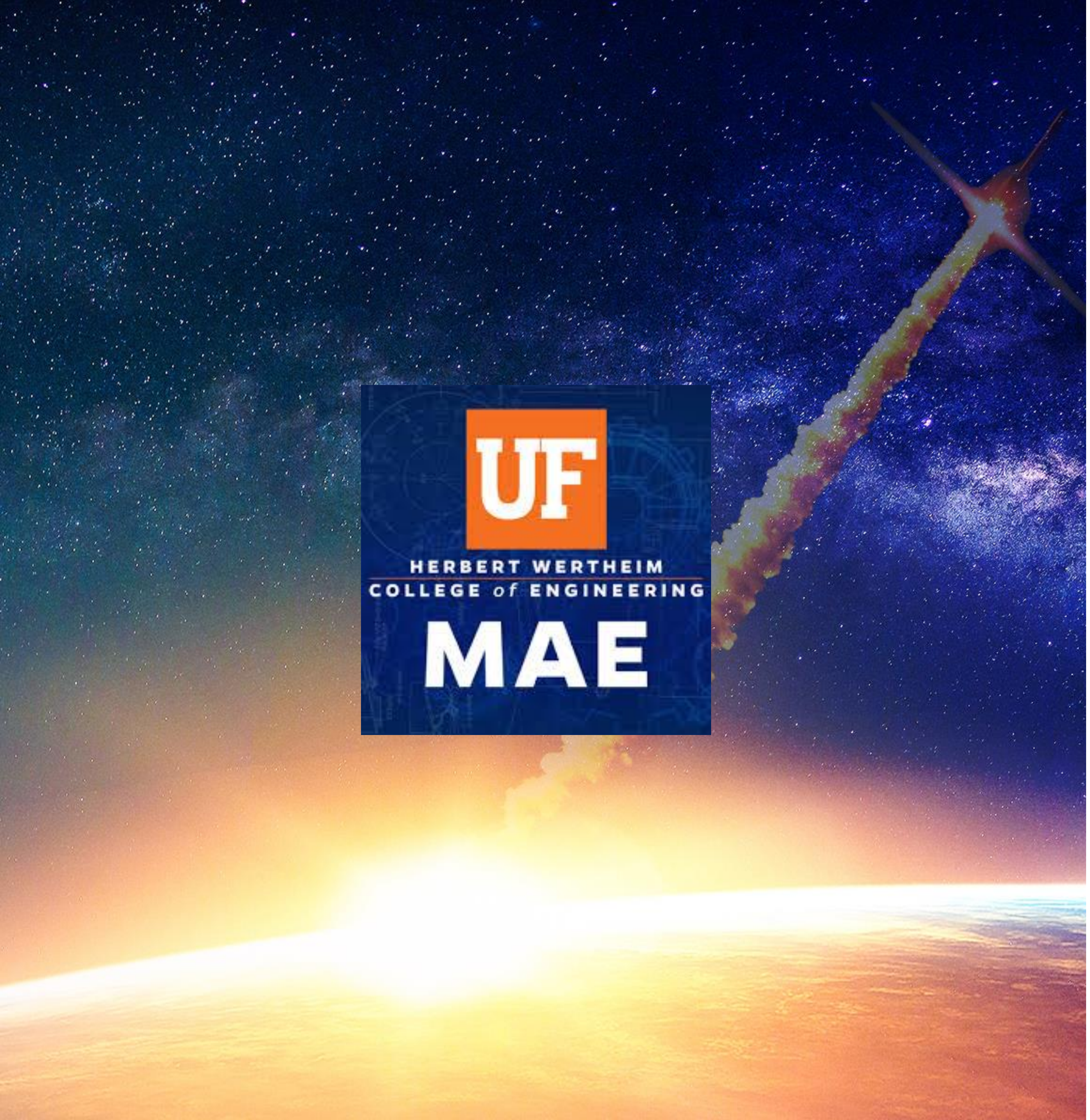


Autonomous Rendezvous and Docking of Spacecraft Using Hierarchical Deep Reinforcement Learning

by Anthony (Allen) Aborizk

UF
Herbert Wertheim
College of Engineering
Department of Mechanical
& Aerospace Engineering
UNIVERSITY of FLORIDA

Acknowledgement

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1842473. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Overview

- Autonomous Spacecraft
  - Autonomous Rendezvous, Proximity Operations, and Docking

- Reinforcement Learning
  - What and why
  - Drawbacks
  - Solutions

- Case Study, Simulation Dynamics and Controller Design
  - Case Study
  - Simulation dynamics
  - Controller Architecture

- Concluding Remarks

Autonomous Spacecraft

Key papers

- **NASA Space Technology Roadmap (Taxonomy)**
  - NASA
  - 2011, 2015 & 2020

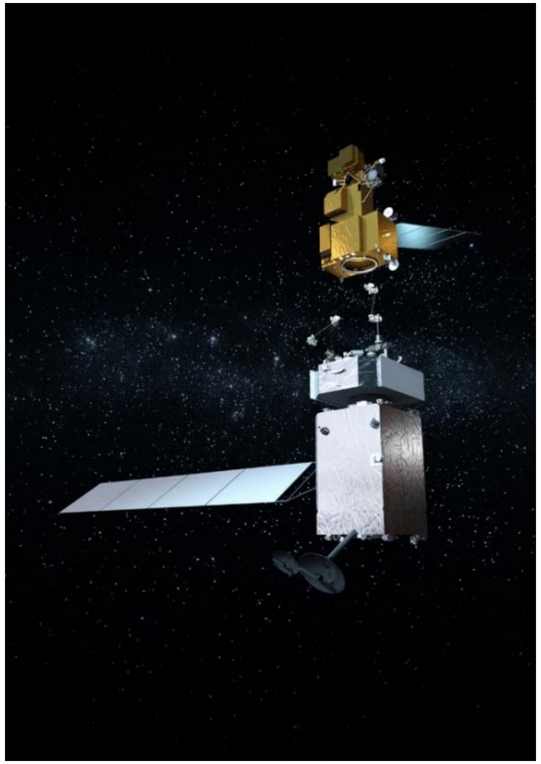
- **A Spacecraft Benchmark Problem for Hybrid Control and Estimation**
  - C. Jewison and R. Scott Erwin
  - *IEEE 55th Conference on Decision and Control*, Las Vegas

Motivation

- Advancements in spacecraft autonomy can impact many areas of research within the space community

- The technology can:
  - Expand the lifespan of space-based assets
    - OSAM-1
    - Orbital Express

  - Enable proximity and time sensitive maneuvers while outside the feasible reaches of ground communication capabilities
    - Mars landers
    - Communication delays



An artist’s rendering of OSAM-1 rendezvousing with a spacecraft in orbit. OSAM-1 will use robotic arms to capture the s/c before docking with it (NASA, 2020)

Advancements in spacecraft autonomy can impact many areas of research within the space community

Specific Areas of Interest

- The key challenges in autonomous rendezvous and docking (AR&D) include:
  - Rendezvous
  - Maneuvering
  - Hazard Avoidance

