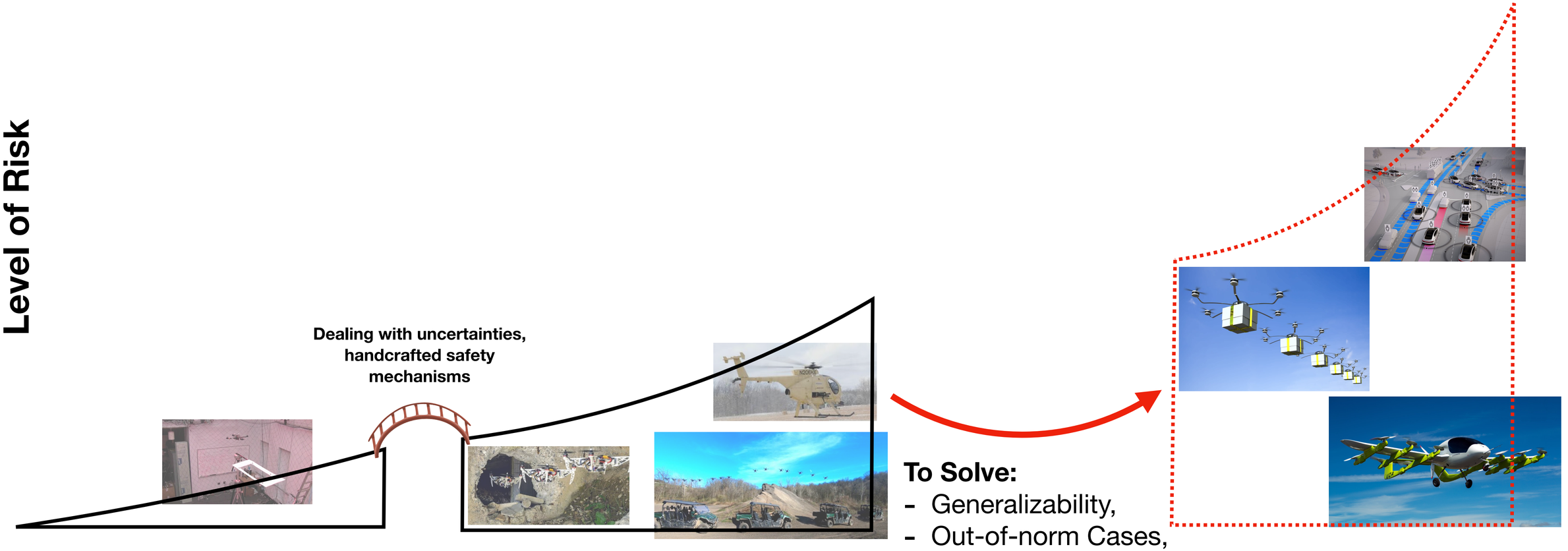
Now:
Specialized
Uncertain Applications

Look mom,
No cage!

Future:
Ubiquitous
Autonomy

Look mom,
No safety pilot!

Level of Risk



Before:
Controlled
Environment

Look mom,
No hands!

Now:
Specialized
Uncertain Applications

Look mom,
No cage!

To Solve:
- Generalizability,
- Out-of-norm Cases,
- Self-accountable

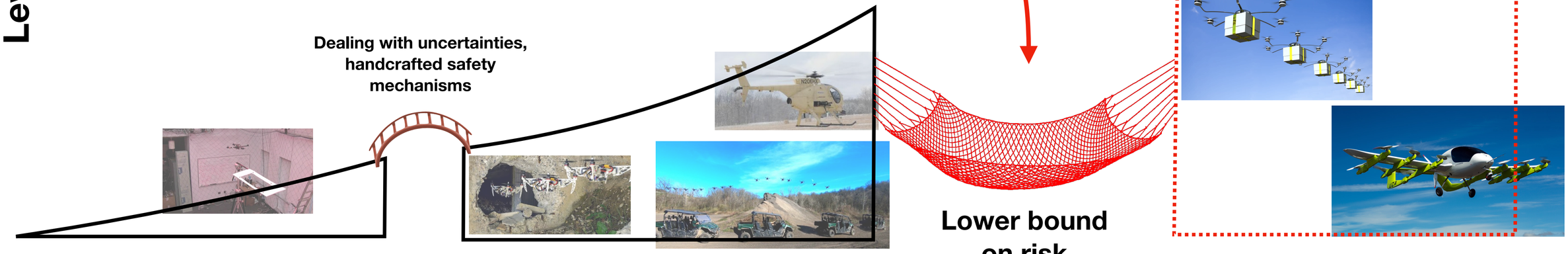
Future:
Ubiquitous
Autonomy

Look mom,
No safety pilot!

Level of Risk

Our Question:

How can we *ensure safety* for *scalable* deployment in the *real world*?



Before:
Controlled Environment

Look mom, No hands!

Now:
Specialized Uncertain Applications

Look mom, No cage!

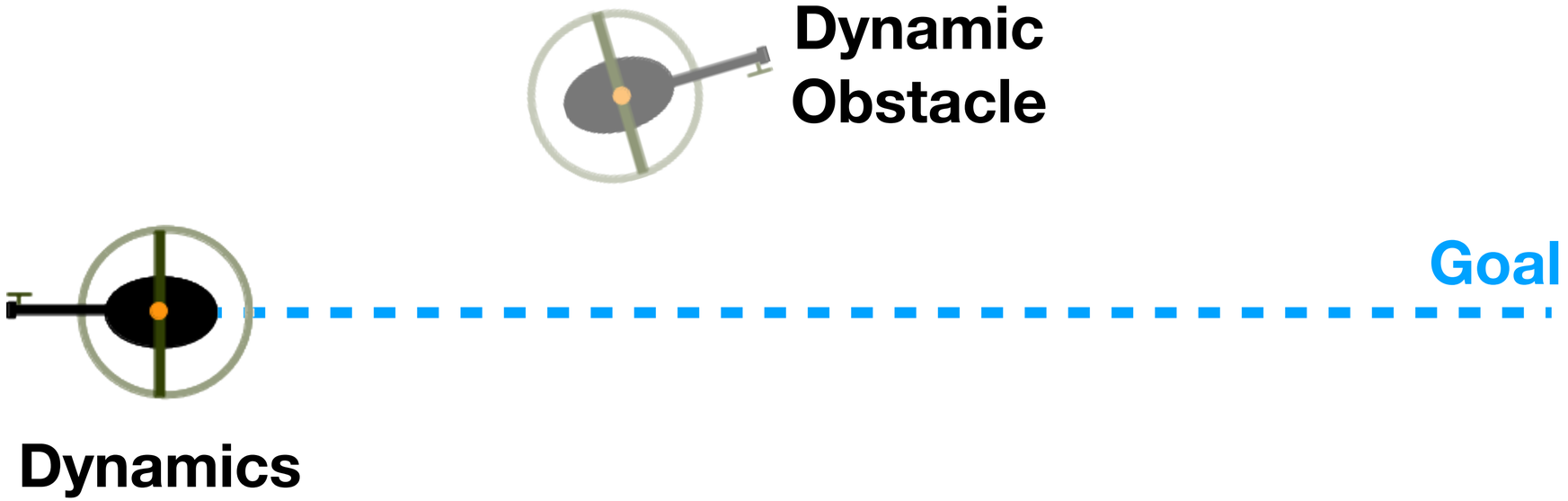
Future:
Ubiquitous Autonomy

Look mom, No safety pilot!

Outline

- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

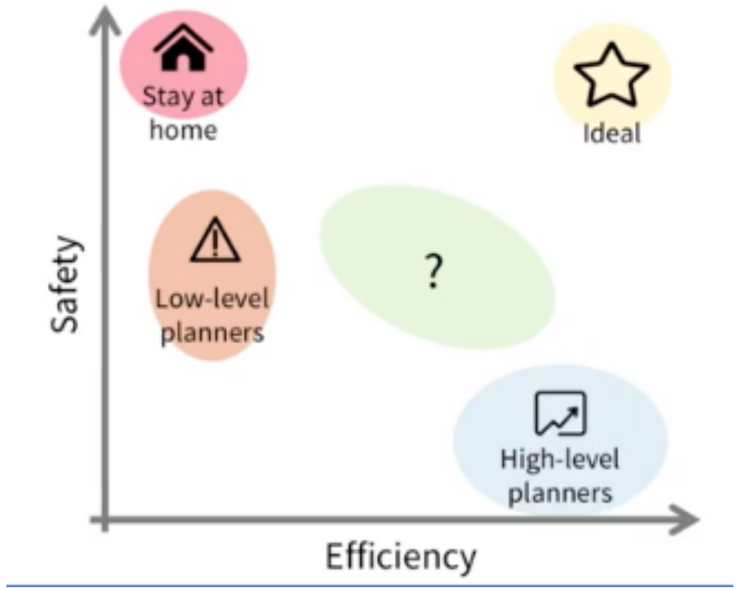
Running Example - Avoid Occupied Space

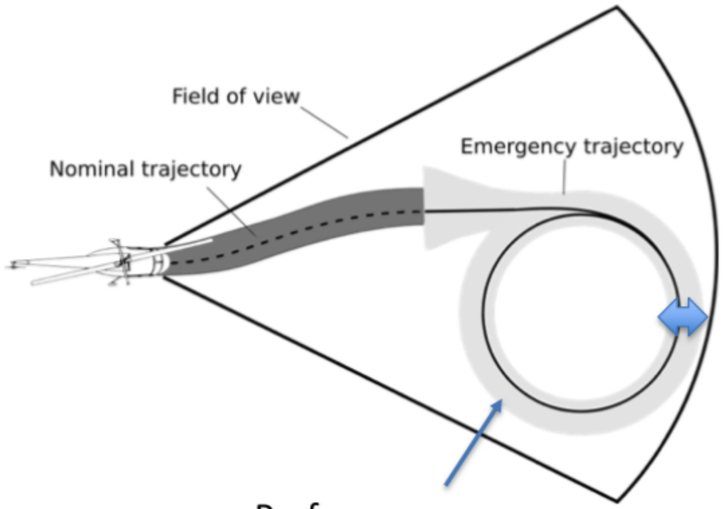


Challenge 1: “How tight can we go?”

Balance safety and performance

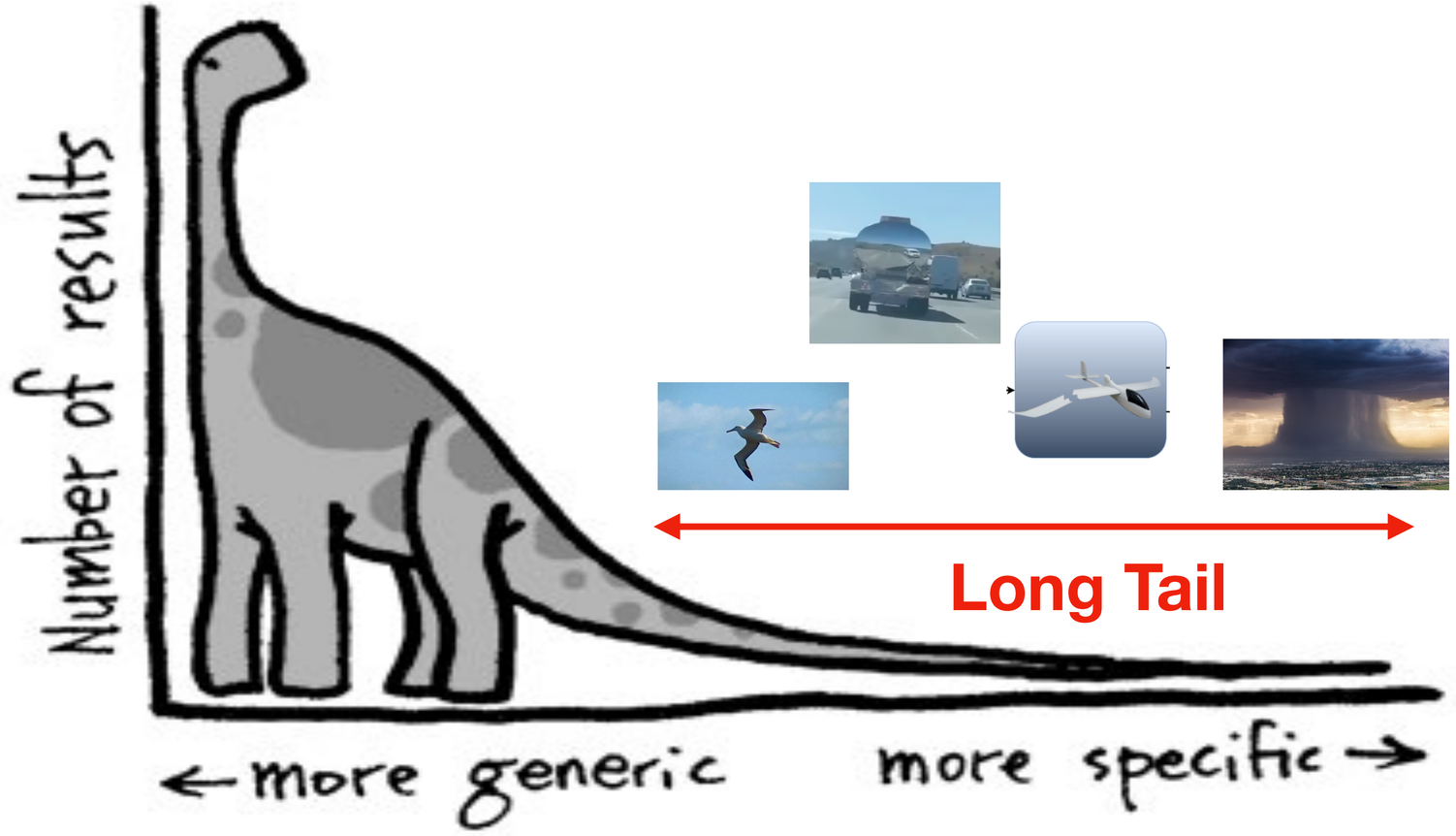
To be safe or get things done?





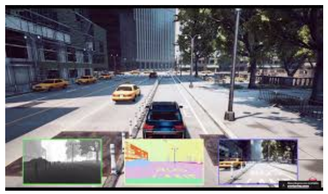
Performance is directly related to size of the reachable set

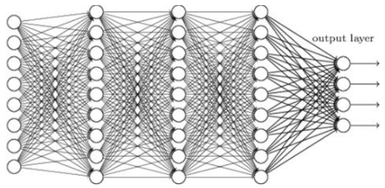
Challenge 2: Many Tail-end Cases

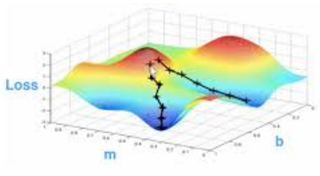


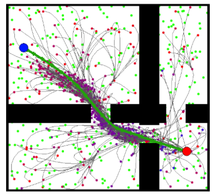
Coverage: How to know we have covered everything?

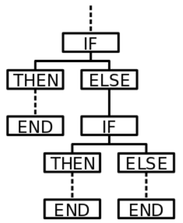
Challenge 3: Complex, Non-deterministic System

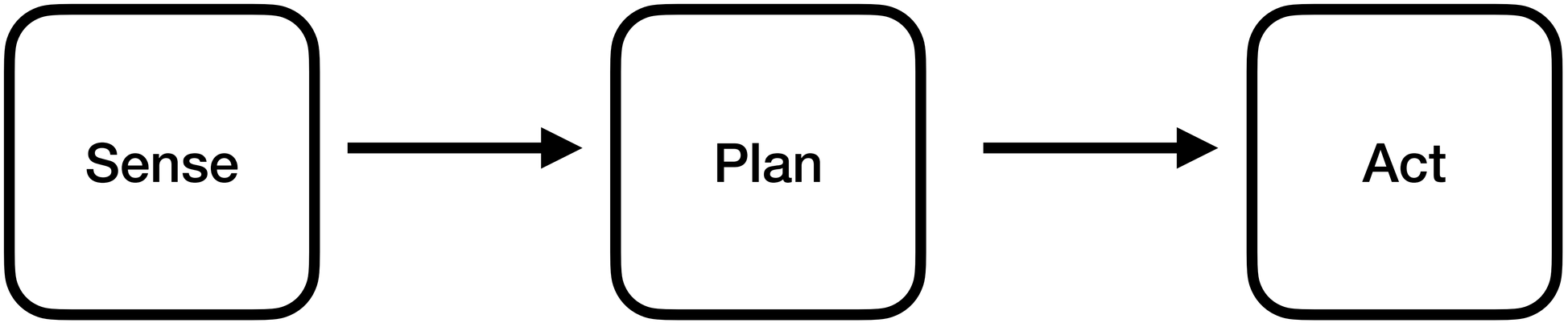






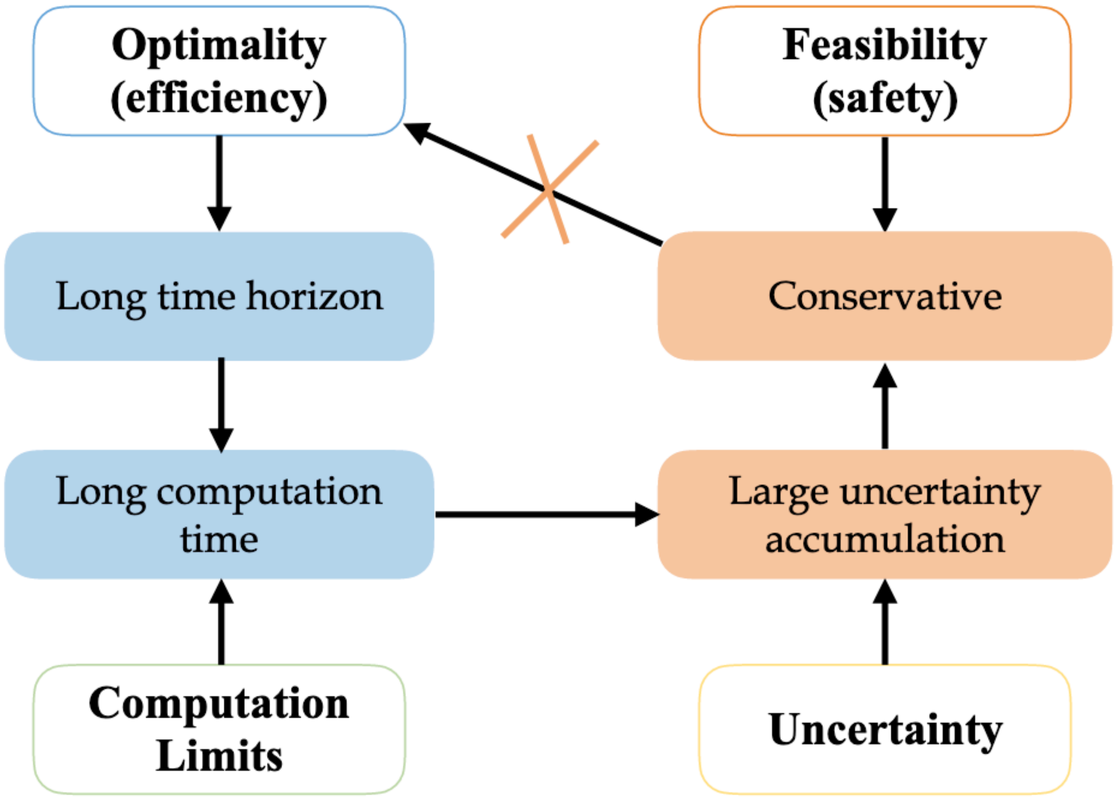






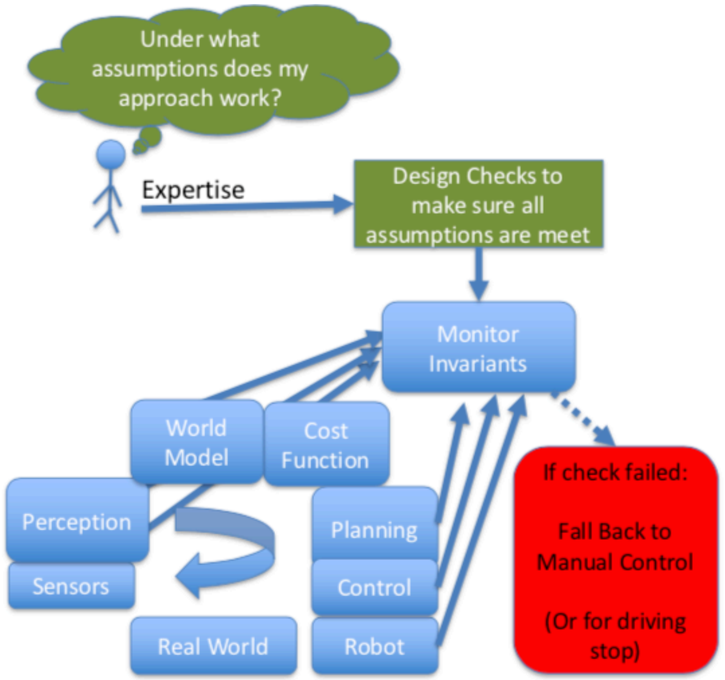
- Limited full-pipeline real world data
- Adapting systems

Challenges



Taken from Changliu Liu's 16-883 Provably Safe Robotics course

Ensuring Safety (Safety Pilot)



If anything goes wrong, pass it to pilot.

Pros: Risk lower bounded by pilot ability (human-in-the-loop).
Cons: Hard to scale, a lot of expertise and training

Taken from Basti's Tutorial Slides

Ensuring Safety (Laundry List)

Nominal behavior

Backup Behaviors

- Actuator Failure, above 2km
- Near ground
- Sharp Corner in tunnel

Pros: Known behavior in most cases
Cons: Specialized applications, hard to generalize.

Ensuring Safety (Formal Guarantees)

- Pros: Generalizable, mathematical claims on safety properties
- Cons: Ultimately relies on models and assumptions
  (to be discussed)

Outline

- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

Ingredients for verifying safe action

If all cases are safe after given action,
verified!

