Learning in-the-loop correct-by-construction control

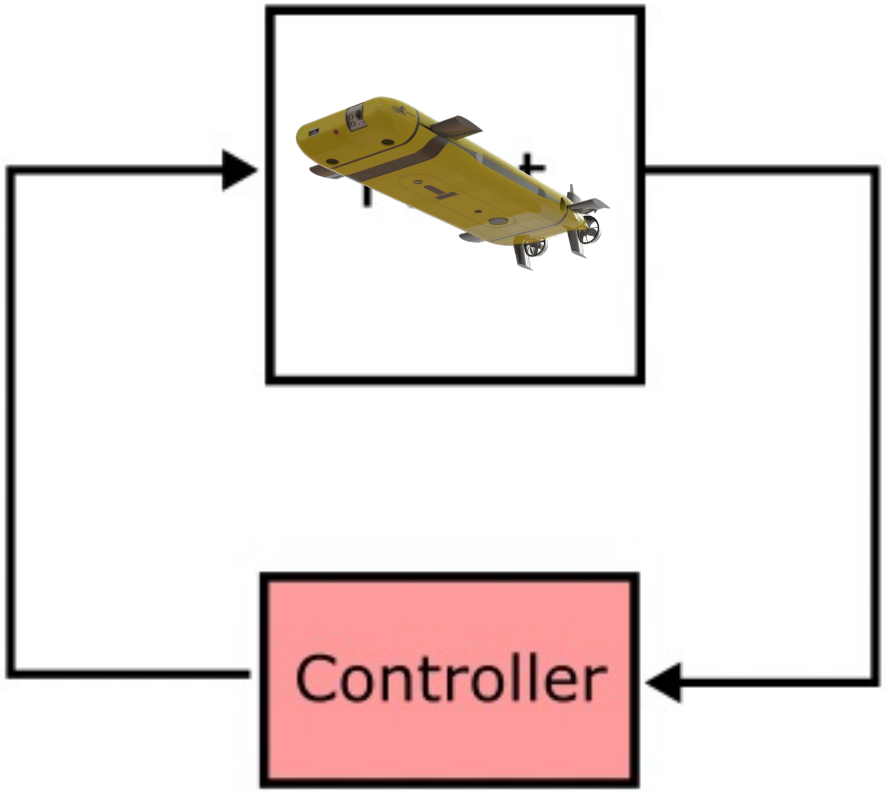
Necmiye Ozay
Electrical Engineering and Computer Science & Robotics
University of Michigan, Ann Arbor

July 9, 2023
IFAC World Congress Preconference Workshop
Data-Driven Verification and Control of Cyber-Physical Systems

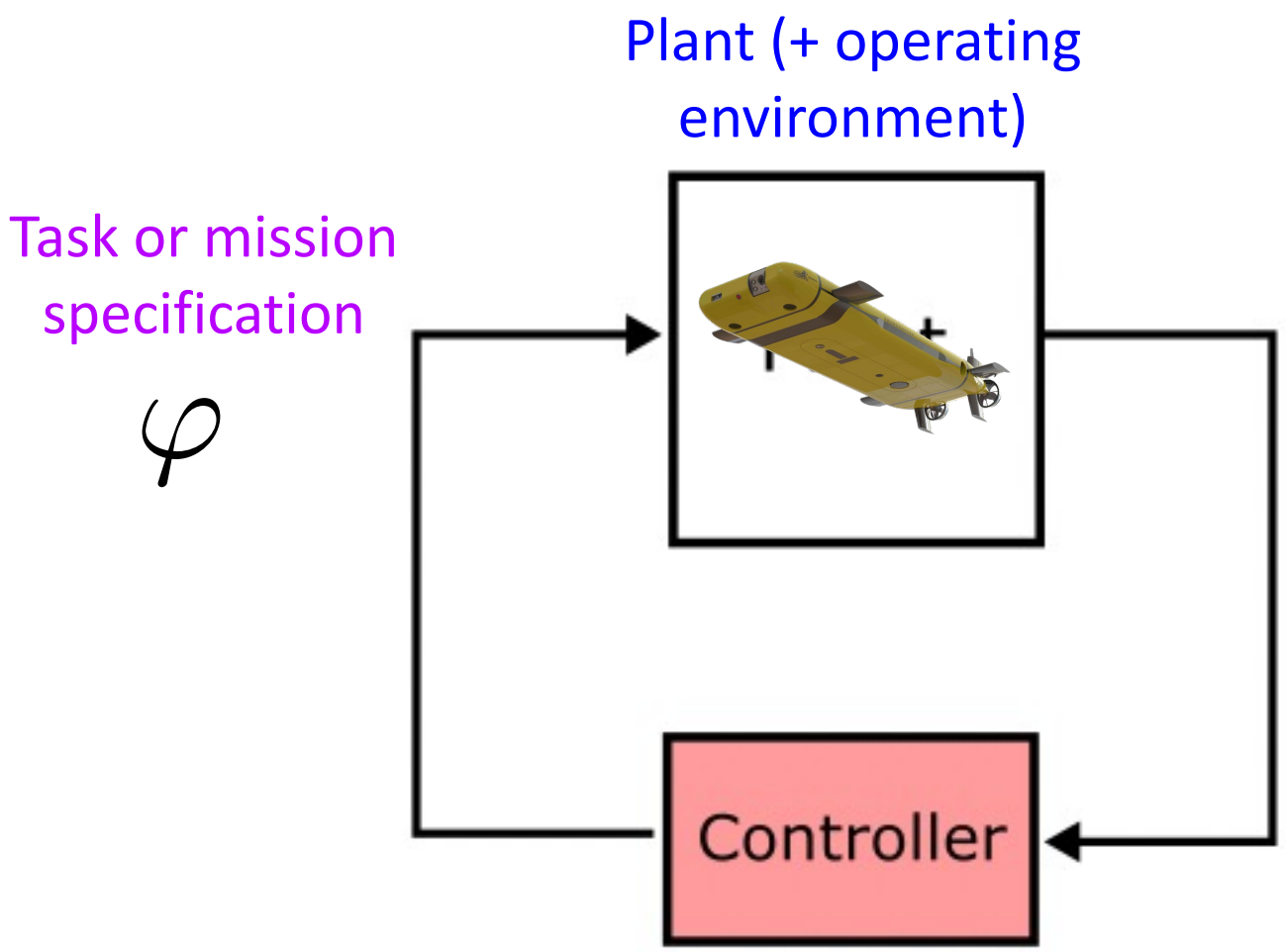
Plant (+ operating environment)

Task or mission specification

φ



Plant model + Task spec → Controller (decision maker)



Learning in-the-loop control:

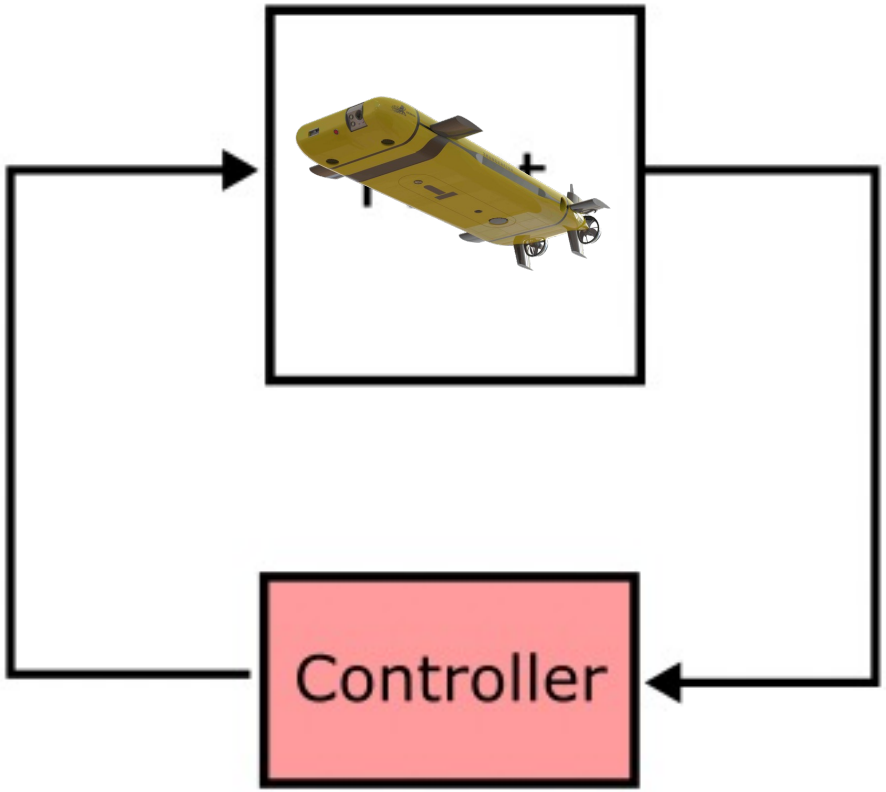
- **Plant** can be unknown and need to be learned (ACC'19, TAC'22, LCSS'23)
- **Task** can be unknown and need to be learned (RA-L'20, RSS'20, AURO'21)
- **Controller** can be hard to design or synthesize and can be learned instead (RA-L'21, CDC'21, EMSOFT'21, LCSS'23)
- **Perception module** typically does not have an analytical solution and is learned (EMSOFT'21, WAFR'22)

Plant model + Task spec → Controller (decision maker)

Plant (+ operating environment)

Task or mission specification

φ



Learning in-the-loop control:

- **Plant** can be unknown and need to be learned (ACC'19, TAC'22, **LCSS'23**)
- **Task** can be unknown and need to be learned (RA-L'20, RSS'20, AURO'21)
- **Controller** can be hard to design or synthesize and can be learned instead (RA-L'21, **CDC'21**, EMSOFT'21, **LCSS'23**)
- **Perception module** typically does not have an analytical solution and is learned (EMSOFT'21, **WAFR'22**)

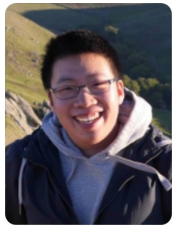
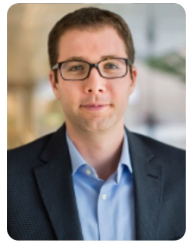
Plant model + Task spec → Controller (decision maker)

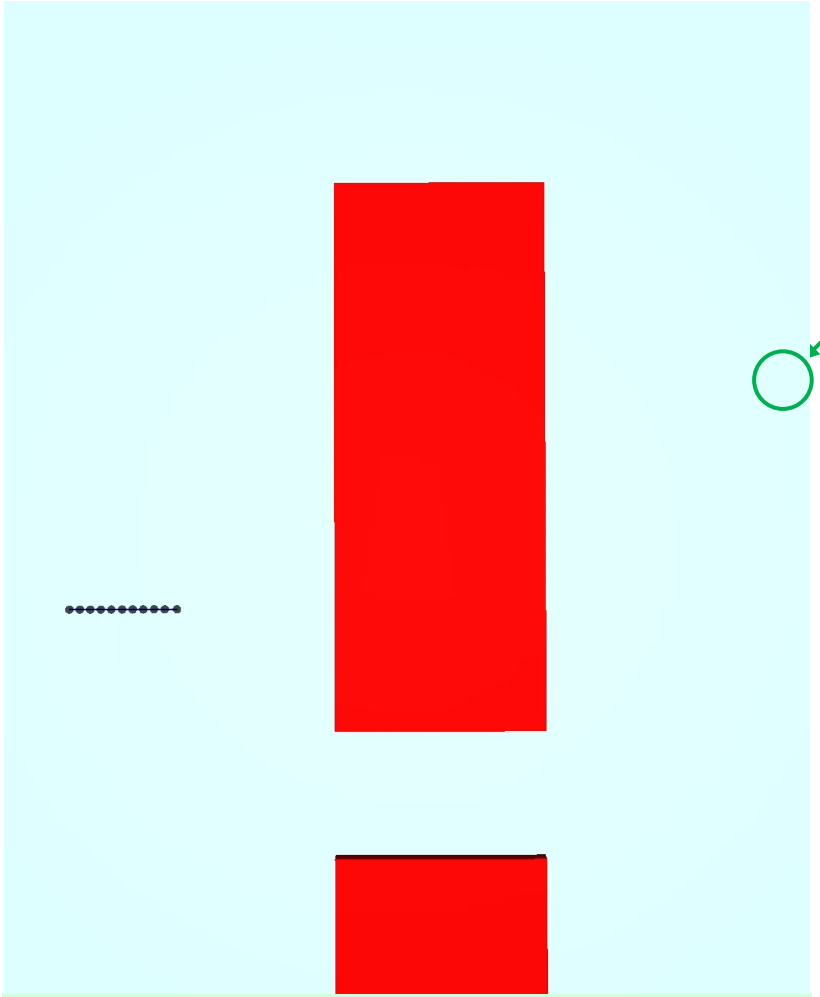
GOAL: **Ensure** system-level safety and performance when some of these pieces are learned!

Learning the model and a controller

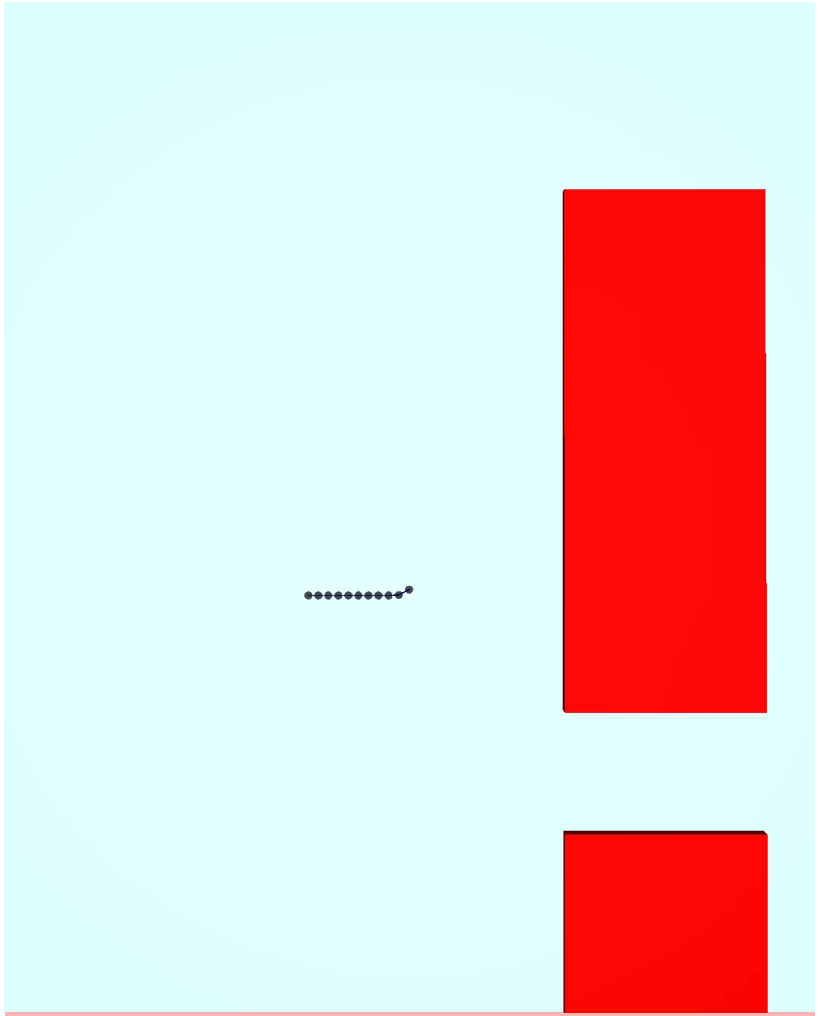
References:

- **Model Error Propagation via Learned Contraction Metrics for Safe Feedback Motion Planning of Unknown Systems (CDC, 2021)**
- Planning with Learned Dynamics: Probabilistic Guarantees on Safety and Reachability via Lipschitz Constants (RA-L, 2021)

Joint work with Glen Chou  and Dmitry Berenson 



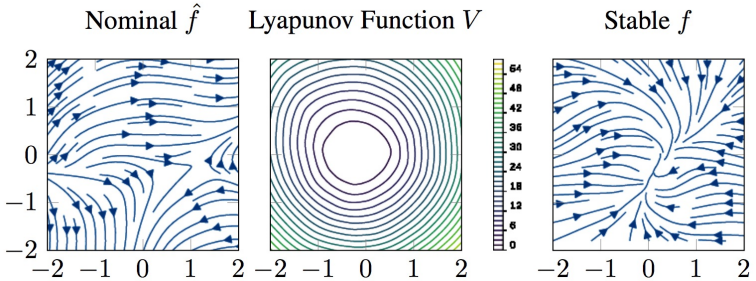
Staying where the *model is valid* and the *controller is stabilizing*



Leaving the domain of model validity and controller stability

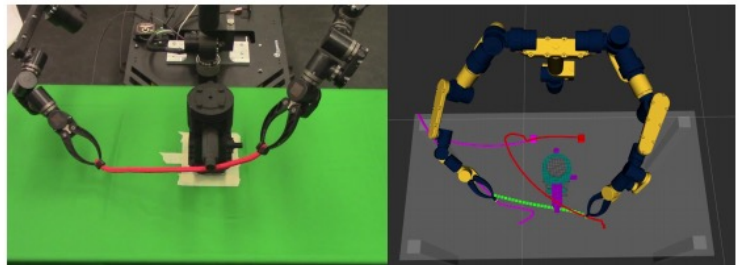
Goal: *long-horizon motion planning* with a *learned model* of *underactuated dynamics*, with guarantees that the system can *safely track the plan* with a *learned feedback controller*?