- Guidance and Control design must be:
  - Capable of trajectory planning
  - Robust
  - Capable of real-time implementation
  - Verifiable

Reinforcement Learning

Key papers

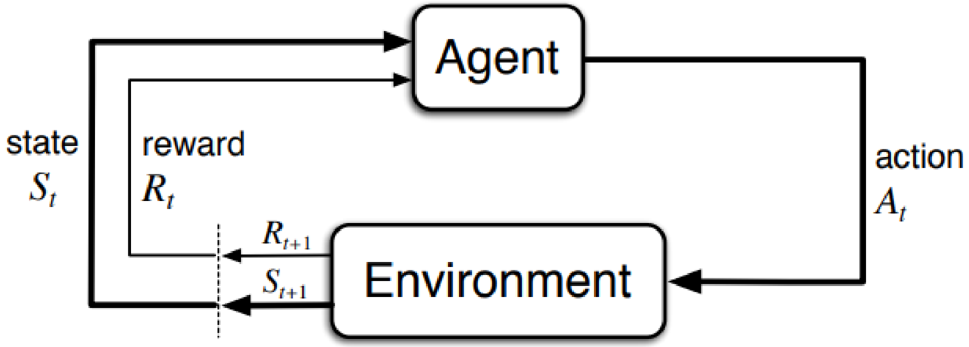
- **Reinforcement Learning: An Introduction**, **2nd ed.,**
  - R. S. Sutton and A. G. Barto
  - Cambridge, MA: The MITPress, 2018

- **Challenges of Reinforcement Learning**
  - Z. Ding and H. Dong
  - Deep Reinforcement Learning Fundamentals, Research and Applications, Singapore, Springer, 2020, pp. 249-272

- **Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models**
  - K. Chua, R. Calandra, R. McAllister and S. Levine
  - *32nd Conference on Neural Information Processing Systems*, Montreal, 2018

What is RL?

- Reinforcement learning (RL) is a sample-based machine learning algorithm
- It is based off the Markov decision process (MDP)
- It learns a controller framework by interacting with an environment (real or simulated)



A flowchart for a simple Markov decision process (Sutton, 2018)

$$p_\theta(\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T) = p(\boldsymbol{s}_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

$$\theta^* = argmax_\theta E_{\mathrm{T} \sim p_\theta(\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{a}_T)} \left[\sum_t r(\boldsymbol{s_t}, \boldsymbol{a_t}) \right]$$

Reinforcement Learning is a sample-based method trained by rewarding desired actions

Why RL?

- It has the potential for better generalizability than traditional optimal control algorithms
- It is adaptive and robust to changes in initial conditions
- It can learn via simulated experience and/or directly sampling from its environment
- This makes it useful for when dynamics are unknown

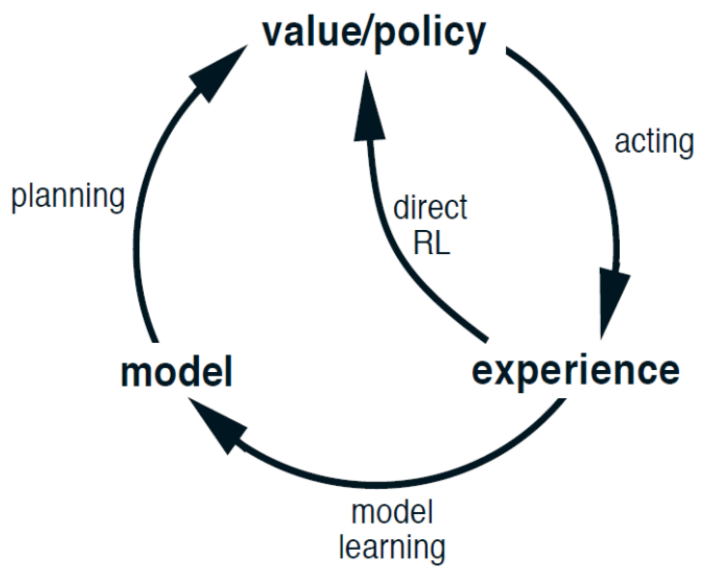
Reinforcement learning is a task focused, robust, adaptable and sample-based trajectory optimization algorithm

Drawbacks of RL

- Model-free reinforcement learning (MFRL) achieves higher asymptotic performance
  - It generalizes better
  - It learns entirely from interacting with the environment

- Model-based reinforcement learning (MBRL) is more sample efficient
  - It typically involves a planning step
  - It reiterates over past experiences

- Integrating neural networks and MBRL is difficult
- MBRL struggles with complex dynamics
- Neural networks overfit small datasets



(Sutton, 2018)

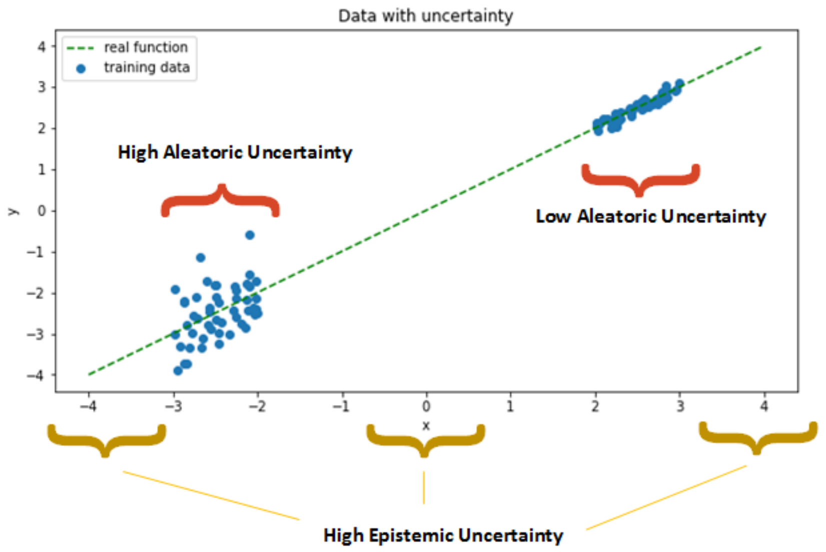
MBRL learns quickly and retains a model of known dynamics. MFRL achieves higher asymptotic performance.

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

- Probability ensemble trajectory sampling (PETS)

- Create uncertainty aware neural network allowing for better generalizability

- PETS can isolate two classes of uncertainty:
  - Epistemic: a subjective uncertainty resulting from gaps in the sample set
  - Further exploration into this can potentially direct exploration efforts
  - Aleatoric: the inherent system stochasticity generated from noise, unknown dynamics, etc.



