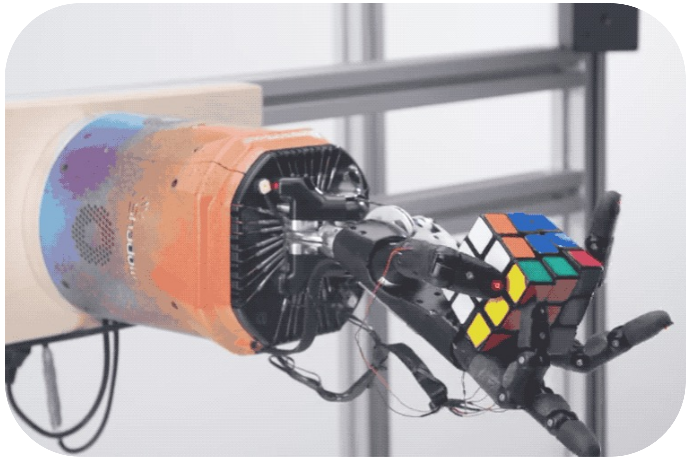
Safety Guarantees for Uncertain Systems in Interactive Settings

SAFE ROBOTICS LABORATORY

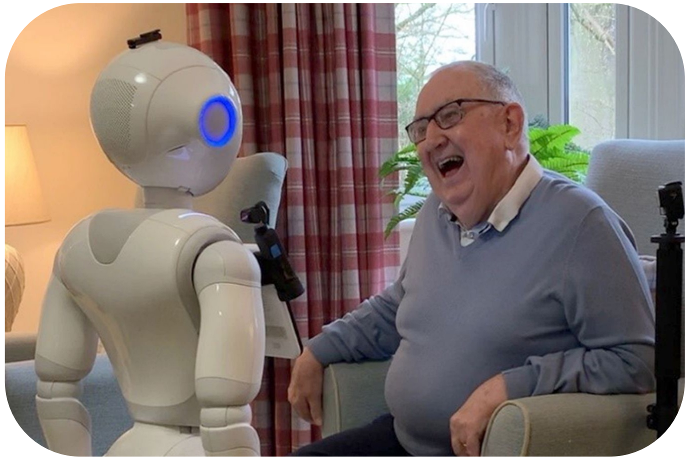
Kai-Chieh Hsu

April 29, 2021

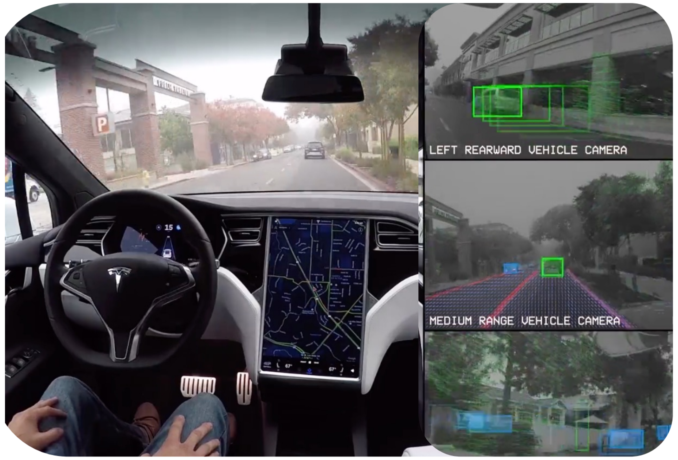
SAFE ROBOTICS LABORATORY



OpenAI: Dactyl



Softbank robotics / RobotLAB: Pepper

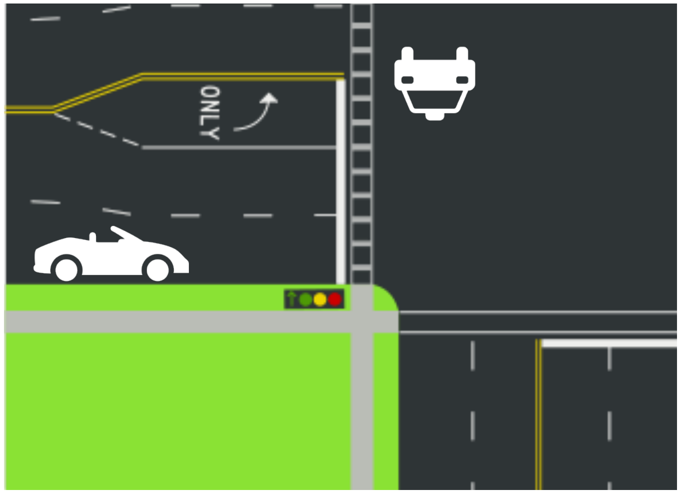


Tesla: self-driving car

NY Times: Boeing 737

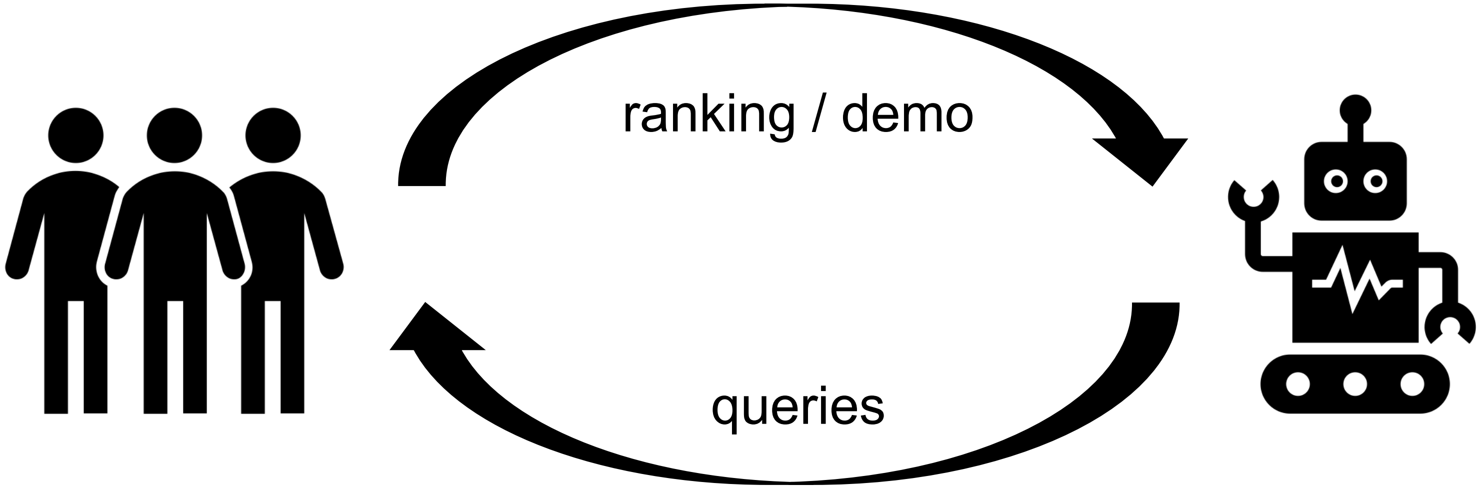
Uber car accident