Safety Definition:
Stay in free space
Stay within lanes

System Dynamics
Initial State
Quadrotor Dynamics
Quadrotor Controller
Pedestrian Modeling

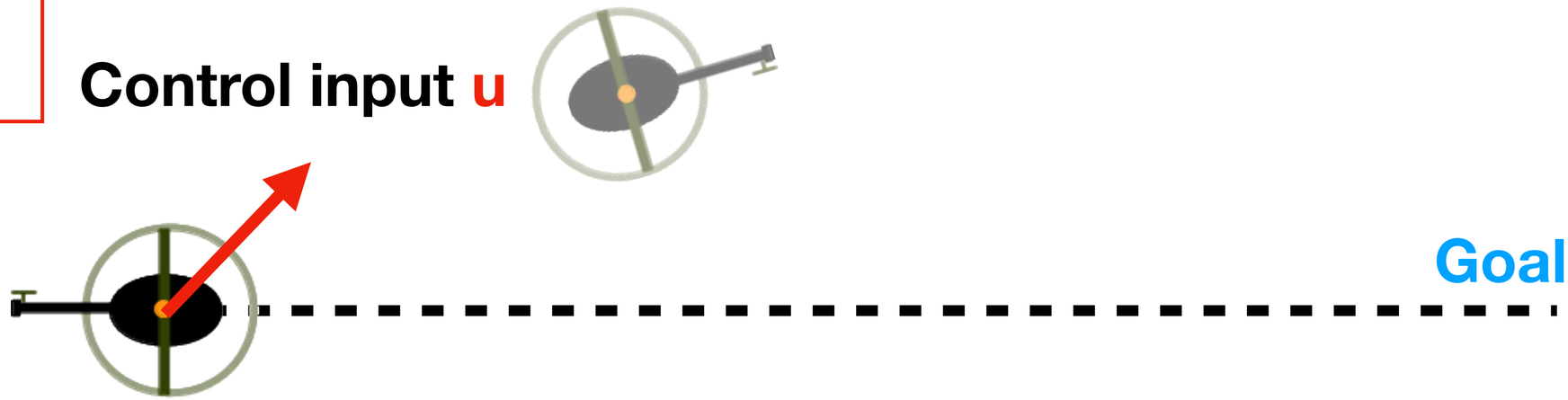
System Limits

Actuator Limits
Sensor Range

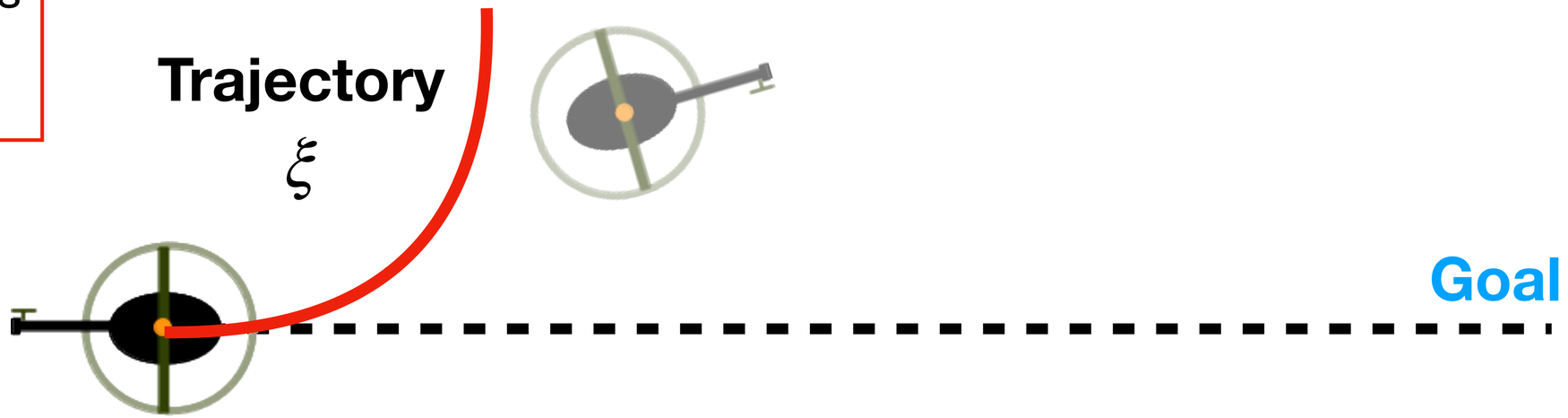
Disturbances

Wind
Sensor Noise

Verifying Action
1. Safety Definition
2. System Dynamics
3. System Limits
4. Disturbances



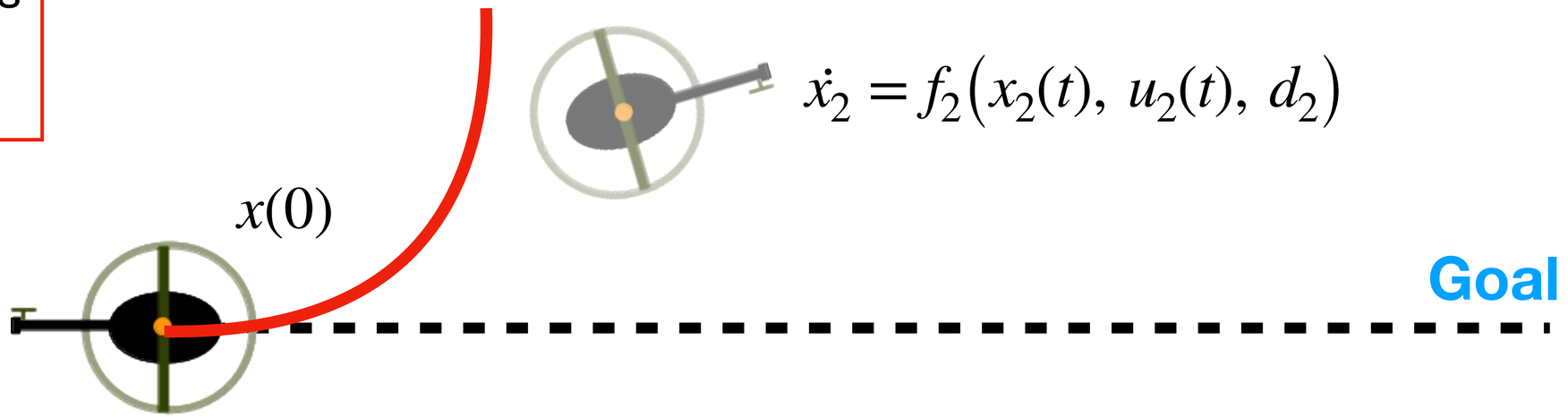
Verifying Action
1. Safety Definition
2. System Dynamics
3. System Limits
4. Disturbances



Definition: Vehicle will stay within free space

$$x(t) \in X_{free}$$

Verifying Action
1. Safety Definition
2. System Dynamics
3. System Limits
4. Disturbances



Dynamics: $\dot{x} = f\left(x(t),\ u(t),\ d\right)$

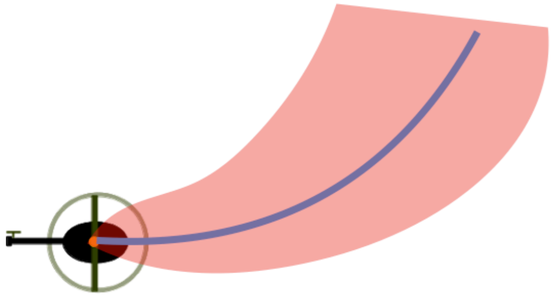
Controller: $u = \pi(x, x_{des})$

Bounded Disturbance:

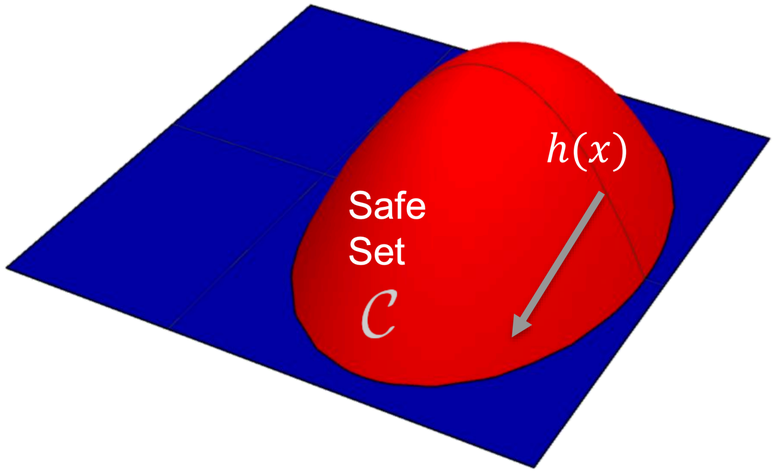
$$d \in [d_{min}, d_{max}]$$

2 Key Tools

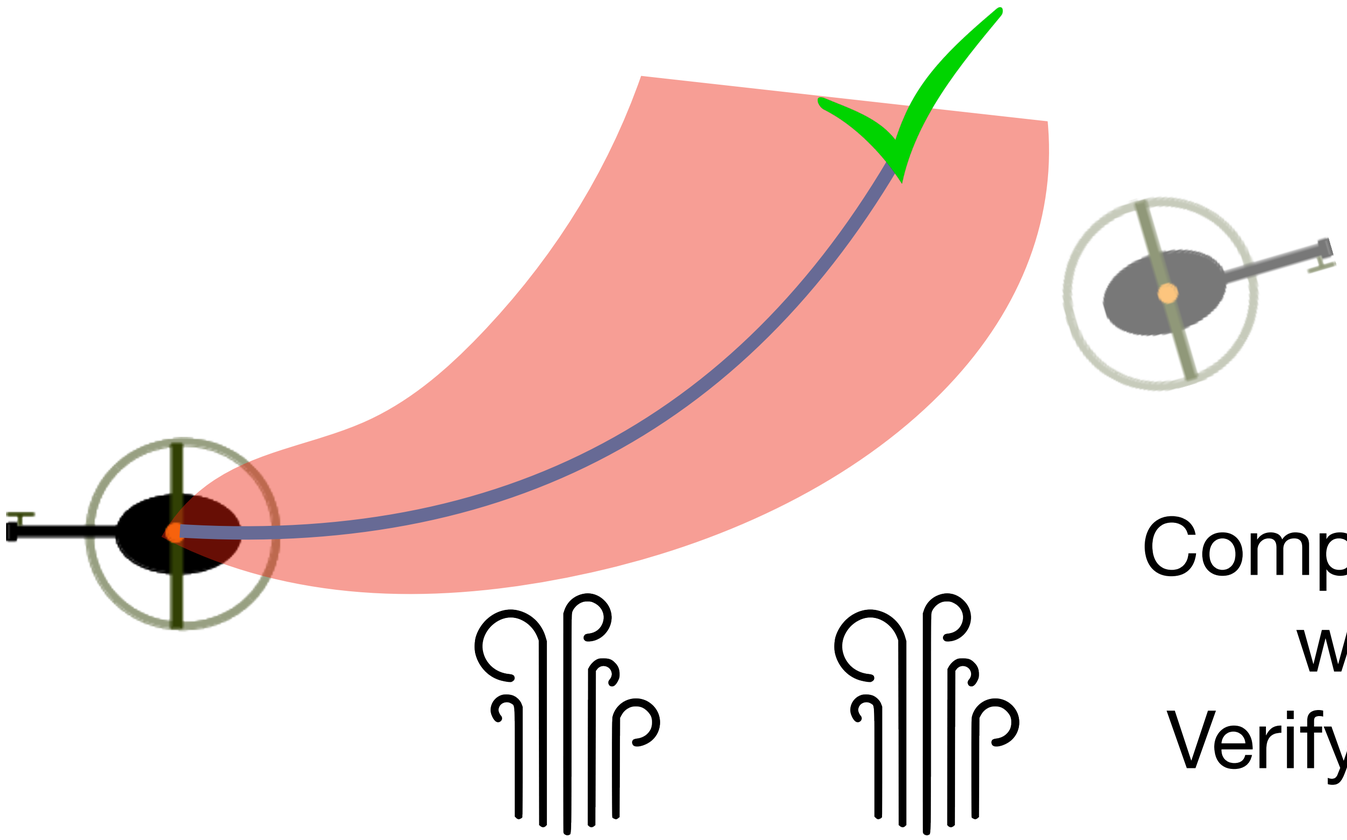
1. Reachable Sets



2. Control Barrier Function



Reachability (Intuition)



Compute set of all possible states
when following trajectory.
Verify that obstacles are outside.

Reachability: Verify Trajectory

Reachable Set of Time Interval when following Trajectory ξ

$$R_\xi([0,t_f]) = \bigcup_{t\in[0,t_f]} R_\xi(t)$$

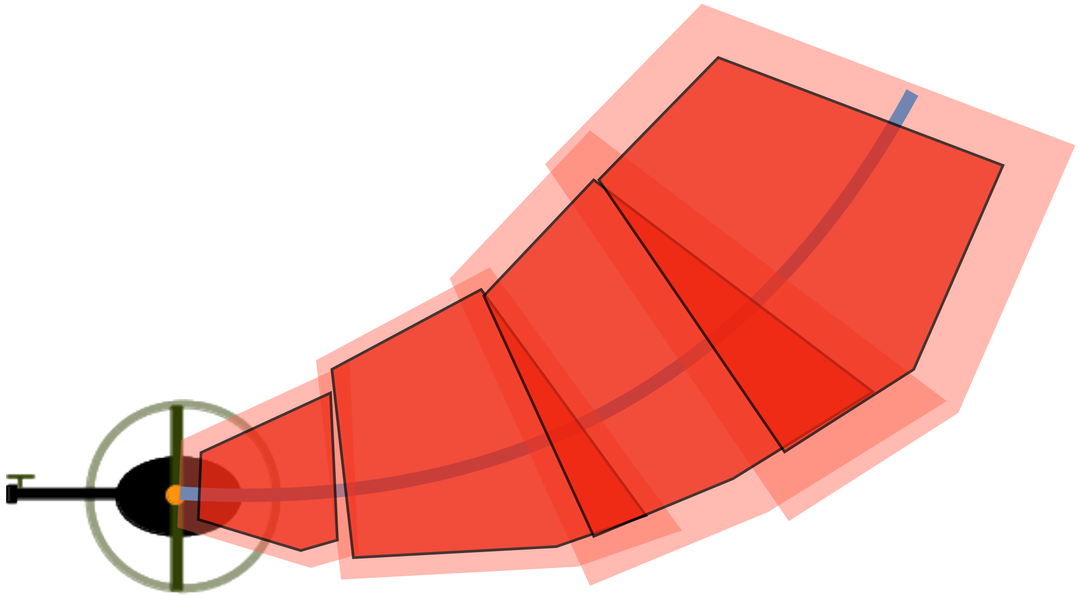
Reachable Set at time *r*:

All possible states at time r given model dynamics all possible initial states, input, disturbance

$$R_\xi(r) = \{ \int_0^r f\big(x(t), u(t), d(t)\big)dt \mid x(0) \in X^0, u(t) = \pi(x(t), \xi(t)) \in U, d \in [d_{min}, d_{max}] \}$$

Key Challenges: high dimensionality, nonlinear dynamics

How to efficiently calculate reachability?
COntinuous Reachability Analysis (CORA)



Overapproximate with linear dynamics

$$\dot{x} = f\big(x(t),\, u(t),\, d(t)\big) \quad \rightarrow \quad \dot{x} = Ax(t) + d(t)$$

Controlled vehicle model

Why start with linear, time-invariant system?

Linear Continuous Model:

$$\dot{x} = Ax + d(t)$$

- Superposition: separate homogenous and inhomogeneous solution (more efficient)

$$x(t) = x^h(t) + x^p(t)$$

- Well known solutions:

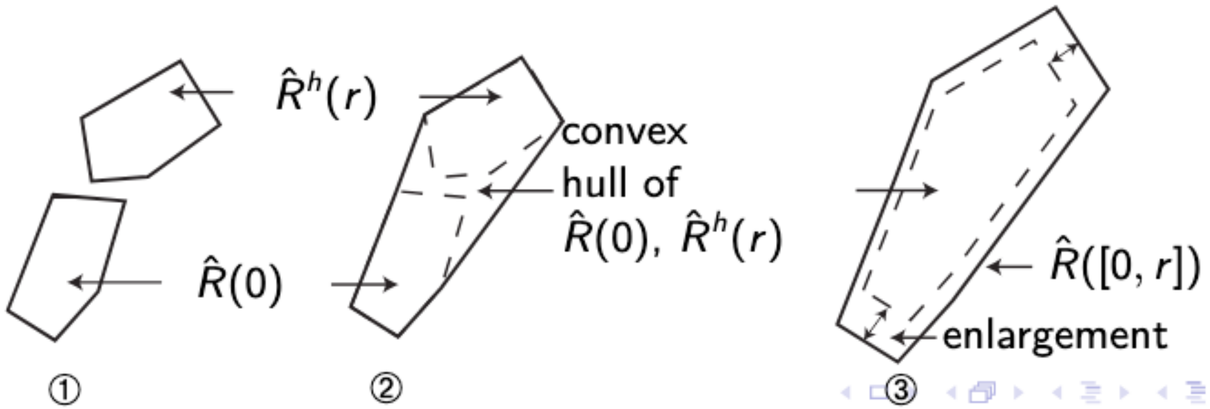
$$x^h(r) = e^{Ar}x(0) \ , \quad x^p(r) = \int_0^r e^{A(r-t)}d(t)dt$$

CORA Overview [Linear System]

1. Compute reachable set $\hat{R}^h(r)$ at time r [no disturbance/noise]

2. Obtain convex hull of $\hat{R}(0)$ and $\hat{R}^h(r)$

3. Link all in-between times:
   Enlarge reachable set to account for curved traj and uncertain input

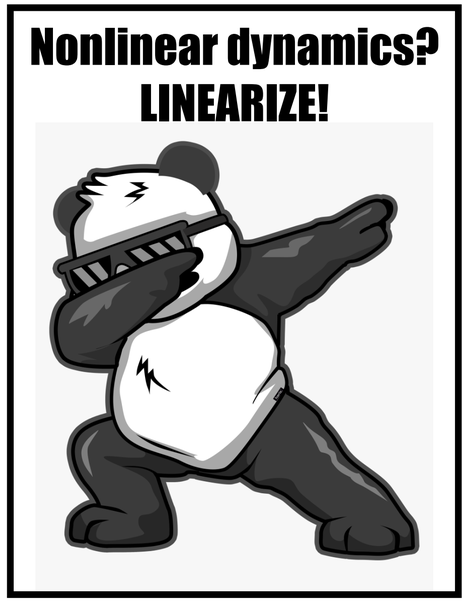


Extending to Nonlinear Systems: Conservative Linearization

1. **Linearize** dynamics for efficient computation

2. **Overapproximate** with linearization error **bounds**

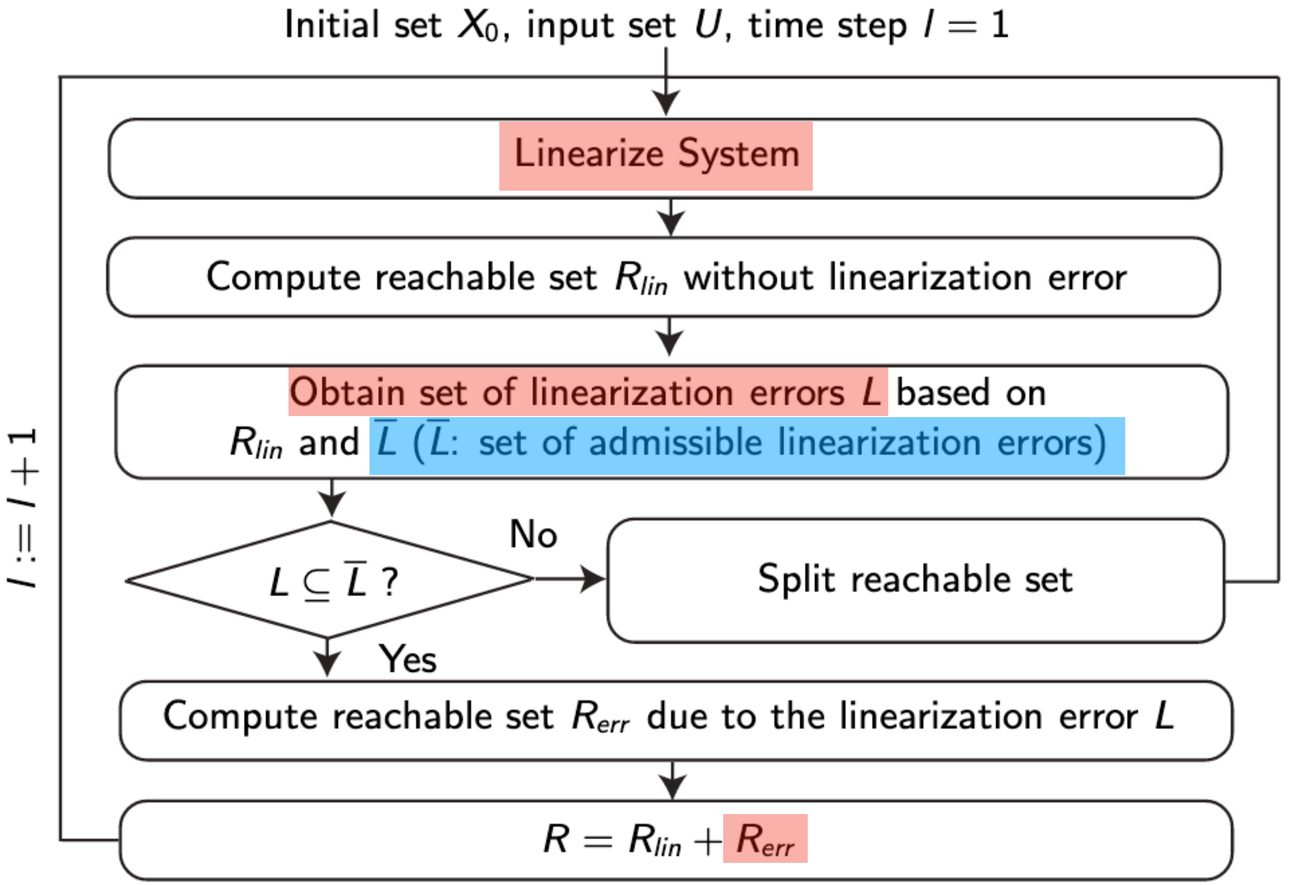
3. **Constrain** maximum allowed linearization error. Split reachable set. (handtune: accuracy vs compute time)



Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

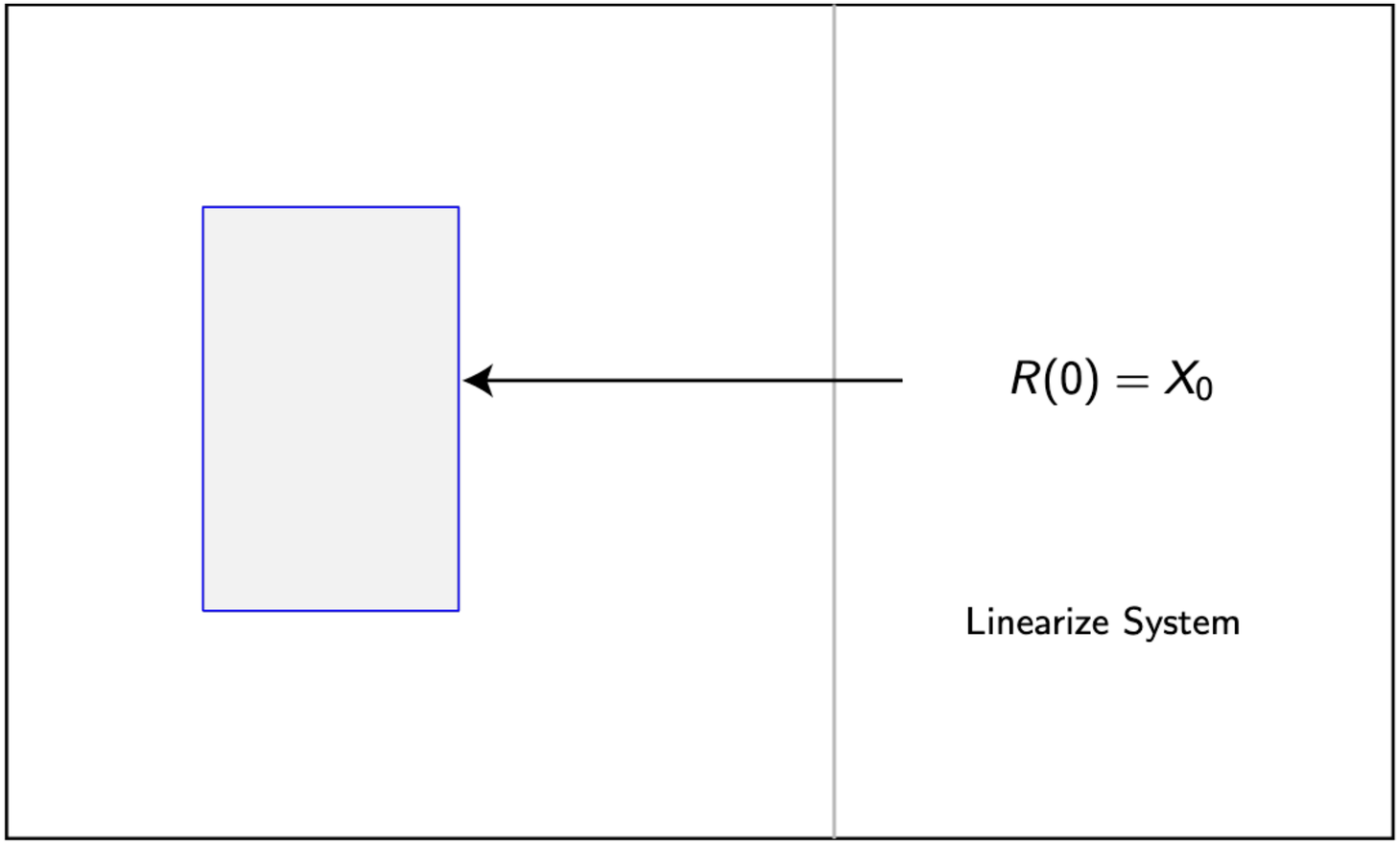
Extending to Nonlinear Systems

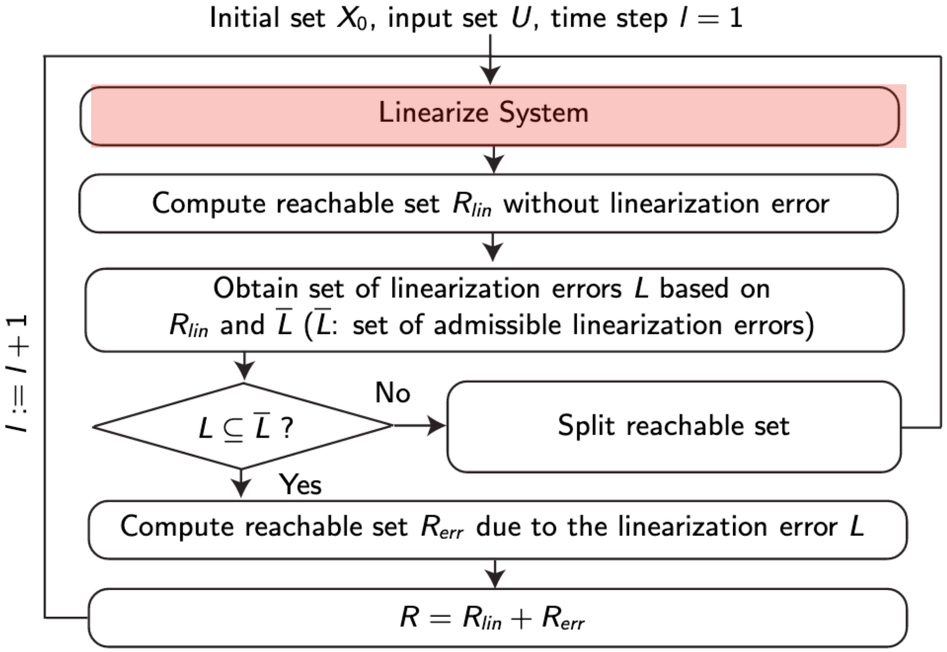
$\overline{L}$: Heuristic Max-allowed Error
(Accuracy vs. Efficiency)



Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

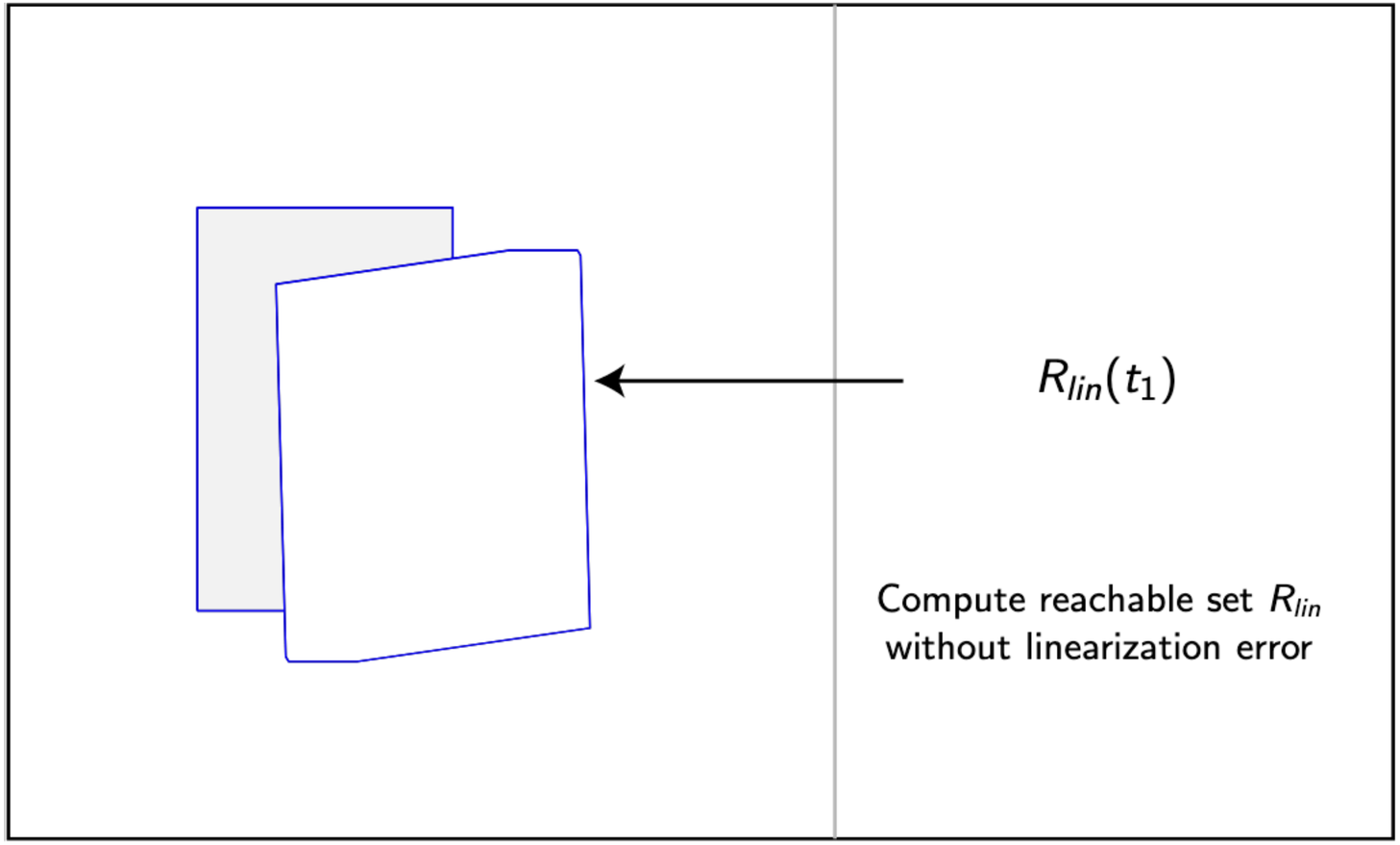
CORA with Nonlinear Systems

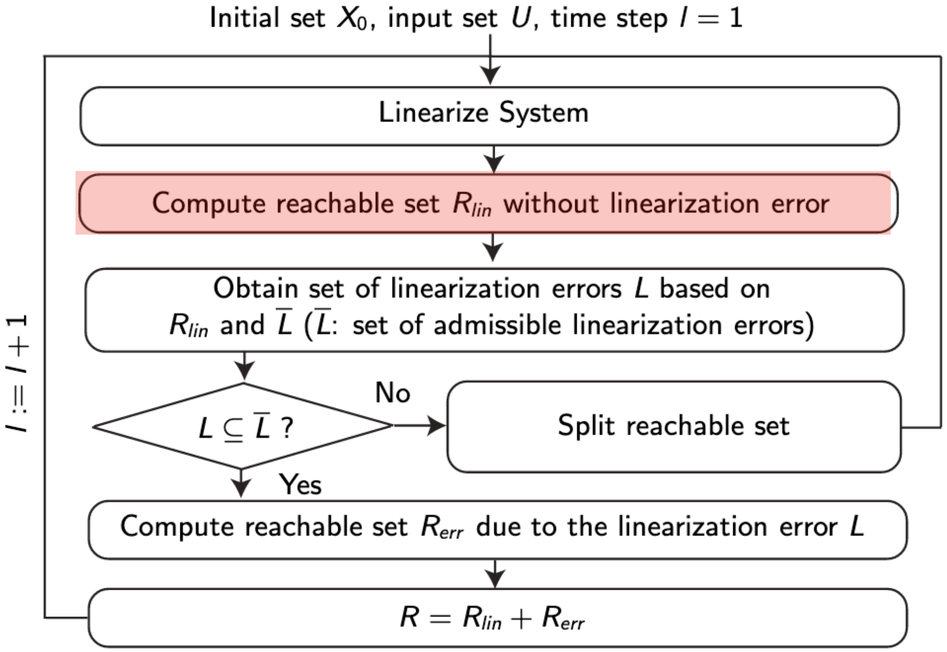




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

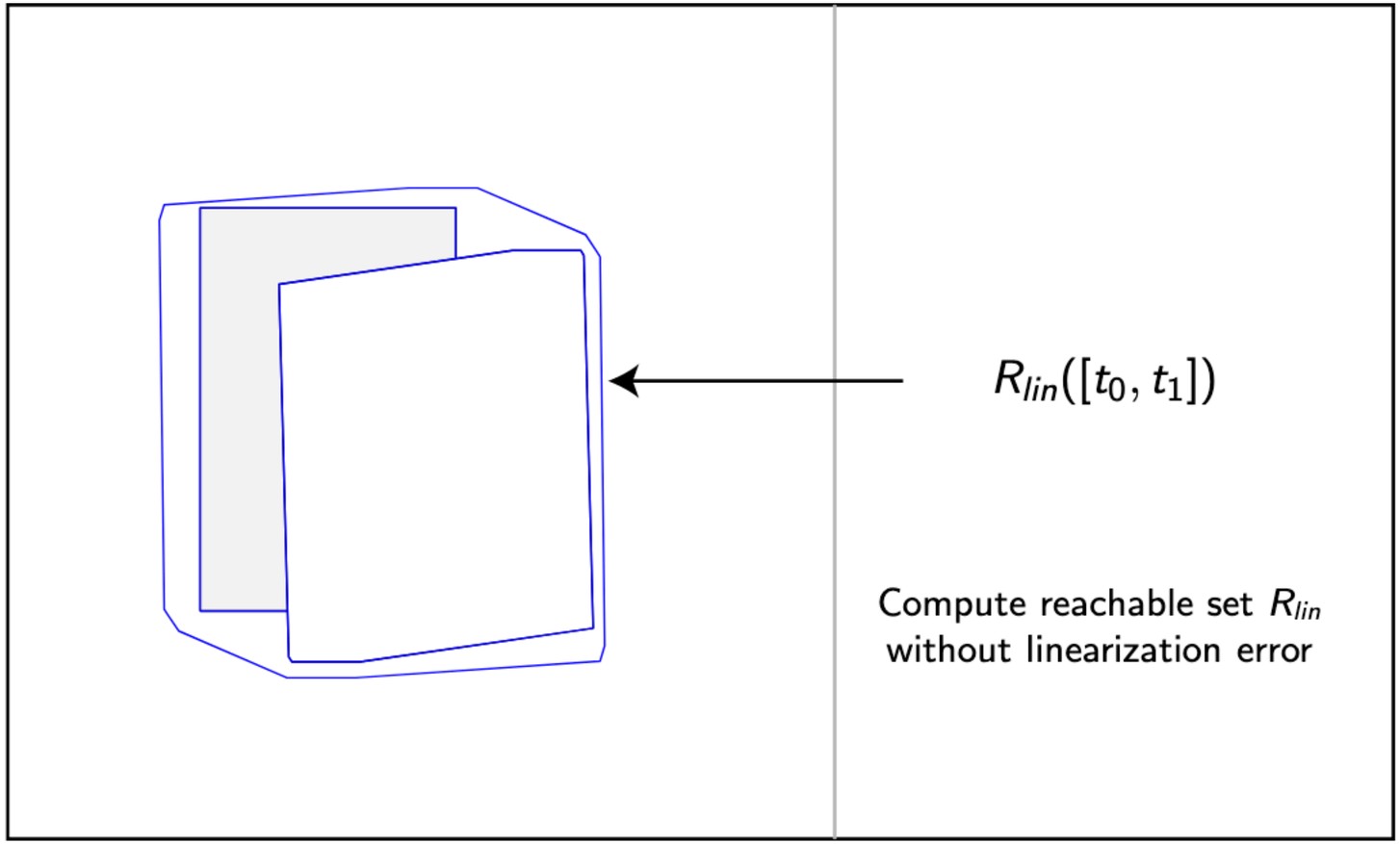
CORA with Nonlinear Systems

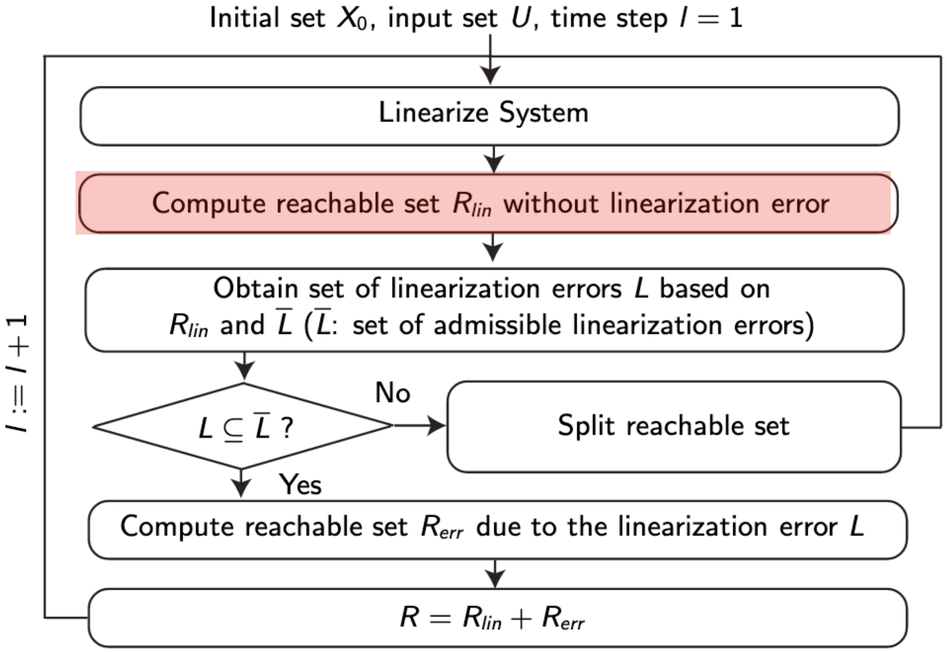




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

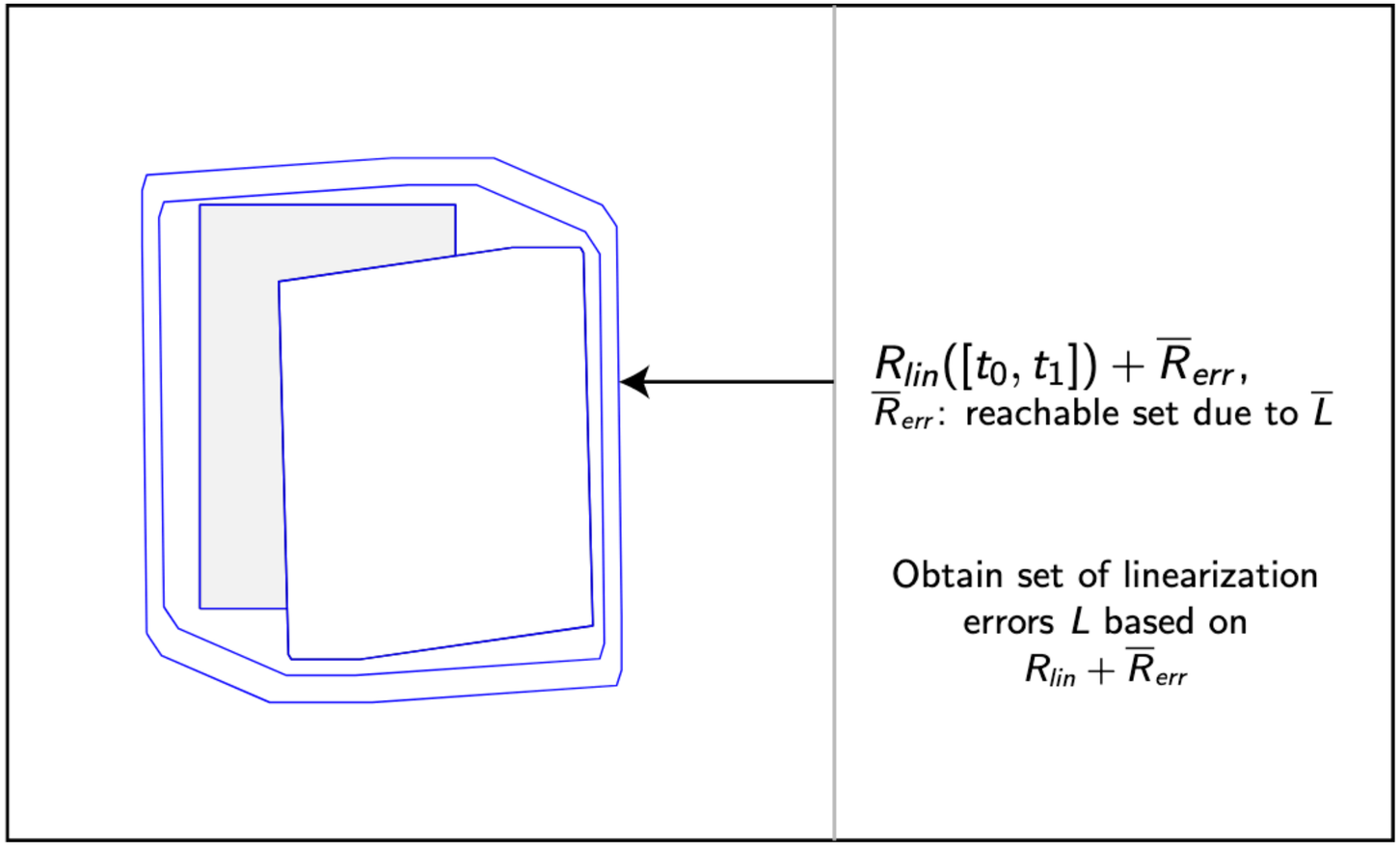
CORA with Nonlinear Systems

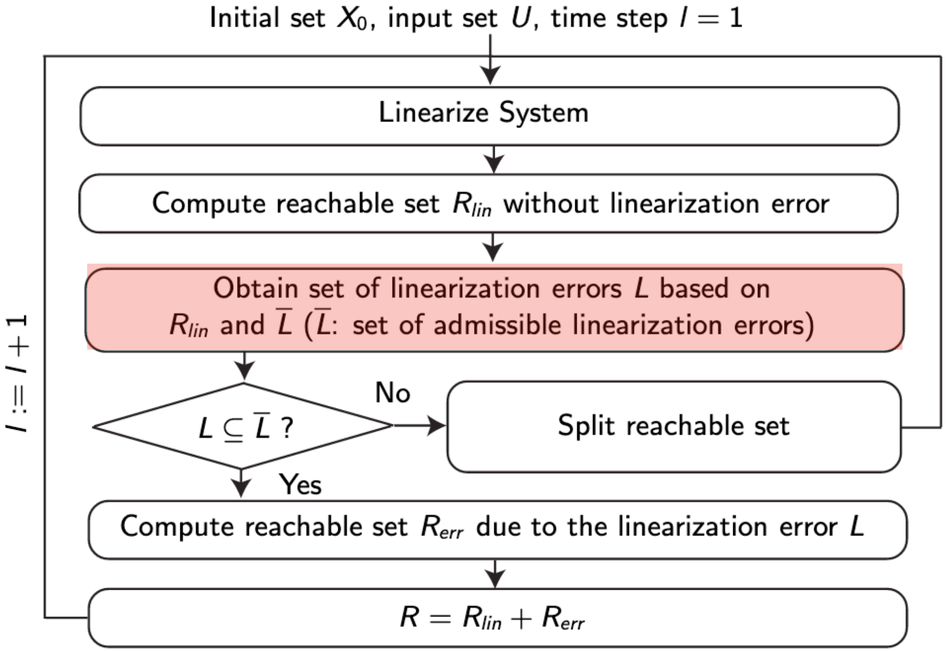




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

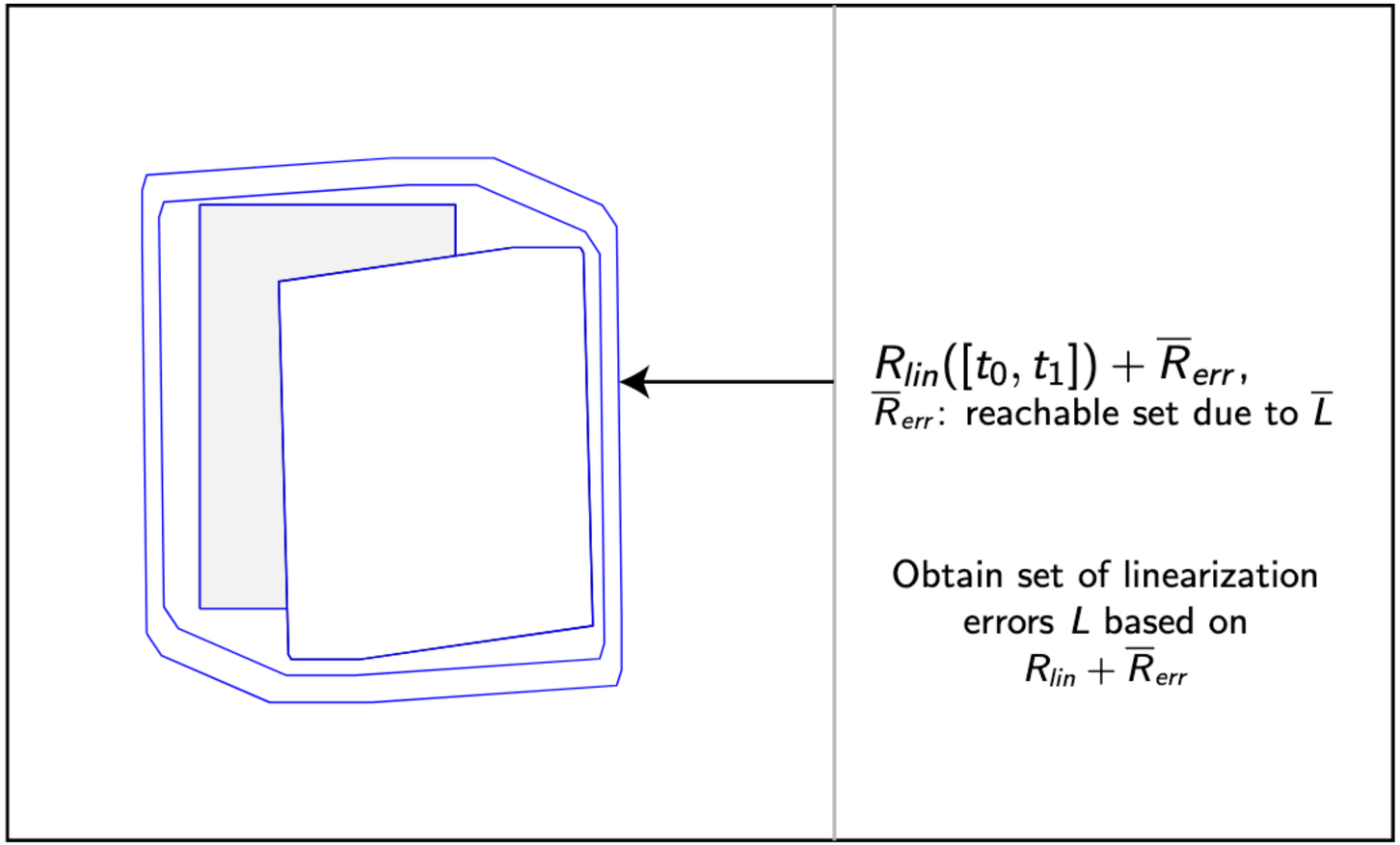
CORA with Nonlinear Systems

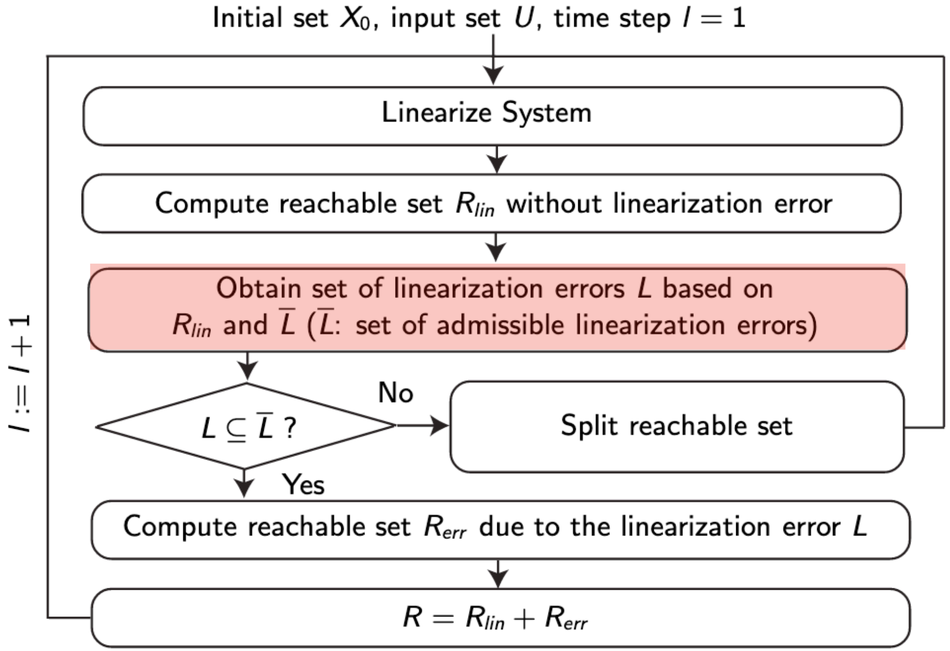




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