Related work



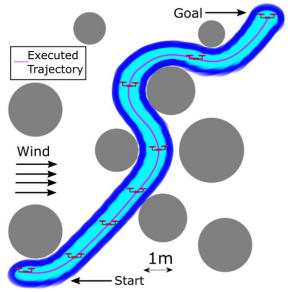
Learning stability certificates from data [Manek and Kolter NeurIPS'19, Boffi et al. CoRL'20, Richards et al. CoRL'18]

- Limited utility for *control* / point-to-point *motion planning*



Long-horizon planning with learned dynamics [Ichter and Pavone RA-L'19, McConachie et al. RA-L'20]

- *No guarantees* on safety / feasibility



Robust feedback motion planning [Singh et al. ICRA'17, Mitchell et al. TAC'05, Majumdar and Tedrake IJRR'17]

- Requires *a-priori known*, tight disturbance bounds

This work:

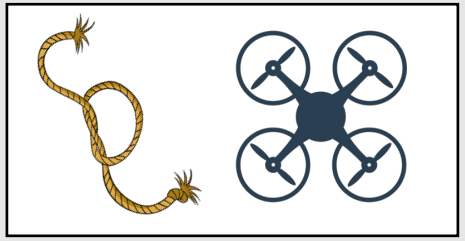
- Model error empirically estimated from data
- Safety and reachability guarantees *in execution* when tracking the computed plans

Problem statement:

Given data $\{(x_i, u_i, h(x_i, u_i))\}_{i=1}^{N}$ from an unknown, deterministic, underactuated system $\dot{x} = h(x, u)$, plan with a learned dynamics model $\dot{x} = g(x, u)$, and ensure safety in execution.

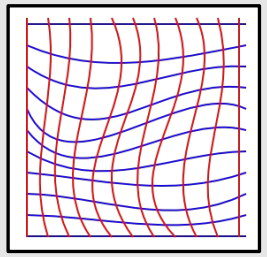
Method:

Learning:

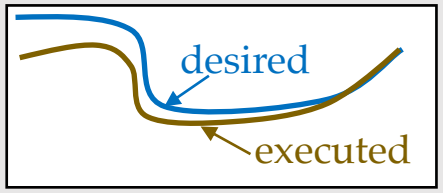


$\dot{x} = f(x) + B(x)u$
(dynamics model)

$M(x)$
(contraction metric)

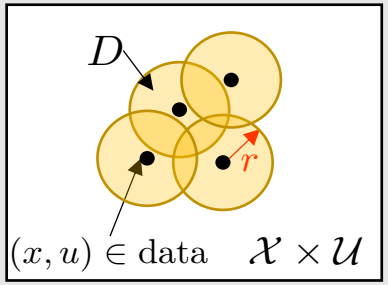


$u(x, x^*, u^*)$
(stabilizing controller)

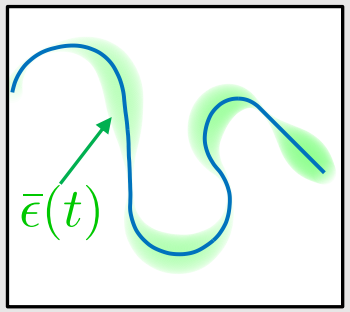


Analysis:

$e(x, u) \leq \bar{e}(x, u),$
$\forall (x, u) \in D$
Bound *model error* inside a trusted domain D

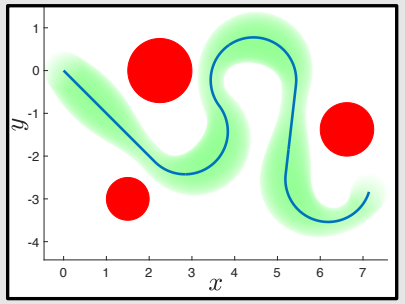


$\left\| \begin{bmatrix} x(t) - x^*(t) \\ u(t) - u^*(t) \end{bmatrix} \right\| \leq \bar{\epsilon}(t),$
$\forall t \in [0, T]$
Bound *trajectory tracking error* when subjected to model error



Planning:

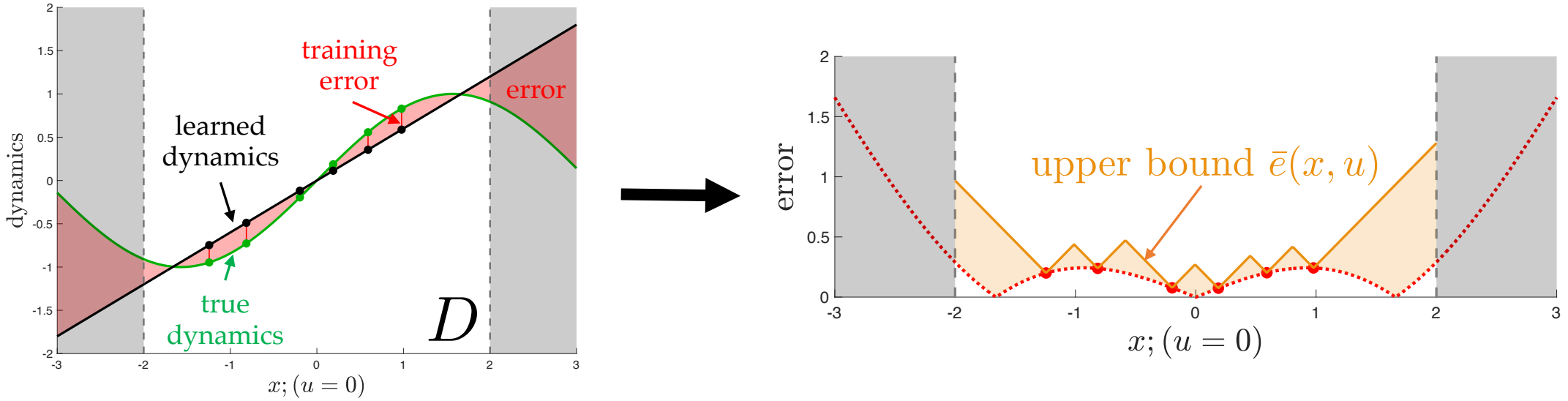
Find plans that remain collision-free under the tracking error bound



Bounding *model error* for a learned *dynamics model*

Problem: Find an upper bound on the model error $\|h(x,u) - g(x,u)\| \leq \bar{e}(x,u)$ inside a domain D around the training dataset $\{(x_i, u_i, h(x_i,u_i))\}_{i=1}^{N}$. Also, determine the domain D.

Key insight: Use the *Lipschitz constant* of the *error between the true and learned dynamics*.



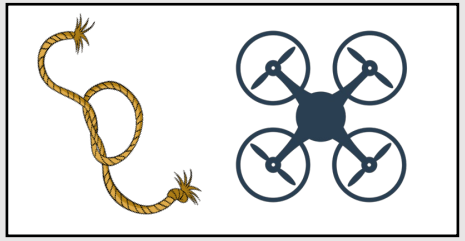
- Implementation details:
  - An "upper bound" on the *Lipschitz constant* *valid in D* is estimated from data (using Extreme value theory): for the rest of the talk, we will assume the bound is valid
  - We *don't fix the domain D a priori*: size is calibrated using dynamics error on a *validation dataset*

Problem statement:

Given data $\{(x_i, u_i, h(x_i, u_i))\}_{i=1}^{N}$ from an unknown, deterministic, underactuated system $\dot{x} = h(x, u)$, plan with a learned dynamics model $\dot{x} = g(x, u)$, and ensure safety in execution.

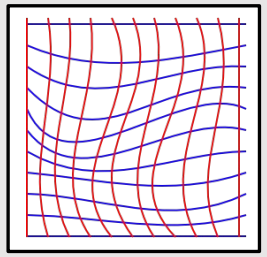
Method:

Learning:

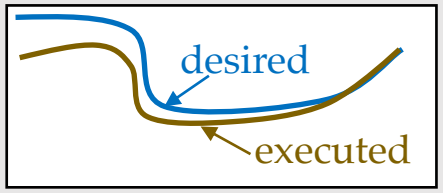


$\dot{x} = f(x) + B(x)u$
(dynamics model)

$M(x)$
(contraction metric)

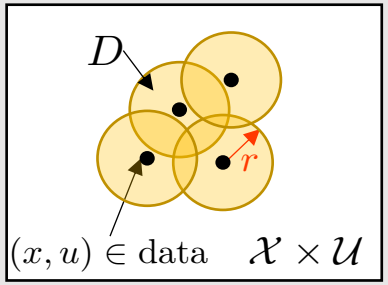


$u(x, x^*, u^*)$
(stabilizing controller)

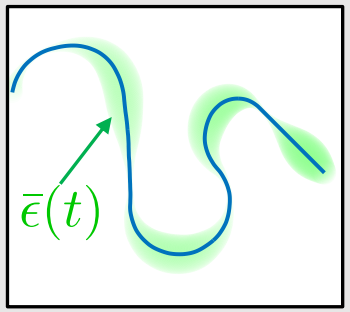


Analysis:

$e(x, u) \leq \bar{e}(x, u),$
$\forall (x, u) \in D$
Bound *model error* inside a trusted domain D

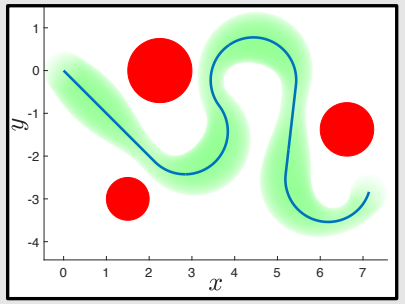


$\left\| \begin{bmatrix} x(t) - x^*(t) \\ u(t) - u^*(t) \end{bmatrix} \right\| \leq \bar{\epsilon}(t),$
$\forall t \in [0, T]$
Bound *trajectory tracking error* when subjected to model error



Planning:

Find plans that remain collision-free under the tracking error bound

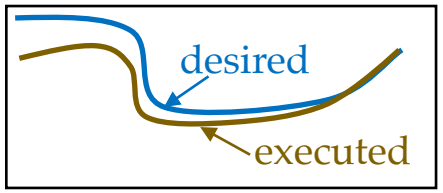


Bounding *tracking error* for a learned *stabilizing controller*

Problem: Learn a *controller* which can stabilize around plans, and bound the *tracking error* in execution caused by model error.

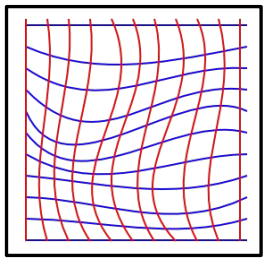
Key insight: Use *contraction theory* to learn a *stable controller* and bound its *tracking error*.

Method:



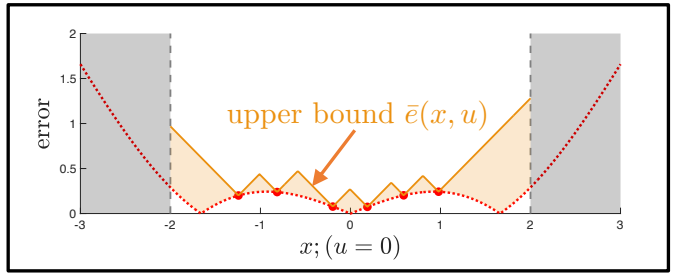
$u(x, x^*, u^*)$

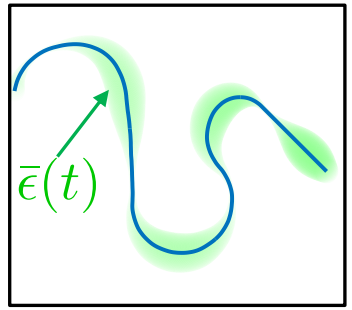
current state desired x desired u



$M(x)$

Learn a *feedback controller* and a *control contraction metric (CCM)* (which certifies closed-loop stability)





Bound *tracking error* due to model error in the *trusted domain*

CCM $M(x)$ defines a *state-dependent distance metric*

Intuition: the closed-loop system under the learned controller is contracting towards the nominal plan, as measured by *M*

We derive a novel bound on how model error affects the *the worst case tracking error* as a function of time:

$$\bar{\epsilon}(t) = \text{function}\big(M(x), \bar{e}(x, u), \bar{\epsilon}(\tau) \text{ for } \tau < t\big)$$

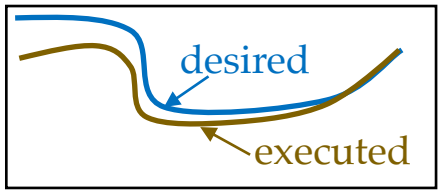
CCM — Model error bound — Tracking tube width at previous timesteps

Bounding *tracking error* for a learned *stabilizing controller*

Problem: Learn a *controller* which can stabilize around plans, and bound the *tracking error* in execution caused by model error.

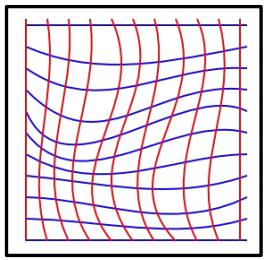
Key insight: Use *contraction theory* to learn a *stable controller* and bound its *tracking error*.

Method:



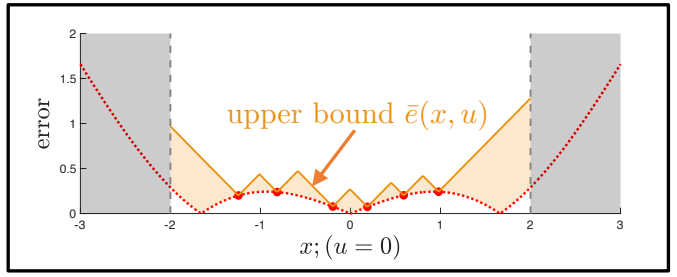
$u(x, x^*, u^*)$

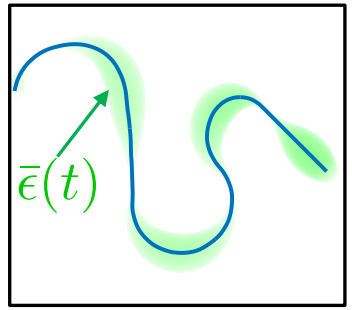
current state desired x desired u



$M(x)$

Learn a *feedback controller* and a *control contraction metric (CCM)* (which certifies closed-loop stability)





Bound *tracking error* due to model error in the *trusted domain*

CCM $M(x)$ defines a *state-dependent distance metric*

Intuition: the closed-loop system under the learned controller is contracting towards the nominal plan, as measured by M

We derive a novel bound on how model error affects the *the worst case tracking error* as a function of time:

$$\bar{\epsilon}(t) = \text{function}\big(M(x), \bar{e}(x,u), \bar{\epsilon}(\tau) \text{ for } \tau < t\big)$$

$$D^+\mathcal{E}(t) \le -2\left(\lambda - L_{h-g}\sqrt{\frac{\bar{\lambda}_D(M)}{\underline{\lambda}_D(M)}}\right)\mathcal{E}(t) + 2\sqrt{\mathcal{E}(t)\bar{\lambda}_D(M)}\left(L_{h-g}\left(\left\|\begin{bmatrix}x^*(t)\\u^*(t)\end{bmatrix} - \begin{bmatrix}x_{i^*(t)}\\u_{i^*(t)}\end{bmatrix}\right\| + \bar{u}_{\text{fb}}(t)\right) + e_{i^*(t)}\right)$$

Comparison lemma

$$\bar{\epsilon}(t) \le \sqrt{\int_{\tau=t_1}^{t_2} \mathcal{E}_{RHS}(\tau)d\tau / \underline{\lambda}_D(M)},$$

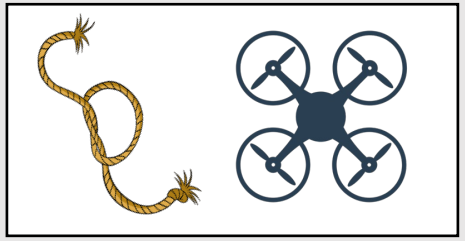
$$\mathcal{E}(t_1) = dist(x(t_1), x^*(t_1))$$

Problem statement:

Given data $\{(x_i, u_i, h(x_i, u_i))\}_{i=1}^{N}$ from an unknown, deterministic, underactuated system $\dot{x} = h(x, u)$, plan with a learned dynamics model $\dot{x} = g(x, u)$, and ensure safety in execution.