OpenAI: Reward Hacking



How to provide **safety guarantees** for **uncertain** systems?

How to **loop humans in** to better understand their preference?



Outline

- Introduction
- Supervisory control in high-dimensional systems
- Inverse specification
- Conclusion and future works

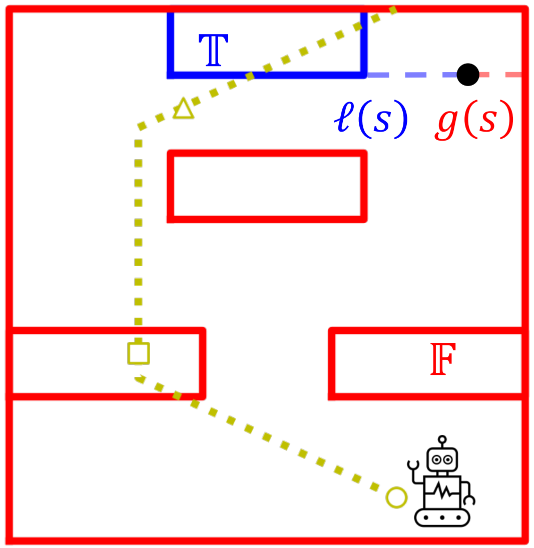
Supervisory Control: Shielding

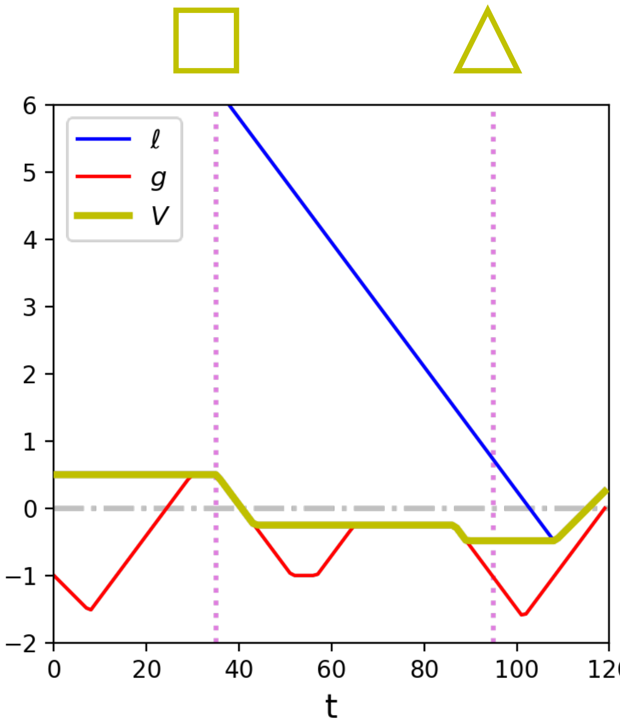
An approximate method to provide **fallback control** to high-dimensional systems