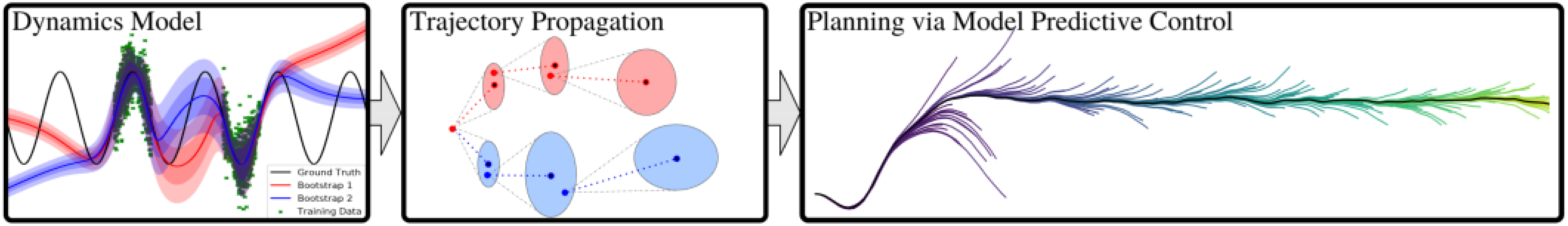
(Chua, 2018)

Epistemic = unexplored regions; Aleatoric = variance of data

How it works

- The probabilistic ensemble (PE) fits B bootstrapped models to the data
- Where bootstrapping involves resampling a limited dataset to approximate a global distribution
- Use trajectory sampling to propagate the model
  - This gives us an expected reward over some action sequence
- B models are aggregated in an Ensemble
  - variance between models will capture uncertainty in unexplored regions
- Implements MPC

(Chua, 2018)



PE's help quantify uncertainty

MPC

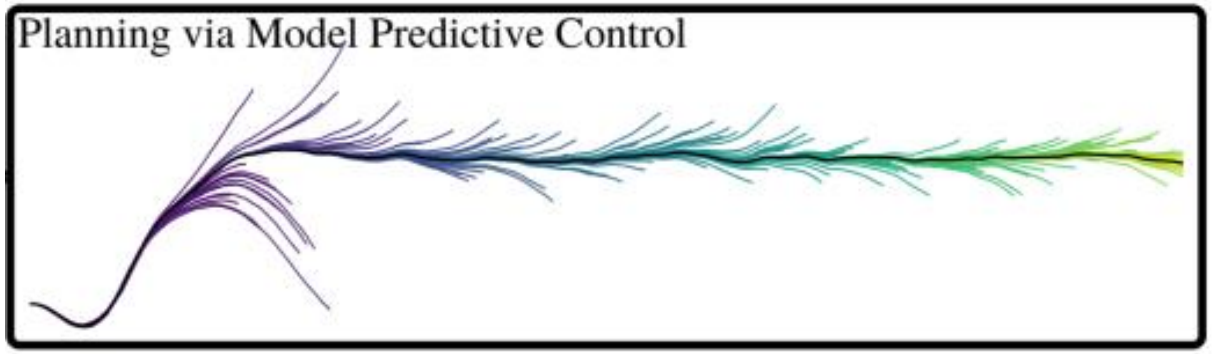
- Action Selection is performed using a Model Predictive Controller (MPC)
- Given the learned dynamics of the model –> choose some action to maximize reward [4]

$$\boldsymbol{A}^* = \arg\min_{\{\boldsymbol{A}^{(0)},\ldots,\boldsymbol{A}^{(K-1)}\}} \sum_{t'=t}^{t+H-1} (\hat{\boldsymbol{s}}_{t'}, \boldsymbol{a}_{t'})$$

$$\text{s.t. } \hat{\boldsymbol{s}}_{t'+1} = \hat{\boldsymbol{s}}_{t'} + f_\theta(\hat{s}_{t'}, \boldsymbol{a}_{t'})$$

$$A^{(k)} = \left(a_t^{(k)}, \ldots, a_{t+H-1}^{(k)}\right)$$

- Where action selection is determined using the cross-entropy method (CEM)
- At each time step, the model predicts H timesteps in the future



(Chua, 2018)

MPC performs multiple action sequences and chooses the best

Trajectory Sampling

- We want to predict the change between the current and next state

$$\hat{\Delta}_{t+1} = f_\theta(\bar{s}_t, \bar{a}_t)$$

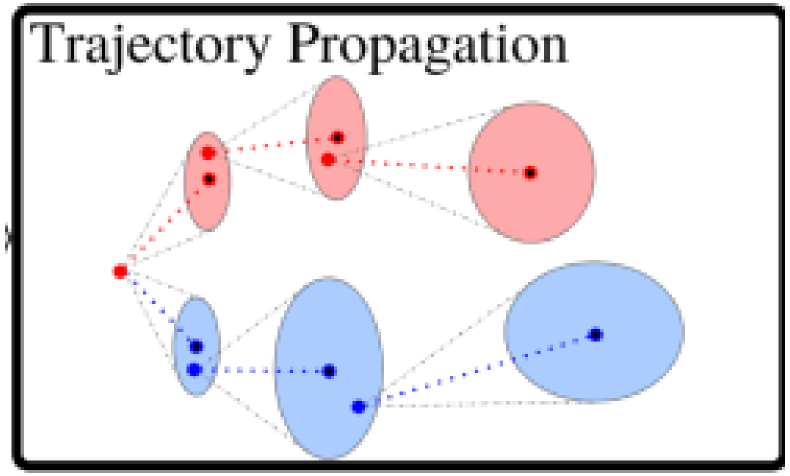
- Then the next predicted state is

$$\hat{s}_{t+1} = s_t + \hat{\Delta}_{t+1}$$

- The outputs for each model are the parameters for a probability distribution
- Clone the current state p times

$$s_{t=0}^{p} = s_0 \forall p$$
$$s_{t+1}^{p} \approx f_{\theta_{b(p,t)}}(s_t^p, a_t)$$

- B is the number of bootstrap models in an ensemble
- Given some action sequence in a particular state s_t each model in the probabilistic ensemble will determine the optimal trajectory.



(Chua, 2018)

Both bootstrapped models are used in trajectory sampling. Result is averaged.

HRL

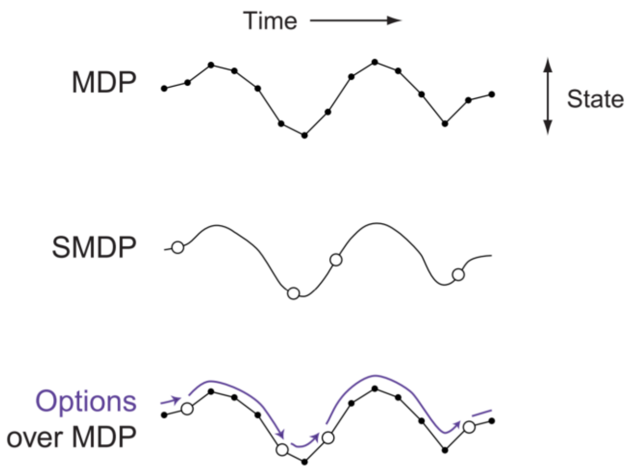
- Hierarchical RL (HRL) can help reduce the curse of dimensionality
- HRL follows a semi-Markov decision process (SMDP)
- It considers the time until the next transition
- Let $\bar{a}$ be the action executed by the hierarchical controller indicating which sub policy to follow