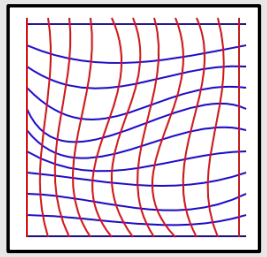
Method:

Learning:

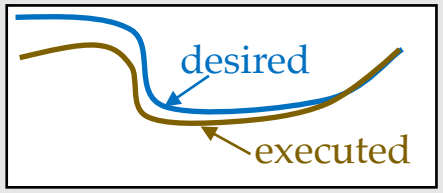


$\dot{x} = f(x) + B(x)u$
(dynamics model)

$M(x)$
(contraction metric)

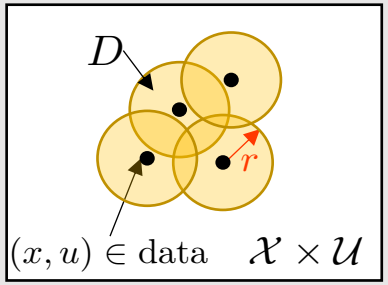


$u(x, x^*, u^*)$
(stabilizing controller)

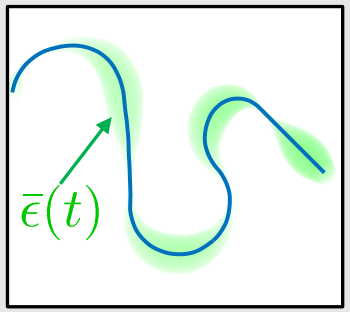


Analysis:

$e(x, u) \leq \bar{e}(x, u),$
$\forall (x, u) \in D$
Bound *model error* inside a trusted domain D

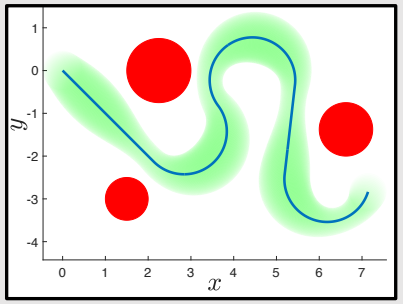


$\left\| \begin{bmatrix} x(t) - x^*(t) \\ u(t) - u^*(t) \end{bmatrix} \right\| \leq \bar{\epsilon}(t),$
$\forall t \in [0, T]$
Bound *trajectory tracking error* when subjected to model error



Planning:

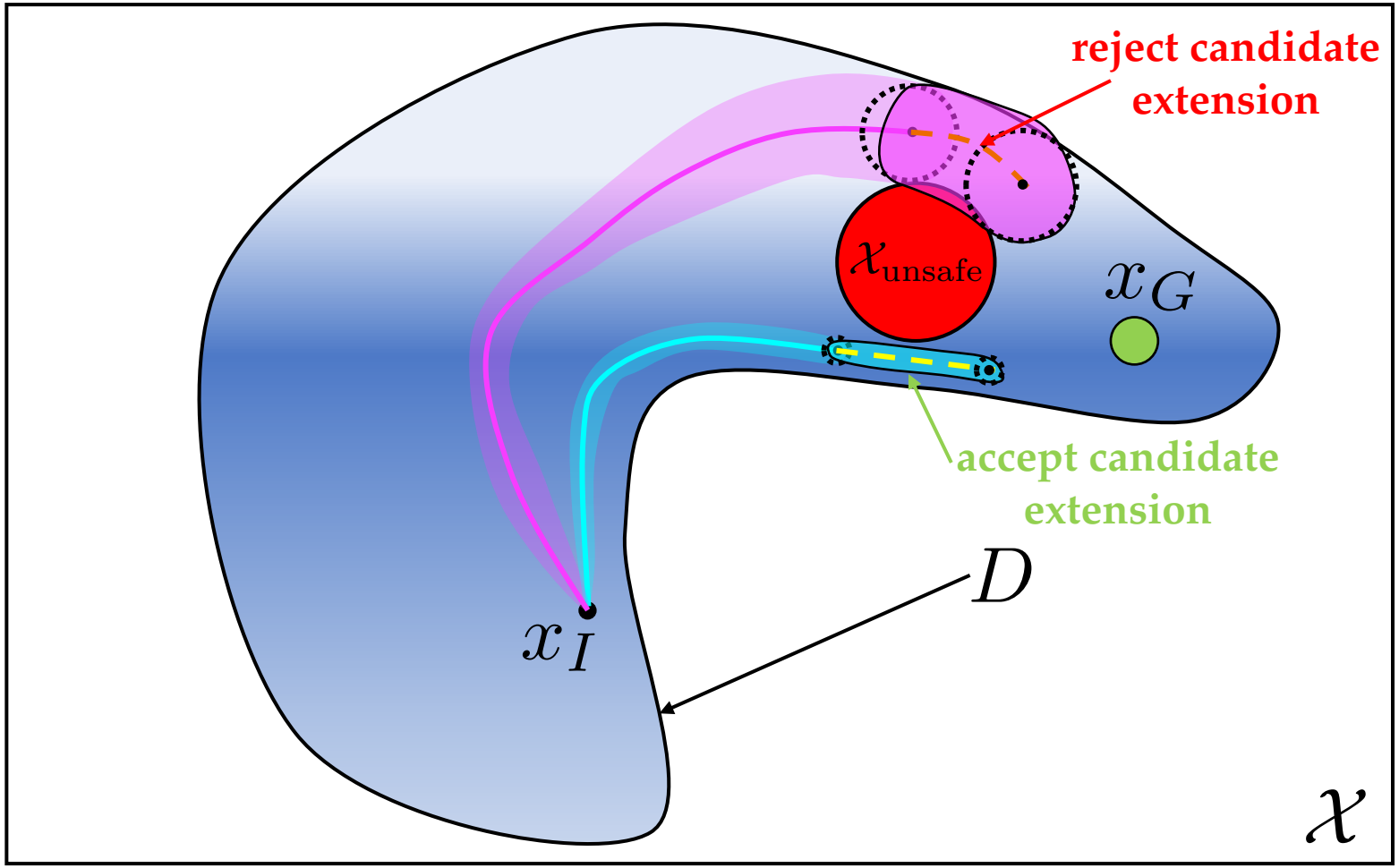
Find plans that remain collision-free under the tracking error bound



Compute safely-trackable plans using the *learned model*

Problem: Plan a trajectory *using the learned dynamics* that *remains safe in execution on the true system*.

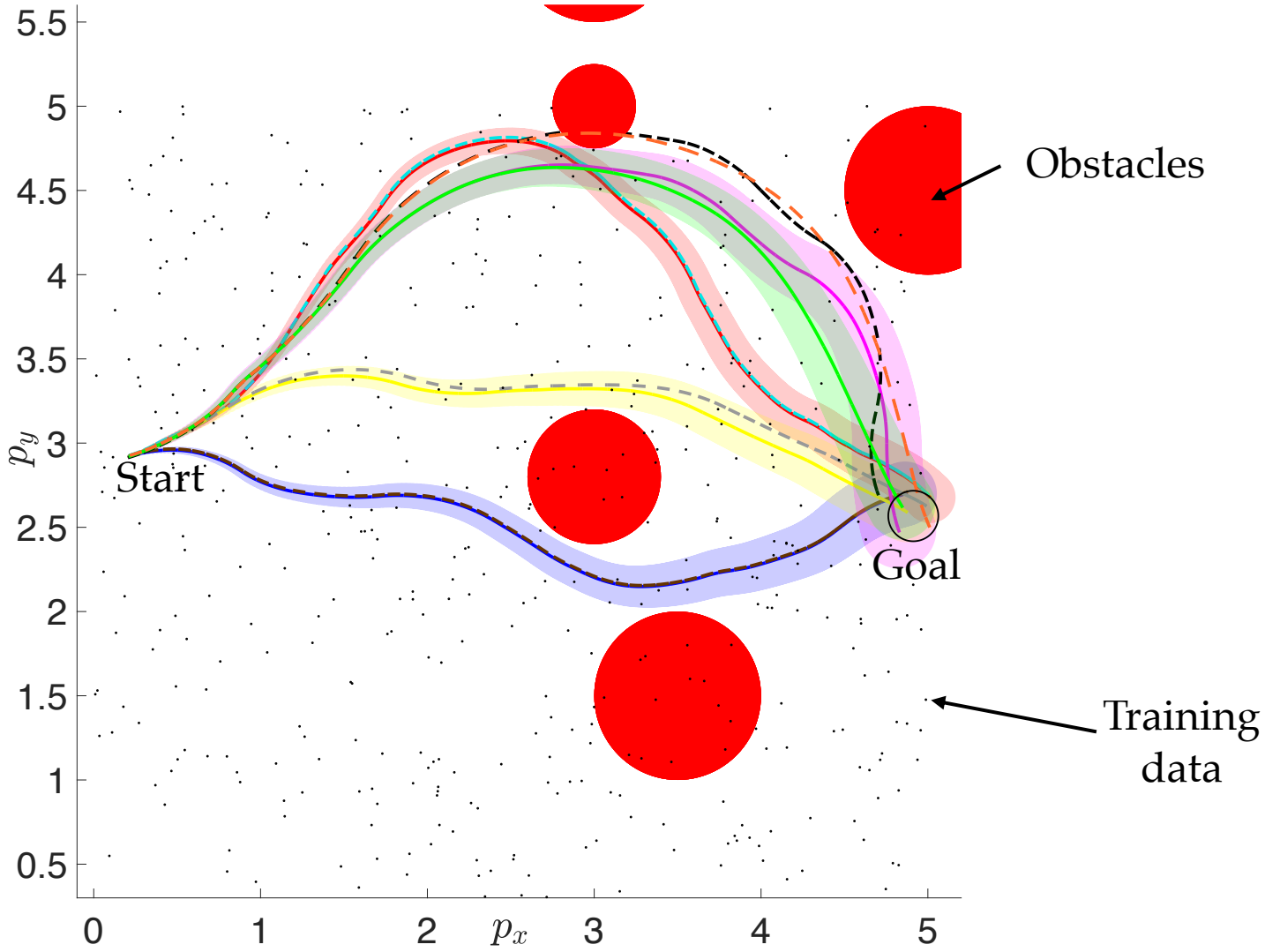
Key insight: *Incrementally propagate* tracking tube around candidate edges and *collision-check it*.



Results: 4D car environment

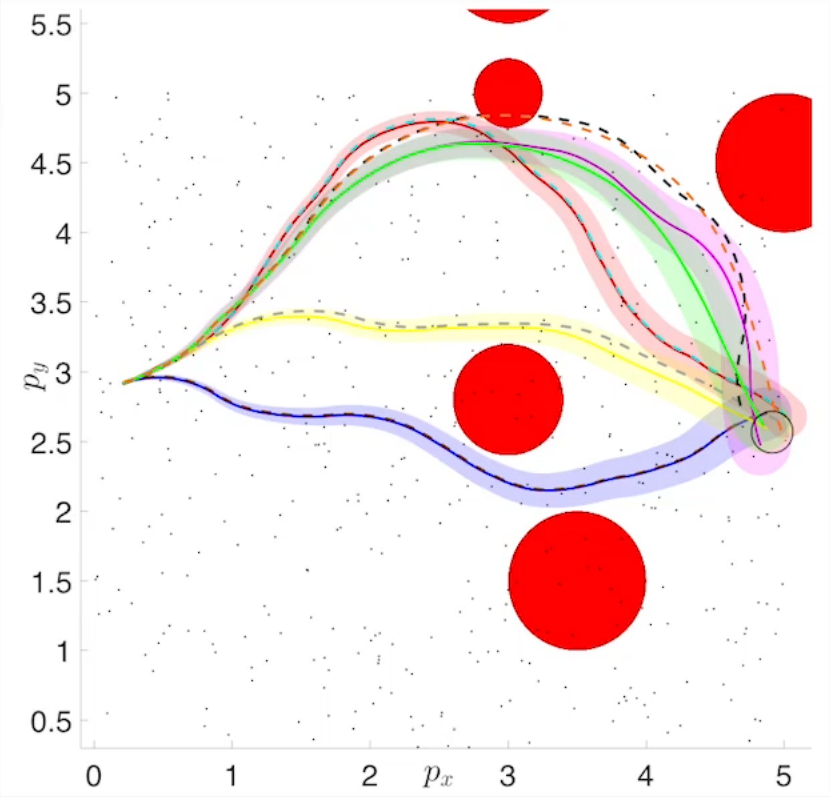
Legend:
Solid lines: planned trajectory
Dotted lines: executed trajectory
Transparent: trajectory tracking error tubes

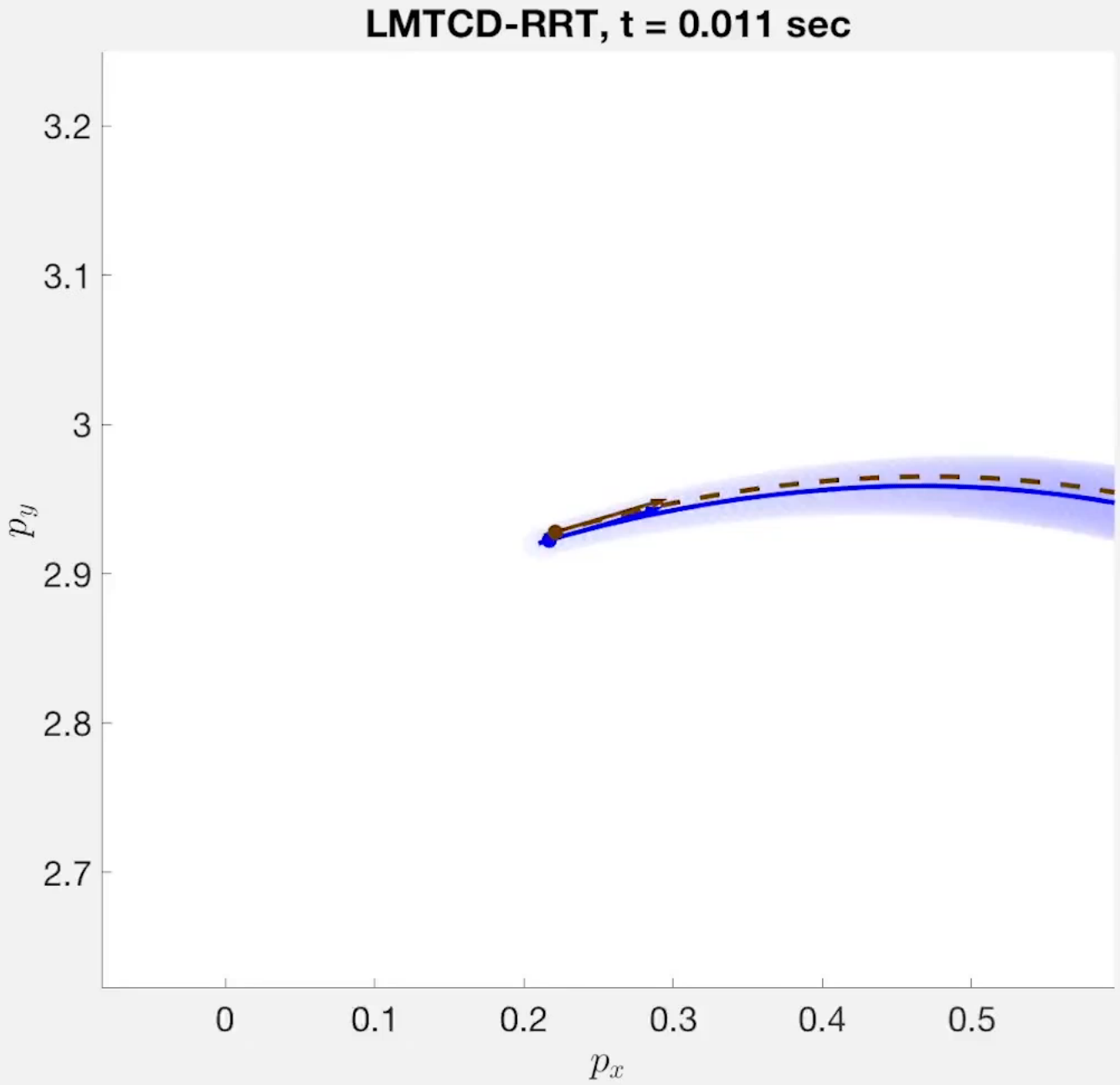
Baselines:
B1: model error bound = *average* training error
B2: model error bound = *maximum* training error
B3: *exit D*; model error bound = *maximum* training error
B4: *exit D*; our *Lipschitz-based* model error bound



Baselines:

B1: model error bound = *average* training error
B2: model error bound = *maximum* training error
B3: *exit D*; model error bound = *maximum* training error
B4: *exit D*; our *Lipschitz-based* model error bound

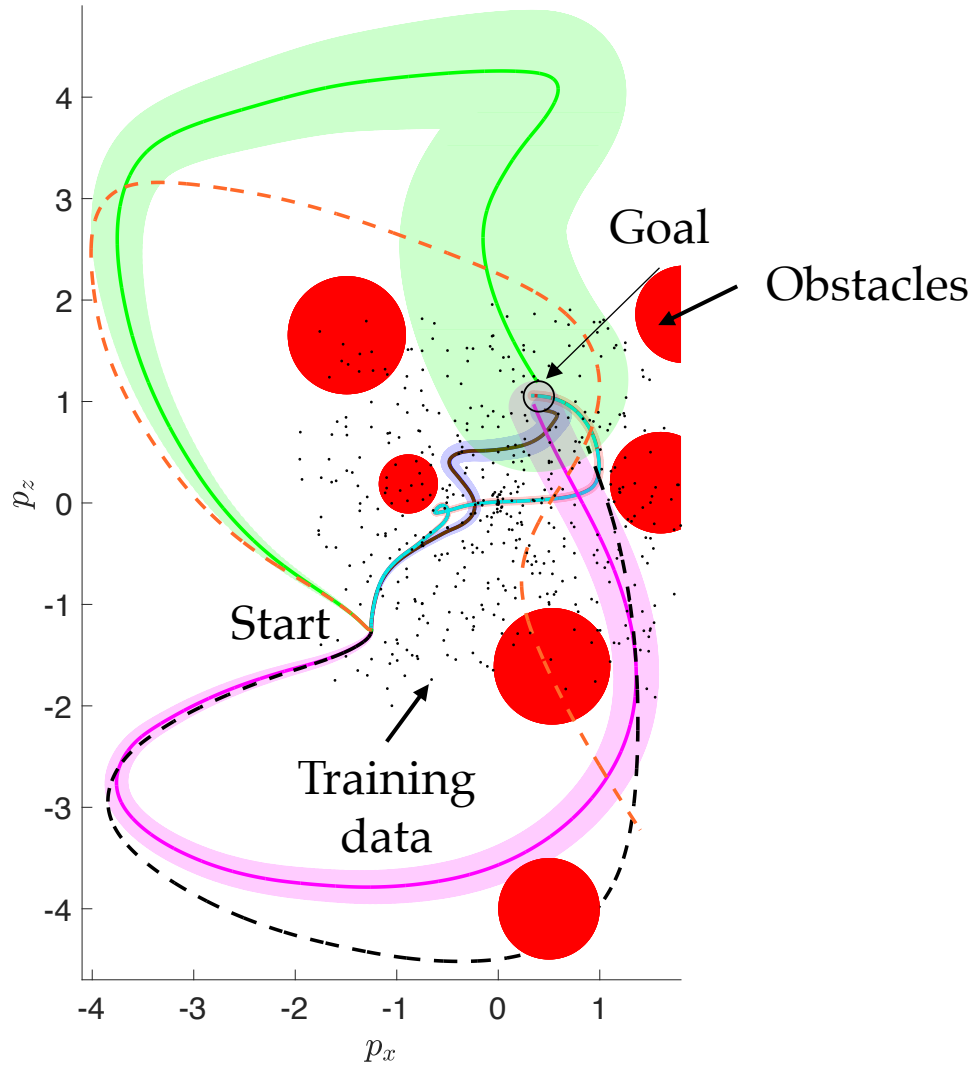




Results: 6D quadrotor environment

Legend:
Solid lines: planned trajectory
Dotted lines: executed trajectory
Transparent: trajectory tracking error tubes

Baselines:
B1: model error bound = *average* training error
B2: model error bound = *maximum* training error
B3: *exit D*; model error bound = *maximum* training error
B4: *exit D*; our *Lipschitz-based* model error bound