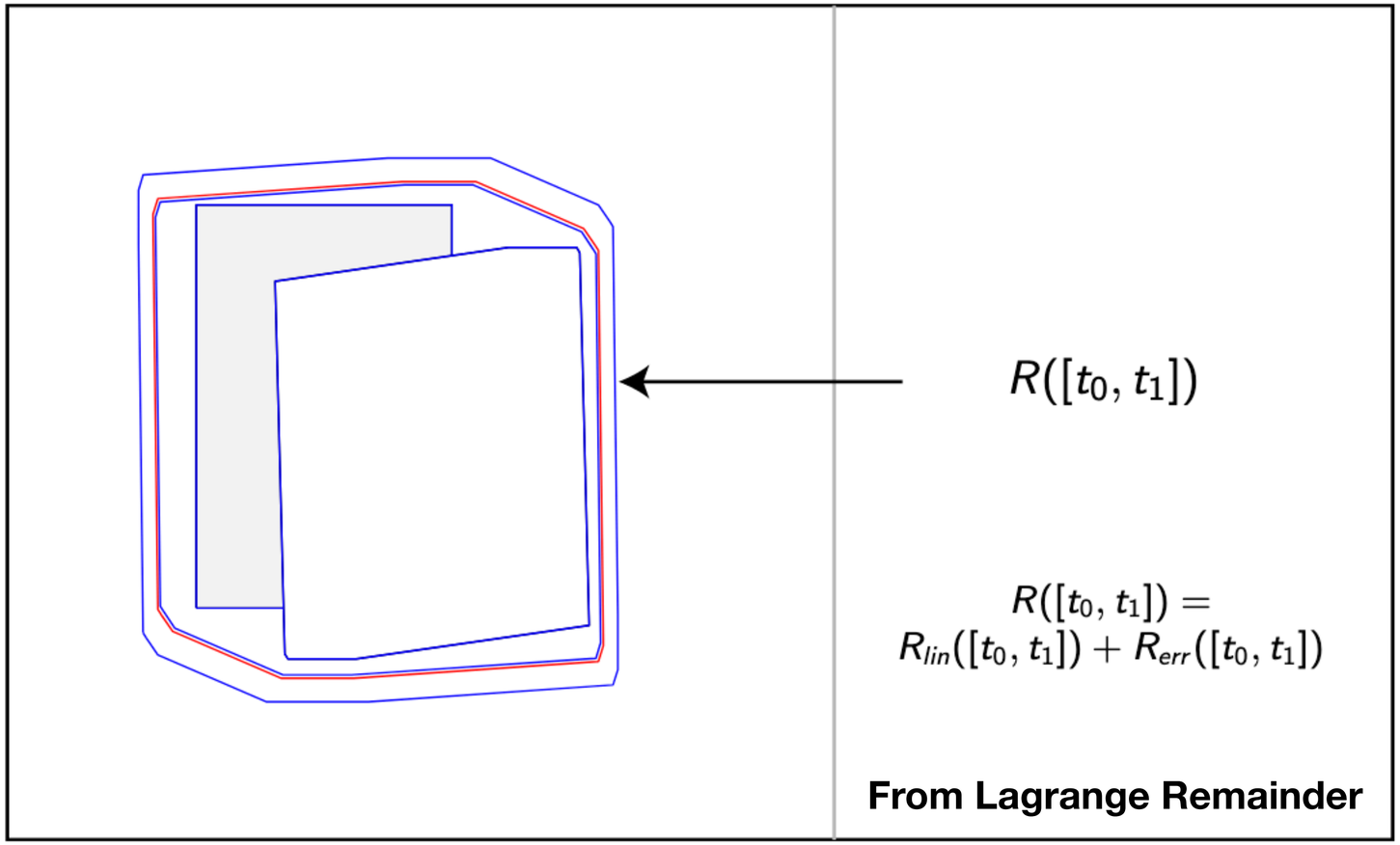
CORA with Nonlinear Systems

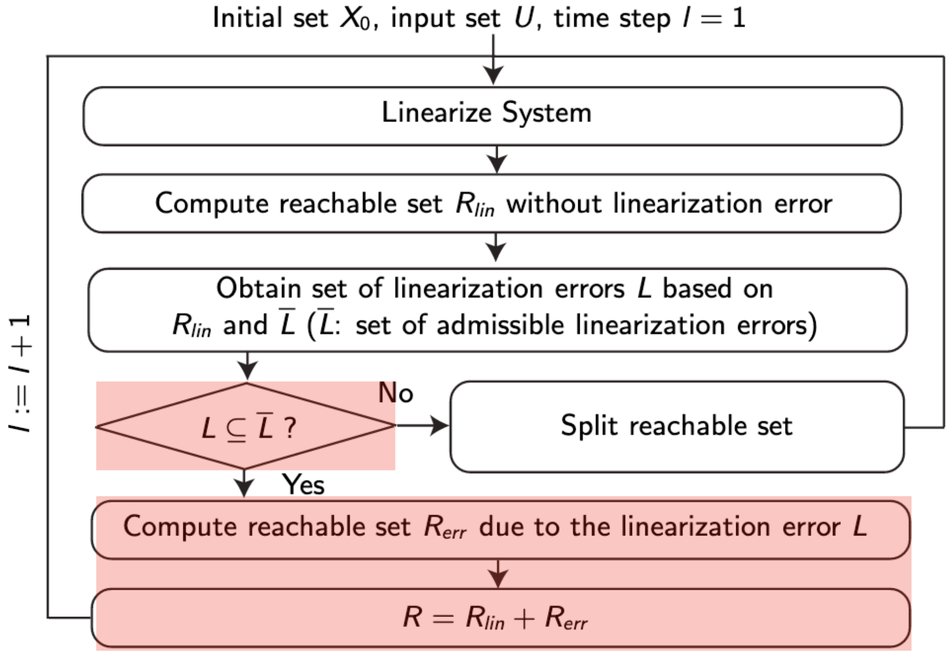




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

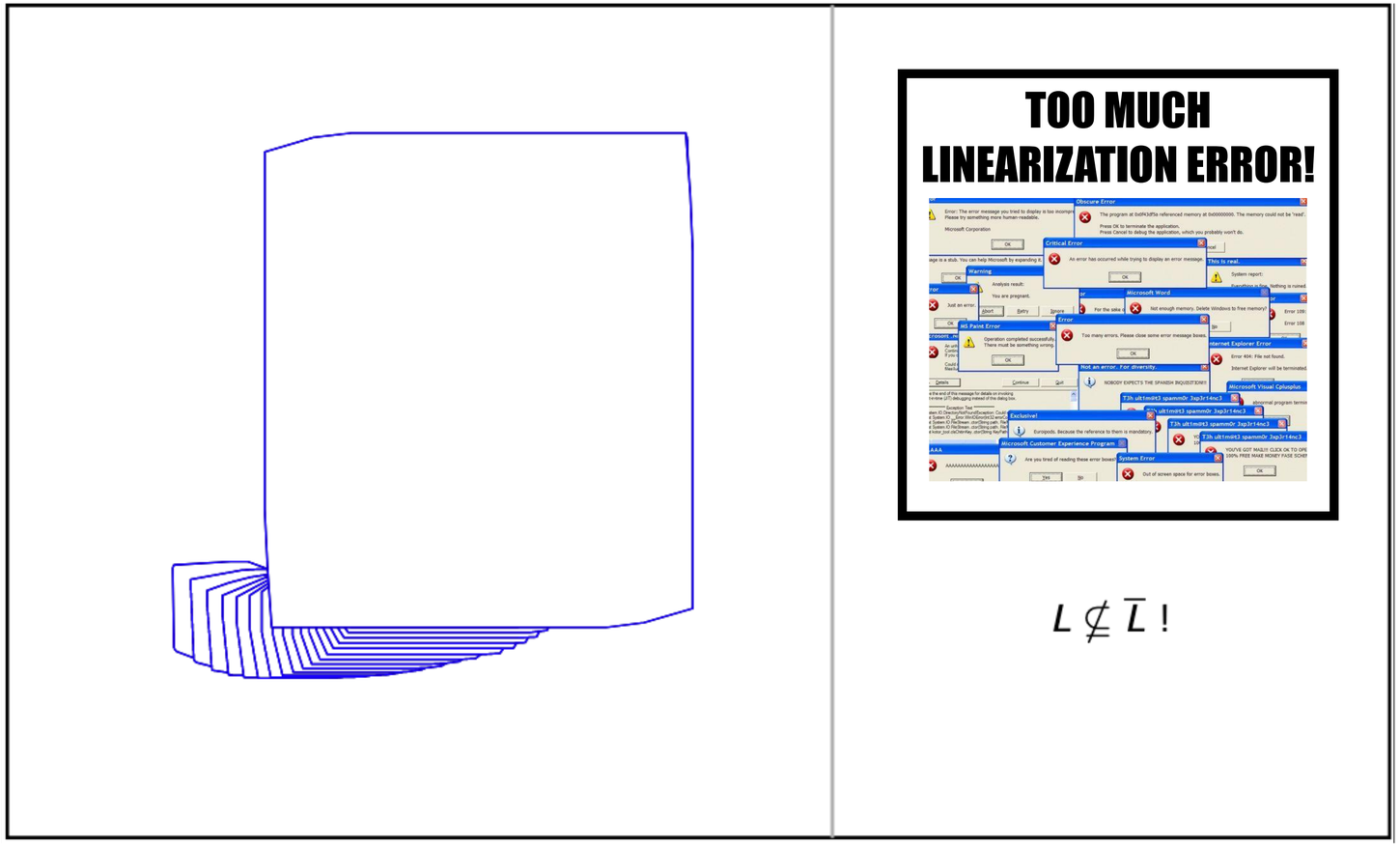
CORA with Nonlinear Systems

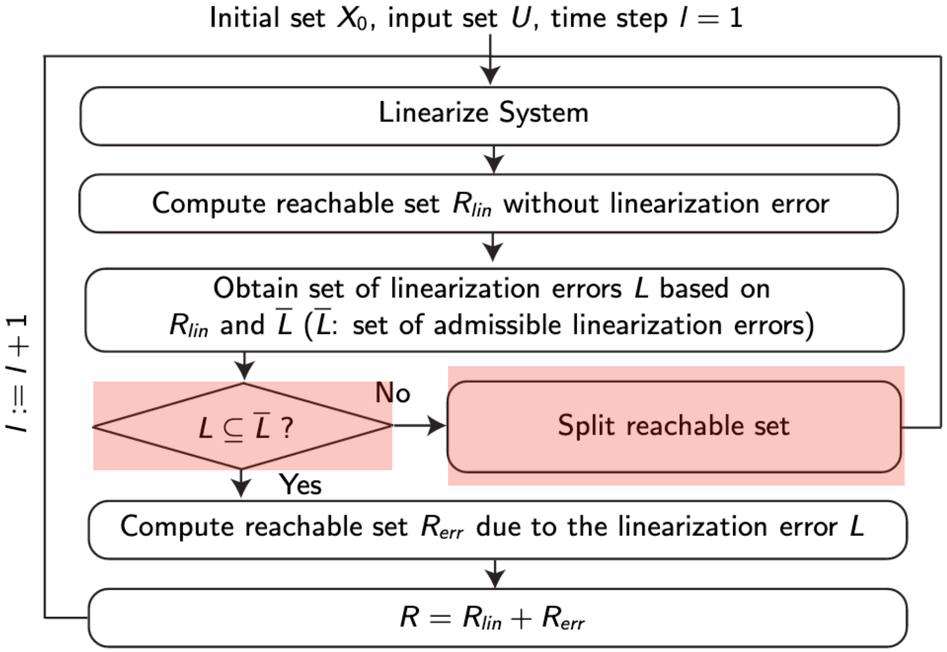




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

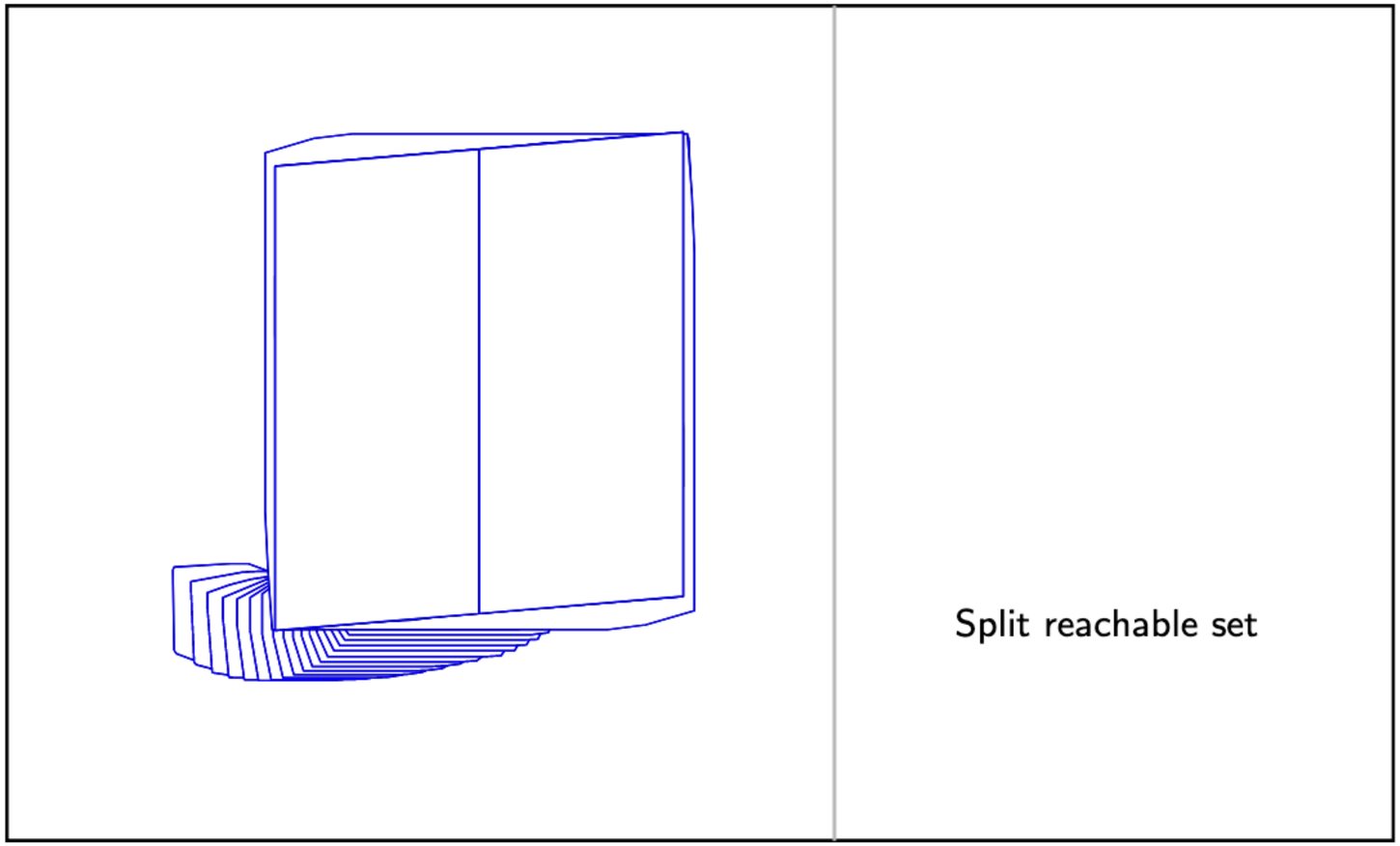
CORA with Nonlinear Systems

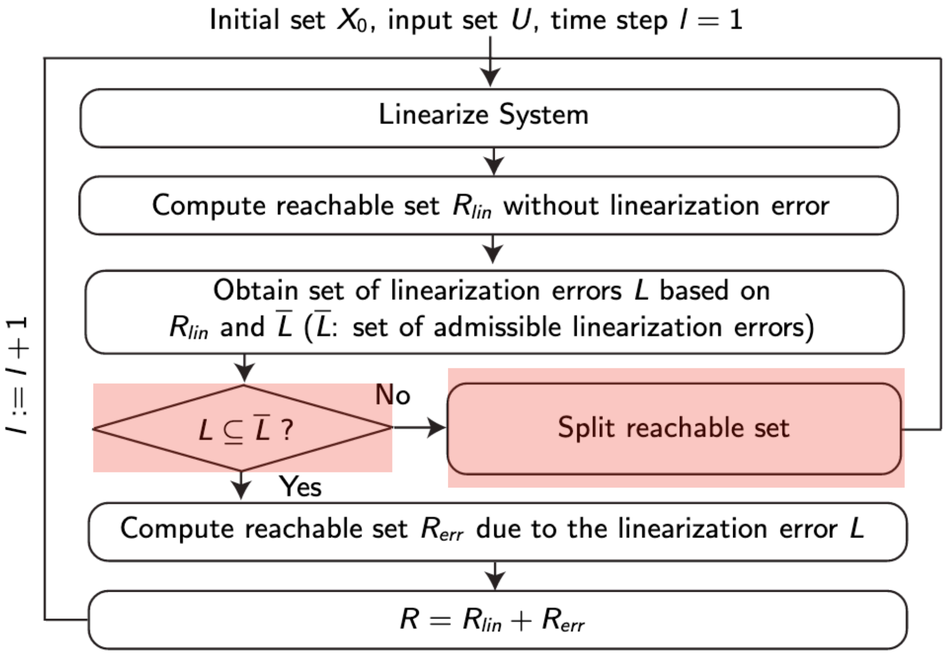




Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

Animation with Running Example

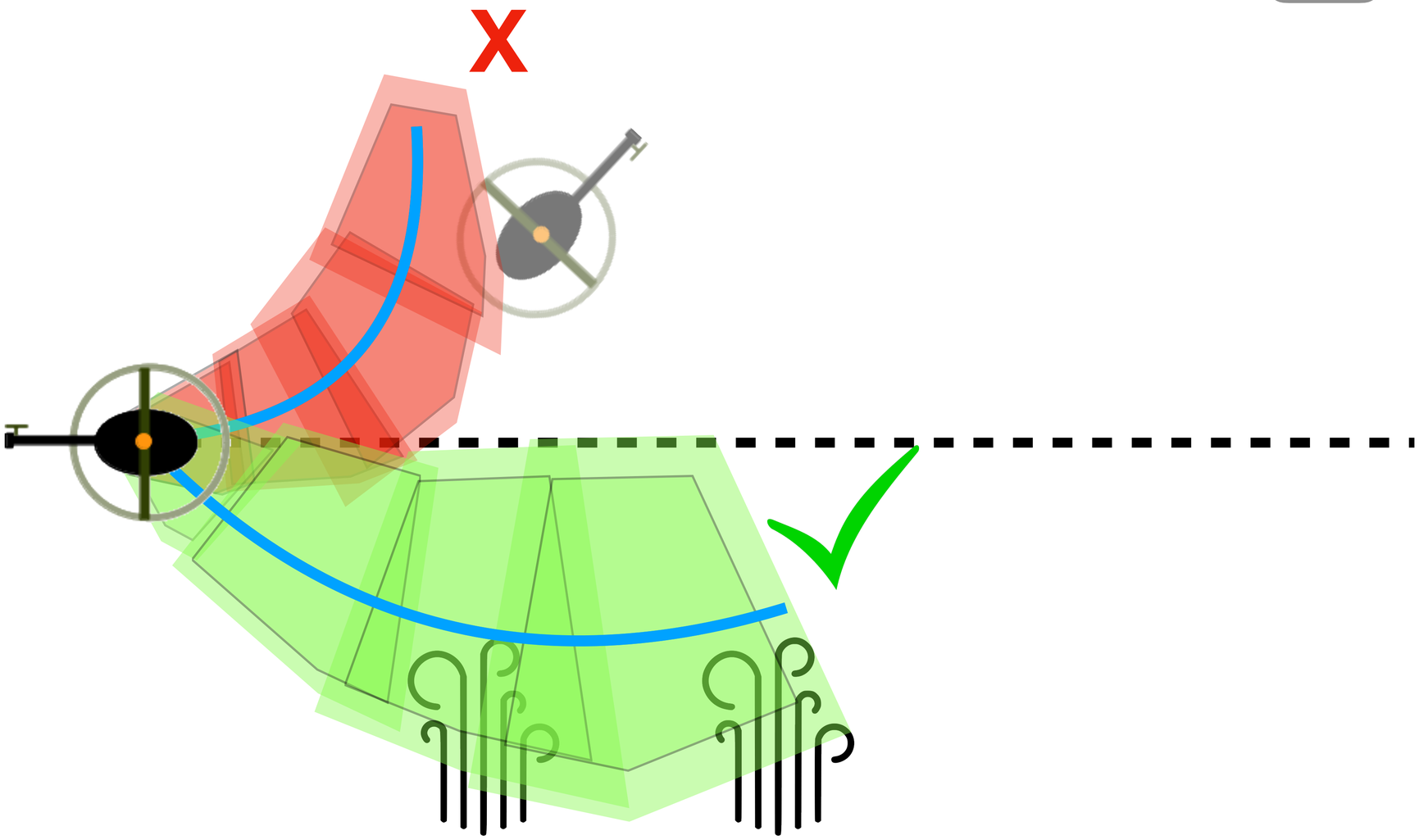




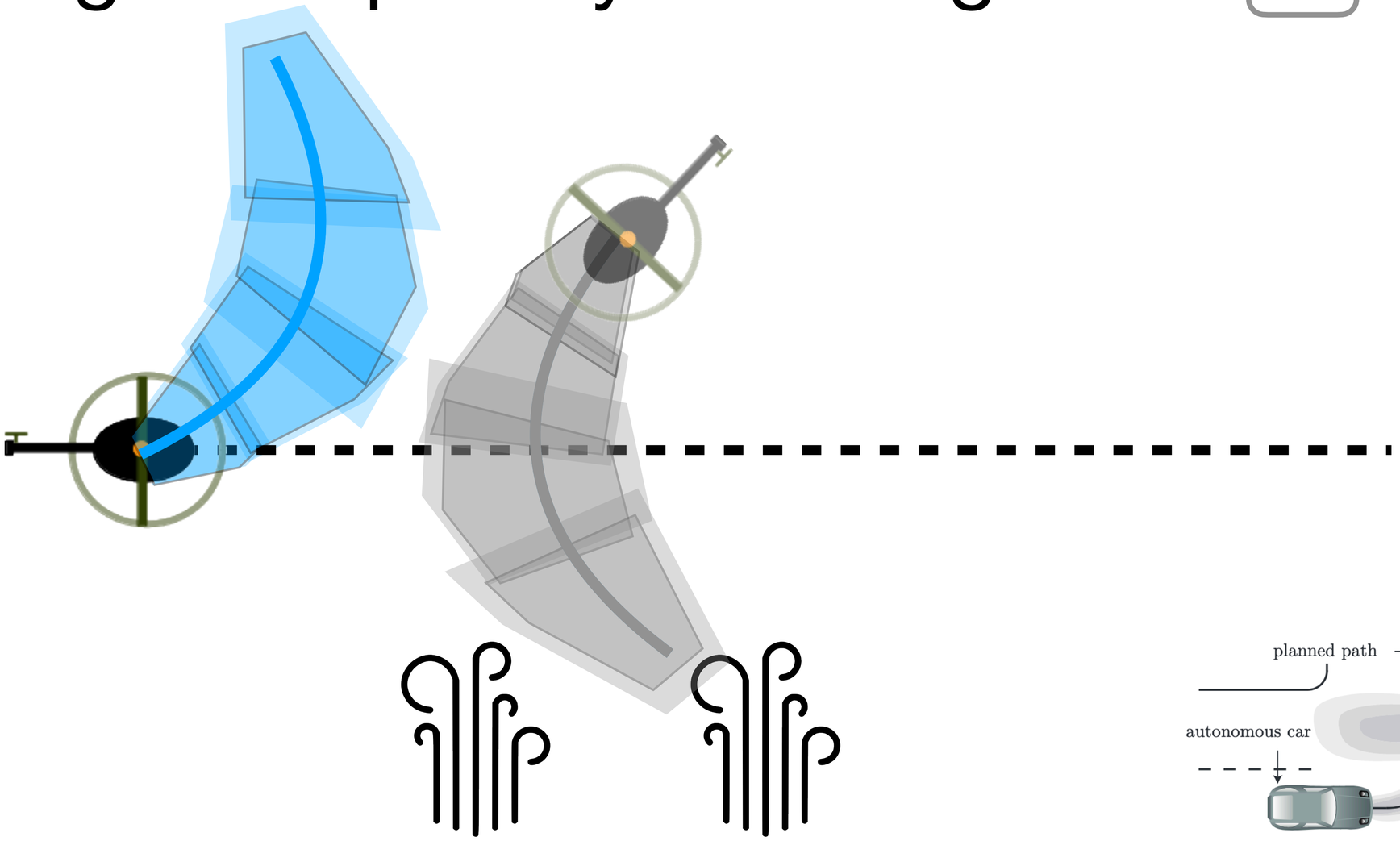
Note: Not done in autonomous driving as will lead to non-deterministic computation times.