$$\forall s_{t+\tau} \in S_{end}: \Pr(s_{t+\tau}, \tau | s_t, \bar{a}) = \Sigma_{s_{t+1:t+\tau-1} \notin S_{end}} \prod_{i=1}^{\tau-1} \Pr(S_{t+i} | S_{t+i-1}, \bar{\pi}(S_{t+i-1}))$$

$$R(s_t, \bar{a}, s_{t+\tau}, \tau)$$

$$= R\big(s_t, \bar{\pi}(s_t)\big) + \gamma \Sigma_{s_{t+1}} \Pr(S_{t+i} | S_{t+i-1}, \bar{\pi}(S_{t+i-1}))[R\big(s_{t+1}, \bar{\pi}(s_{t+1})\big) + \ \dots$$

$$+ \gamma \Sigma_{s_{t+\tau}} \Pr(S_{t+\tau} | S_{t+\tau-1}, \bar{\pi}(S_{t+\tau-1}))\big[R\big(s_{t+\tau}, \bar{\pi}(s_{t+\tau})\big)\big] \dots]$$



If we consider a two-layer hierarchy, let the SMDP represent the time step of the highest level and the MDP represent that of the lower-level (Sutton, 1998)

HRL reduces the complexity of a controller by breaking it up into subtasks

Case Study, Simulation Dynamics and Controller Design

Key Papers

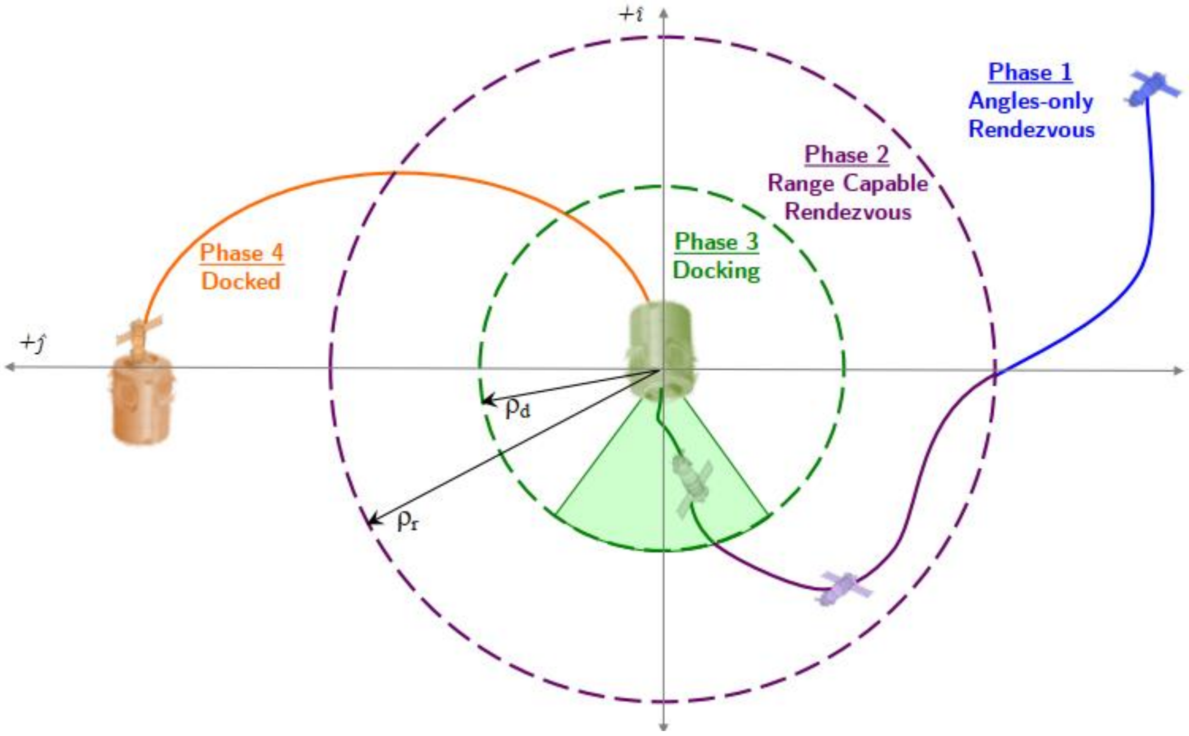
- **Autonomous Six-Degree-of-Freedom Spacecraft Docking Maneuvers via Reinforcement Learning**
  - C. Oestreich, R. Linares and R. Gondhalekar
  - AAS/AIAA, Lake Tahoe, 2020

- **Adaptive Continuous Control of Spacecraft Attitude Using Deep Reinforcement Learning**
  - J. Elkins, R. Sood and C. Rumpf
  - *AAS/AIAA Astrodynamics Specialist Conference*, Lake Tahoe, 2020.

- **Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning**
  - A. Harris, T. Teil and H. Schaub
  - 29th AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, HI, 2019.

Case Study: On Orbit Assembly

- On orbit assembly
- A tug spacecraft performs ARPOD maneuvers to collect pre-assembled components
- Four phase ARPOD problem
  - Phase 1 – 10km, angles only measurements
  - Phase 2 – LYDAR, range capable
  - Phase 3 – Proximity and docking operations
  - Phase 4 – Relocation under new dynamics
- The case study accounts for:
  - Gaussian white noise
  - 2DOF & 3DOF cases
  - Various thruster designs



(Jewison, 2016)

Simulated Translational Dynamics

- 2DOF or 3DOF docking environment in local vertical local horizontal (LVLH) reference frame
- translational motion in training is governed by Clohessy-Wiltshire equations

$$\ddot{x} - 2n\dot{y} - 3n^2x = \frac{F_x}{m}$$

$$\ddot{y} + 2n\dot{x} = \frac{F_y}{m}$$

$$\ddot{z} + n^2z = \frac{F_z}{m}$$

$$\dot{x} = A\boldsymbol{x} + B\boldsymbol{u} = CWH(\boldsymbol{x}, \boldsymbol{u}, n, m)$$

- Where position and acceleration can be determined using fourth order Runga-Kutta
- Considering docking port position

$$\boldsymbol{r}_p = \boldsymbol{r} + \boldsymbol{R}(\boldsymbol{q})\boldsymbol{r}_c - \boldsymbol{r_t}$$

$$\boldsymbol{v}_p = \boldsymbol{v} + (\boldsymbol{\omega} \times \boldsymbol{R}(\boldsymbol{q})\boldsymbol{r}_c)$$