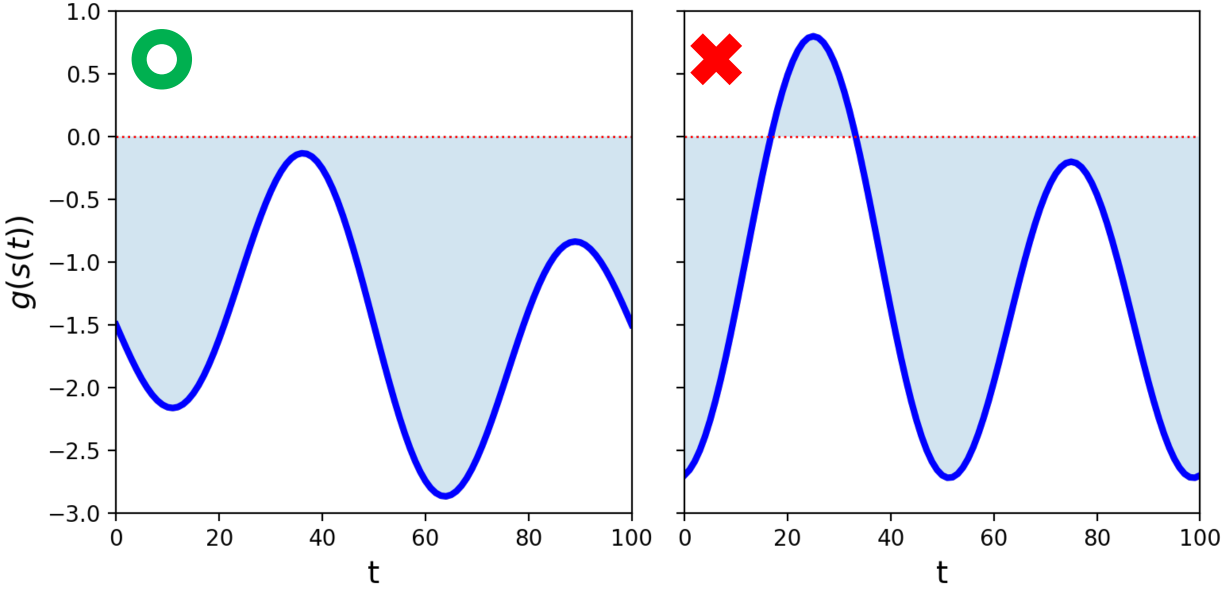
Keep **safe** away from forbidden states but
maintain **liveness** to reach target states

$s \in \mathbb{T} \leftrightarrow \ell(s) \leq 0$: reachability

$s \in \mathbb{F} \leftrightarrow g(s) > 0$: safety







Dynamics: $\dot{s} = f(s,u)$

Goal: find $\boldsymbol{u}$ such that

$\exists$ t, $\boldsymbol{s}(\mathrm{t}) \in \mathbb{T} \wedge \forall \tau \in [0,t], \boldsymbol{s}(\tau) \notin \mathbb{F}$

Reach-Avoid (RA)

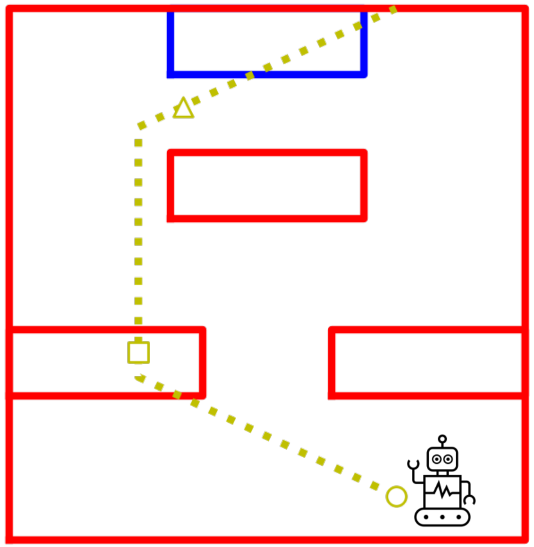
$$L(\boldsymbol{s}^{\mathbf{u}}) = \min_{t\in[0,T]} \max\{\ell(\boldsymbol{s}(t)), \max_{\tau\in[0,t]} g(\boldsymbol{s}(\tau))\}$$

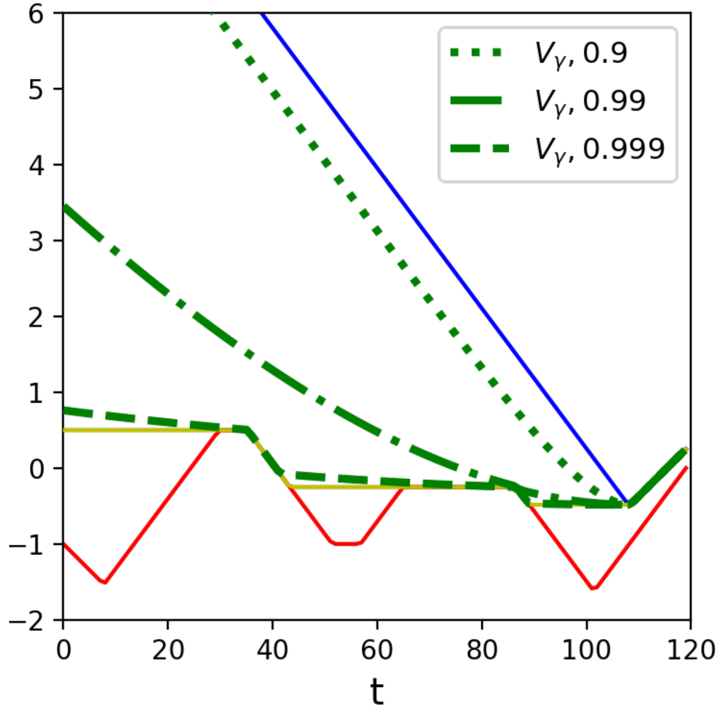
$$\mathrm{V}(\mathrm{s}) = \min_{\mathbf{u}} L(\boldsymbol{s}^{\mathbf{u}})$$
$$= \max\{ \mathrm{g}(\mathrm{s}), \min\{\ell(s), \min_{u} \mathrm{V}(s + f(s,u)\Delta t)\} \}$$