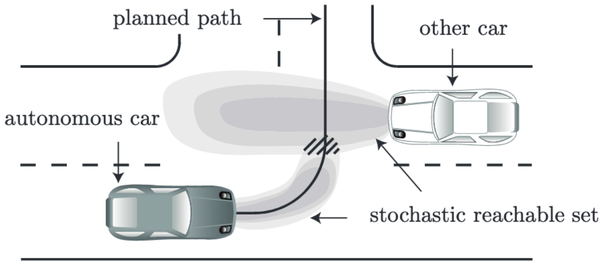
Adapted from Althoff's Slides on Reachability Analysis using Zonotopes (2010)

Running Example: Static Agents



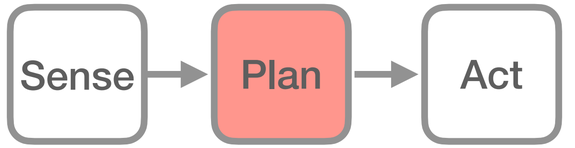
Running Example: Dynamic Agent





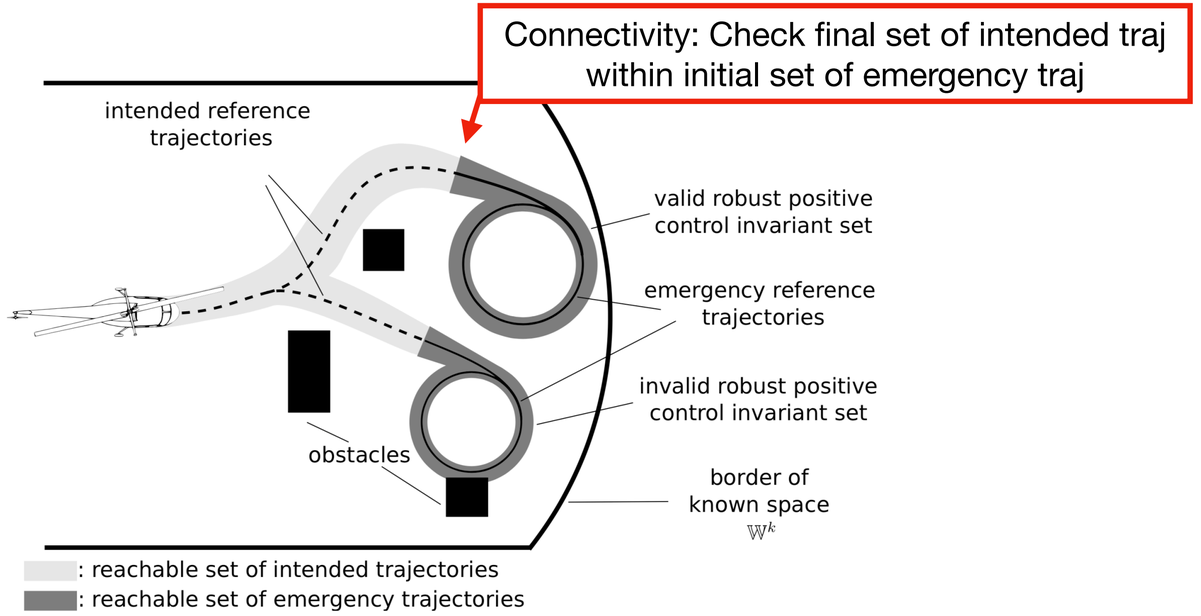
Extensions: Stochastic Reachability Analysis [Althoff 2010]

Verifying beyond planning horizon



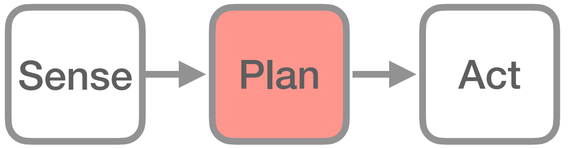
- Verified safe if it can enter free loiter circle (stay safe indefinitely)
- Verify intended trajectory (online) and emergency loiter circle (offline)

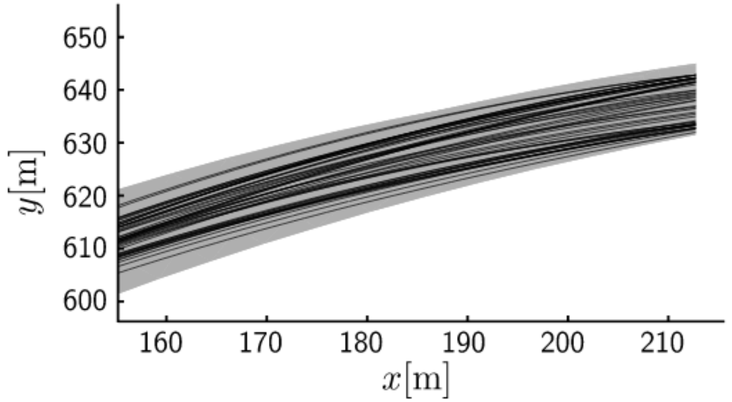
Safety Definition	Stay in known free space
Dynamics	Fixed wing model, P Control
Limits	Sensor range, actuator limits
Disturbances	Bounded Additive Velocity, Bounded Sensor Noise



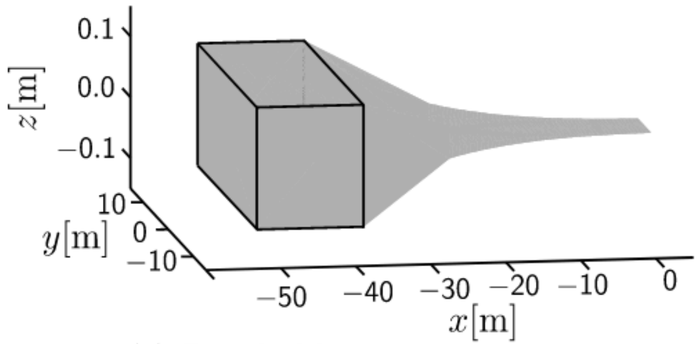
[Althoff, Althoff, Scherer 2015]

Verifying beyond planning horizon

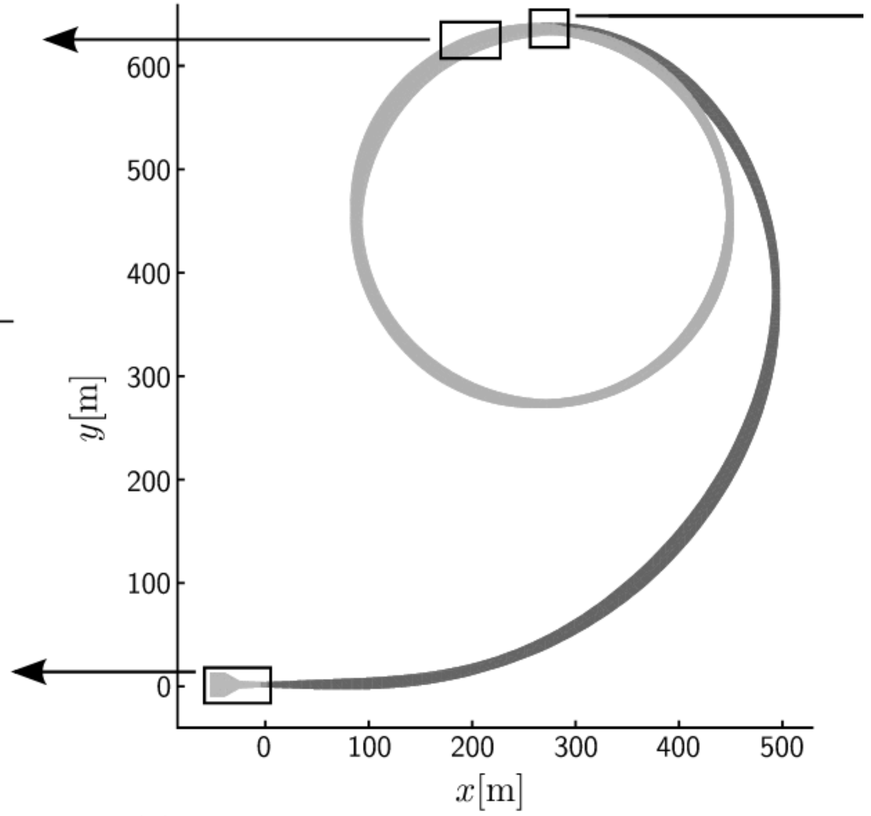




(b) Monte Carlo simulation



(c) Reachable set snap trajectory



(a) Reachable set entire emergency maneuver

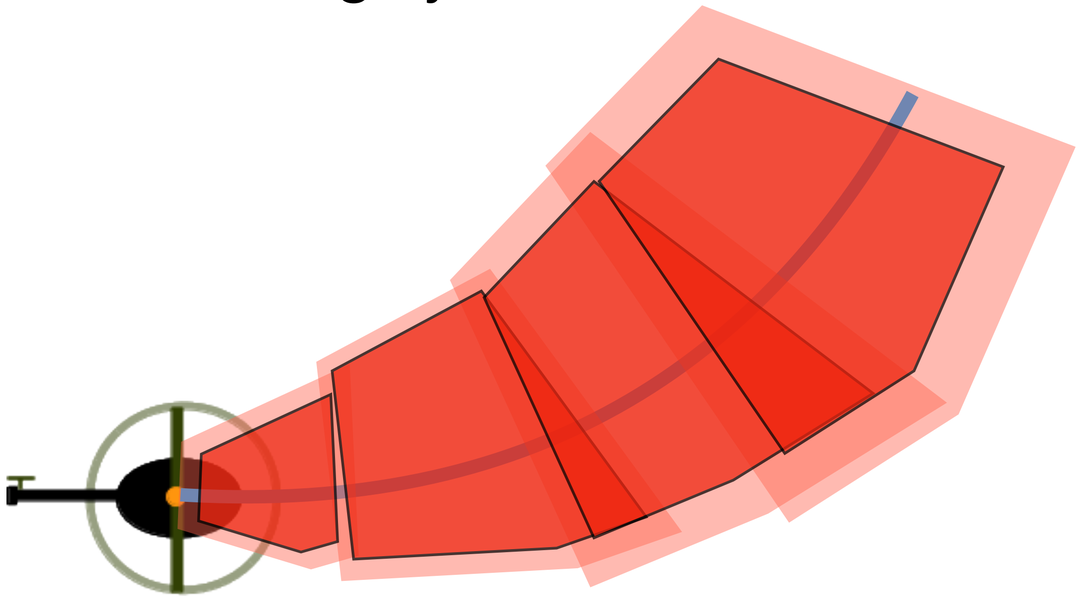
[Althoff, Althoff, Scherer 2015]

CoRA Takeaways

- **Key Insight:** Overapproximate linear dynamics
  (Linear + linearization error + uncertainties + curve traj)

- **Pros:** Scales well with # state dimension

- **Cons:** Does not work well in highly nonlinear cases —> much splitting

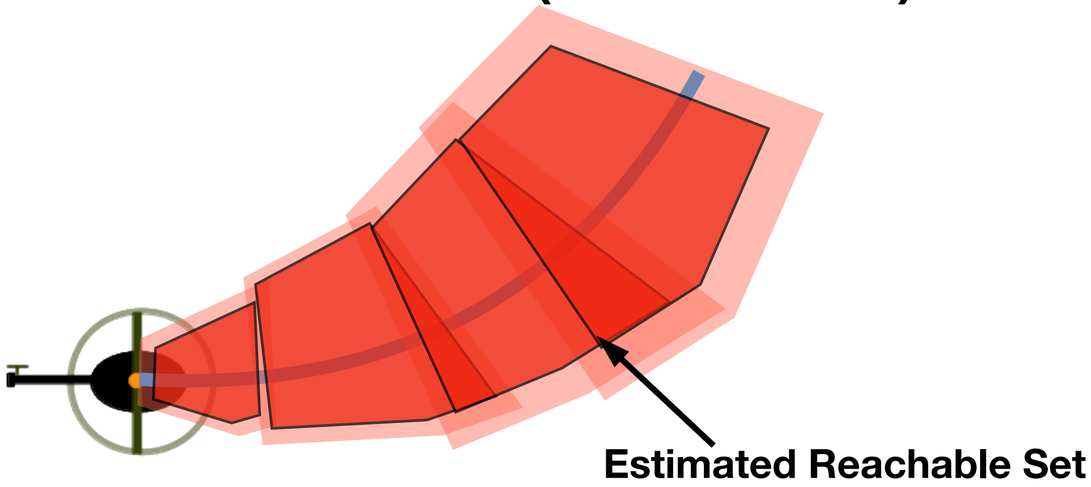


Outline

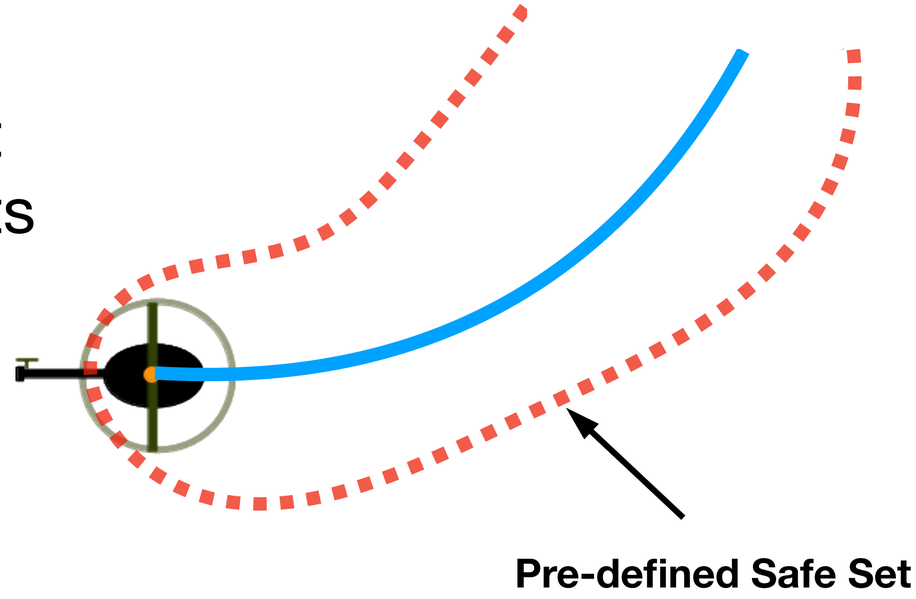
- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

Formal Tool 2 : Control Barrier Functions (Intuition)

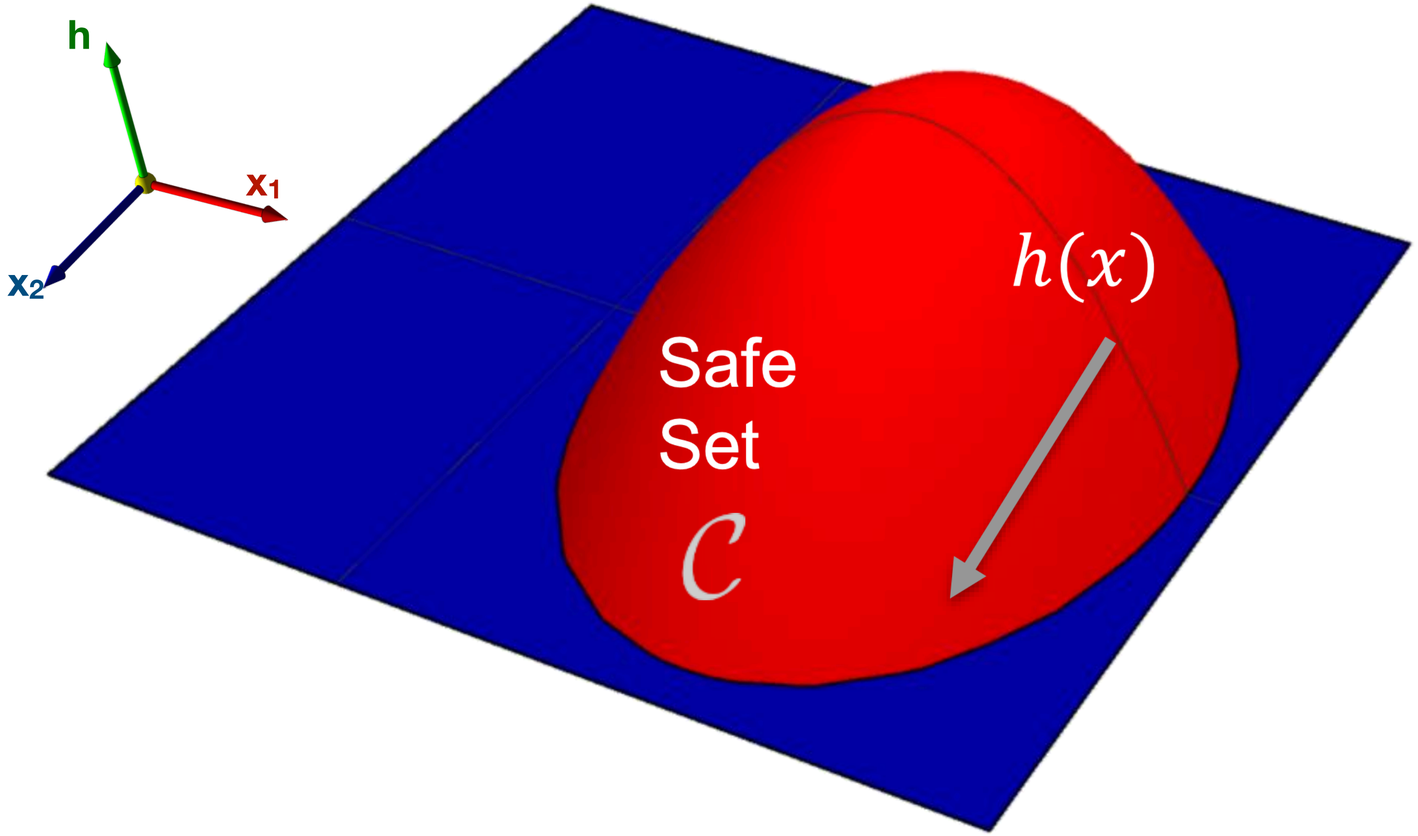
- Reachable Sets: Estimates future states to remove unsafe set of actions



- Control Barrier Function: Pre-defines safe set and imposes corresponding safety constraints



Tool 2 : Control Barrier Functions



Dynamics

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

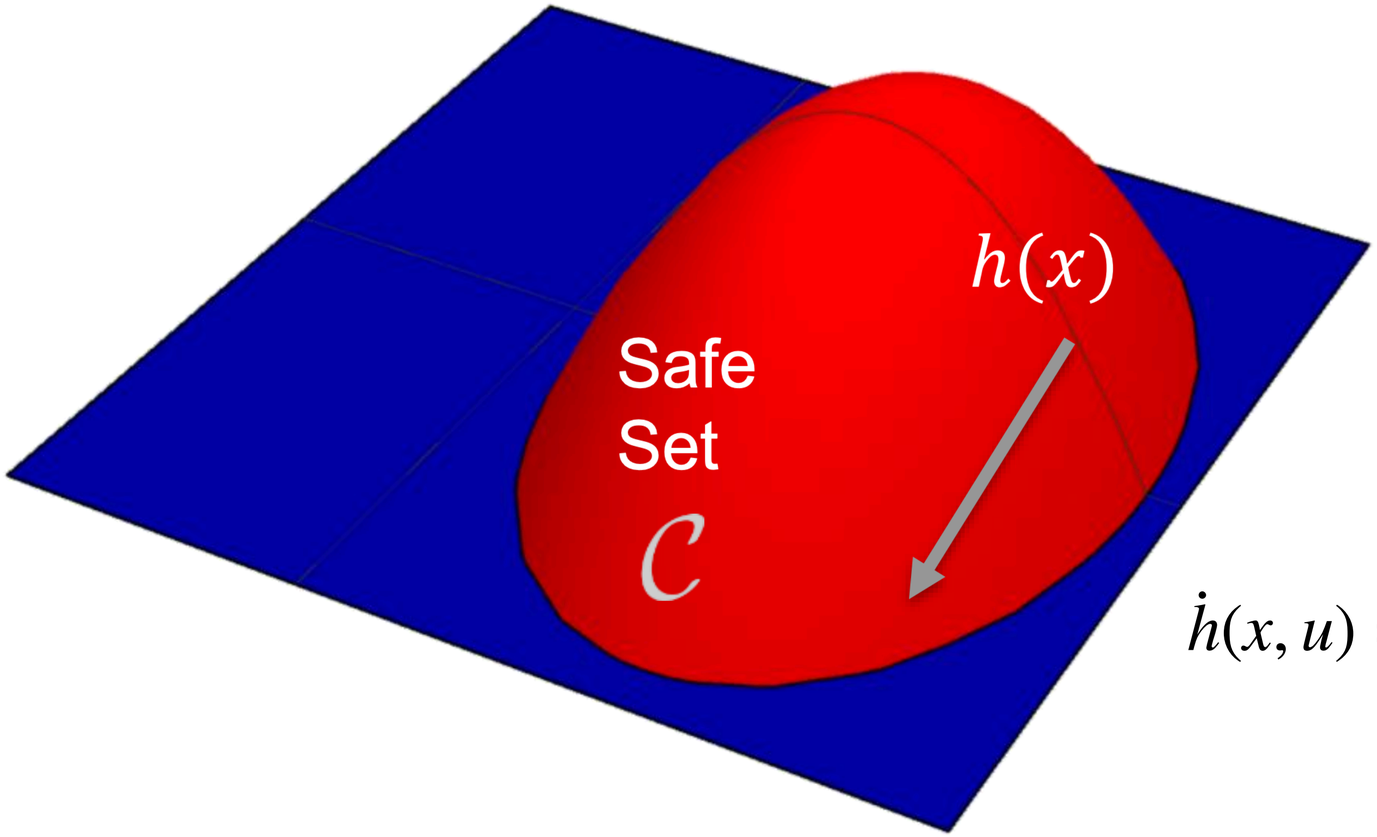
Safe Set (safe if above zero)

$$\mathscr{C} = \{x \in R^n \ : \ h(x) \geq 0\}$$

$$h : \mathbb{R}^n \longrightarrow \mathbb{R}$$

What does it mean to stay safe?

Tool 2 : Control Barrier Functions



Safety Check (Nagumo Theorem):

When we don't have controller

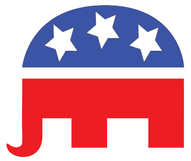
$$\dot{h}(x) = \frac{\partial h}{\partial x}(x)f(x) \geq 0, \forall x \in \partial\mathscr{C}$$

Extension to whole set:

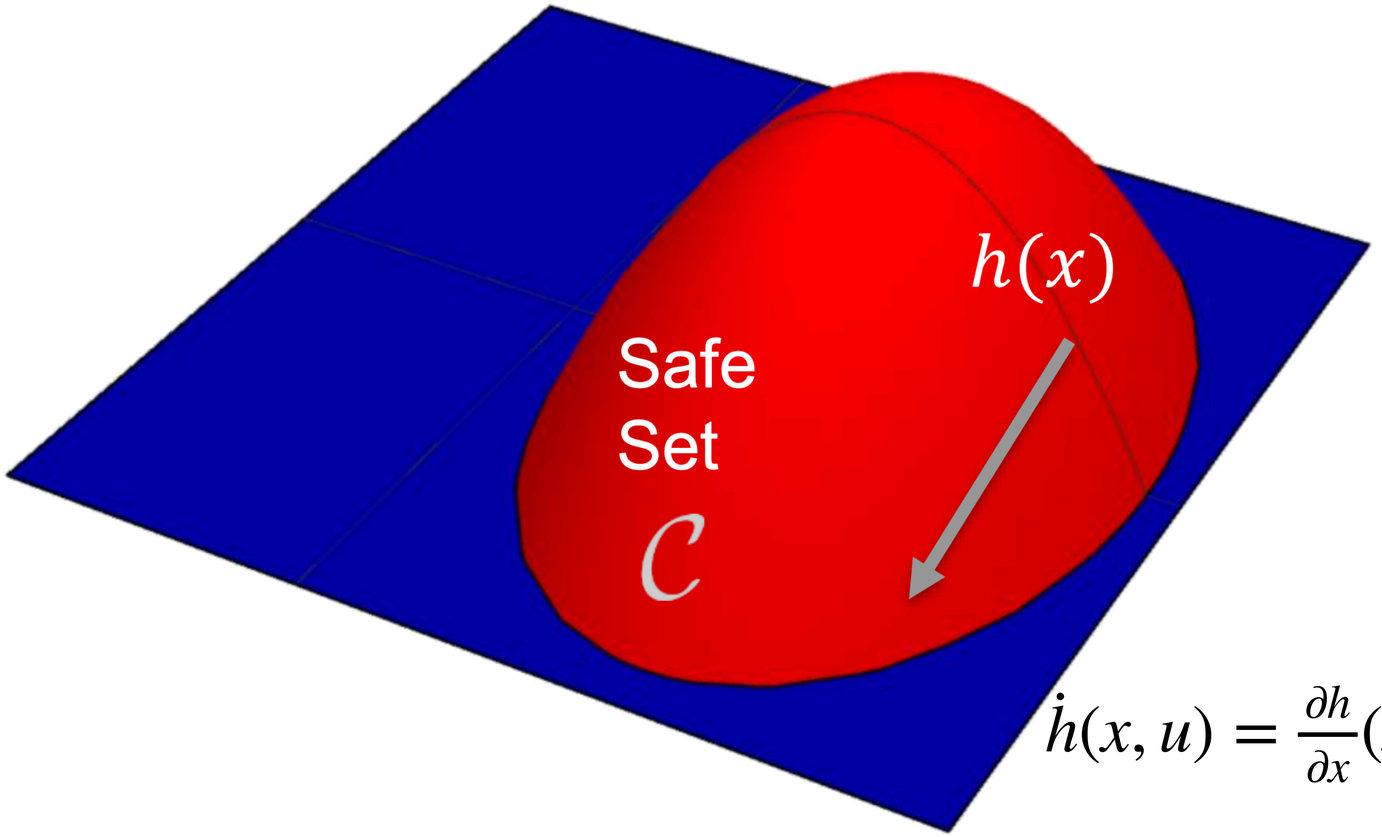
(Control Barrier Function-CBF)

$$\dot{h}(x,u) = \frac{\partial h}{\partial x}(x)(f(x) + g(x)u) \geq 0, \forall x \in \mathscr{C}$$

Problem: Too Conservative!