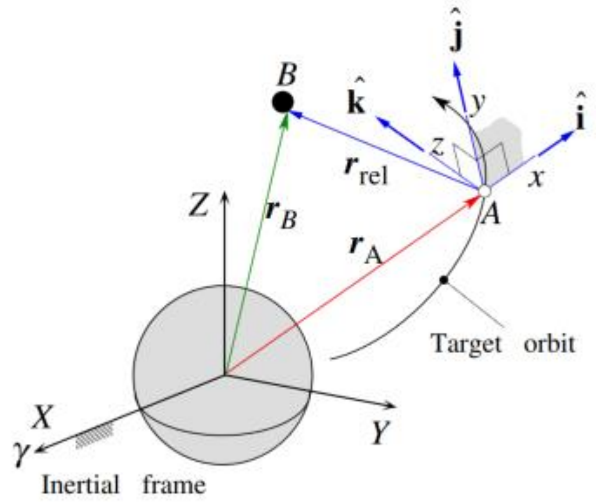


Illustration of LVLH from Comparing Run Time Assurance Approaches for Safe Spacecraft Docking by K. Dunlap (Curtis, 2014)

Relative dynamics are used to guide the chaser to the target

Simulated Rotational Dynamics

- Euler's rotation equations

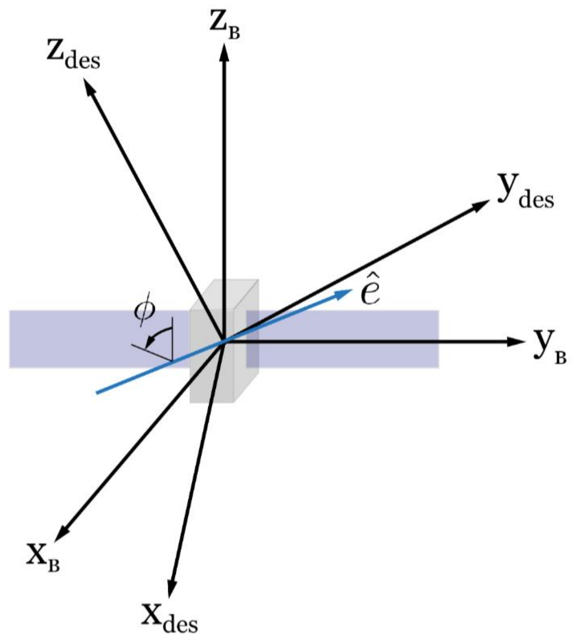
$$\boldsymbol{M} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\times}\boldsymbol{I}\boldsymbol{\omega}$$

- Where the body fixed angular velocity at the next timestep can be determined using a fourth order Runga-Kutta scheme
- The error quaternion

$$\boldsymbol{q_e} = \begin{bmatrix} \hat{\boldsymbol{e}} sin\left(\frac{\phi}{2}\right) \\ cos\left(\frac{\phi}{2}\right) \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} \\ q_s \end{bmatrix}$$

$$\dot{\boldsymbol{q}}_{\boldsymbol{e}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q_e}$$

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}^{\times} & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{T} & \boldsymbol{0} \end{bmatrix}$$

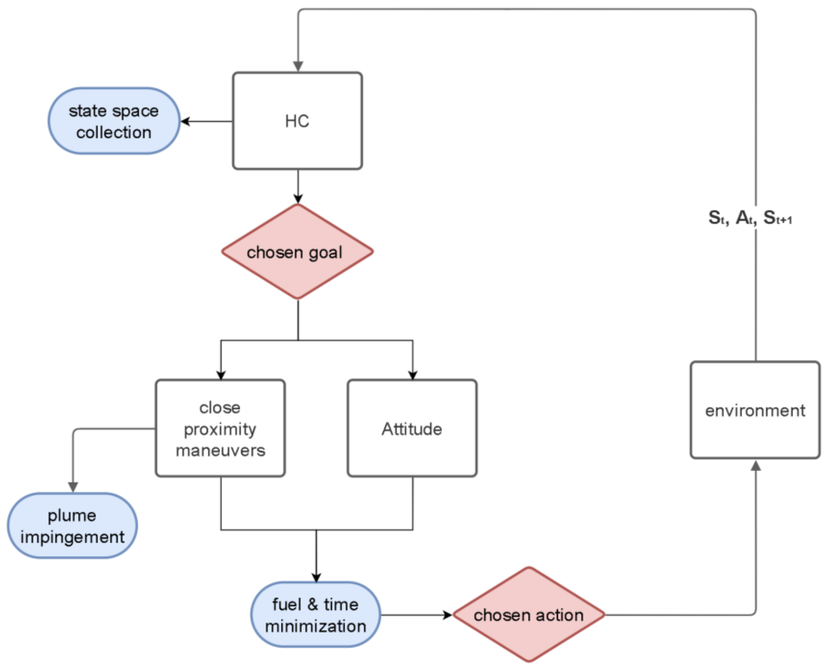


Schematic of orientation components as defined by. Where $\hat{e}$ is the axis of rotation, and ϕ is the angle of rotation. (Elkins, 2020)

Rotational dynamics follow Euler's equations

Controller

- The hierarchical controller (HC) will receive input
$$S_t = [\boldsymbol{r_t}, \boldsymbol{\dot{r}_t}, \boldsymbol{q_t}, \boldsymbol{\omega}]$$
- It will determine a goal state and distribute state space components to the necessary sub-controllers
- Both sub-controllers will receive the desired state from the HC and reduce the error
- Sub-controllers will produce an action
$$A_t = [\boldsymbol{F_t}, \boldsymbol{M_t}]$$
- where the subscript “t” indicates the timestep
- Each sub-controller will implement PETS

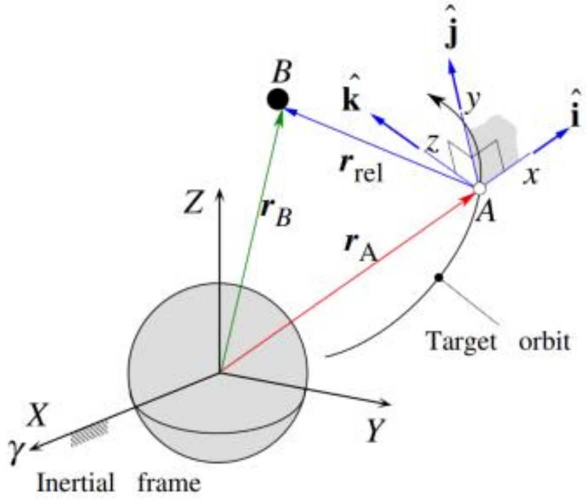


This is an illustration of the initial controller design. It will implement MBRL using the PETS-CEM algorithm in a hierarchical structure. The HC will determine the error values that the two sub-policies must minimize through action selection.

QUESTIONS?