Sum of costs, Lagrange

$$L(\boldsymbol{s}^{\mathbf{u}}) = \Sigma_{t=0}^{T} c(\boldsymbol{s}(t))$$

$$\mathrm{V}(s) = \min_{\mathbf{u}} L(\boldsymbol{s}^{\mathbf{u}})$$
$$= \min_{u} c(s,u) + \mathrm{V}(s + f(s,u)\Delta t)$$





Goal: find $\boldsymbol{u}$ such that

$$\exists\, t,\ \ \boldsymbol{s}(t) \in \mathbb{T} \wedge \forall \tau \in [0,t], \boldsymbol{s}(\tau) \notin \mathbb{F}$$

$$s_+^u := s + f(s,u)\Delta t$$

Reach-Avoid Bellman Equation:

$$V(s) = \max\left\{g(s), \min\left\{\ell(s), \min_u V(s_+^u)\right\}\right\}$$

Curse of dimensionality → deep RL

Lagrange or SUM

$$L(\xi) = \Sigma_t\, \gamma^t c(\xi(t))$$

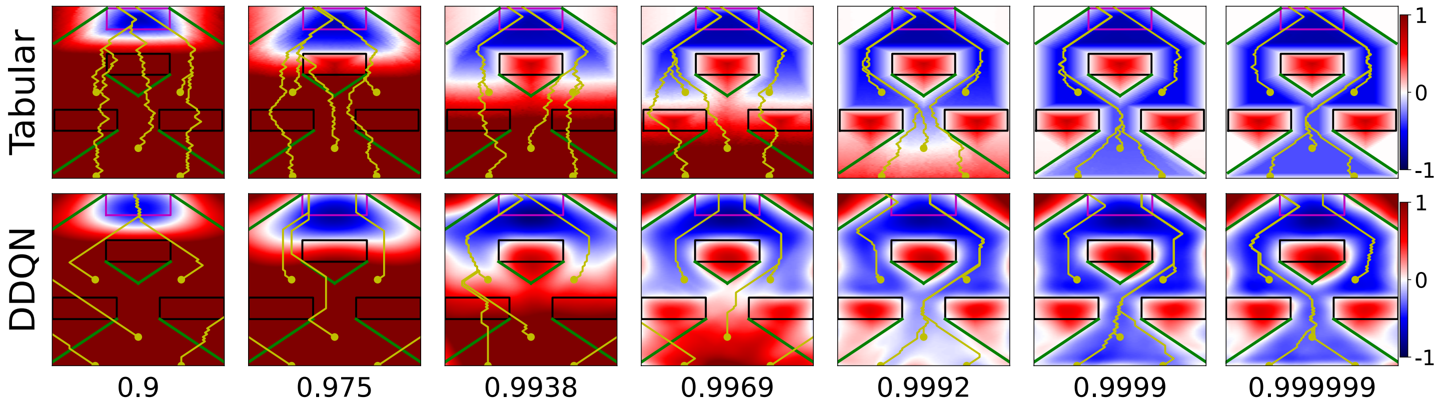
$$V(s) = \min_u \gamma \left(c(s,u) + V(s_+^u)\right) + (1-\gamma)c(s,u)$$

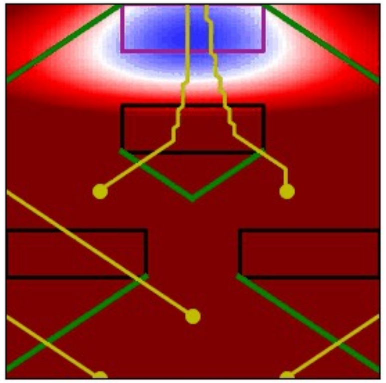
Discounted Reach-Avoid Bellman Equation:

$$V_\gamma(s) = \gamma \max\left\{g(s), \min\left\{\ell(s), \min_u V(s_+^u)\right\}\right\} + (1-\gamma)\max\{g(s), \ell(s)\}$$