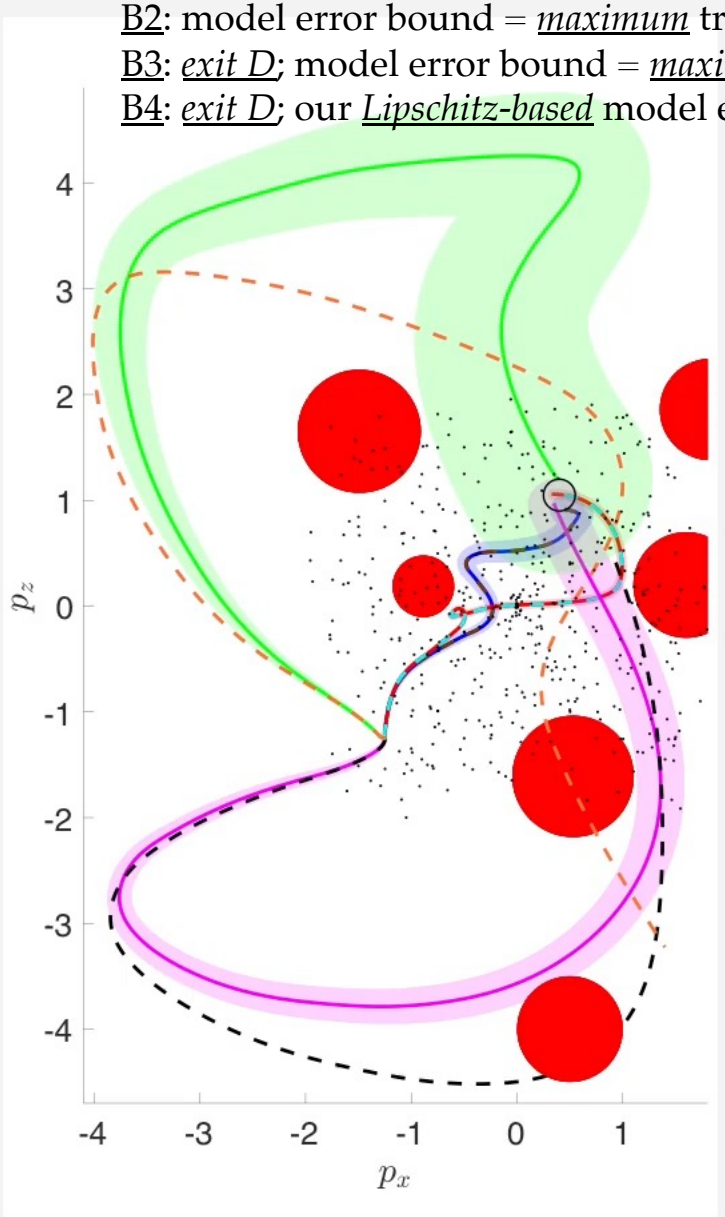
Baselines:

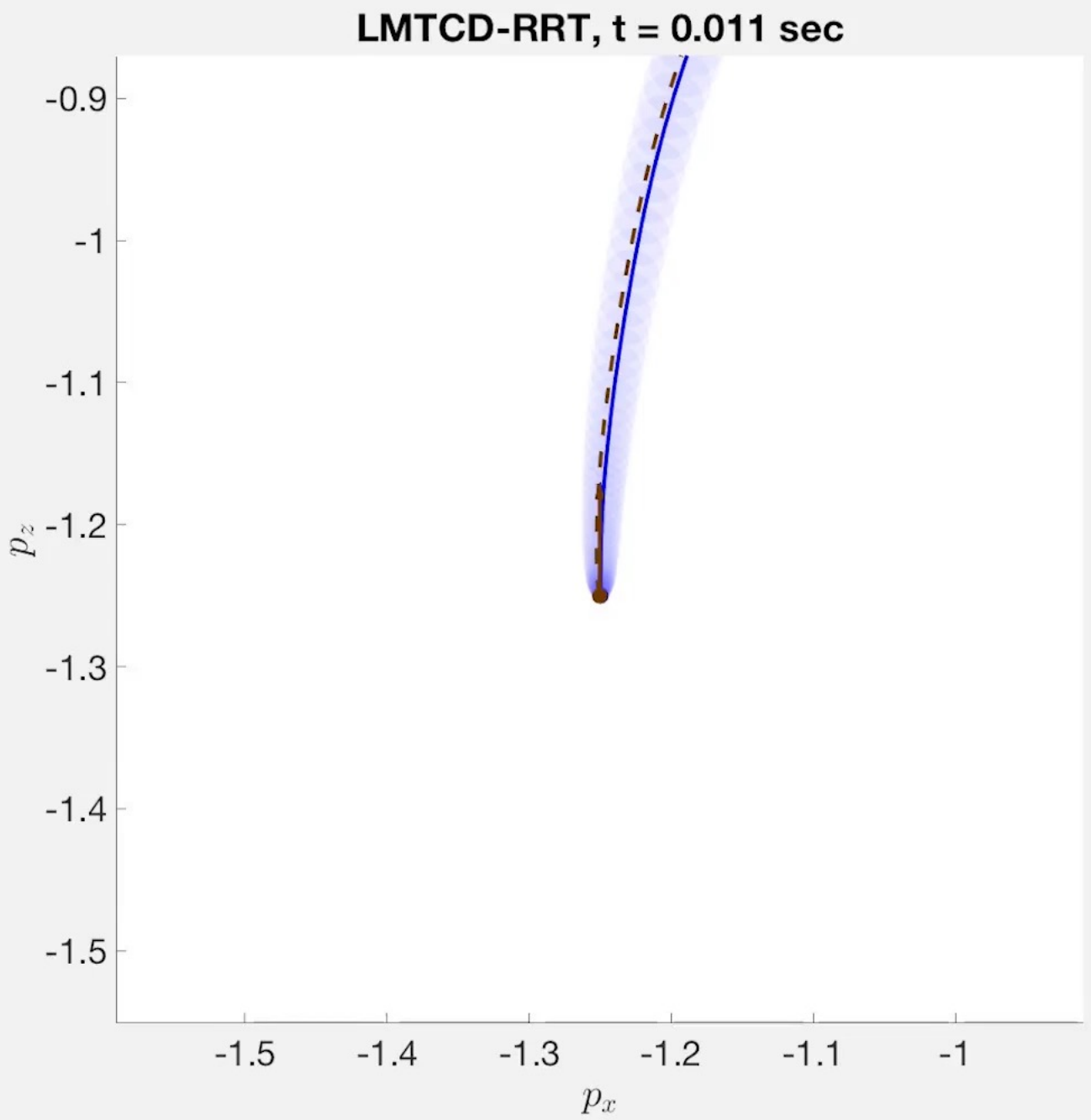
B1: model error bound = *average* training error

B2: model error bound = *maximum* training error

B3: *exit D*; model error bound = *maximum* training error

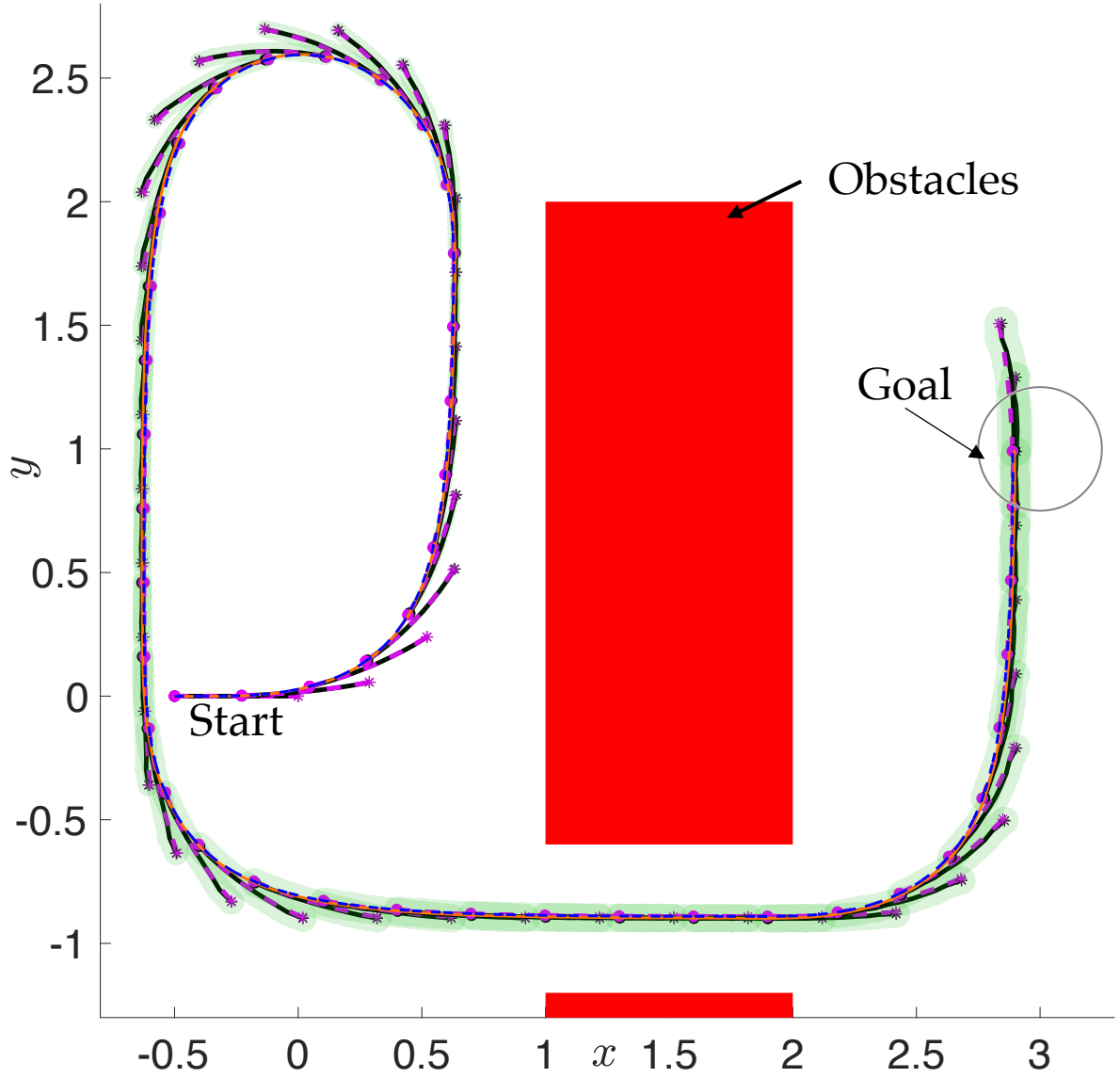
B4: *exit D*; our *Lipschitz-based* model error bound

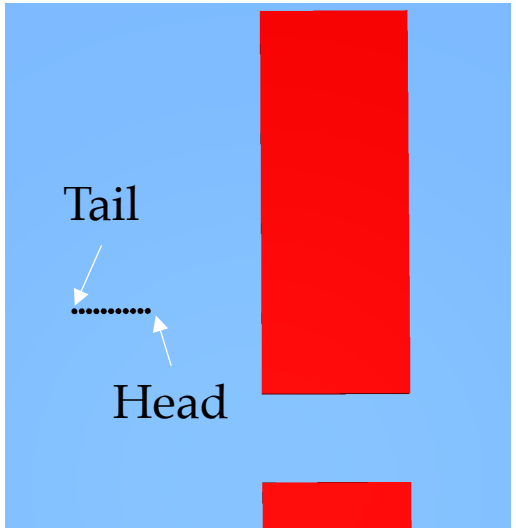




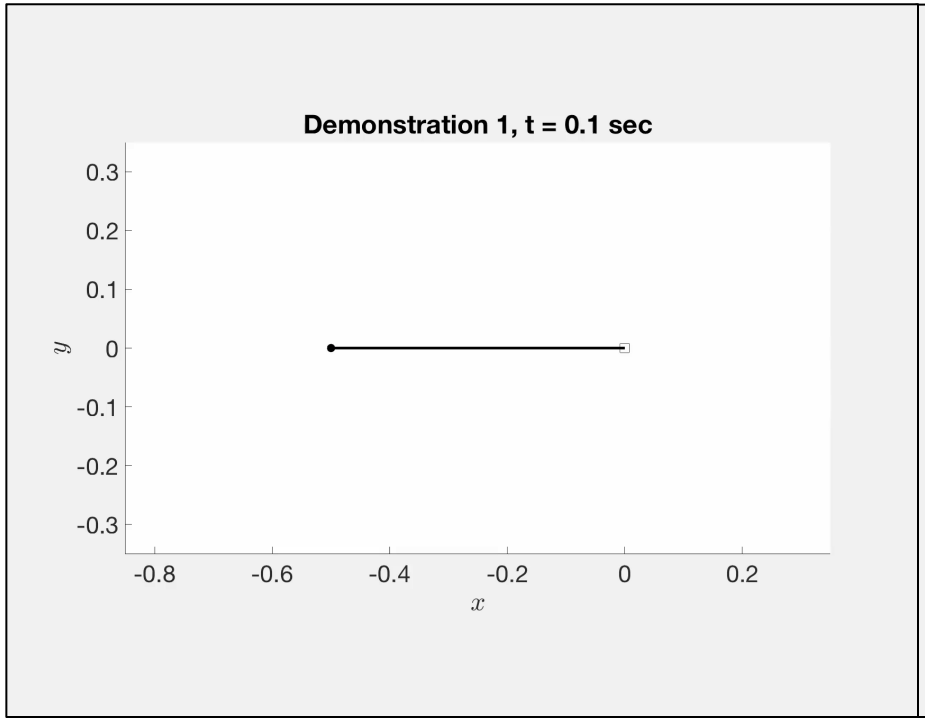
Results: 22D rope environment

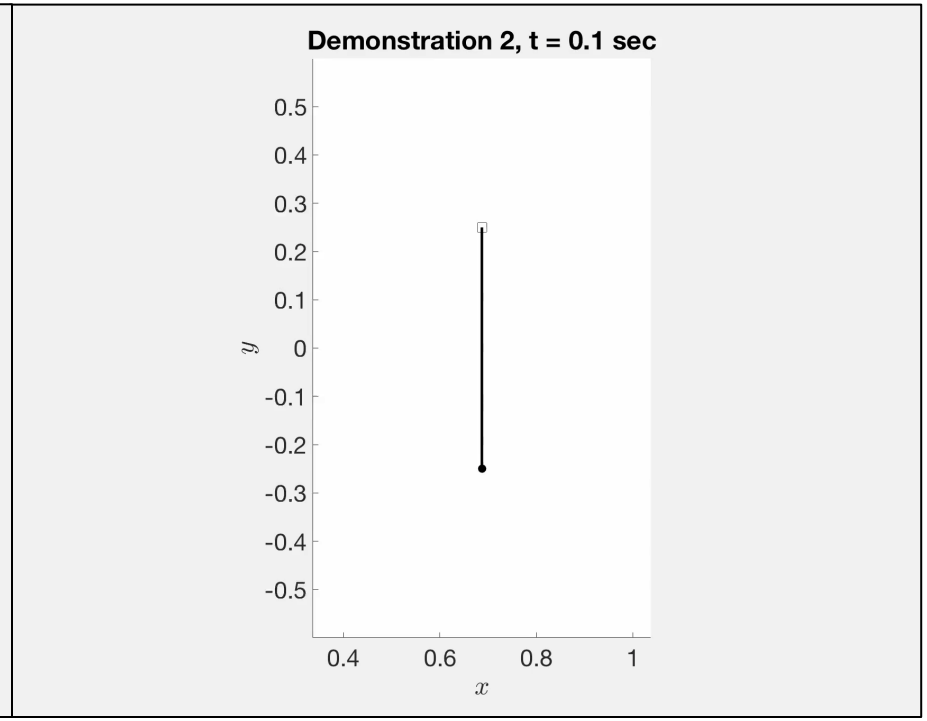
Legend:
: planned trajectory of the tail
: executed trajectory of the tail
: snapshots of the full rope in planning
: snapshots of the full rope in execution
: trajectory tracking error tubes
* * : rope head at a given snapshot
● ● : rope tail at a given snapshot