Preliminary Findings

- I used a modified version of a 2D spacecraft docking environment created by a team of interns at AFRL
- The code uses local vertical local horizontal (LVLH) reference frame
- Environment dynamics are calculated using Clohessy-Wiltshire equations
  - The origin of the reference frame is located at the COM of the chief spacecraft
  - $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$
  - Where $\boldsymbol{x} = [x, y, \dot{x}, \dot{y}]$ is the state vector of position and velocity
  - $\boldsymbol{u} = [F_x, F_y]$ **,** where F is the applied thrust of the deputy
- A successful dock is achieved when the deputy is <= 0.1 m with a speed <= 0.2 m/s



[47]

The simulation environment focuses on 2D relative motion

Preliminary Findings

- Ran roughly the same algorithm on a simulated docking environment
- Did not predict distribution for next state
- Used random action selection in MPC instead of CEM

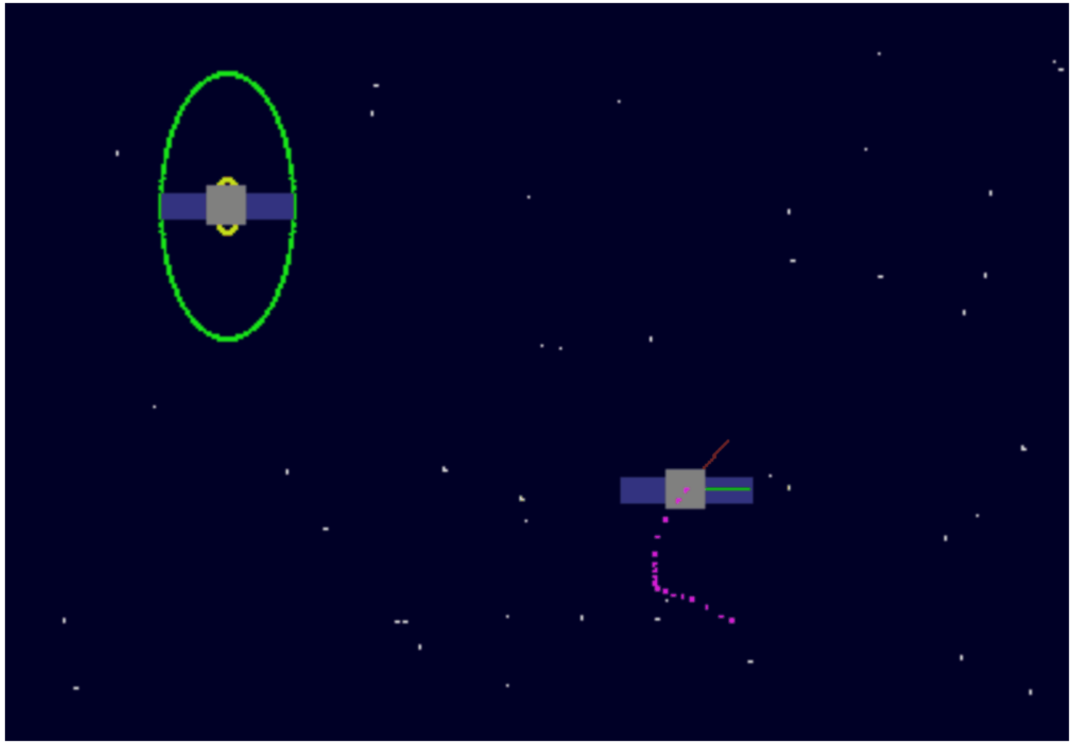


Figure 3-2. Simulation environment of 3DOF close proximity maneuvers provided by the AFRL Scholars Program [47]. The pink dotted line tracks the random path of the chaser satellite. The white dots emitting from the chaser indicate the applied impulse thrust.

Preliminary Findings

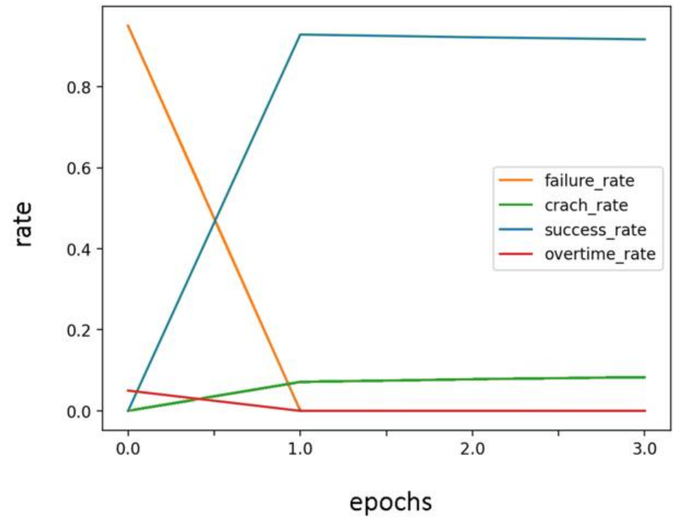
Task	Description	Reward
Successful docking	Position <= 0.1m Velocity <= 0.2m/s	+1
Traveling out of frame	The environment expands 1500m from COM of chief in x, y, -x, -y	-1
Running out of time	Max time = 4000s	-1
Running out of fuel	Control input > 2500N	-1
Crashing	Position <= 0.1m Velocity > 0.2m/s	-0.001
Get closer to chief	Small negative dense reward	$\frac{\left(-1+\left\|\|r_{old}\right\|\|-\left\|\|r\right\|\|\right)}{2000}$
Travel at a safe speed	Do not exceed max velocity constraint	$-0.0035 * \left\|\left\|\|v\right\|\|-\left\|\|v_{\max}\right\|\|\right\|$
Move	Ensure velocity is greater than min requirement	$-0.0075 * \left\|\left\|\|v\right\|\|-\left\|\|v_{\min}\right\|\|\right\|$

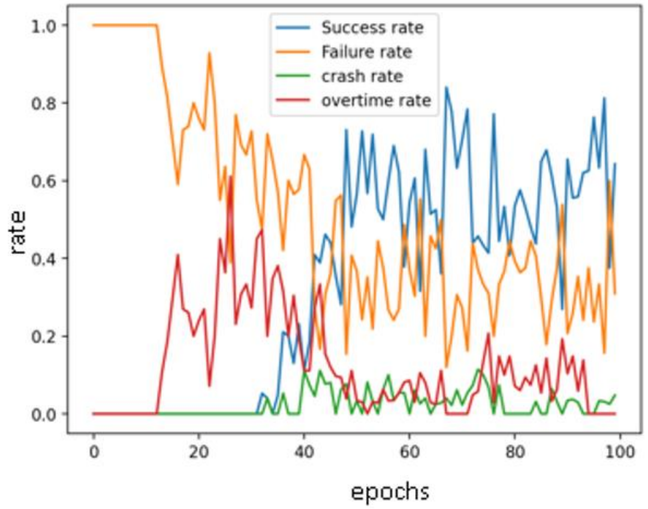
[47]

Preliminary Findings

Success rate	Avg # of successful docks
Failure rate	Avg # of times spacecraft runs out of frame
Crash rate	Avg # of times spacecraft crashes
Overtime rate	Avg # of times spacecraft runs out of time/fuel
Destroy rate	Avg # of times spacecraft is hit with 3+ items

- MBRL reached a higher asymptotic performance than PPO
- MBRL = 88% success rate after 3 episodes of training
- PPO = 81% success rate after 65 episodes