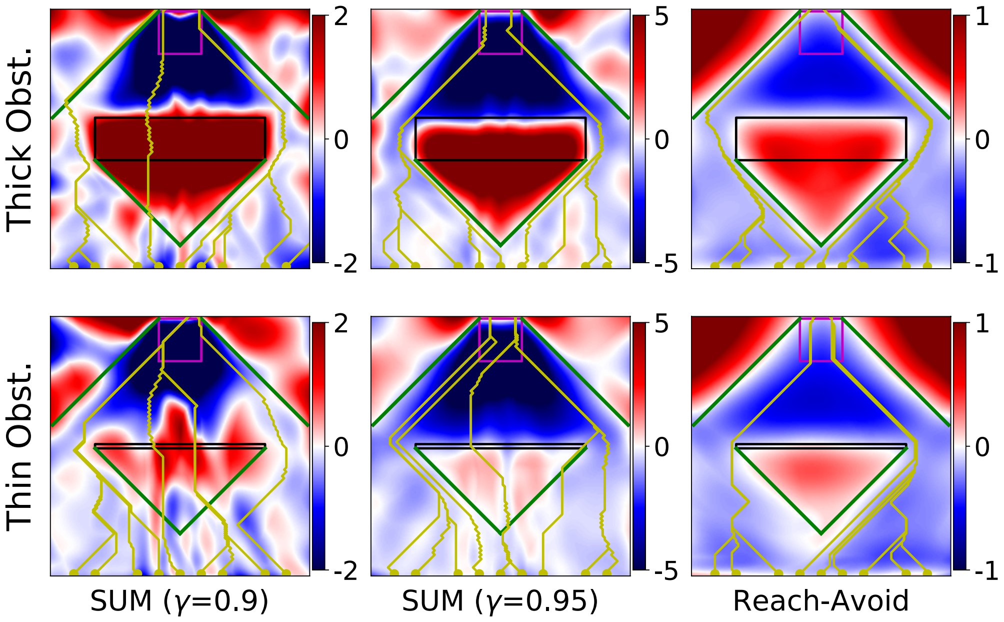
Double Deep Q-Network

Conservativeness of Discounted Reach-Avoid Set

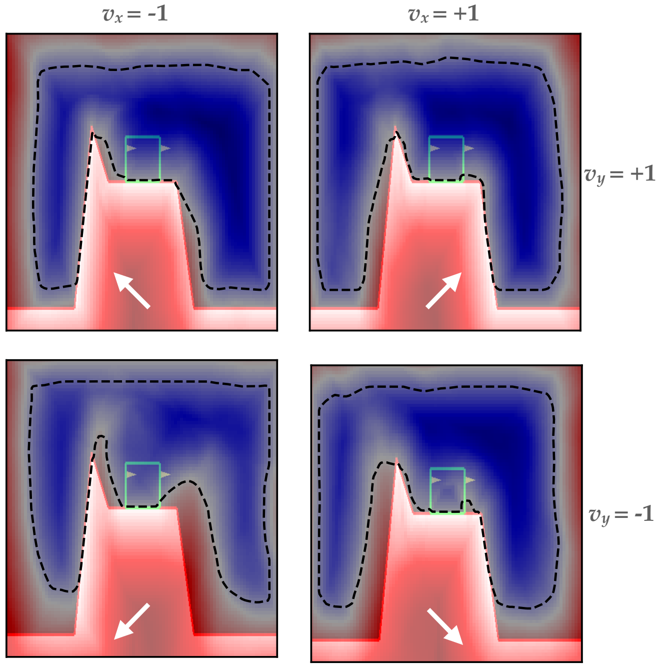


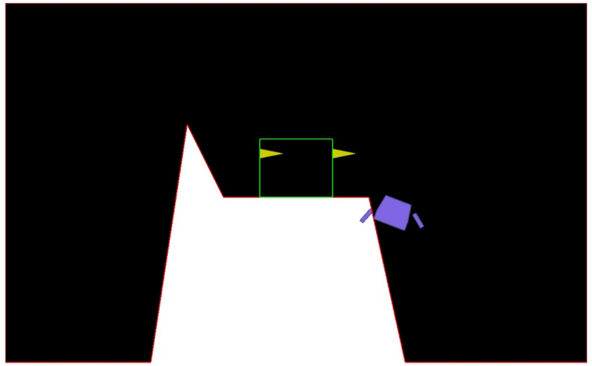


Reach-Avoid vs. Lagrange



Lunar Lander (6D: $x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}$)