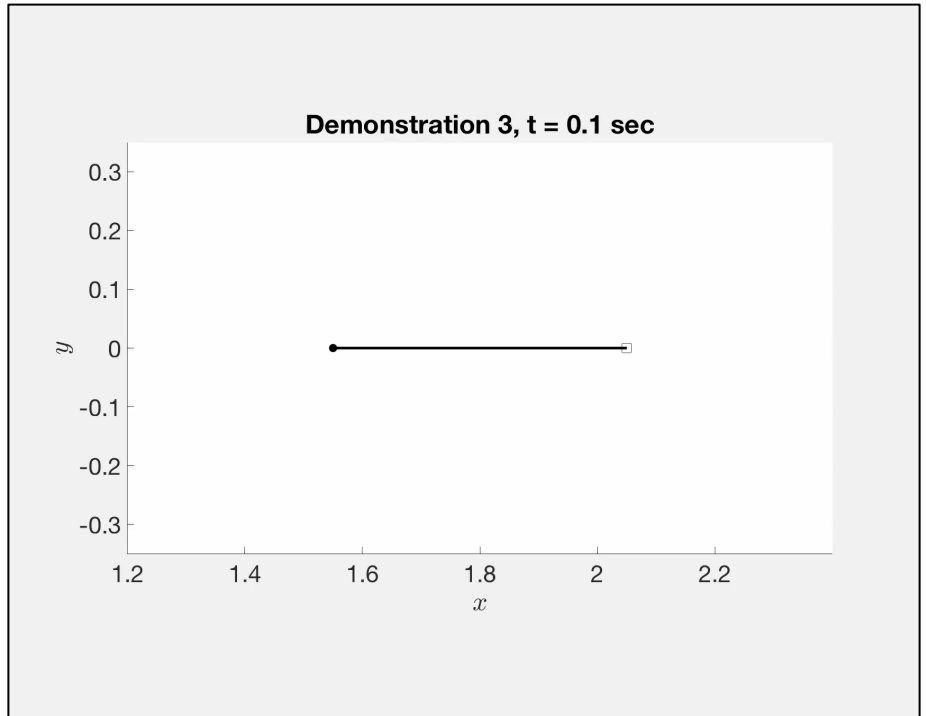




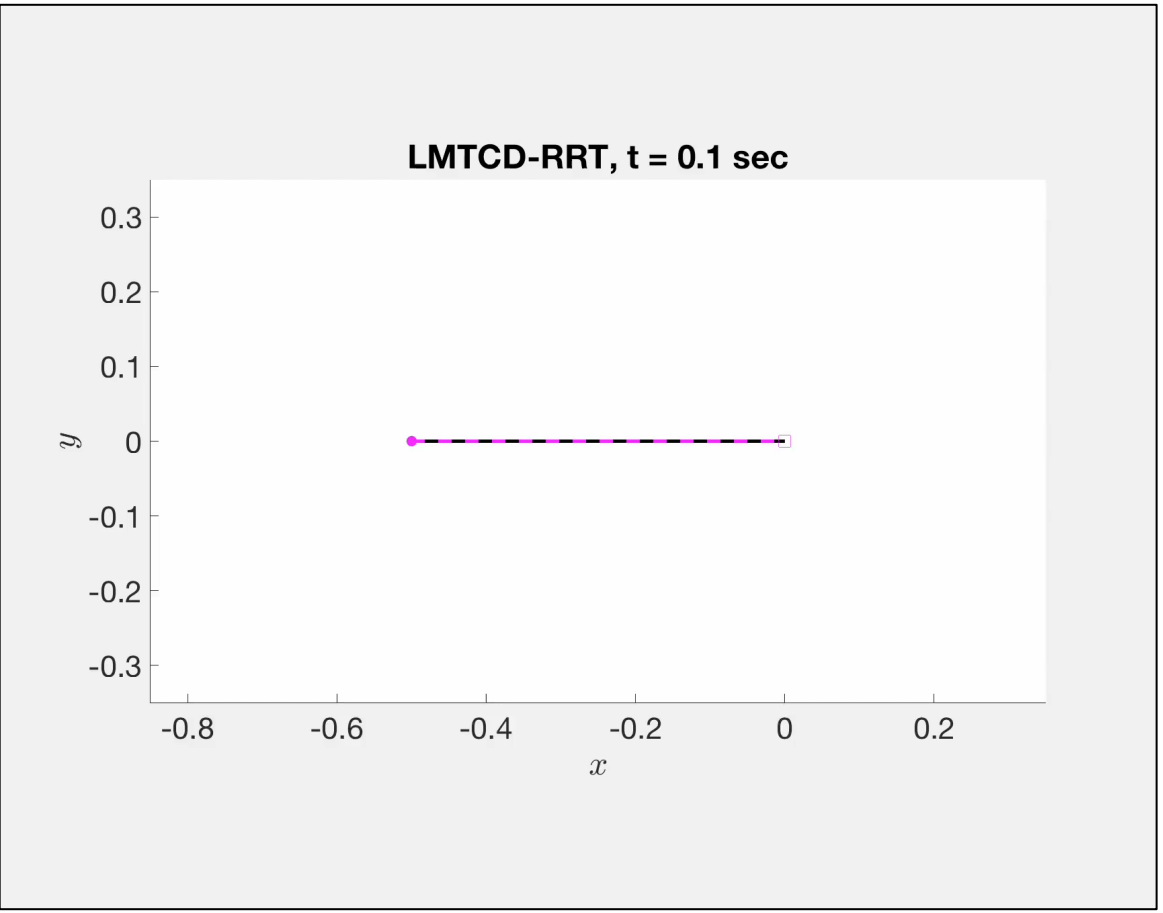
Baselines:
B1: model error bound = *average* training error
B2: model error bound = *maximum* training error
B3: *exit D*; model error bound = *maximum* training error
B4: *exit D*; our *Lipschitz-based* model error bound

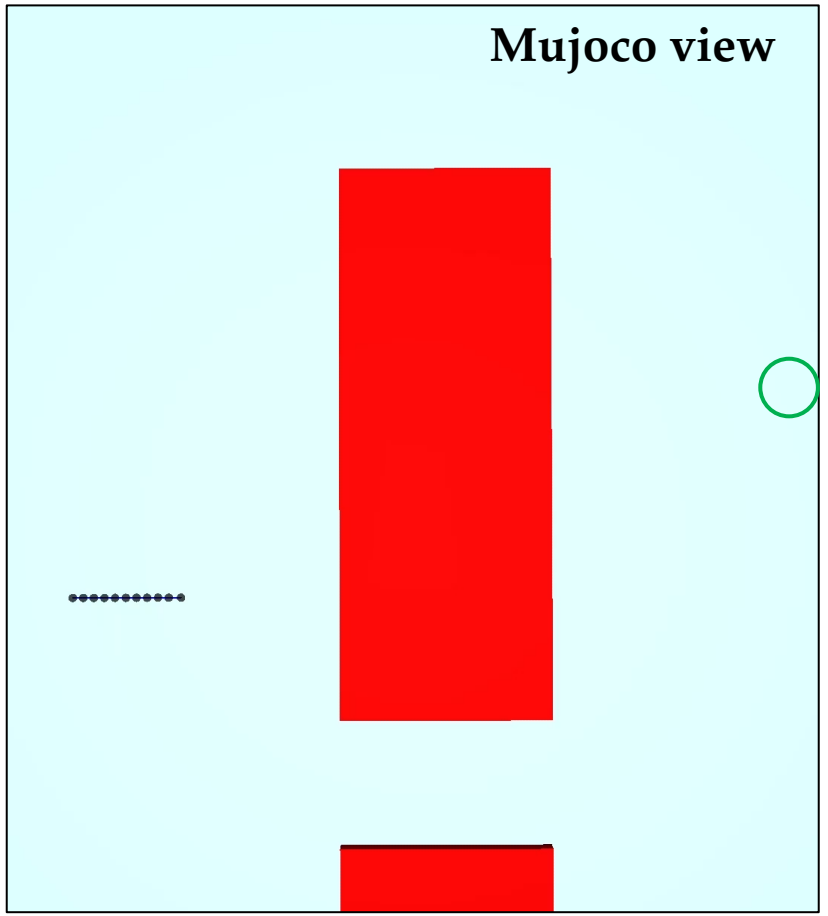






Given rope dynamics data





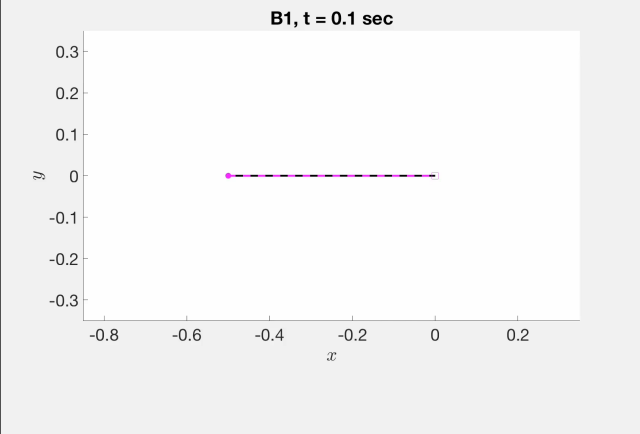
Baselines:

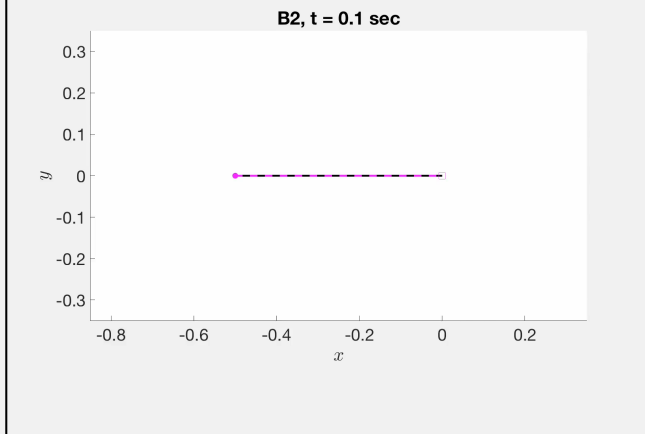
B1: model error bound = *average* training error

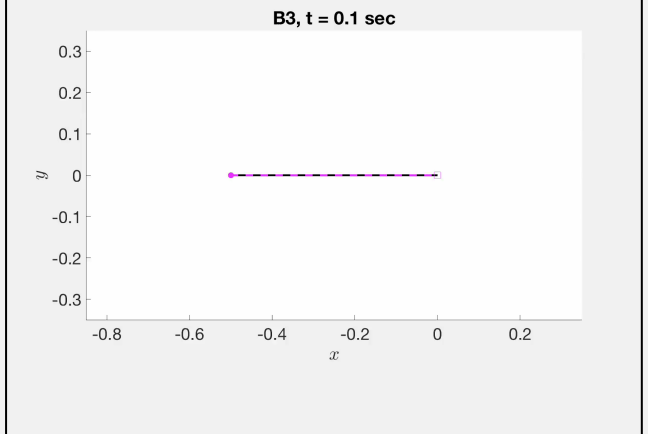
B2: model error bound = *maximum* training error

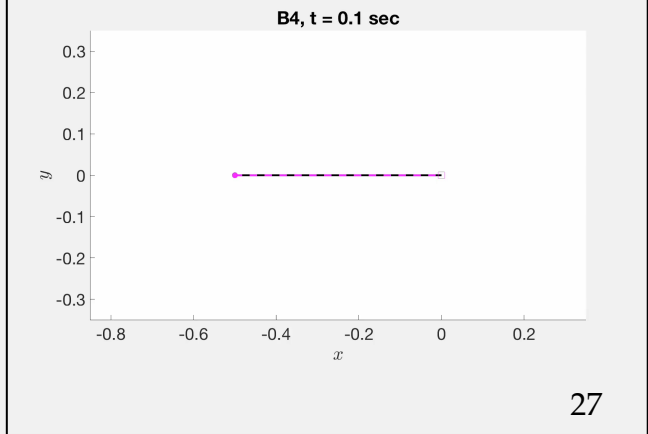
B3: *exit D*; model error bound = *maximum* training error

B4: *exit D*; our *Lipschitz-based* model error bound





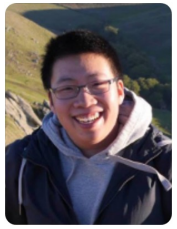
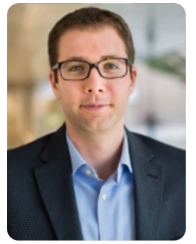




Safe output feedback control with a learned perception module

Reference:

- **Safe Output Feedback Motion Planning from Images via Learned Perception Modules and Contraction Theory (WAFR'22)**

Joint work with Glen Chou  and Dmitry Berenson 

Problem setting

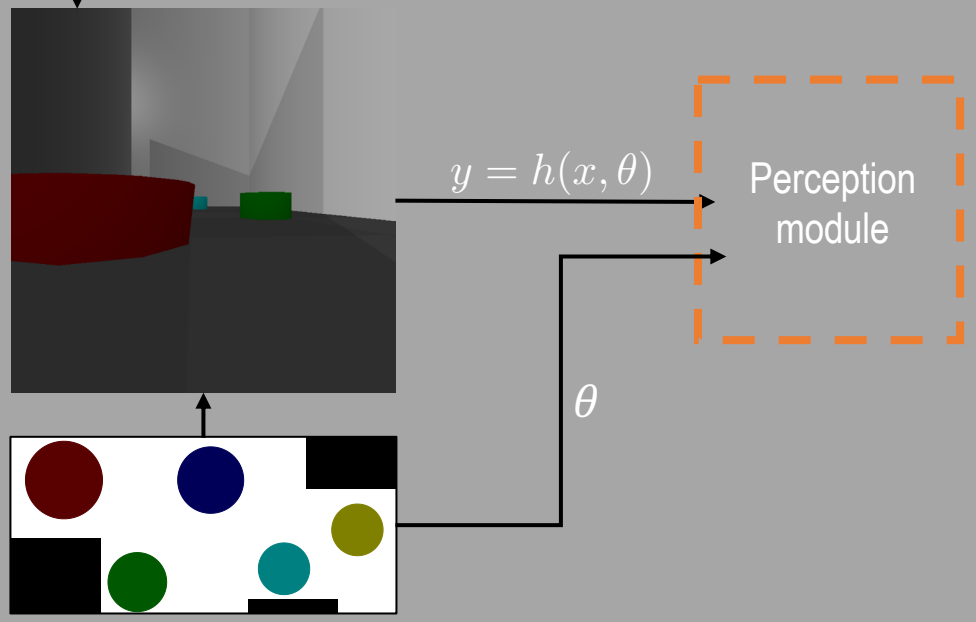
state

$$\dot{x}(t) = f(x(t)) + Bu(t) + w_{\mathrm{dyn}}(t)$$
$$y(t) = h(x(t), \theta)$$

control

model error

high-dimensional (e.g., an RGB-(D) image)



context vector θ (e.g., map)

Assume: incremental exponential stabilizability, observability

(exists controller that shrinks tracking error exp. quickly)

identical image trajectories implies identical state trajectories

Known: nominal dynamics f, B

bound on model error $\|w_{\mathrm{dyn}}(t)\| \leq \overline{w}_{\mathrm{dyn}}$

Unknown: h . But, given a dataset: $\{(x_i^{\mathrm{obs}}, \theta_i, y_i)\}_{i=1}^{N}$

And an iid validation dataset: $\{(x_i^{\mathrm{obs}}, \theta_i, y_i)\}_{i=1}^{N_v}$

Objective:

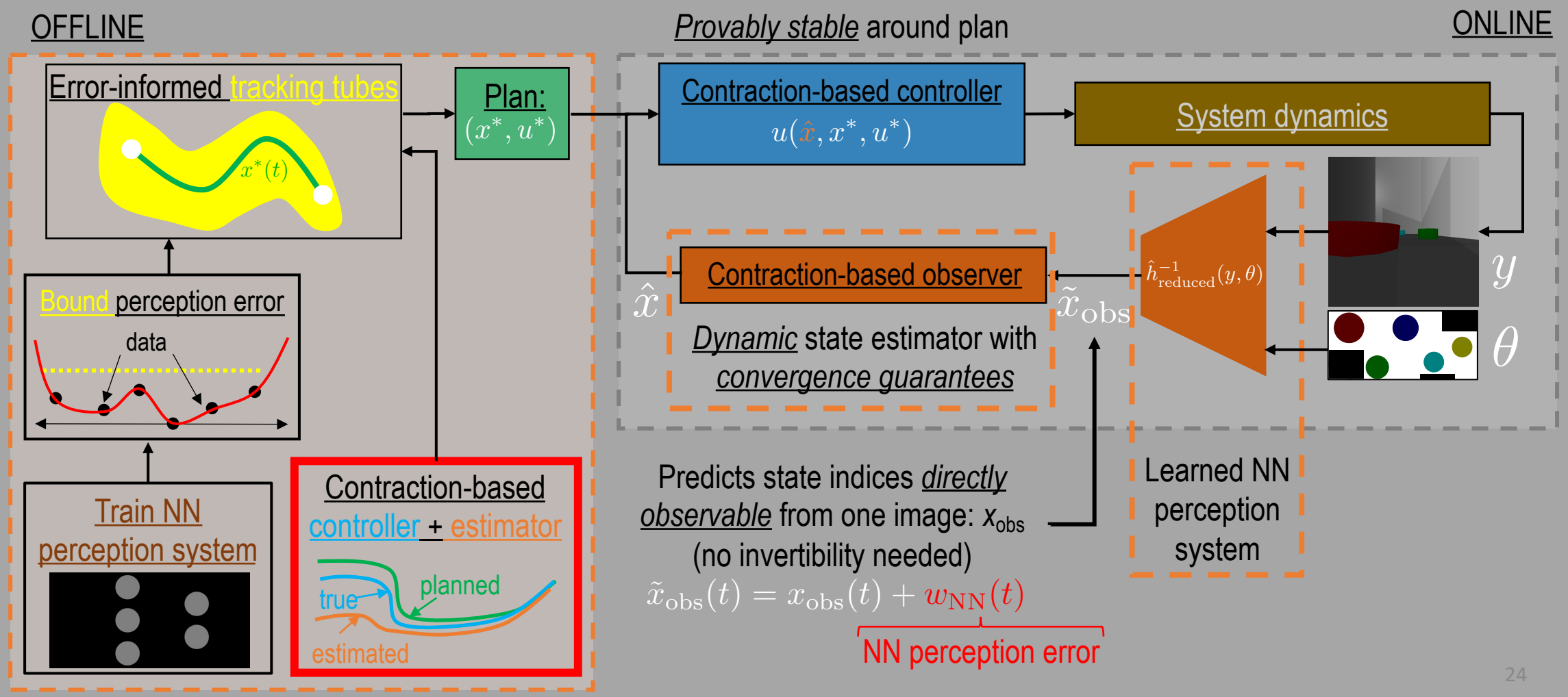
given start, goal, θ, safely reach the goal at runtime using observations *y(t)* and the learned perception module.

Key challenge:

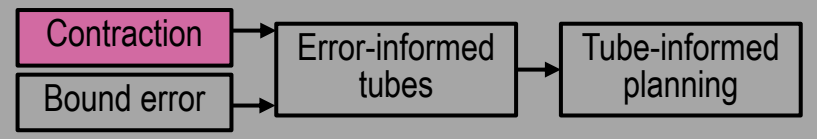
bounding all error sources and propagating to tracking error

Method

Key insight: Estimate bound on error in the learned perception module, and propagate it to low-level control via contraction theory.