Tool 2 : Control Barrier Functions (Intuition)



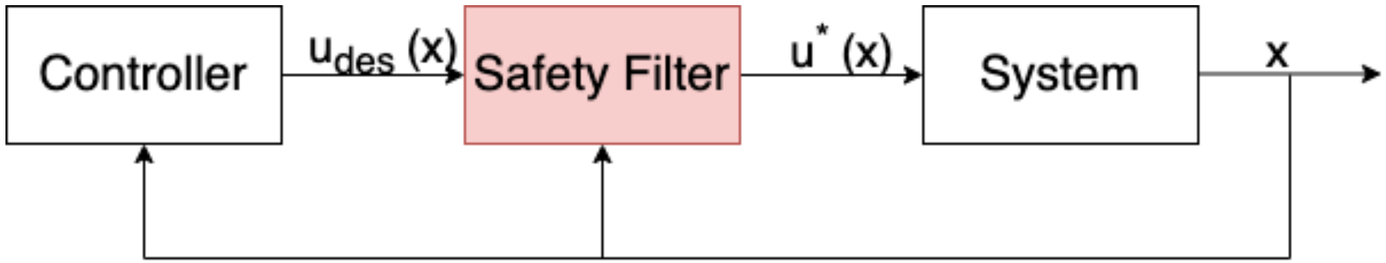
Solution: Allow $\dot{h}$ to go negative

$$h(x) = 0$$

$$\dot{h}(x,u) = \frac{\partial h}{\partial x}(x)(f(x) + g(x)u) \geq -\gamma(h(x)), \forall x \in \mathscr{C}(1)$$

Now what?!!!

Tool 2 : Control Barrier Functions (Intuition)



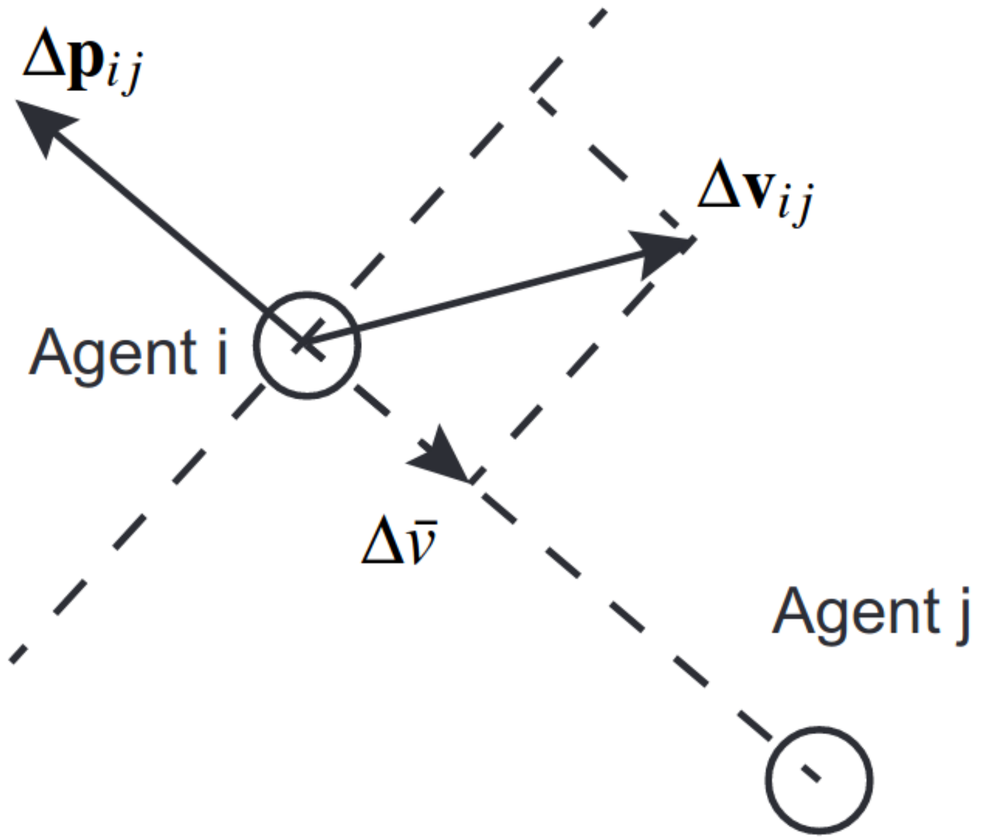
Optimization Problem: (Safety Filter)

$$u^* = \arg\min ||u - u_{des}||^2$$

$$s.t \quad \dot{h}(x,u) \geq -\gamma(h(x))$$

Computationally efficient since it's a QP problem
Minimally Invasive Control: Only modify if unsafe

Tool 2 : Control Barrier Functions (Example)



Dynamics: (Double integrator)

$$\begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\mathbf{v}}_i \end{bmatrix} = \begin{bmatrix} 0 & I_{2\times 2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{v}_i \end{bmatrix} + \begin{bmatrix} 0 \\ I_{2\times 2} \end{bmatrix} \mathbf{u}_i$$

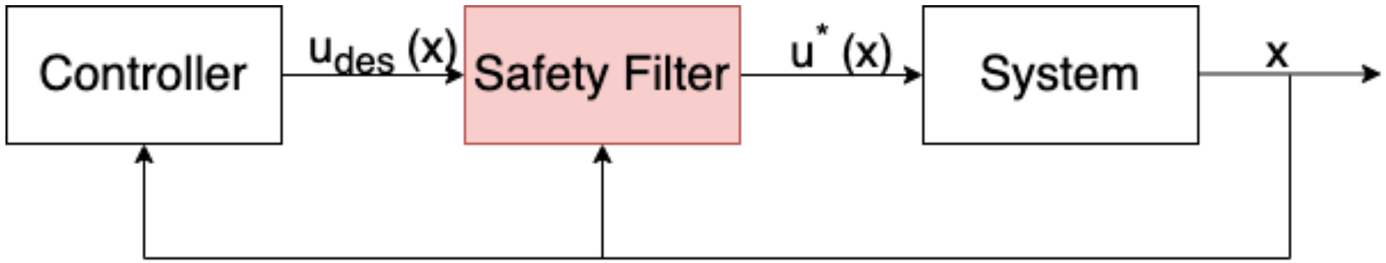
Safe Set: Defining safe set as

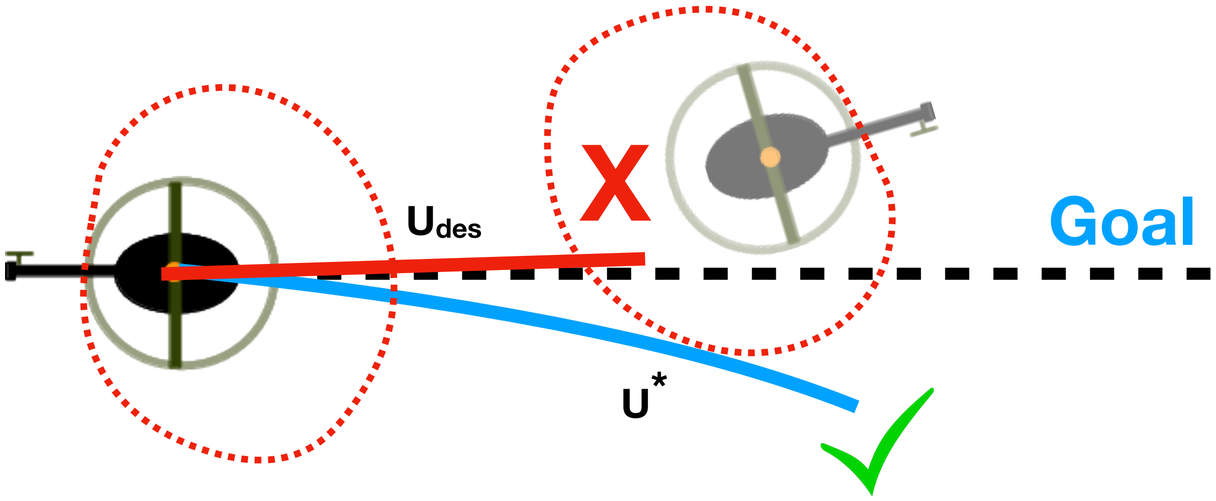
$$||\Delta p|| + \int_T \Delta \bar{v}(t) dt \geq D_s$$

Velocity-based Safety Distance

$$h(p, v) = \sqrt{(||\Delta p|| - D_s)} + \frac{\Delta p^T}{||\Delta p||} \Delta v$$

Tool 2 : Control Barrier Functions (Example)





Optimization Problem: (Safety Filter)

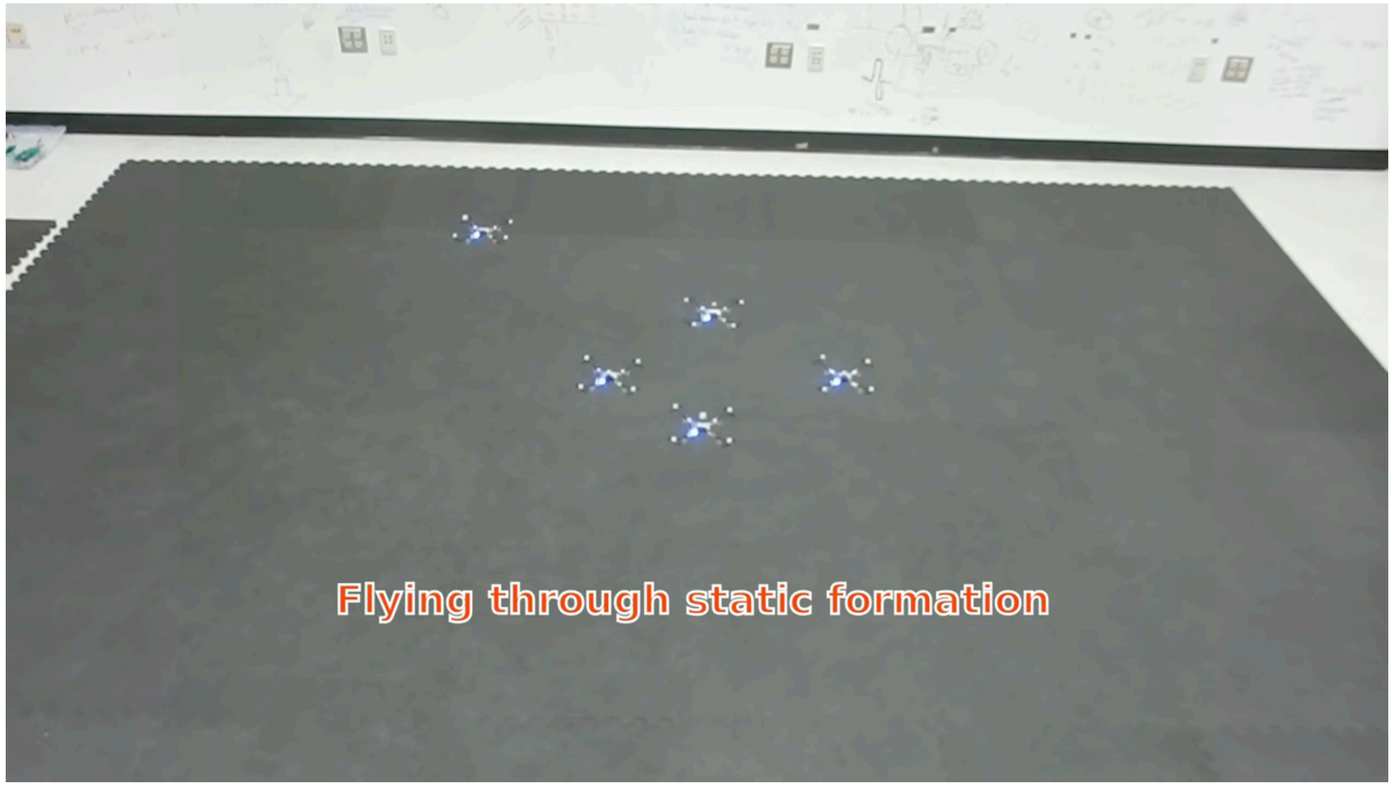
$$u^* = \arg\min ||u - u_{des}||^2$$

$$s.t \quad A_{ij}u \leq b_{ij} \quad \forall i \neq j$$

$$A = [0,\ldots,-\Delta p^T,\ldots,\Delta p^T,\ldots,0]$$

$$b = \gamma h^3 ||\Delta p|| - \frac{(\Delta v^T \Delta p)^2}{||\Delta p||^2} + \frac{\Delta v^T \Delta p}{\sqrt{||\Delta p|| - D_s}} + ||\Delta v||^2$$

Application - Aggressive Multi-Robot



https://www.youtube.com/watch?time_continue=23&v=pmZZzIXIIsc&feature=emb_logo

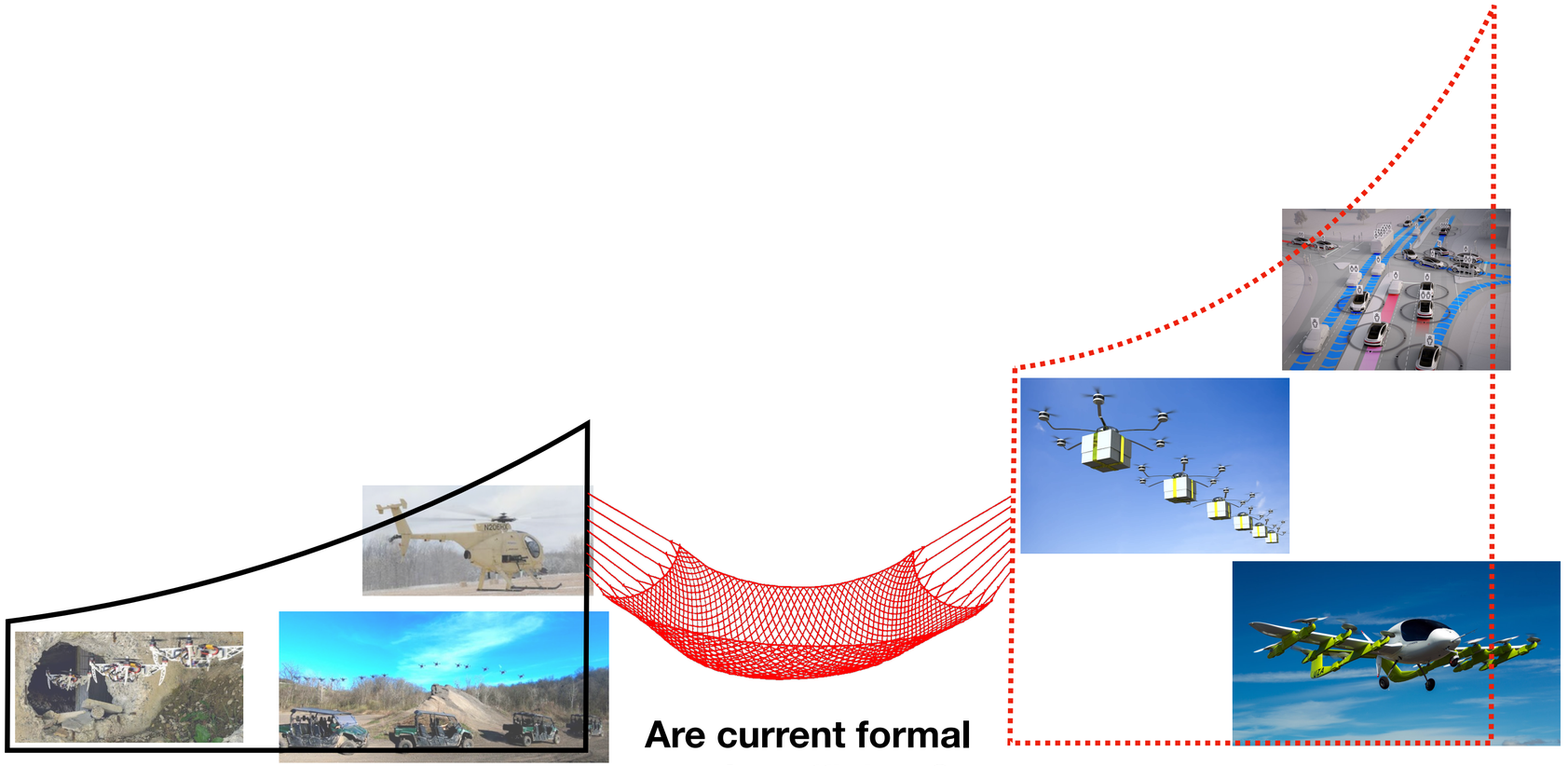
CBF Takeaways

- **Key Insight:** Predefine the safe set and always stay in safe set
- **Implementation:** Minimally invasive optimization i.e. It only modifies the nominal control if system is unsafe
- **Pros:** CBF is used to verify safety properties of a set without doing the difficult task of computing the systems reachable set i.e. computationally more affordable
- **Cons:** Doesn't work when assumptions are violated and deadlock in symmetrical scenarios. Designing a custom safe set for each application.

Outline

- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

How do formal guarantees hold in real world?



Are current formal tools sufficient?

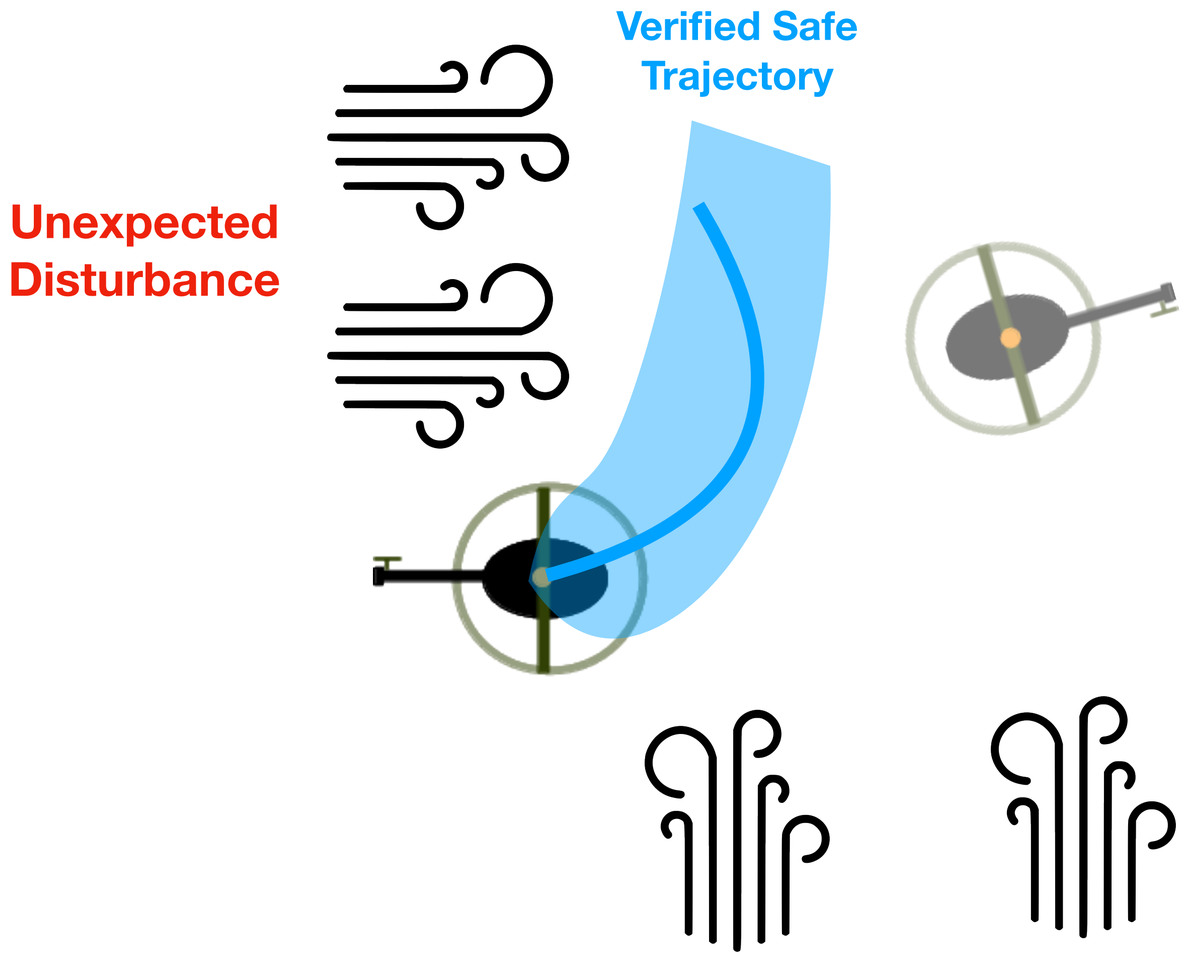
Look mom,
No cage!

Look mom,
No safety pilot!

Common Model vs. Real World

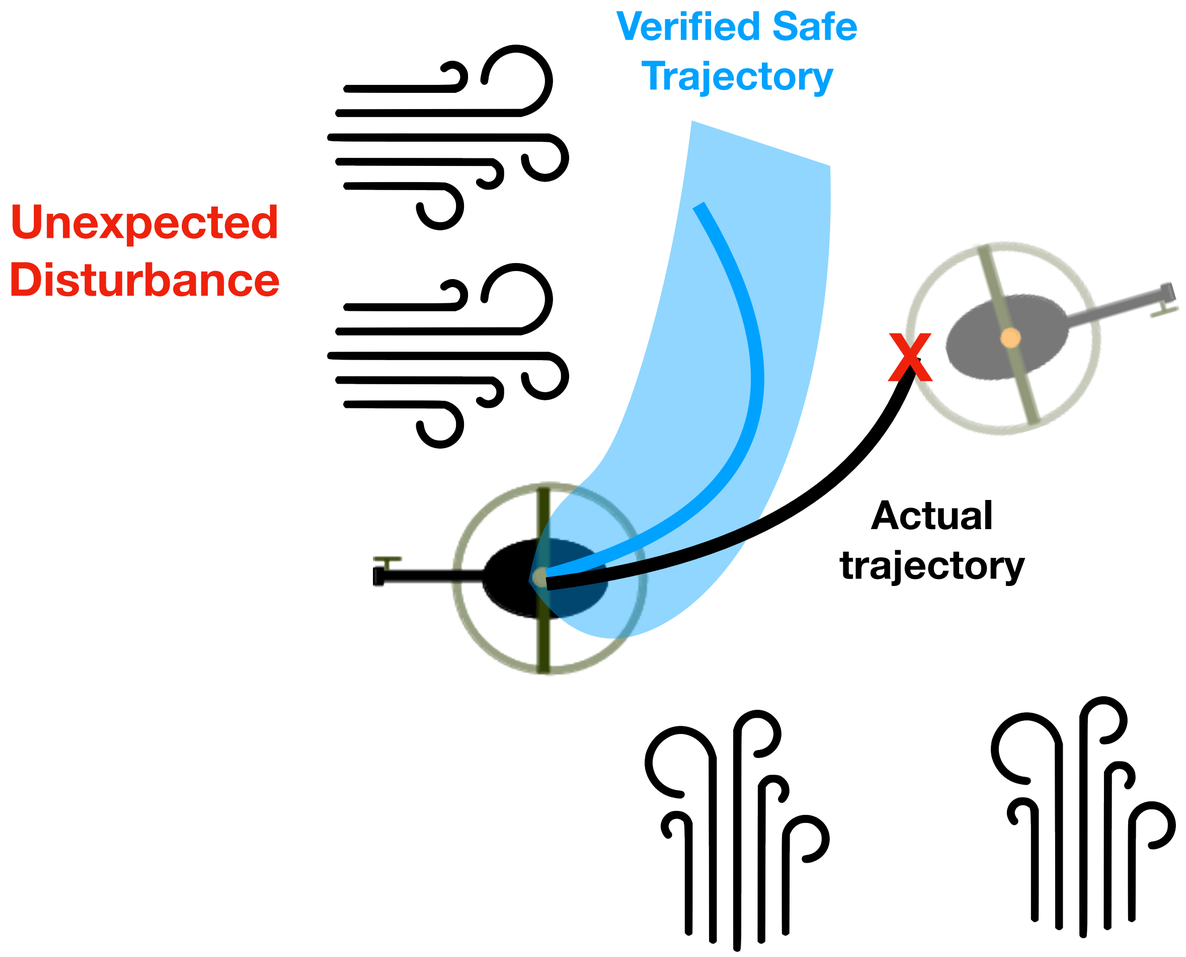
Ingredient	Common Abstracted Model	Real-world Deviation
Vehicle Dynamics	2nd-order integrator	Nth-Order
External Disturbance	Bounded additive wind	State-based disturbance (e.g. drag) Sudden gusts
State Estimation Error	Gaussian Noise, Often None	Noisier with dynamic objects

Violated Models - Preset Bounded Disturbance



What if new disturbance outside of bounds?