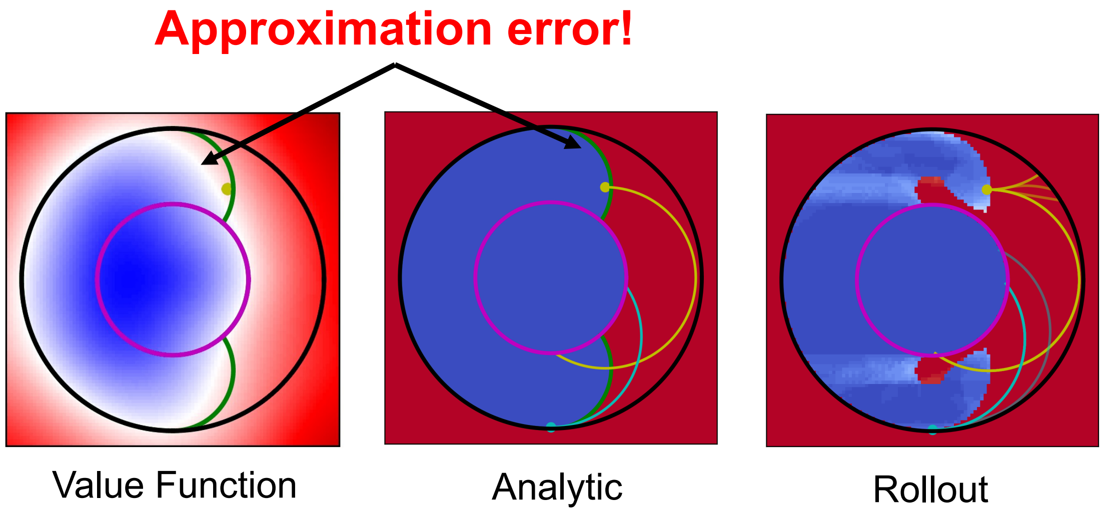




Untrusted Oracles

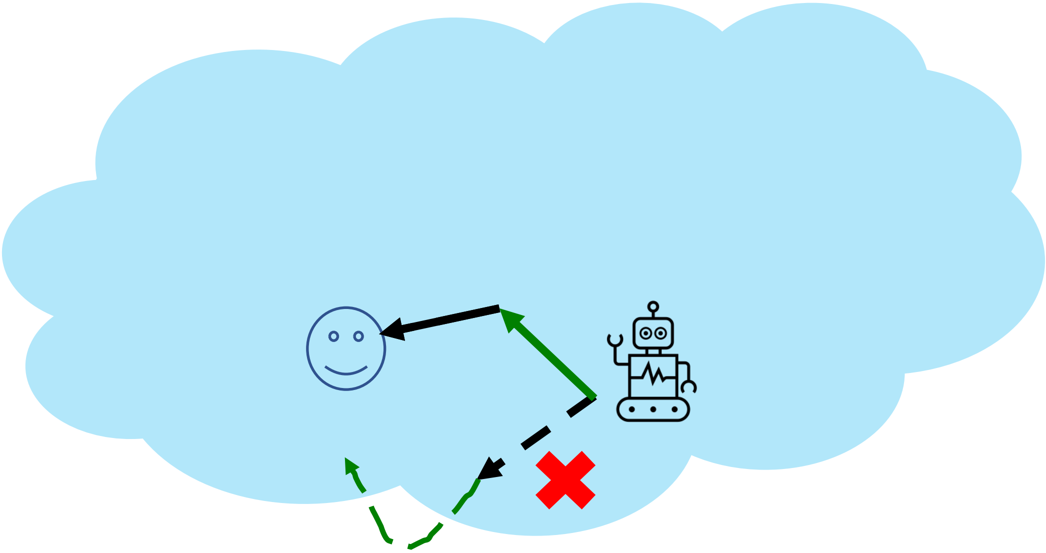
Dubins Car

- State: x-pos, y-pos, heading angle
- Actions: straight, left turn and right turn



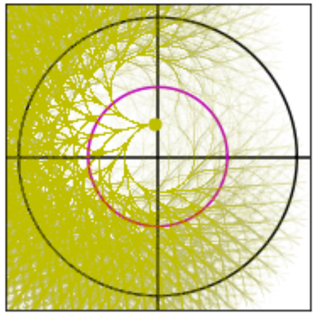
Shielding scheme:

→ obtain a candidate action
→ simulate a short trajectory forward
→ if not, reach-avoid action
→ if remaining in the reach-avoid set, we execute the candidate action

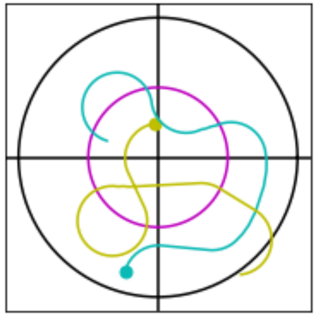


Future Works

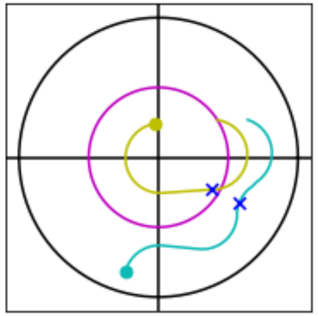
- Learned policy degrades in the no-discount limit
  - LL's actions: Right, Left, Main thruster on, Thruster off
  - Actor-Critic algorithms, e.g., soft actor-critic
- Zero-sum differential game
  - $V(s) = \max\left\{g(s), \min\left\{\ell(s), \min_{u} \max_{d} V(s_{+}^{u,d})\right\}\right\}$
  - Principle of iterative adversarial improvements



Exhaustive Search



Rollout: -0.145

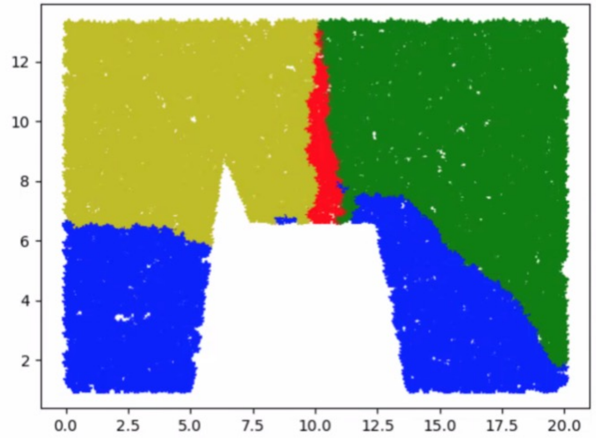


Exhaustive: 0.13

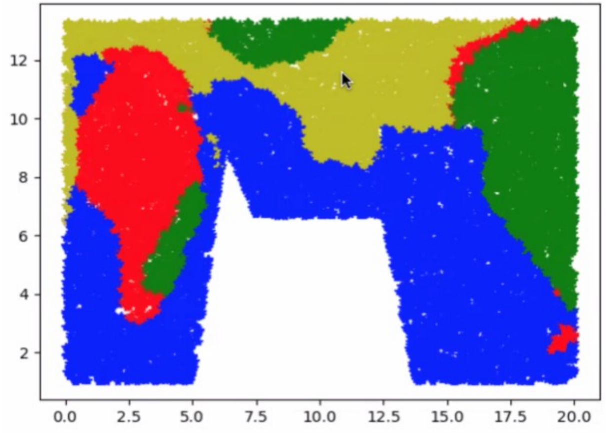
Attacker
Defender

$\theta = \dot{x} = \dot{y} = \dot{\theta} = 0$

$\gamma = 0.99$



$\gamma = 0.9999$



SAFE ROBOTICS LABORATORY

Future Works

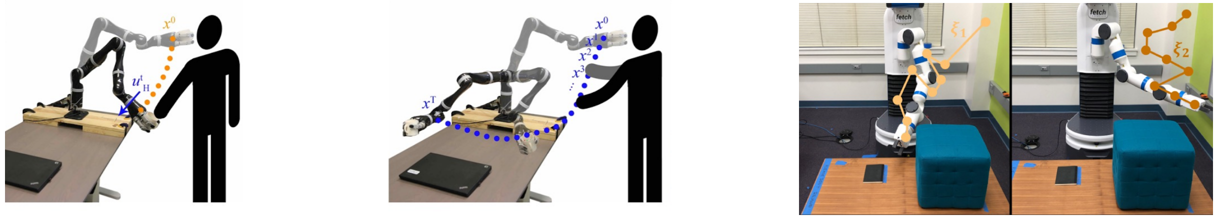
- Unknown Environment Exploration: PAC-Bayes Control framework
  - Assumption:
    - An underlying distribution D of environments
    - We have a set of N sample environments, S
  - PAC-Bayes Bound: with probability $1-\delta$,