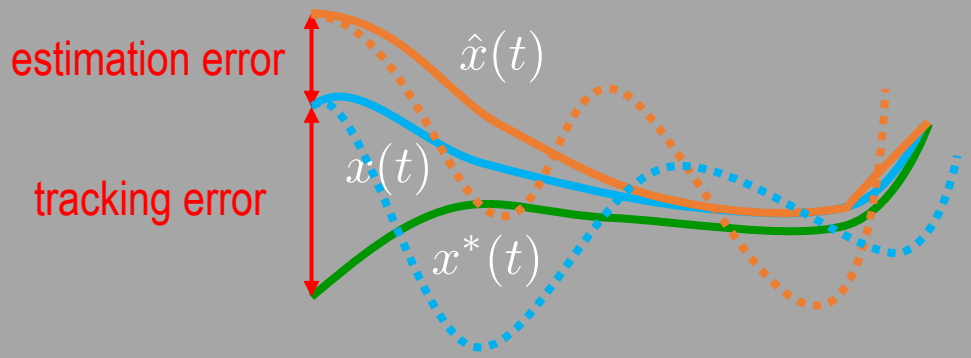


Contraction-based control & estimation

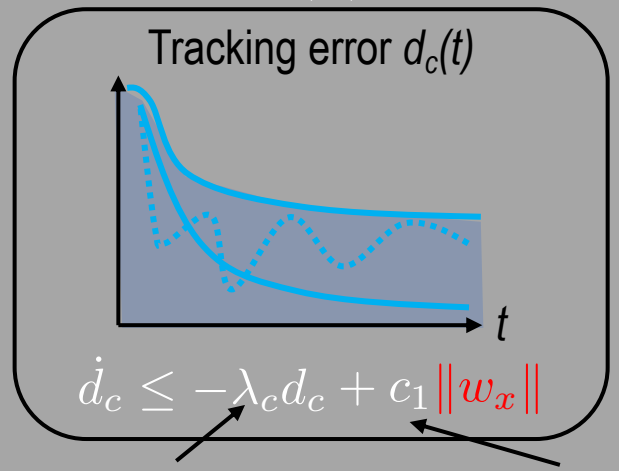


Guarantees (for system with no disturbance):

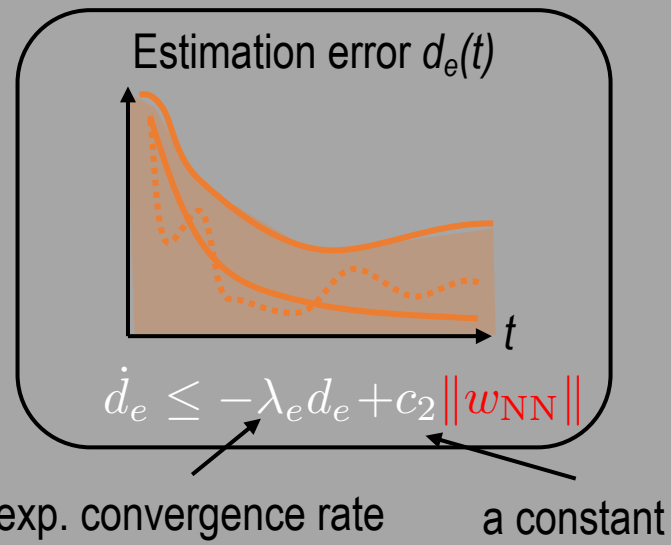
- controller converges exponentially quickly around all feasible plans *(x*(t), u*(t))*
- state estimator converges exp. quickly to the true state, for all feasible *x(t)*
- prior work: obtain via convex optimization [Manchester and Slotine '14, Singh et al. '17]



$$\text{dynamics} = f(x) + Bu + w_x$$



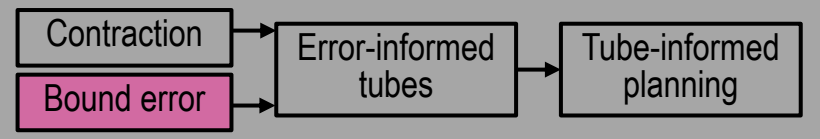
$$\text{measurement} = x_{\text{obs}} + w_{\text{NN}}$$



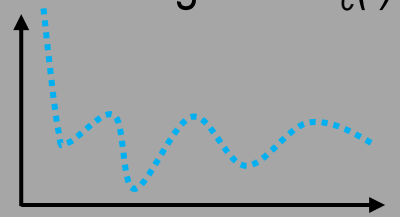
With disturbance: state/state estimate remains in a *tube* that scales with *disturbance size*

Key: for valid tubes, must get valid upper bound on disturbance.

From error sources to tracking error

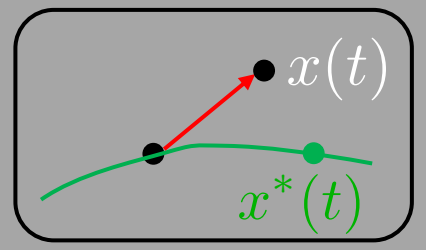


Tracking error $d_c(t)$

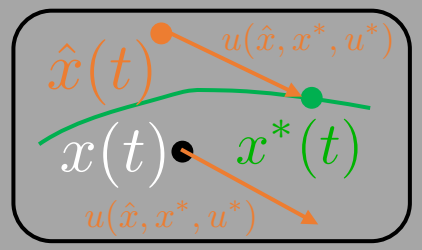


$$\dot{d}_c(t) \le -\lambda_c d_c(t) + c_1 \|w_x(t)\|$$

Dynamics error:



Control mismatch error:



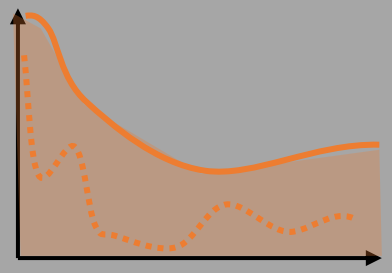
(driven by state estimate error)

$$w_x = w_{\text{dyn}} + w_{\text{mismatch}}$$

$$\|w_{\text{dyn}}(t)\| \le \bar{w}_{\text{dyn}}$$

EVT-β estimated Lipschitz constant

State estimate error $d_e(t)$



$$\|w_{\text{NN}}\| \le \bar{w}_{\text{NN}}$$

$$\dot{d}_e(t) \le -\lambda_e d_e(t) + c_2 \|w_{\text{NN}}(t)\|$$

- Bound error in a *trusted domain* $D \subseteq \mathcal{X} \times \Theta$, which we want to stay in at runtime.
- Select *D* to be a set *near the training data*.
- Use iid validation data + extreme value theory (EVT) [1, 2] to get upper bound $\bar{w}_{\text{NN}}$, valid with user-specified probability β *(statistical test)*.

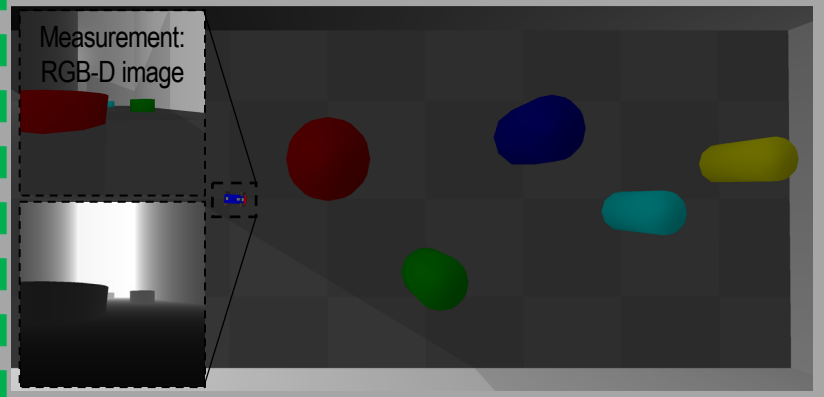
$$\begin{bmatrix} \dot{d}_c \\ \dot{d}_e \end{bmatrix} \le \begin{bmatrix} -\lambda_c & L_{\Delta k} \\ (*) & -\lambda_e \end{bmatrix} \begin{bmatrix} d_c \\ d_e \end{bmatrix} + \begin{bmatrix} \sqrt{\bar{\lambda}_{D_c}(M_c)}\bar{w}_x \\ \sqrt{\bar{\lambda}(W_e)}\bar{w}_x + \frac{\rho}{2}\bar{\lambda}(M_e)^{1/2}\left(L_{\hat{h}^{-1}}\bar{w}_y + \bar{\epsilon}_{\{1,\tilde{2},3\}}(x^*,\theta)\right) \end{bmatrix}$$

[1] Chou, Ozay, Berenson; CDC'21.

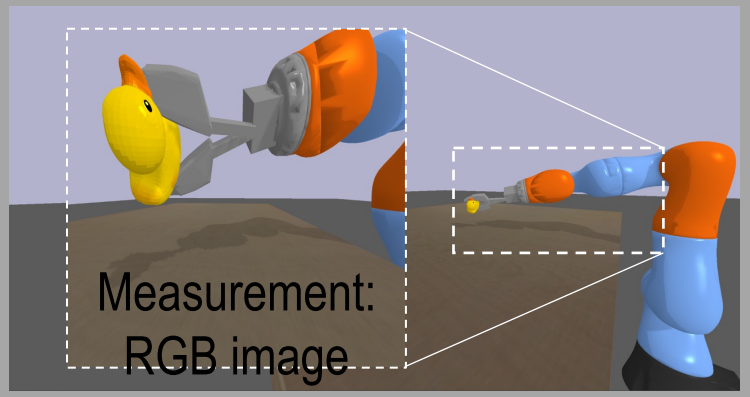
[2] De Haan and Ferreira; Springer'07.

Results

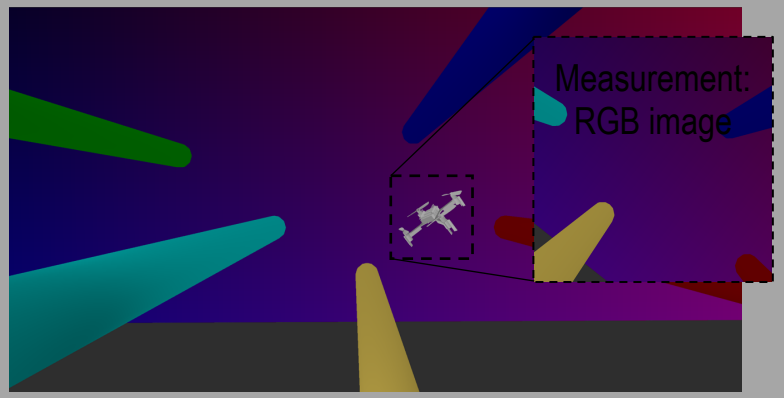
4D nonholonomic ground vehicle



14D acceleration-controlled arm



6D underactuated quadrotor



Nonholonomic car

Nominal dynamics:

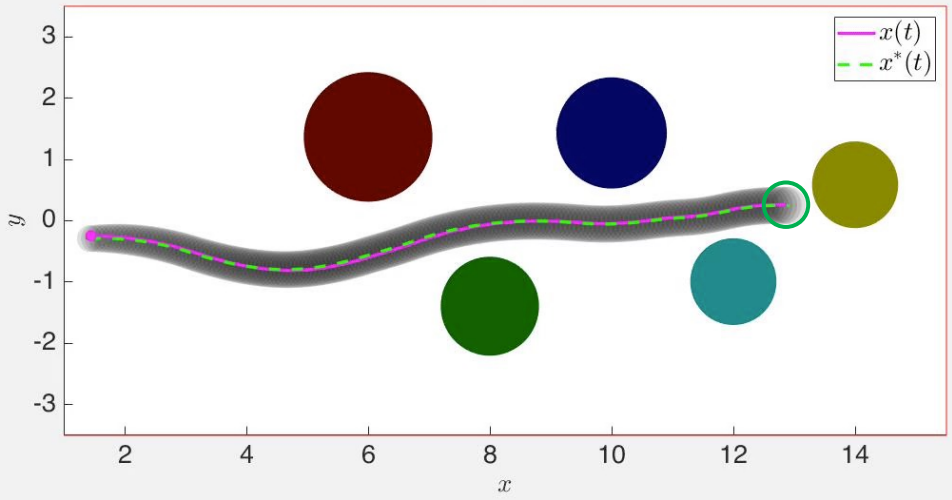
$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\phi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\phi) \\ v\sin(\phi) \\ u_1 \\ u_2 \end{bmatrix}$$

Goal: safely reach goal using RGB-D images at runtime

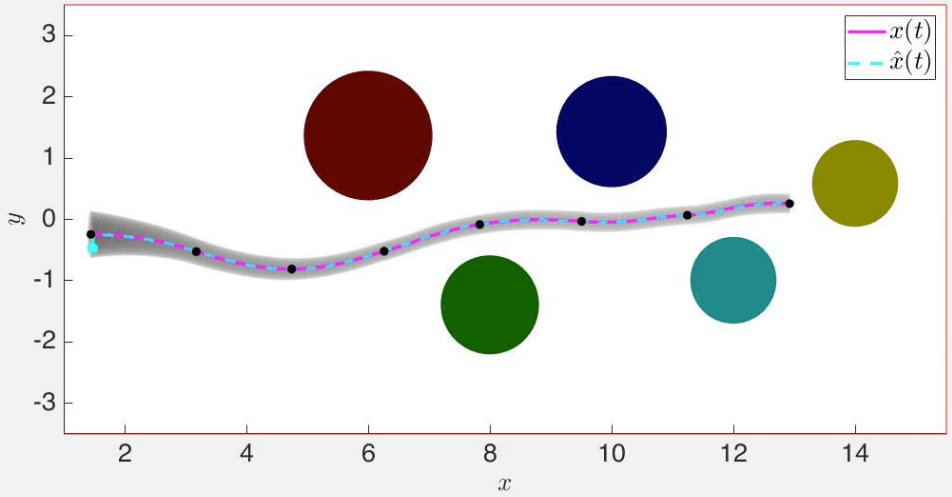
Perception module:
y = RGB-D image
θ = obstacle locations
output = p_x, p_y, ϕ .

xy training data restricted to:
p_x in [0, 13.5]
p_y in [−2.5, −2.5]

Trajectory tracking

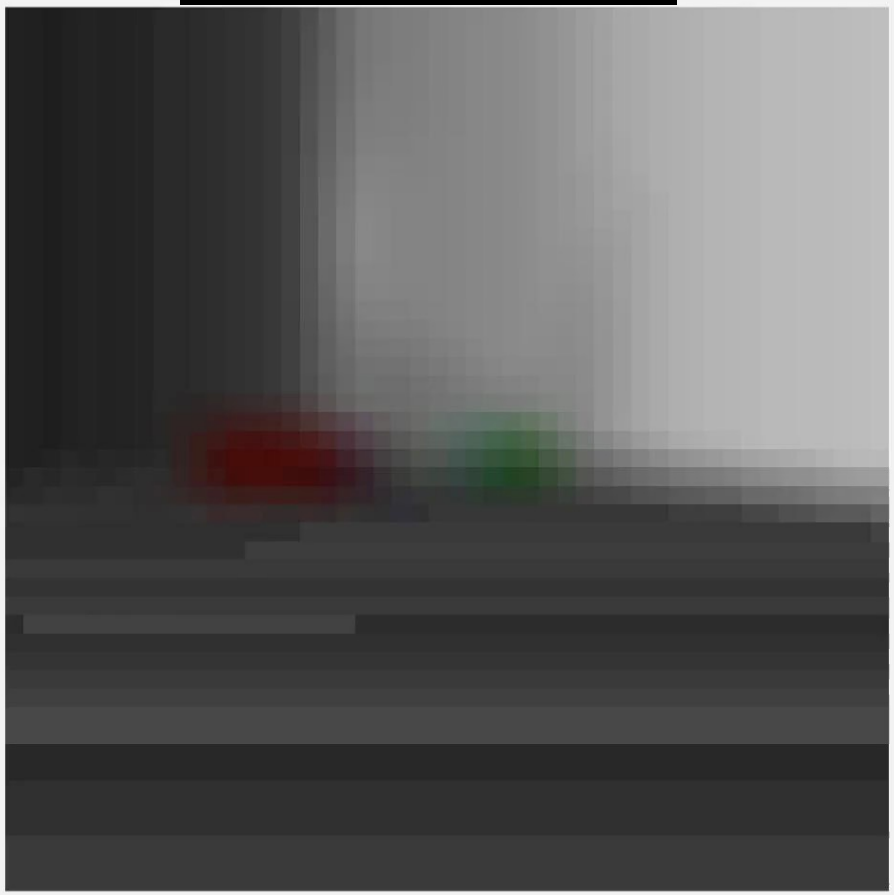


State estimation



Our method

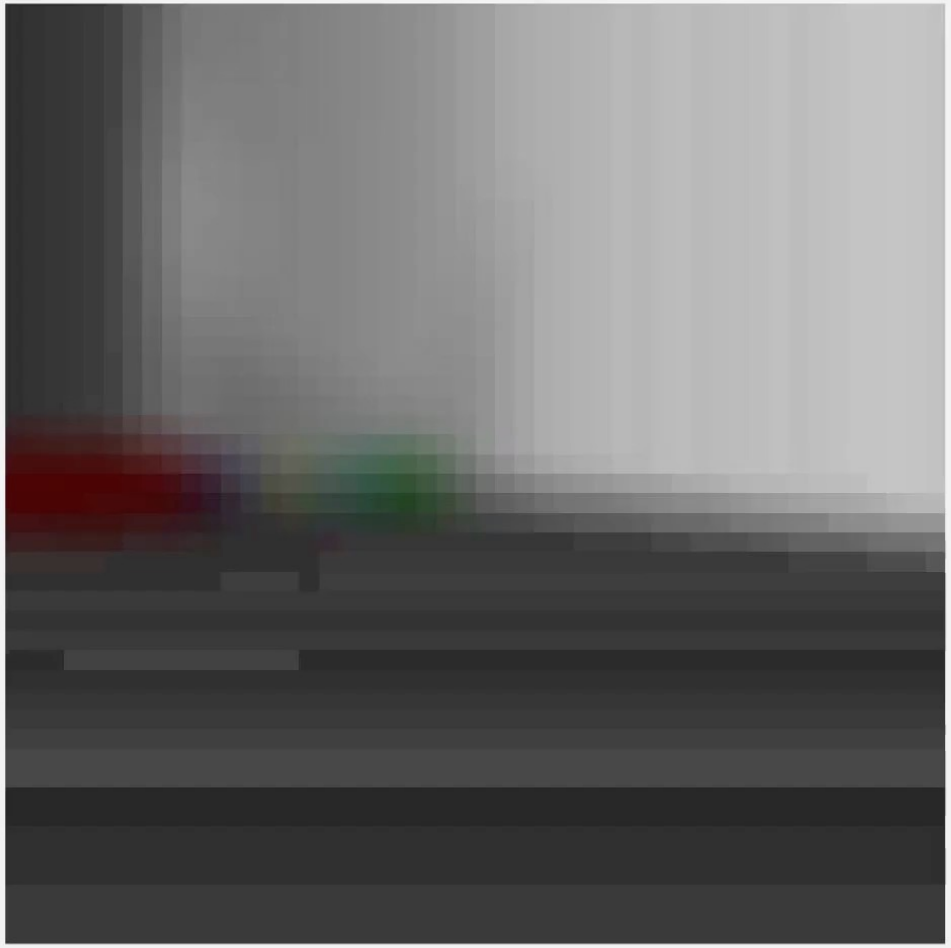
Runtime observations (RGB part)