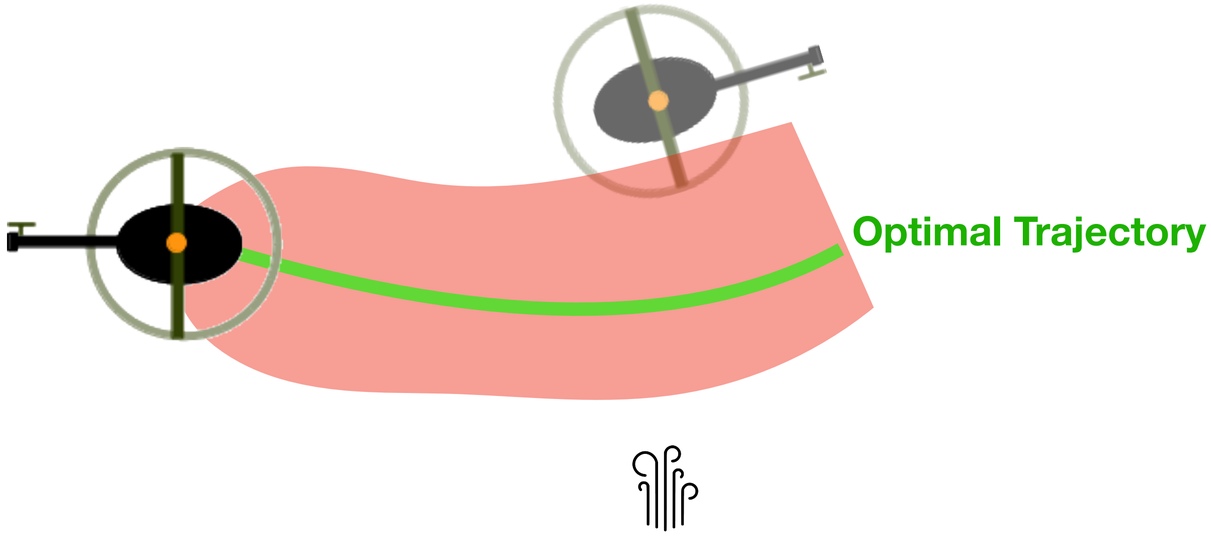
Violated Models - Preset Bounded Disturbance



What if new disturbance outside of bounds?

Leaves reachable set. **Unsafe!**

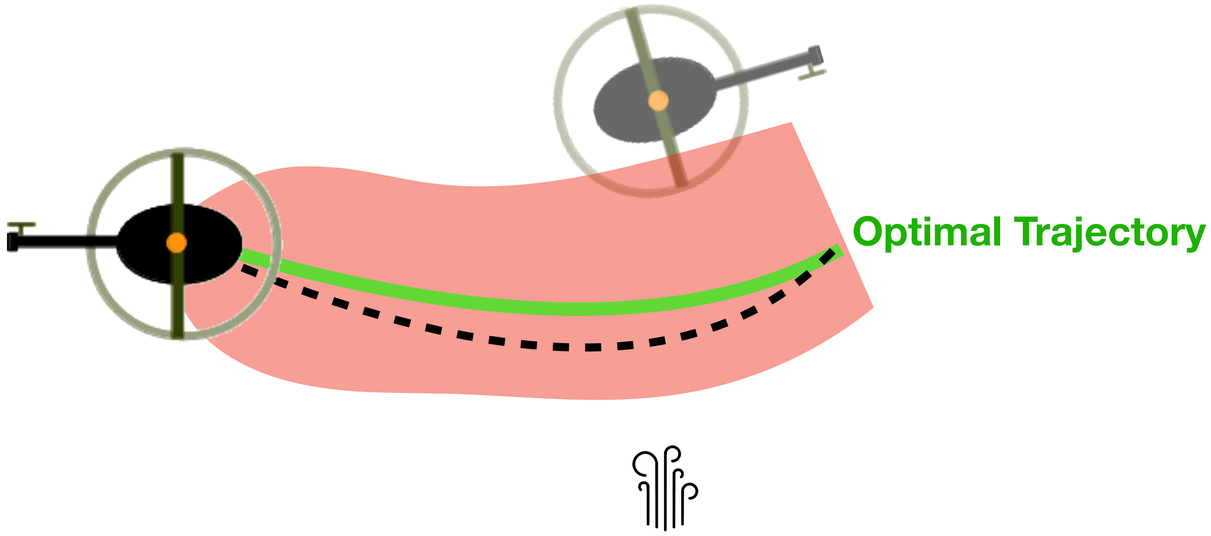
Violated Models - Preset Bounded Disturbance



Why not verify for a higher disturbance?
What if disturbance much smaller?

Performance loss!

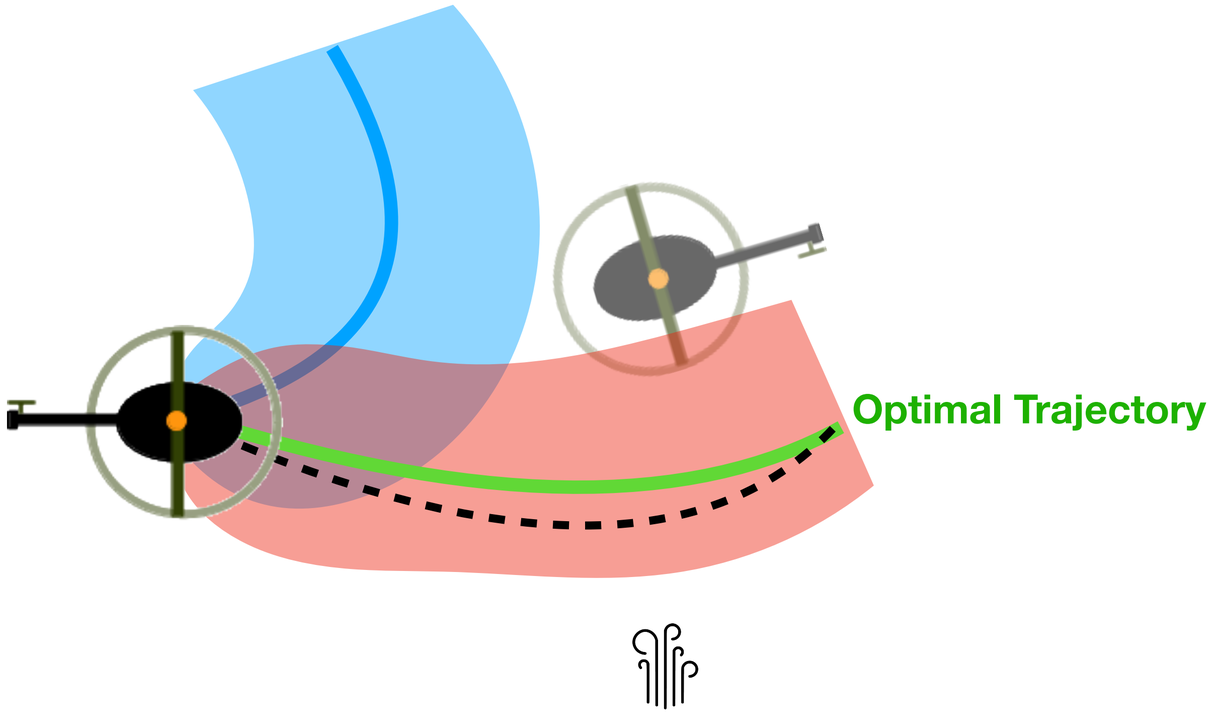
Violated Models - Preset Bounded Disturbance



Why not verify for a higher disturbance?
What if disturbance much smaller?

Performance loss!

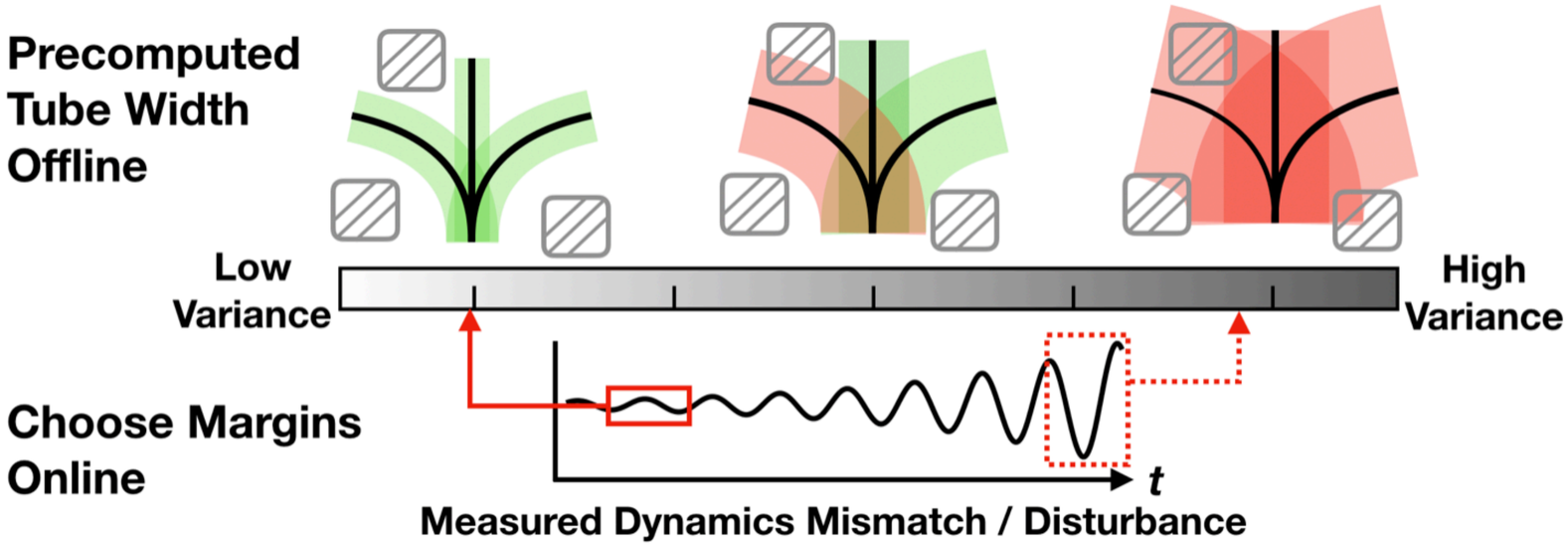
Violated Models - Preset Bounded Disturbance



Why not verify for a higher disturbance?
What if disturbance much smaller?

Performance loss!

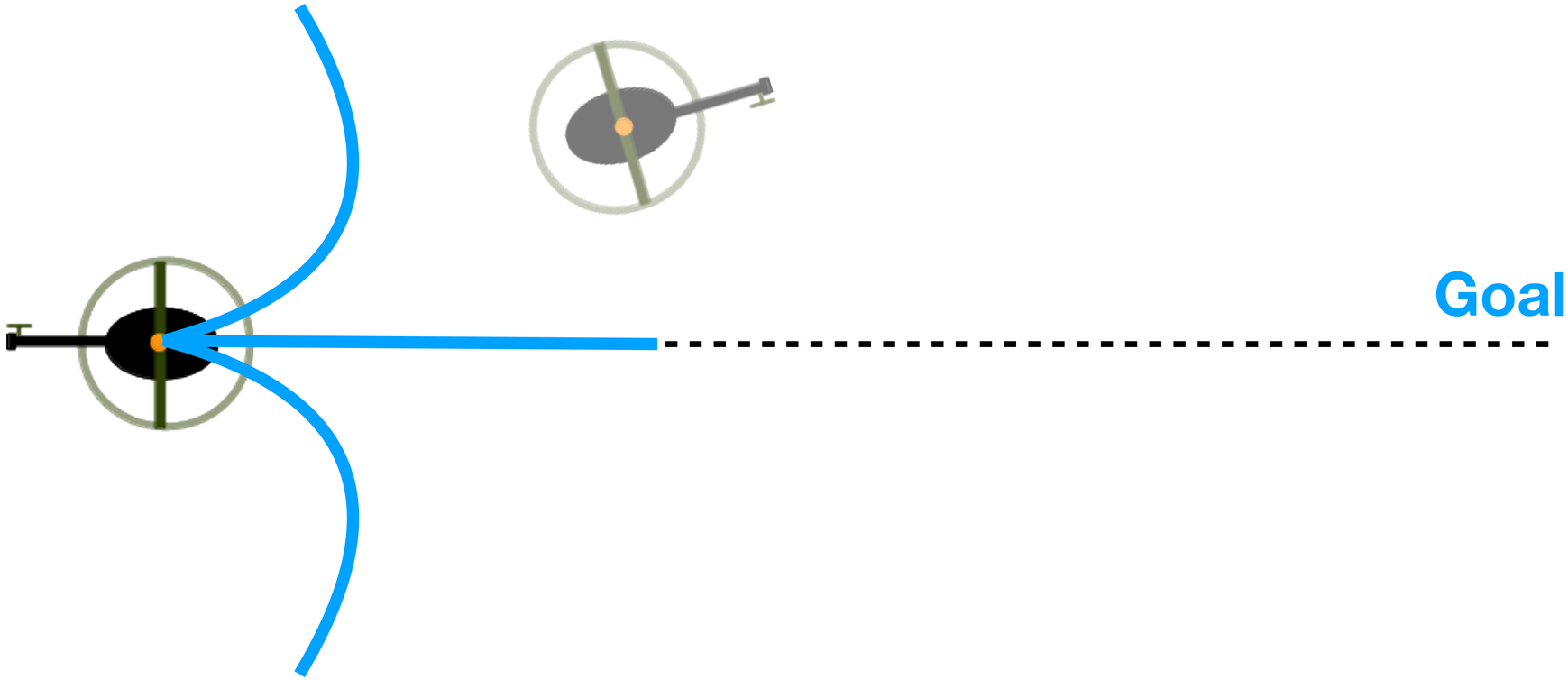
Make it adaptive! Balance safety and performance.



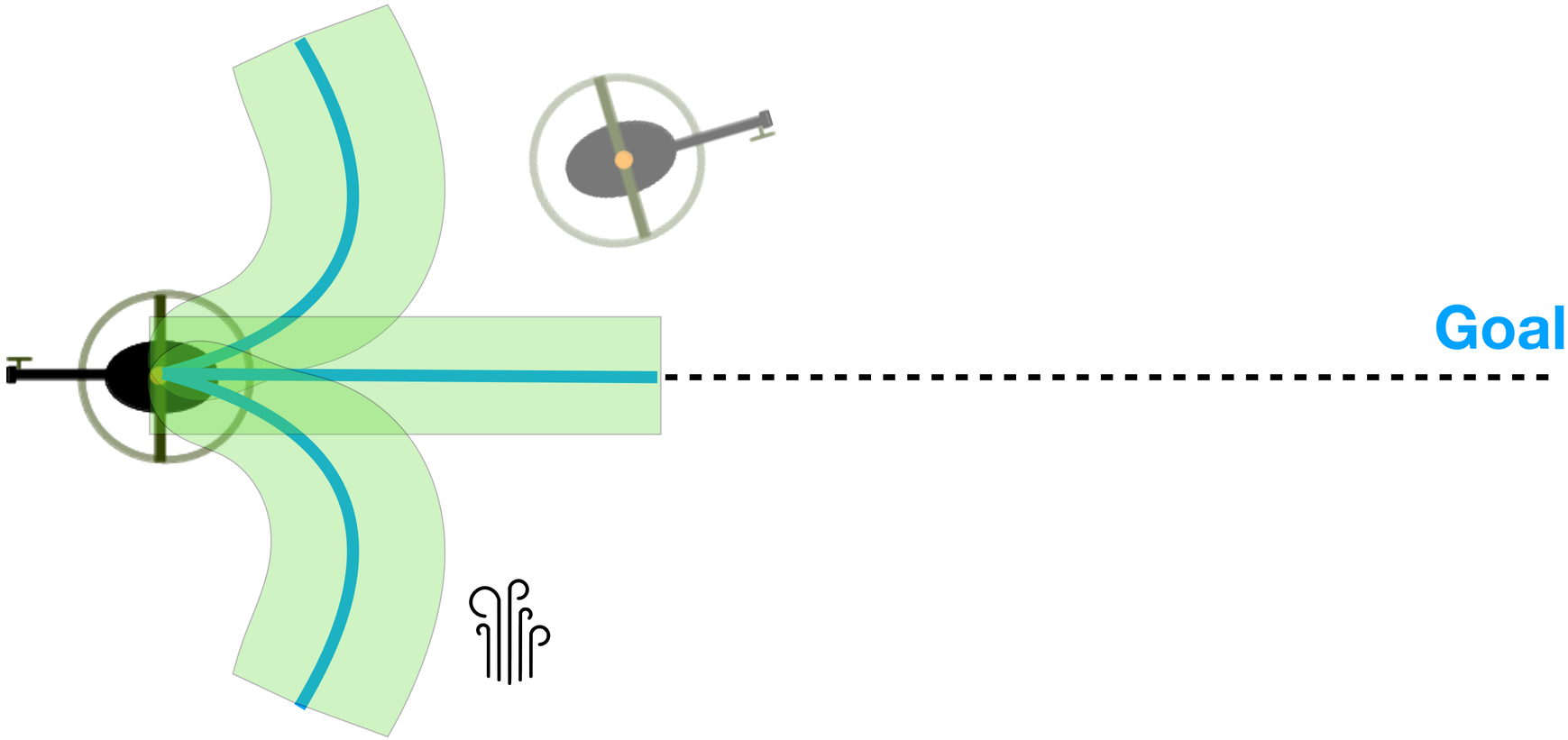
Precompute safety margins for multiple disturbances. Adapt on-the-fly.

[Ho et. al. 2020, in submission]

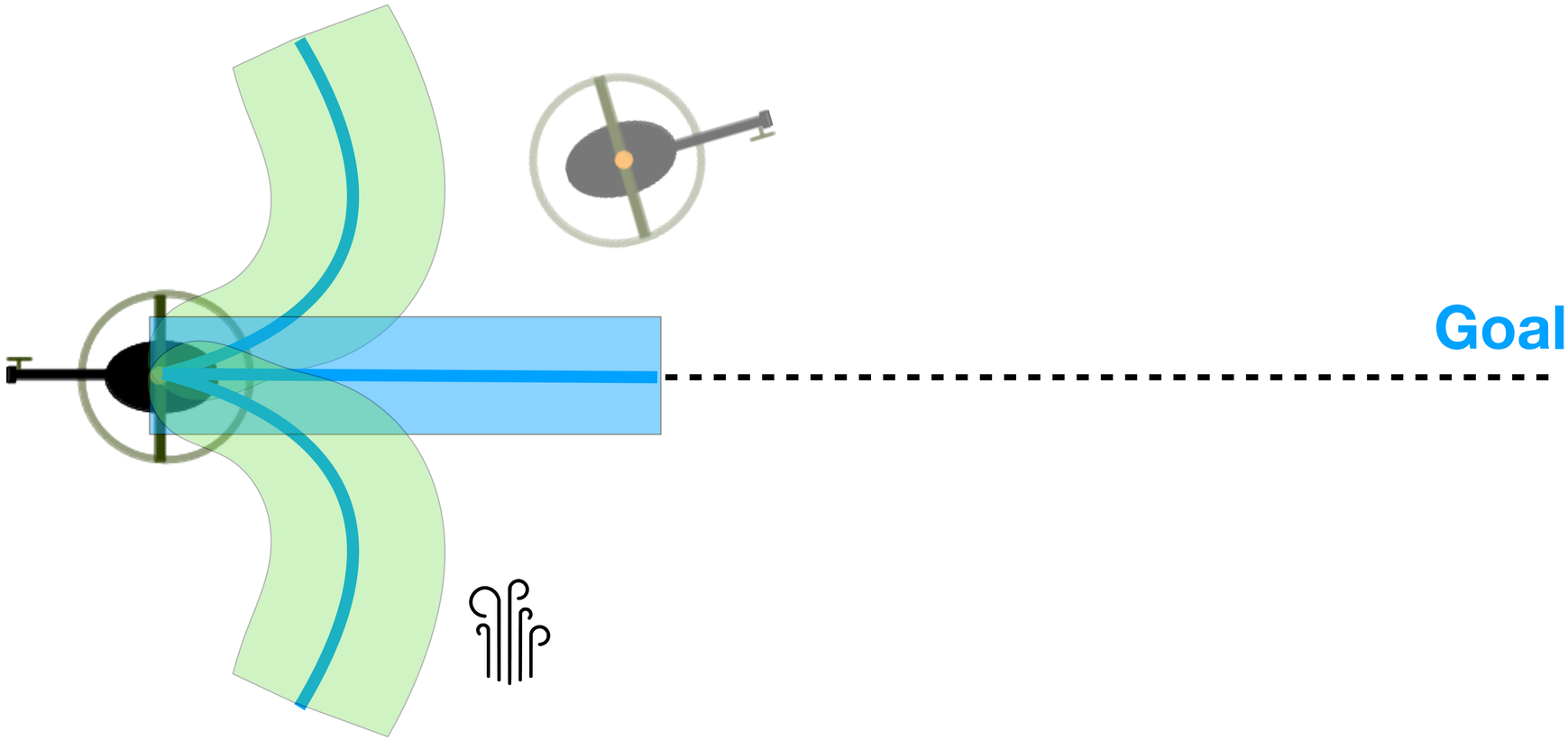
Example (Adaptive Tube)