$$C_D(P) \leq C_S(P) + Reg(P, P_0)$$

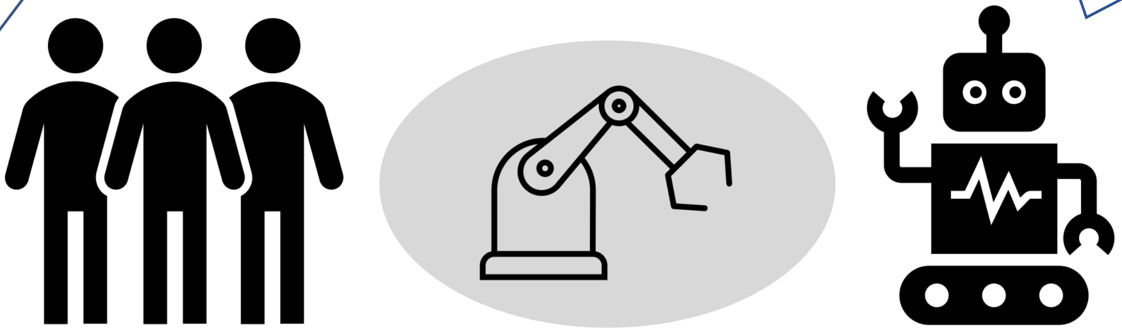
$$Reg(P, P_0) = \sqrt{\left(\frac{KL(P||P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right)}$$

  - How can we add shielding to improve the bound?

Correction Demonstration Ranking



Bobu et al. TRO’20
Bıyık et al. IJRR’20



Inverse Specification

Interact with human to better understand their preference

↑ send queries to human and receive their ranking feedback

↑ (1) components of objective
(2) constraints

Mental States

Noisy translation ↓ ↑ Inverse Specification

Specification

Previous Works – inverse reinforcement learning / inverse optimal control

- Maximum Entropy IRL [Ziebert et al. AAAI'08]
  - Based on demonstrations
  - The trajectory distribution only relies on the human utility
  - $P_{w_H}(q_i) \propto \exp(u_{w_H}(q_i))$, u_{w_H}: human utility

- IRL by human preferences [Christiano et al. NeurIPS'17]
  - Given a query, $\mathbf{q} \coloneqq (q_i, q_j)$
  - Provide feedback (f): $q_i > q_j, f = [1,0]^T$; else, $f = [0,1]^T$
  - Loss: $L = -\sum f[0] \log P_w(\text{pick } q_i) + f[1] \log P_w(\text{pick } q_j)$

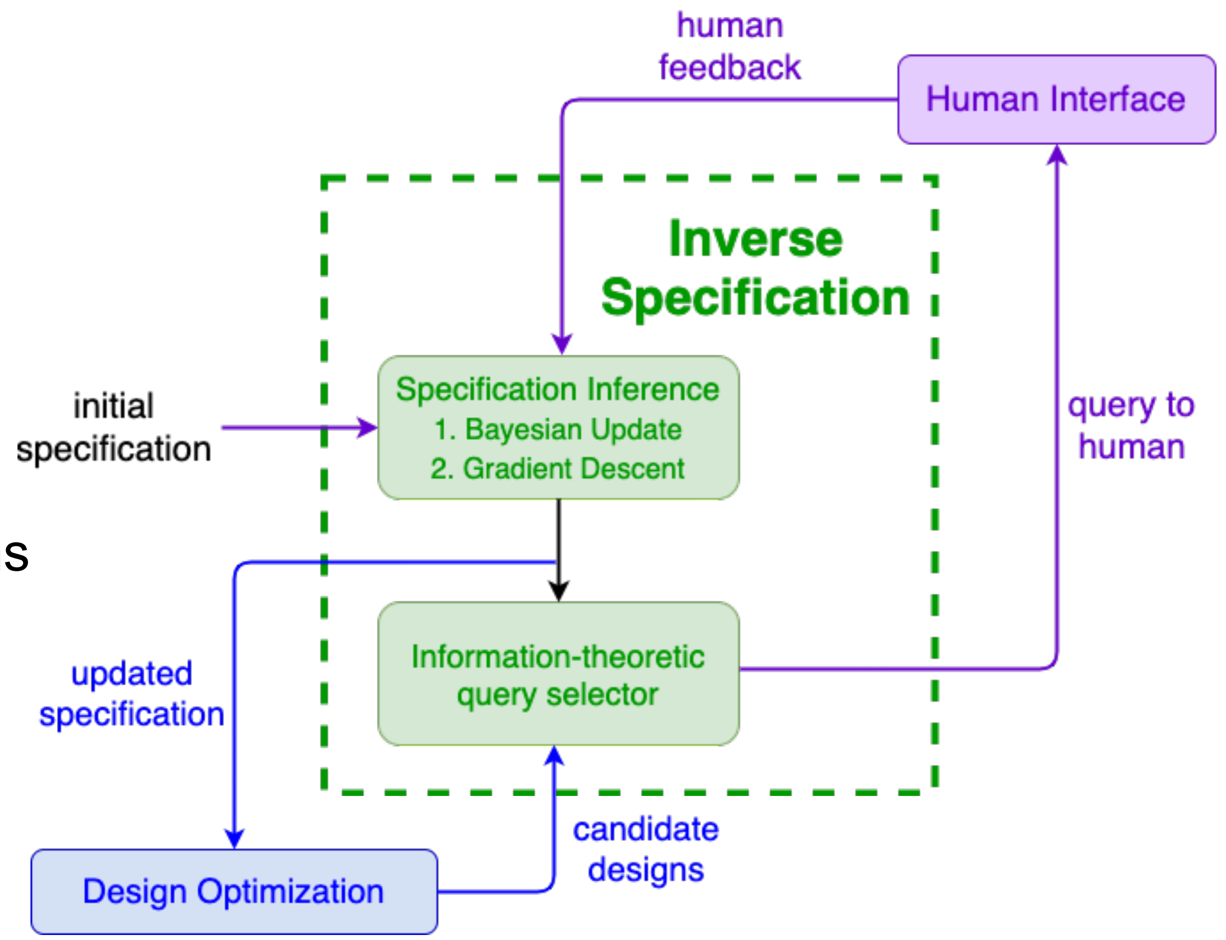
- Constraint inference for IRL [Scobee et al ICLR'20]
  - Assume nominal reward ($\tilde{\boldsymbol{w}}$) and N available demonstrations (Q_D)
  - Maximize $P(C) = \frac{1}{Z(C)^N} \prod_{q \in Q_D} \exp(u_{\tilde{w}}(q)) \mathbb{1}_C(q)$