Preliminary Findings

- It is possible to bridge the gap between MB and MF algorithms
- Epistemic and aleatoric uncertainty paly a role in the discrepancy between the two algorithms
- While planning algorithms have many benefits, they are not always real-time implementable due to computational cost
- Policy optimization integration into this method may prove useful

Biggest issue with probabilistic MBRLs is real time implementability

Related works to handful of trials

- Most MBRLs use Gaussian Process to model dynamics
  - Good for low dimensional models
  - Assume smoothness
- NN are also useful
  - Constant time inference
  - Tractable training in large data regime
  - Can represent more complex functions
  - Non-smooth
- NN can be improved via ensemble, dropout, alpha divergence

NN were chosen because they are non-smooth and good for high dimensional models

Uncertainty aware neural network dynamics models

- A Probabilistic NN is one that outputs the parameters for a probability distribution
- Chua uses a negative log prediction probability as the loss function

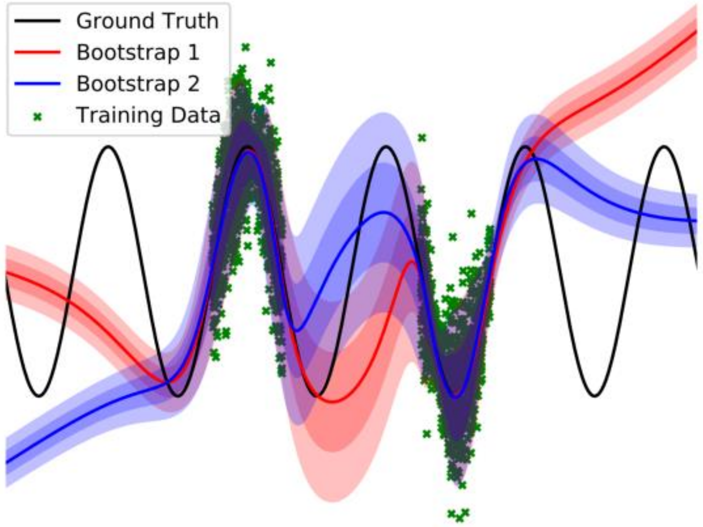
$$loss_p = -\Sigma_{n=1}^{N} \log \widetilde{f}_\theta(s_{n+1}|s_n, a_n)$$

- For Gaussian probability

$$loss_{Gauss}(\theta) = \Sigma_{n=1}^{N}[\mu_\theta(s_n, a_n) - s_{n+1}]^T \Sigma_\theta^{-1}(s_n, a_n)[\mu_\theta(s_n, a_n) - s_{n+1}] + \log\det\Sigma_\theta(s_n, a_n)$$

- Ensemble is defined as

$$\widetilde{f}_\theta = \frac{1}{B}\Sigma_{b=1}^{B}\widetilde{f}_{\theta\,b}$$

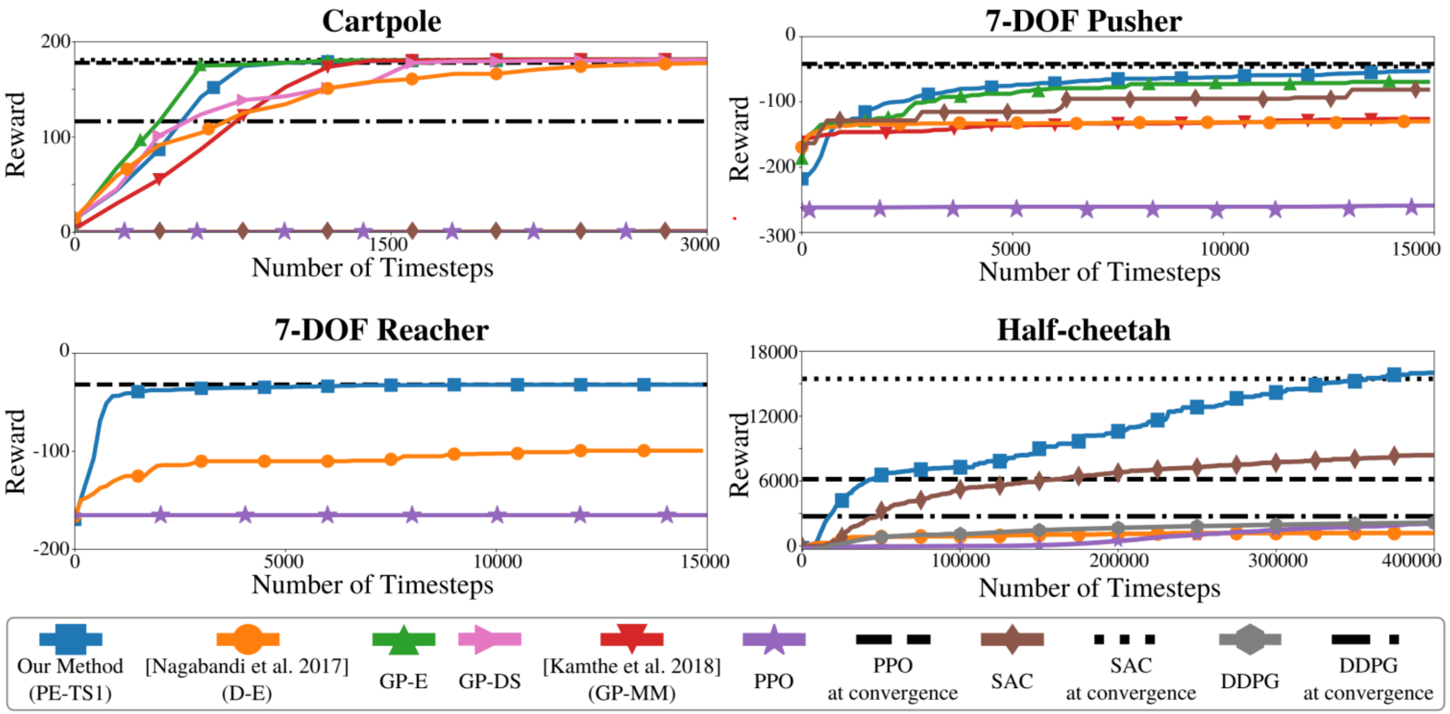


[28]

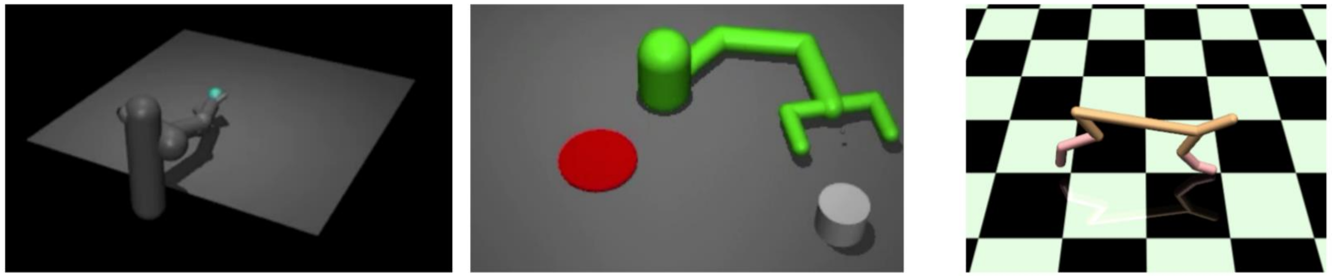
Chua assumes a gaussian distribution for his Δ prdictions