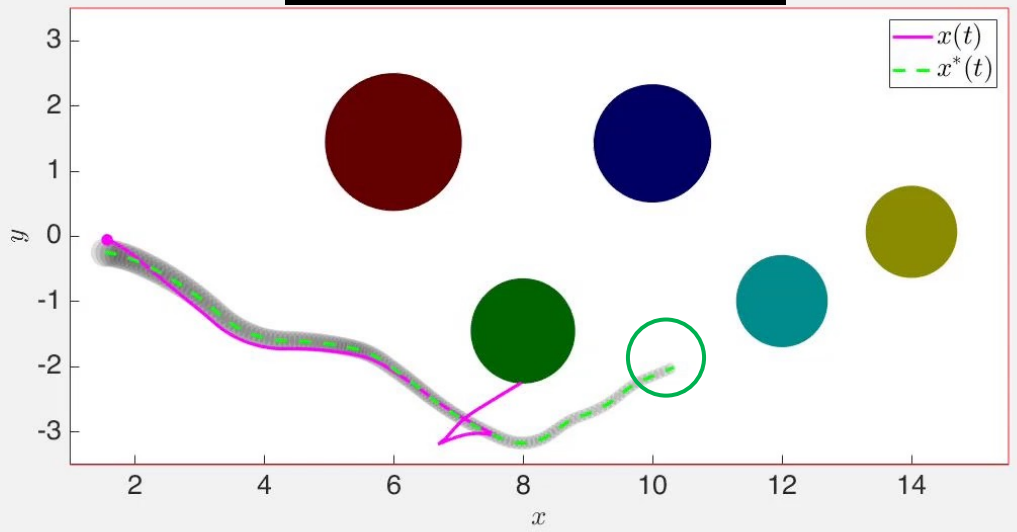
Legend:
Tubes, executed trajectory, planned trajectory, estimated trajectory

Baseline: ignoring the effect of perception error during planning

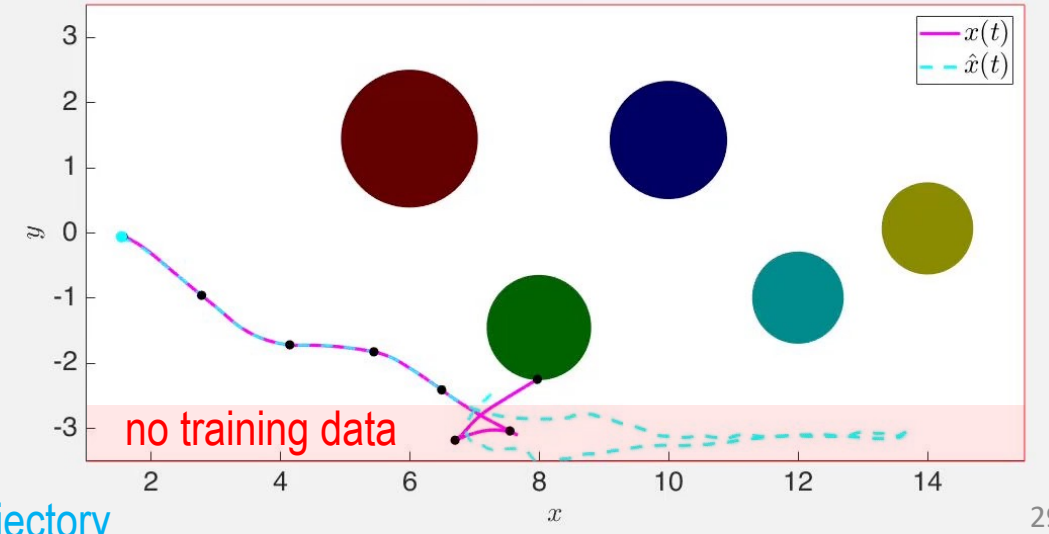
Runtime observations



Trajectory tracking



State estimation (no tubes)



Legend:

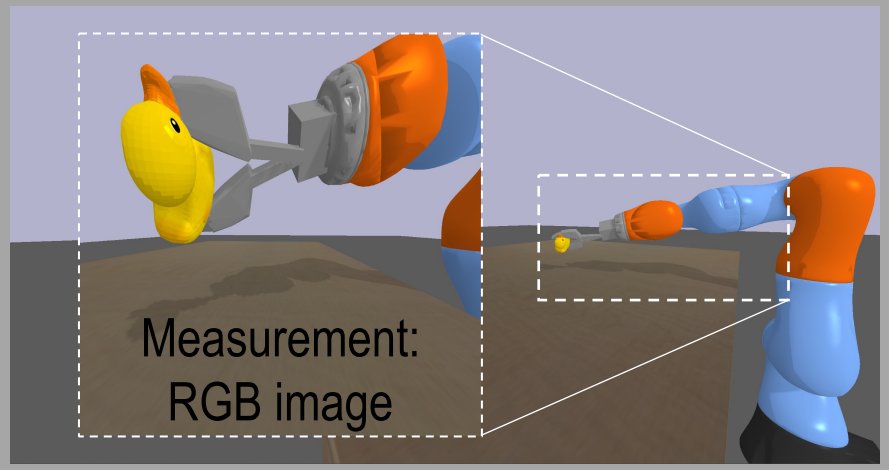
Tubes executed trajectory; planned trajectory; estimated trajectory

Guaranteed active perception on a 7DOF arm

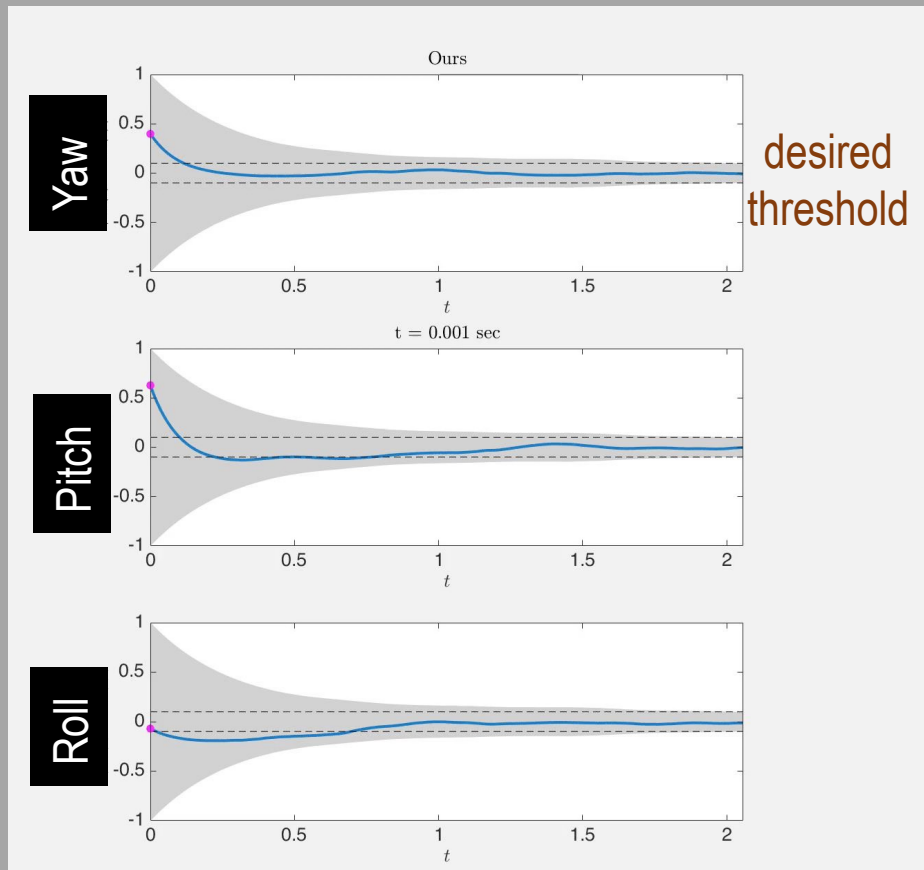
Goal: estimate duck pose relative to end effector, within a guaranteed accuracy; you can move the arm to get better readings

Perception module:
y = RGB image
θ = joint angles
output = (yaw, pitch, roll)

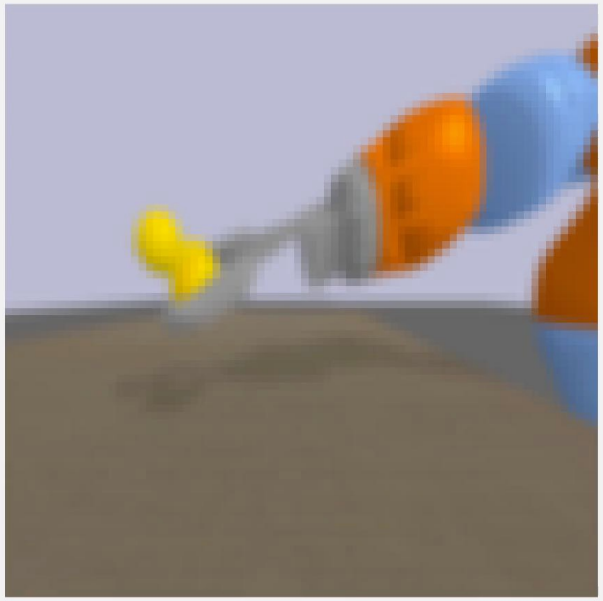
Result: tube encourages the arm move to low-error regions of the space



State estimate tubes



Runtime observations

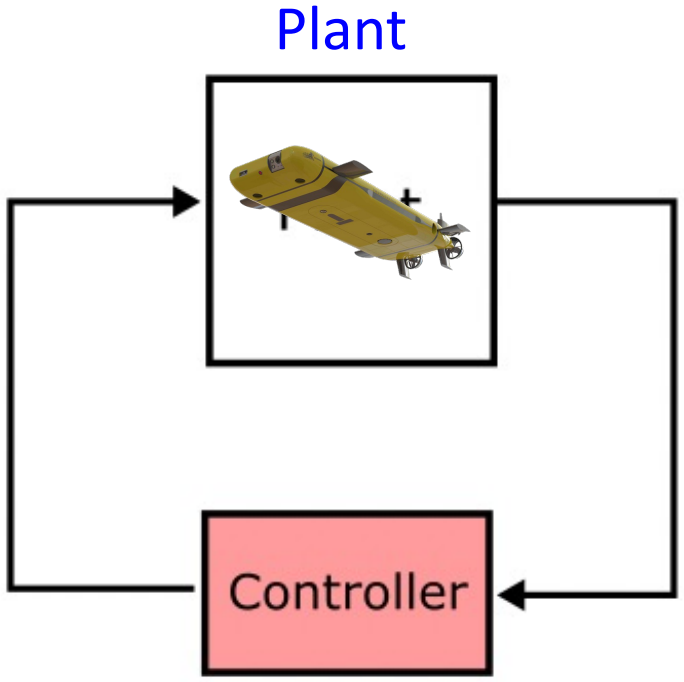


Legend: state estimate error tube (projections)

Task or mission specification

φ

LTL = High-level plan (logic structure) + Low-level predicates



High-level decisions
Trajectory planning
Low-level feedback control

Learning in-the-loop control:

- **Plant** can be unknown and need to be learned (ACC'19, TAC'22)
- **Task** can be unknown and need to be learned (RA-L'20, RSS'20, AURO'21)
- **Controller** can be hard to design or synthesize and can be learned instead (RA-L'21, **CDC'21**, EMSOFT'21, LCSS'23)
- **Perception module** typically does not have an analytical solution and is learned (EMSOFT'21, **WAFR'22**)

Observations

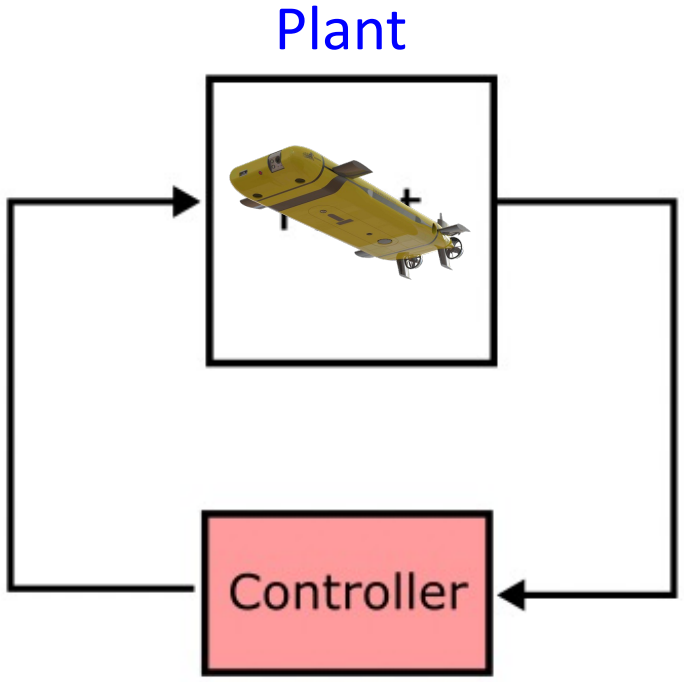
Learning while respecting architectures ➔ system-level guarantees

- Separation of dynamics, low-level control, planning, and high-level decision-making
  - "Easier" characterization and propagation of uncertainty between layers

Task or mission specification

φ

LTL =
High-level plan (logic structure) + Low-level predicates



High-level decisions
Trajectory planning
Low-level feedback control

Learning in-the-loop control:

- **Plant** can be unknown and need to be learned (ACC'19, TAC'22)
- **Task** can be unknown and need to be learned (RA-L'20, RSS'20, AURO'21)
- **Controller** can be hard to design or synthesize and can be learned instead (RA-L'21, **CDC'21**, EMSOFT'21, LCSS'23)
- **Perception module** typically does not have an analytical solution and is learned (EMSOFT'21, **WAFR'22**)

Observations

Mostly planning, what about control with a larger domain of validity?

- Recent work on hybrid Koopman-inspired liftings for learning-based backward reachability (LCSS'23)

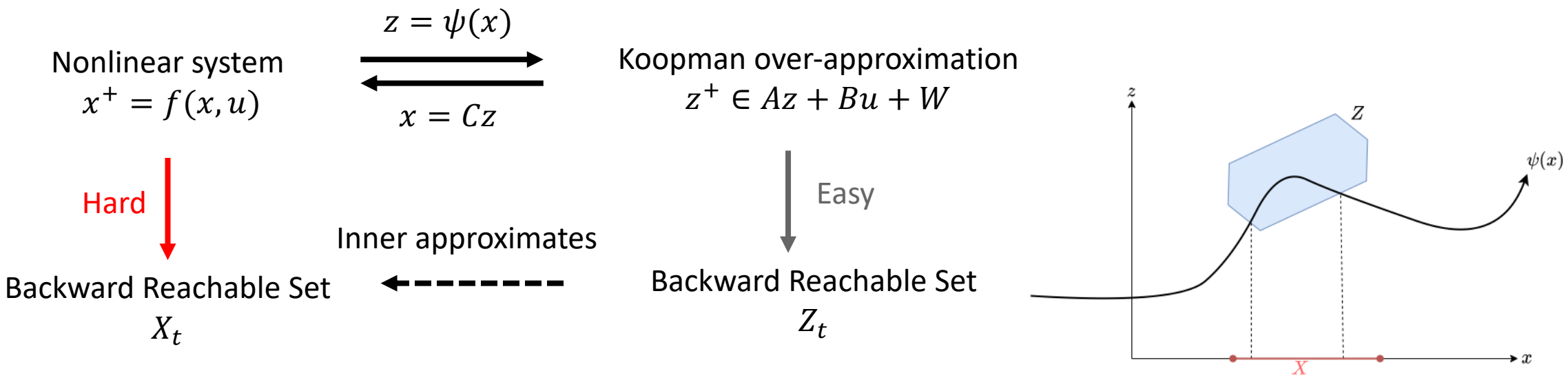
Learning the Model and Computing Backward Reachable Sets

Reference:

- **Koopman-inspired Implicit Backward Reachable Sets for Unknown Nonlinear Systems (L-CSS, 2023)**