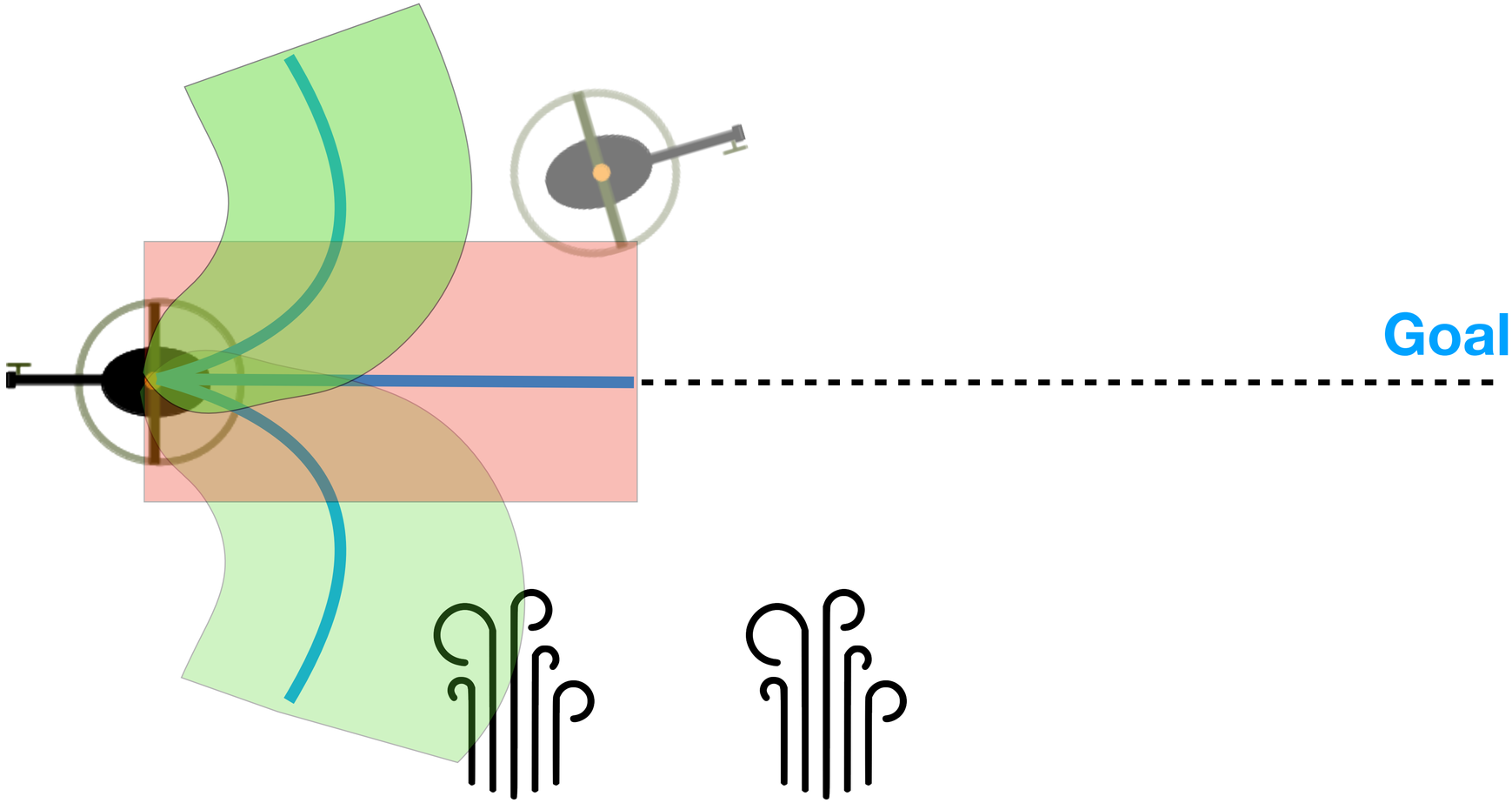
Example (Adaptive Tube) - Low Wind



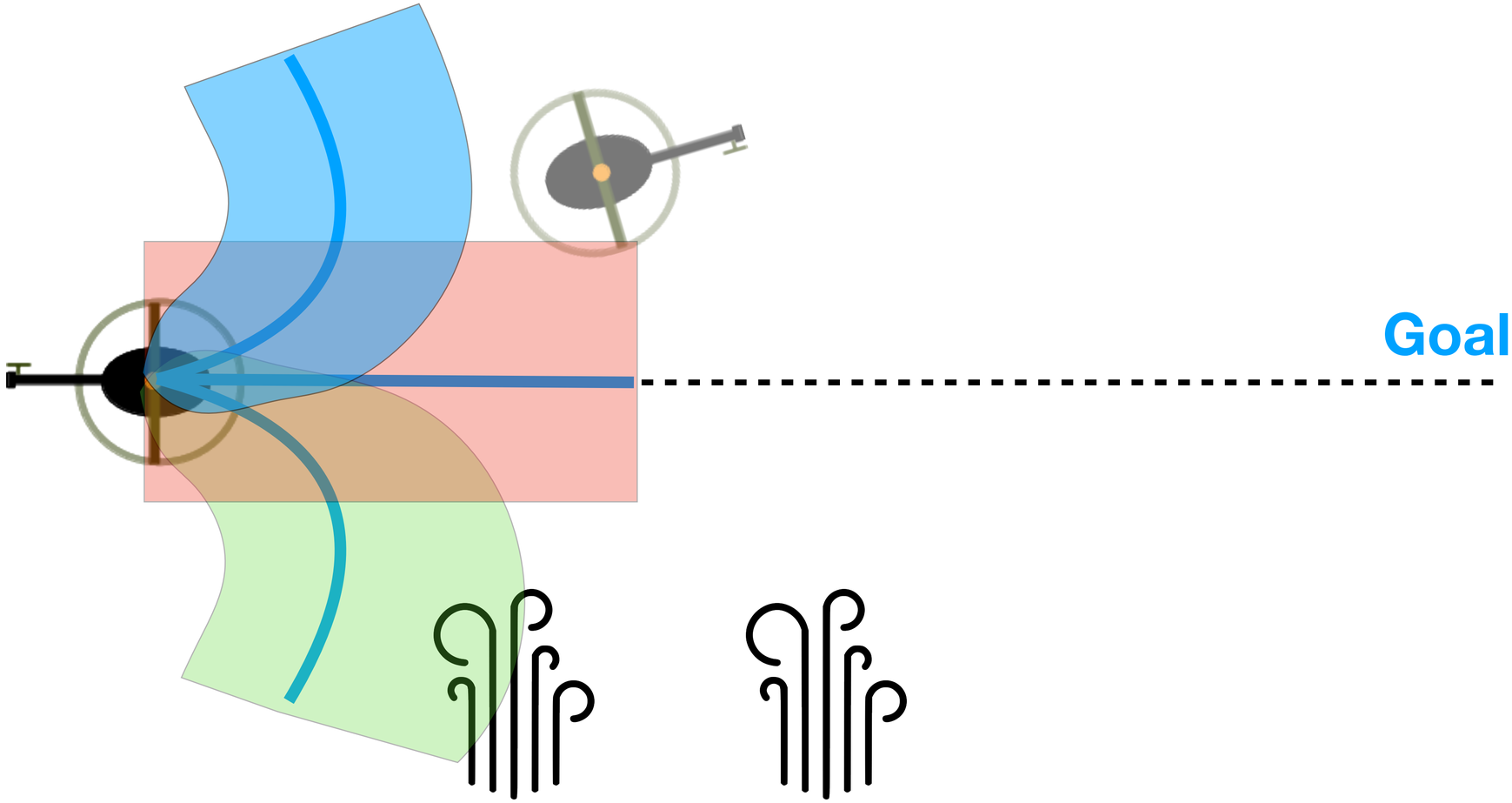
Example (Adaptive Tube) - Low Wind



Example (Adaptive Tube) - High Wind

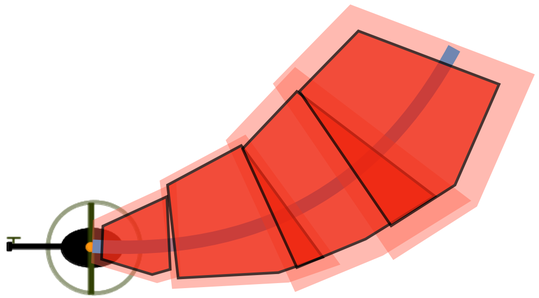


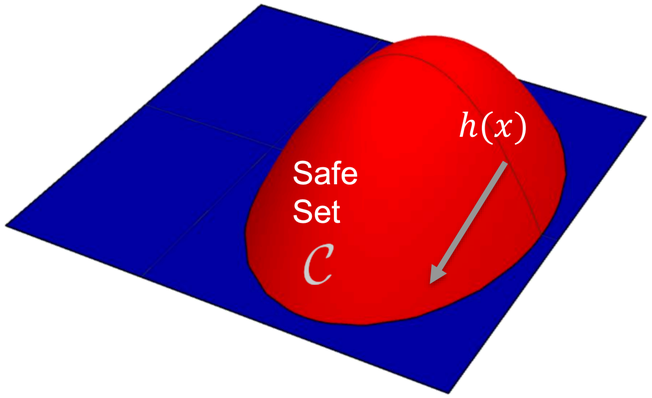
Example (Adaptive Tube) - High Wind



Summary

- Why ensuring safety is important? Towards ubiquitous autonomy
- Key challenges: Safety vs. Performance, Complexity, Tail-end Cases
- Key formal verification tools and where they may fail





Open Questions

- How to know how well we have covered cases for complex systems?
- How to be safe even when our assumptions are violated?
- How to best design simulator to properly stress test and how statistical proofs apply in real life?

Sterling Anderson @ Aurora:
"We're going to drive about half as many miles this year," Sterling said at the event, adding that the company is focused on testing the vehicles in simulated environments.
[Aurora20]

ANSYS & BMW:
"ANSYS and BMW Group Partner to Jointly Create the Industry's First Simulation Tool Chain for Autonomous Driving"
[Ansys19]

Dmitri Dolgov @ Waymo:
"At Waymo, we've driven more than 10 million miles in the real world, and over 10 billion miles in simulation,"
[Waymo19]

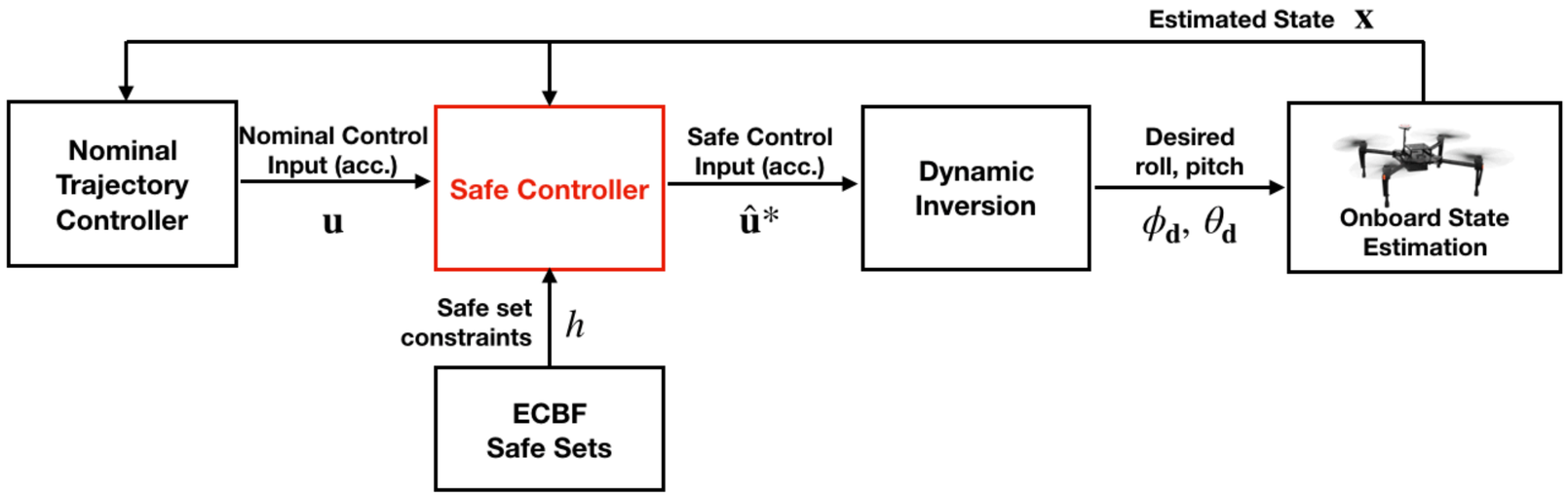
Trend - Simulated Statistical Proofs

From Pavone's AA274 Slides.

Outline

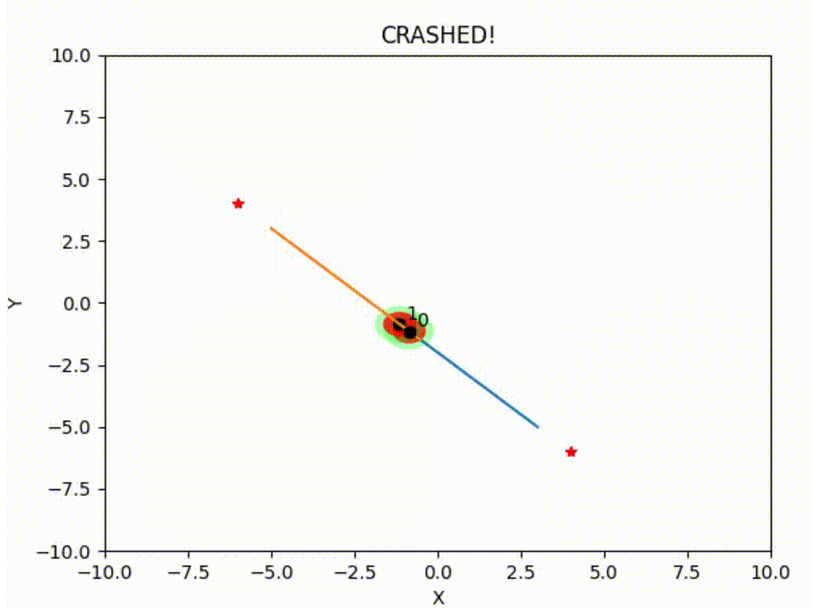
- Introduction
- Challenges
- Formal Tools
  - Math Formulation
  - Reachability Analysis
  - Control Barrier Function
- Formal Guarantees in the Real World
- Interactive Session - Control Barrier Functions

Interactive Session - Control Barrier Functions



Exercise 1 (30 min)

Run `exercises.py` you should see:



Fill out `compute_safe_control()` function from `ecbf_control.py`

to implement the optimization problem:

$$u^* = \arg\min ||u - u_{des}||^2$$

$$s.t \quad A_{ij}u \leq b_{ij} \quad \forall i \neq j$$

You can use `compute_A` and `compute_b` to derive the constraints

Exercise 1-Quadratic Programming using `cvxopt`

`qp` solver from `cvxopt` solves the following optimization problem:

$$\min \quad (1/2)x^T Px + q^T x$$

$$s.t. \quad Gx \leq h$$

$$u^* = \arg\min ||u - u_{des}||^2$$

$$s.t \quad A_{ij}u \leq b_{ij} \quad \forall i \neq j$$

Usage:

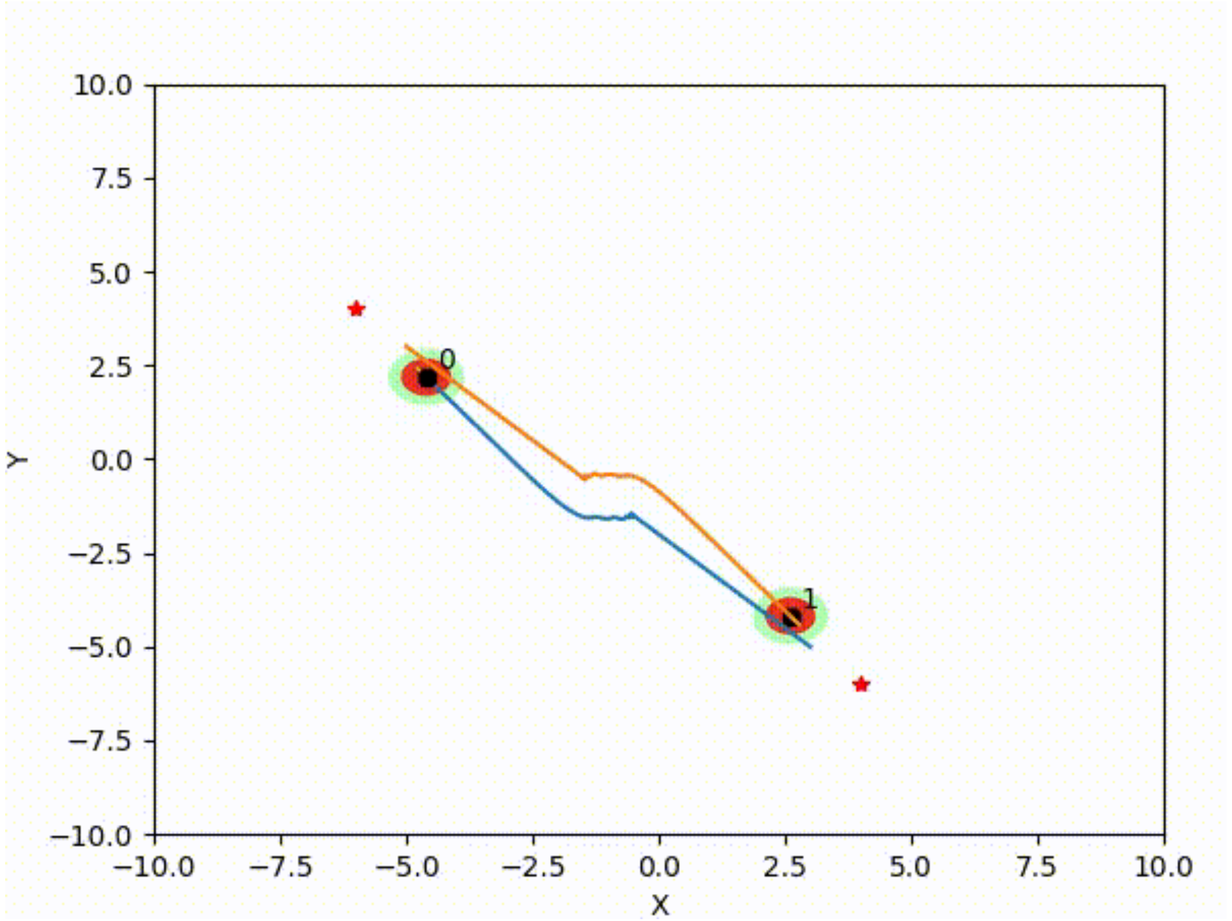
```
Sol = solve_qp(P, q, G, h)
```

Where:

```
Sol['x']: argmin of the optimization problem
P, q, G, h are all numpy arrays
```

Exercise 1

Once you’re done, you should get the following result:

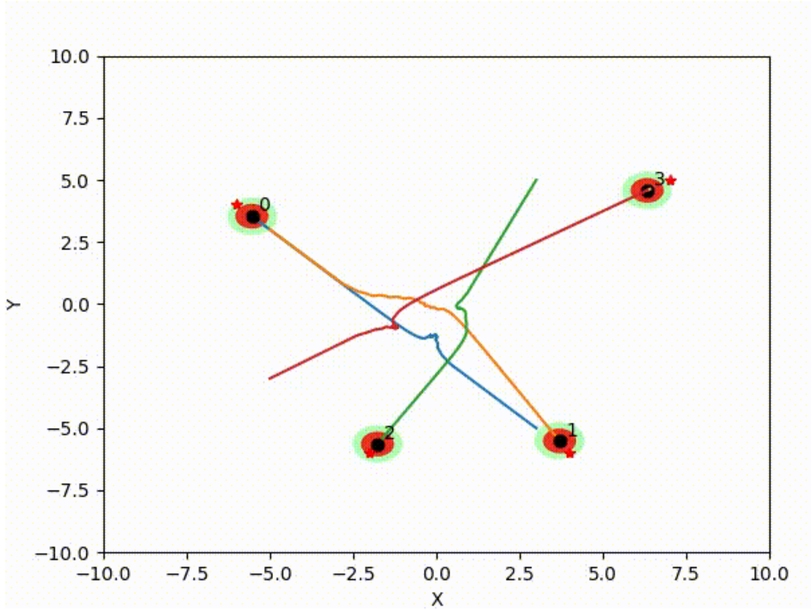


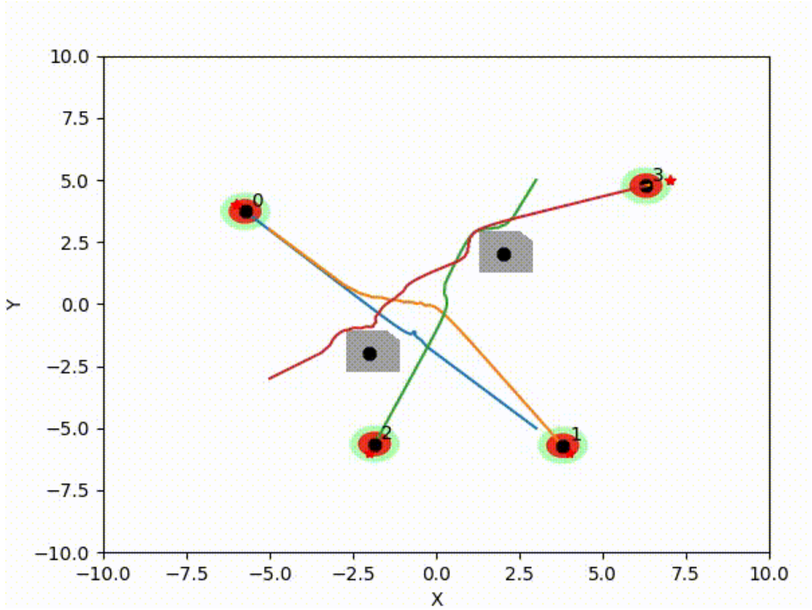
Exercise 2 (15 min)

A. In `exercises.py` add 2 more robots
   - Initial positions: R2=[3,5] , R3=[-5, -3] and goals: G2=[-3, -7] , G3=[7, 5]
   - Remember to append new robots in `Robots`

B. Uncomment obstacle lines and see the results

After you're done you should see results similar to these:



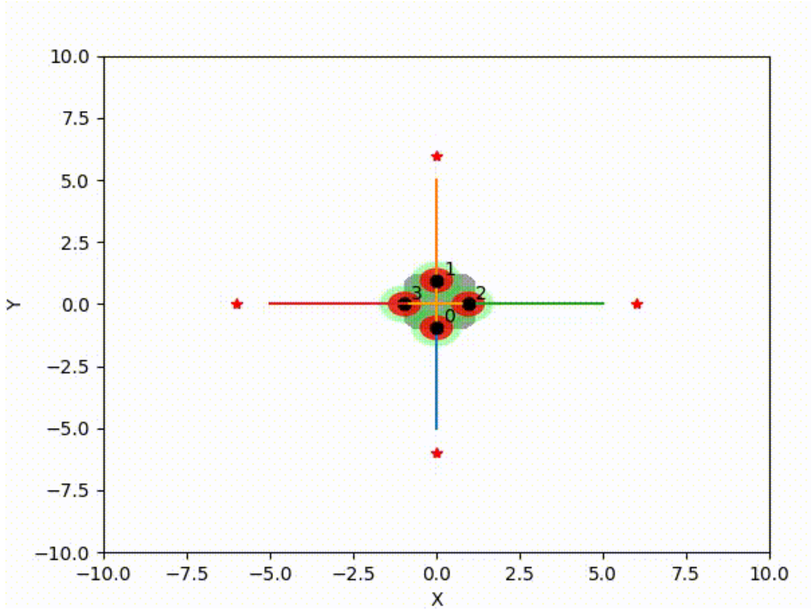


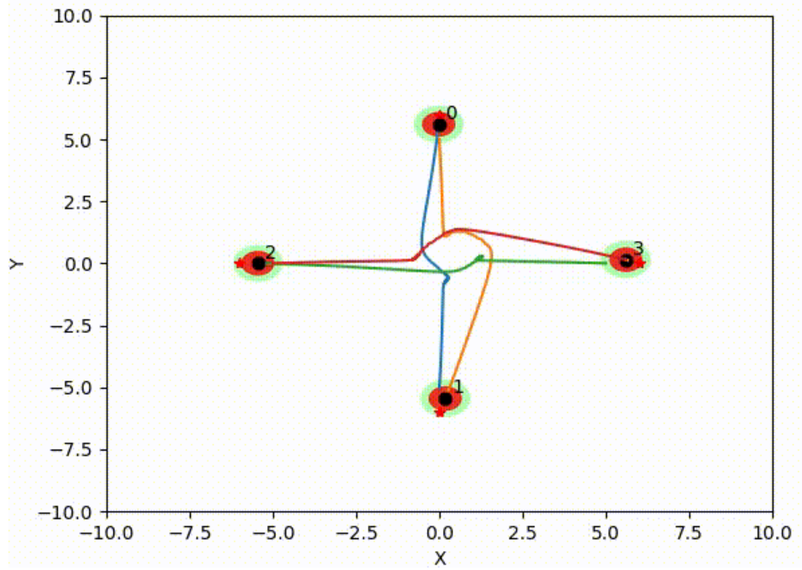
Exercise 3 (20 min)

A. In `exercises.py` change the initial positions and goals of all 4 robots to create a **symmetrical** situation to observe deadlock (No obstacles)

B. Try adding some small (10%) random value (using `np.random.random()`) to u^* (in `compute_safe_control()`) to make it non-symmetrical and resolve the deadlock

After you're done you should see results similar to these:





Backup


BACKUPS,
BACKUPS EVERYWHERE!
makeameme.org


When someone says “guaranteed safe”
and I suddenly pay attention

Backup

FOOLING THE AI

Deep neural networks (DNNs) are brilliant at image recognition — but they can be easily hacked.

These stickers made an artificial-intelligence system read this stop sign as 'speed limit 45'.