Results from a handful of trials

- PETS was compared to multiple MB and MF algorithms
- Outperformed all algorithms in all benchmarking experiments except Cartpole
- Faster learning and higher asymptotic performance
- Cartpole is an extremely simple dynamic model



[28]



(a) 7-DoF Reacher (b) Pusher (c) Half Cheetah

[19]

Overview

- Spacecraft Autonomy
- Autonomous Rendezvous & Docking
- Reinforcement Learning
  - What and why
  - Drawbacks
  - Solutions
  - Other considerations
    - Adaptability,   proposing a marriage of t RL NN and Controls to leverage advantages of both.
  - How HRL satisfies constraints of AR&D
- Detail controller design to explore and improve
  - PETS algorithm because it quantifies uncertainty for complex algorithms AND leverages classical control techniques.
  - Need improvement in uncertainty classification, trajectory sampling , potentially controller choice
- Propose implementation of PETS on 2D docking and orientation separately and then compare with performance in HRL structure

Other Areas of Consideration

- Sim-to-real gap
  - Training in simulation can save money
  - This typically results in a discrepancy between real and simulated inputs
  - This can be addressed by adding noise to input
  - or by creating an algorithm to transform input to a canonical basis

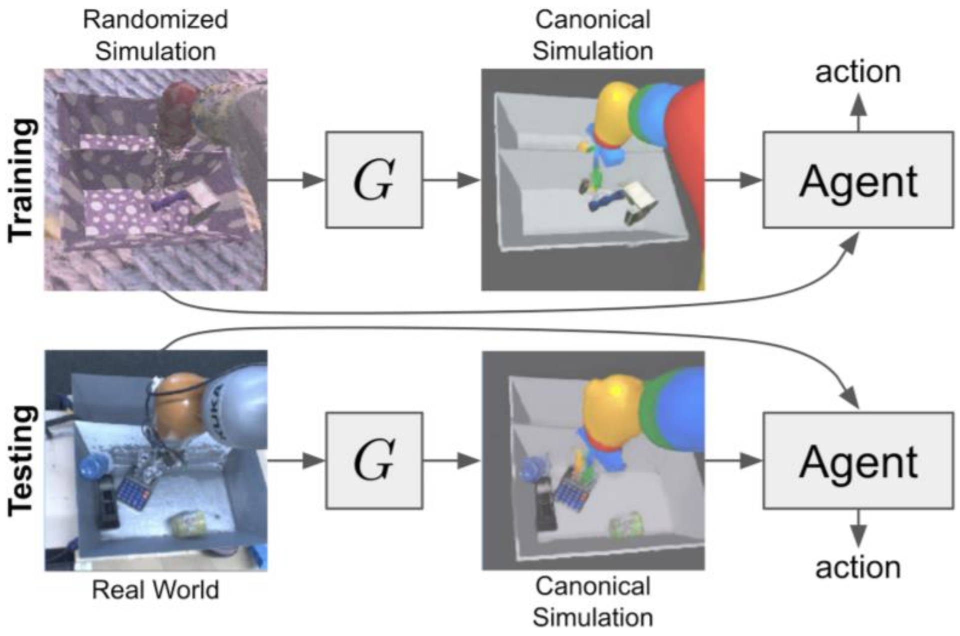
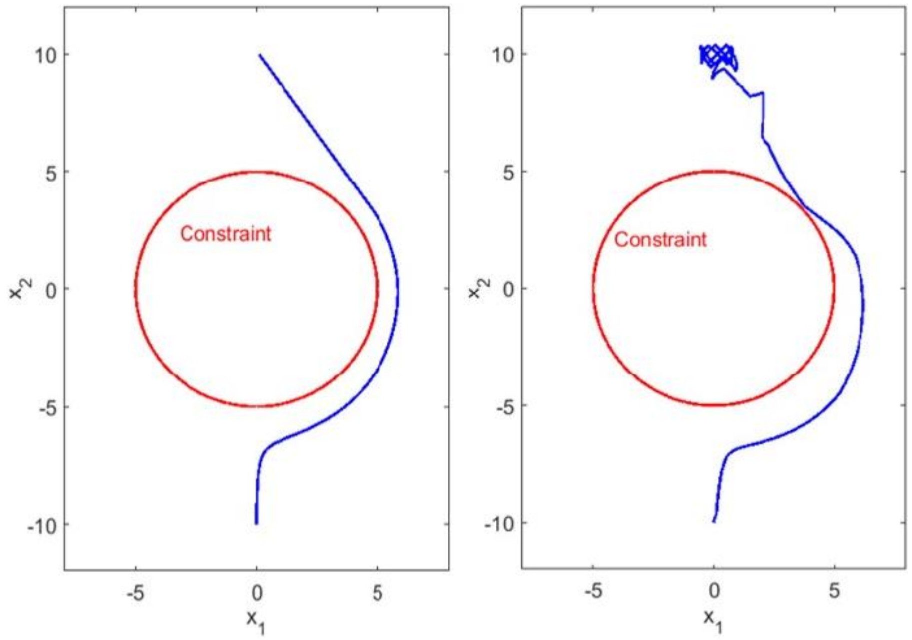


Illustration of input transformation from Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks

Transformation algorithms seem to produce better results than input randomization

Other Areas of Consideration

- Safety
  - guarantee safety without restricting exploration
  - Barrier functions offer a means of restricting the state space to guarantee safety
  - Exploration is still impeded, but it errs on the side of caution



Control Barrier Functions for Constrained Control of Linear Systems with Input Delay by M. Jankovic

Barrier functions can guarantee safety, but restrict exploration

Controller

- We will continue to look at the change in state as before

$$\hat{\Delta}_{t+1} = f_\theta(\bar{s}_t, \bar{a}_t)$$

- Previously listed methods do not account for free flow and uncertain dynamics

$$\hat{\Delta}_{t+1} = f_\theta(\bar{s}_t, \bar{a}_t) + \ g_\theta(\bar{s}_t, \bar{a}_t) + u_\theta(\bar{s}_t, \bar{a}_t)$$

- Thus, the cost function for the learned dynamics model becomes

$$L(\theta) = \sum_{(\bar{s}, \bar{a}, \bar{s}_{t+1}) \in D} \left|\left|(\bar{s}_{t+1} - \bar{s}_t) - f_\theta(\bar{s}_t, \bar{a}_t) + \ g_\theta(\bar{s}_t, \bar{a}_t) + u_\theta(\bar{s}_t, \bar{a}_t)\right|\right|_2^2$$

- Additionally, CEM will be used to increase computation time of the MPC
- Some exploration into policy-based methods may be beneficial to amortize the cost of training

The Dynamics model should account for free flow dynamics