Joint work with Haldun Balim  Antoine Aspeel  and Zexiang Liu 

Koopman-Inspired Safety Control for Unknown Nonlinear Systems



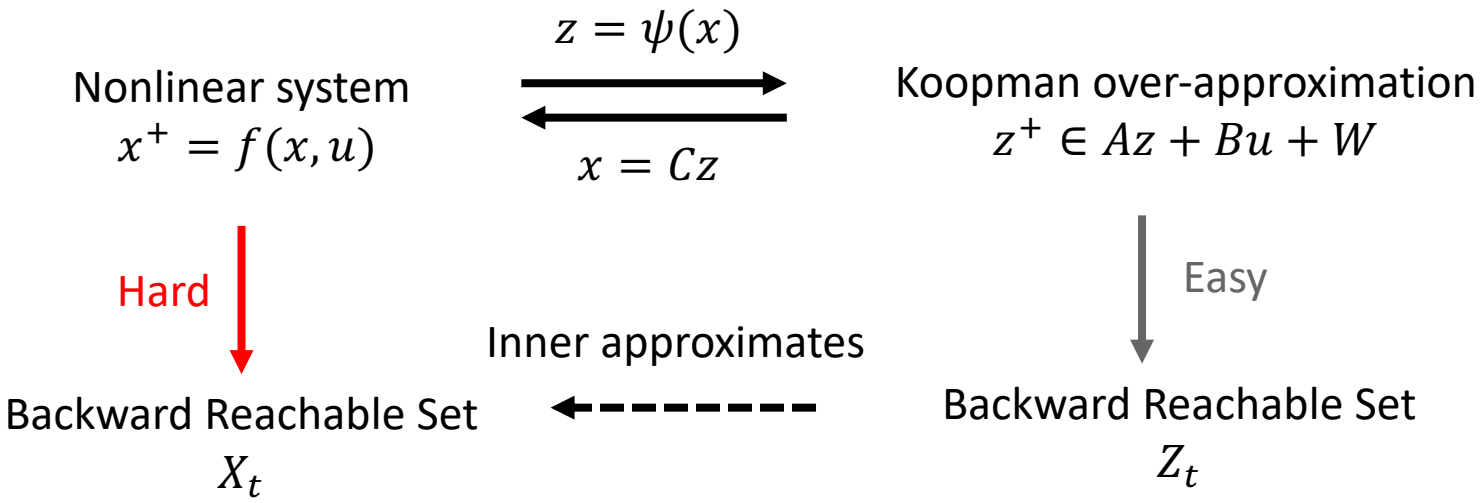
Two key ingredients:

- **Koopman over-approximation (KoA):** a simulation-like relation between the original system and Koopman-inspired abstraction

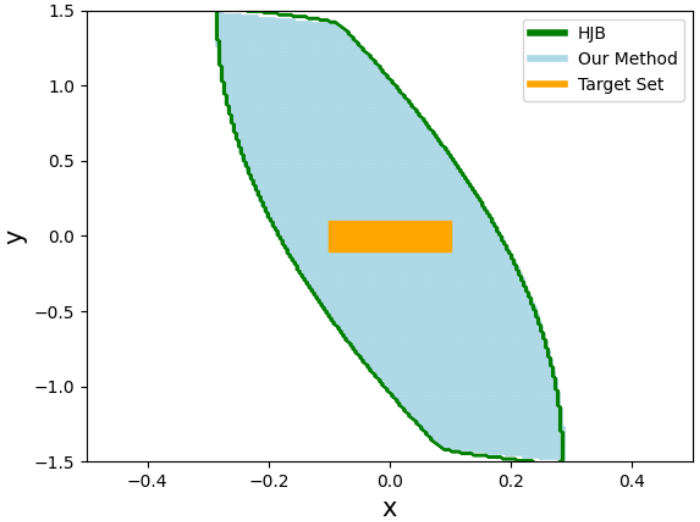
- **Implicit inner-approximation** Z of target set X where $\{x|\ \psi(x) \in Z\} \subseteq X$.

Balim, Aspeel, Liu, Ozay, L-CSS'23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems



Forced Duffing Oscillator

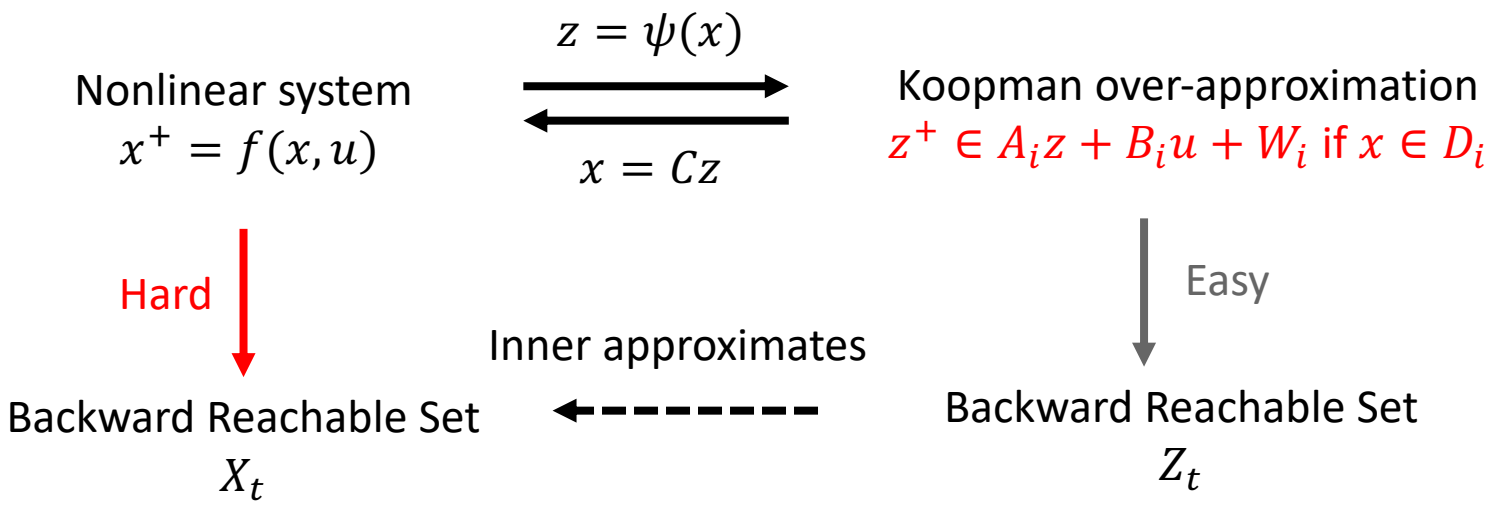


Some properties:

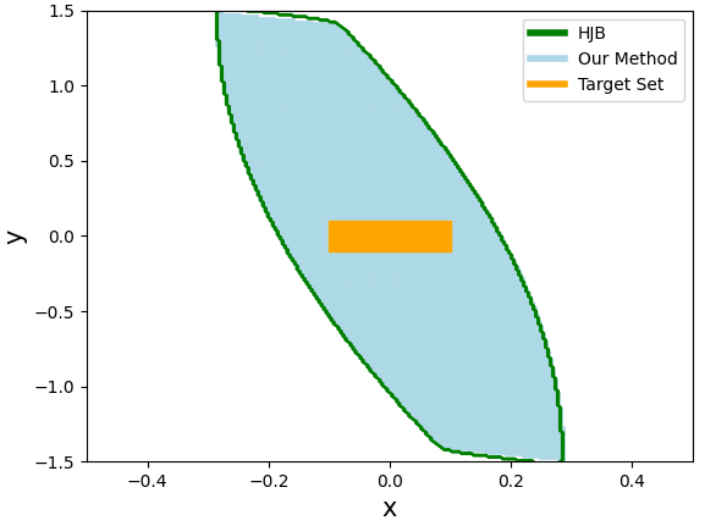
- Any lifting function ψ including x in its coordinates can be used;
- The Koopman over-approximation is learned from data;
- If we can estimate local Lipschitz constants, can improve computation further by updating the linear representation locally

Balim, Aspeel, Liu, Ozay, L-CSS'23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems



Forced Duffing Oscillator

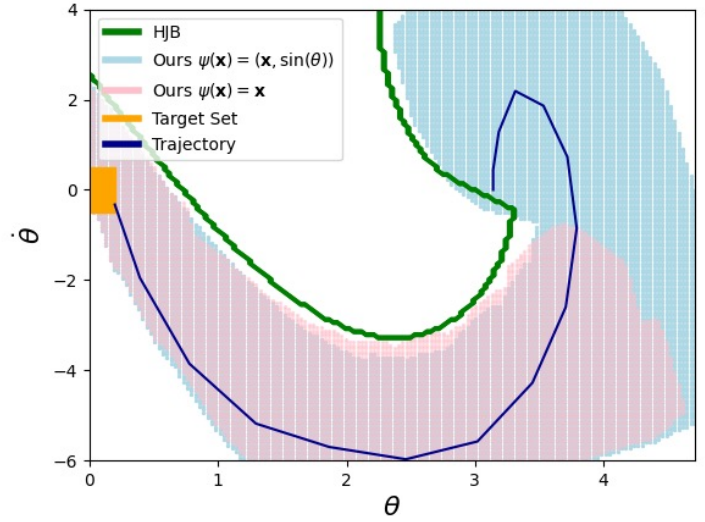


Single linearization is not enough:

- Different over-approximations are learned over local subdomains (leading to a PWA system) for better accuracy:
  - Experiments show that to obtain BRSs with similar sizes, the Koopman over-approximation requires less pieces than direct linearization (hybridization).

  Why do we need hybridization in the lifted space? See Zexiang Liu's talk tomorrow morning at 10am on non-existence of linear immersions for systems with multiple omega limit sets

Inverted Pendulum



Balim, Aspeel, Liu, Ozay, L-CSS'23

Koopman-Inspired Safety Control for Unknown Nonlinear Systems

$z = \psi(x)$

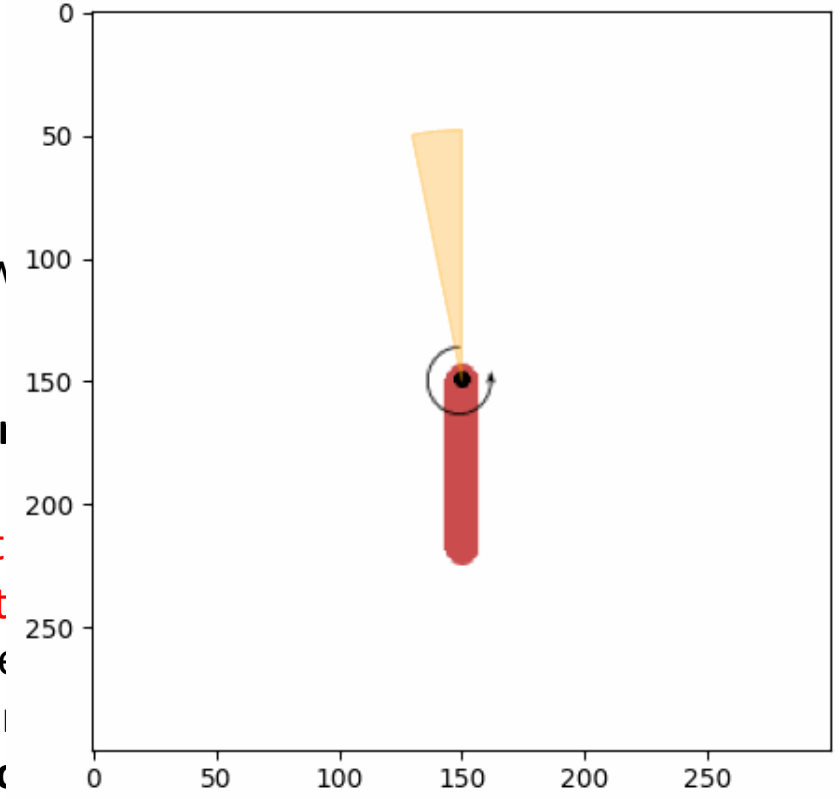
Nc

Backw

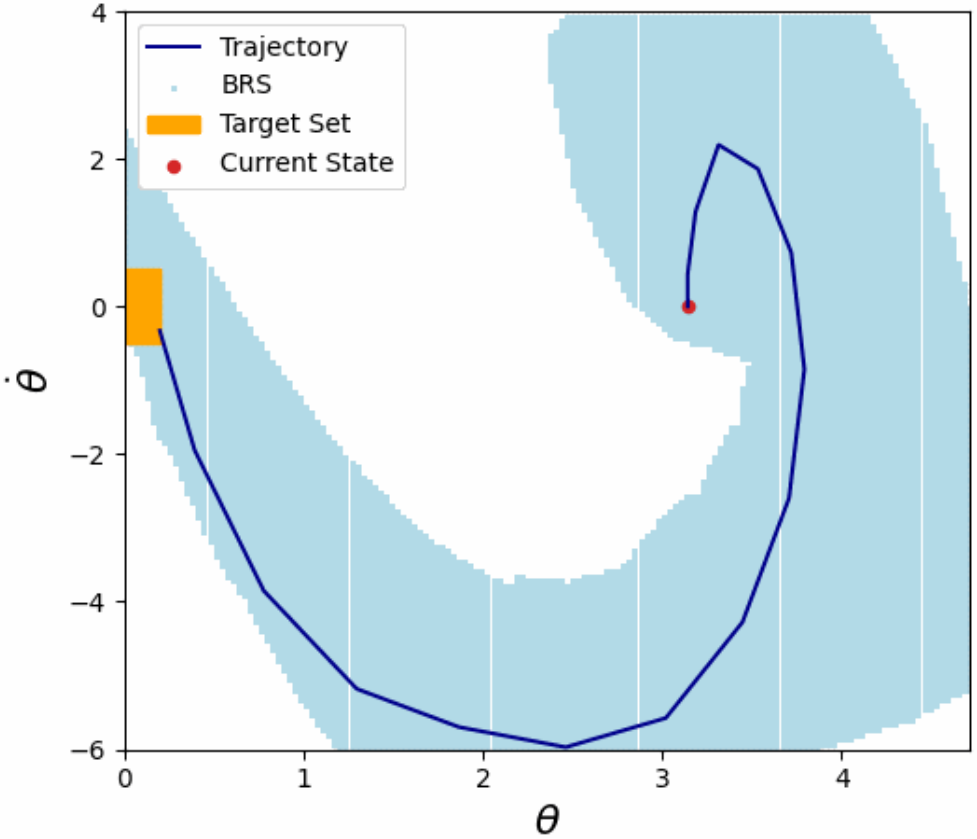
Single linear

- Different
  PWA syst
  - Expe
    appr

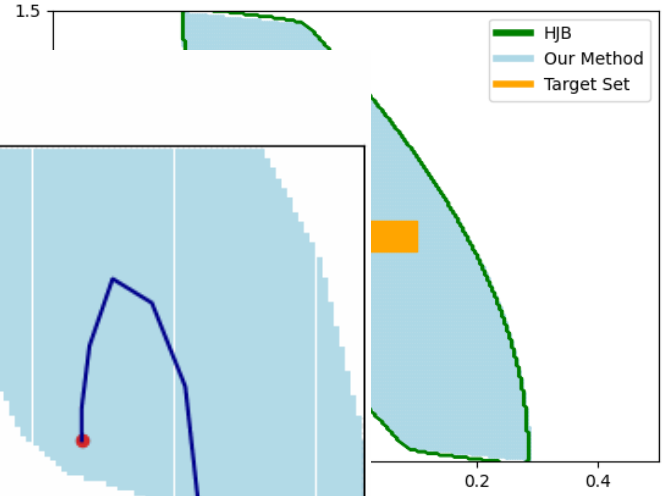
Why do
tomorr
with multiple omega limit sets



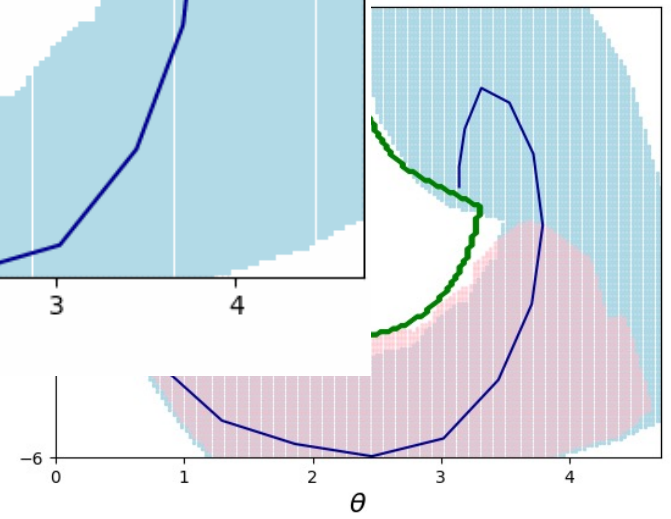
t = 0



Forced Duffing Oscillator



ndulum

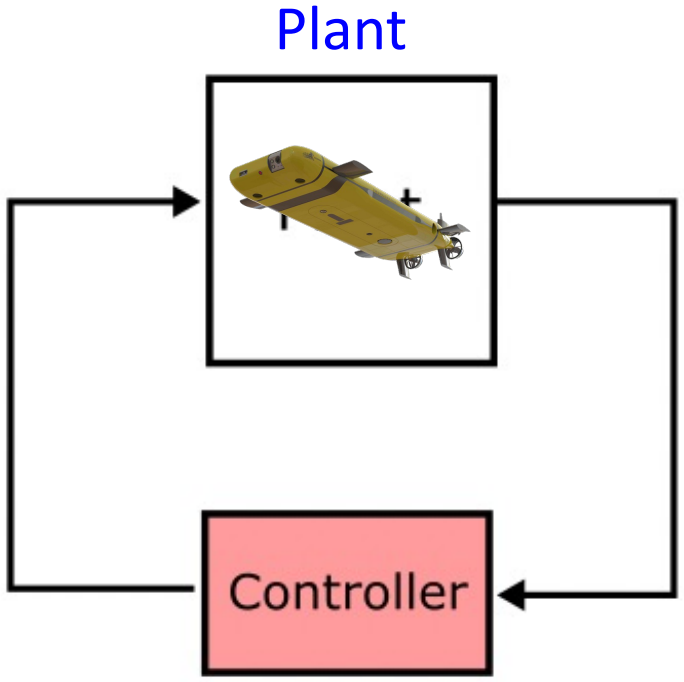


Balim, Aspeel, Liu, Ozay, L-CSS'23

Task or mission specification

φ

LTL =
High-level plan (logic structure) + Low-level predicates



High-level decisions
Trajectory planning
Low-level feedback control

Learning in-the-loop control:

- **Plant** can be unknown and need to be learned (ACC'19, TAC'22, **LCSS'23**)
- **Task** can be unknown and need to be learned (RA-L'20, RSS'20, AURO'21)
- **Controller** can be hard to design or synthesize and can be learned instead (RA-L'21, **CDC'21**, EMSOFT'21, **LCSS'23**)
- **Perception module** typically does not have an analytical solution and is learned (EMSOFT'21, **WAFR'22**)

Thanks!

More info and preprints available @ web.eecs.umich.edu/~necmiye/