Demonstrations can be hard to generate

Preference by reward and constraint is more succinct

Overall Structure

- **Inverse specification**
  - We interact with humans to refine the problem specification and accelerate exploration
- Design optimization
  - We pick candidate designs by genetic algorithms or trained policies by reinforcement learning
- Human interface
  - We pick informative queries from the candidate designs or trajectories



Experiment Details

- Human preference
  - $P_{\boldsymbol{w_H},\boldsymbol{C}}(\text{pick } q_i) \propto \exp(u_{\boldsymbol{w_H}}(q_i)) \cdot \mathbb{1}_C(q_i)$
- Human model in inverse specification machinery
  - $P_{\boldsymbol{w},\,\boldsymbol{\theta}}(\text{pick } q_i) \propto \exp(u_{\boldsymbol{w}}(q_i)) \cdot\ h_{\boldsymbol{\theta}}(q_i)$
- Design space
  - Each design: $\mathrm{q} \in \mathbb{R}_+^6$
  - The true optimal design is obtained by

$$\arg\max_{\mathrm{q}} \boldsymbol{w}_{\mathrm{H}}^T q \cdot \mathbb{1}_C(q)$$

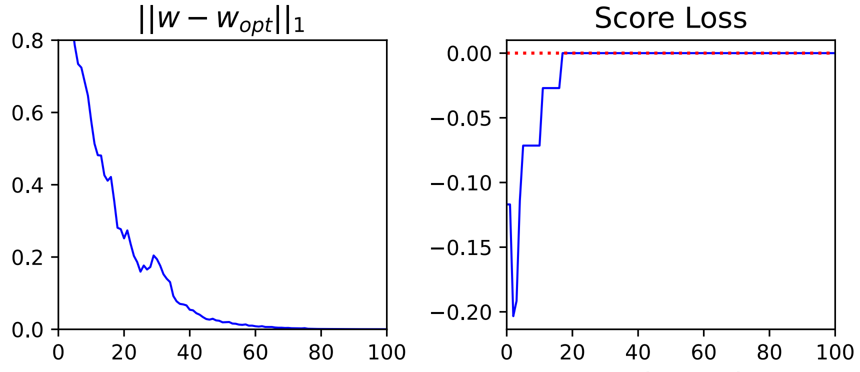
  - The predicted optimal design is obtained by

$$\arg\max_{\mathrm{q}} \boldsymbol{w}^T q \cdot h_\theta(q)$$

Infer utility, assume no explicit constraints

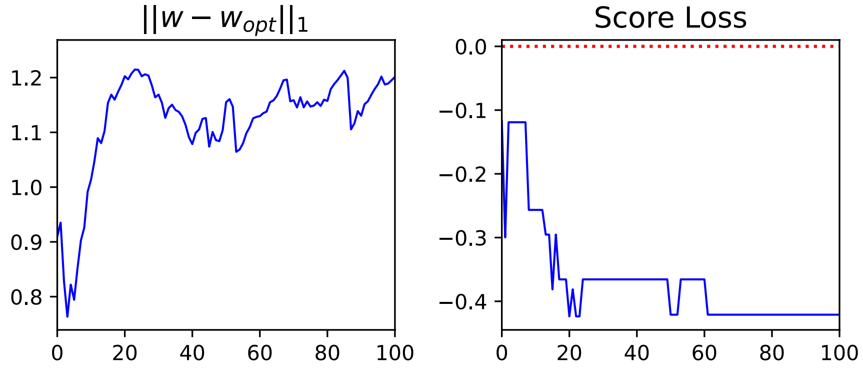
- Bayesian Update: $P(\mathbf{w} \mid \mathbf{q}, f) \propto P(f \mid \mathbf{q}, \mathbf{w}) \cdot P(\mathbf{w})$

no hard constraints



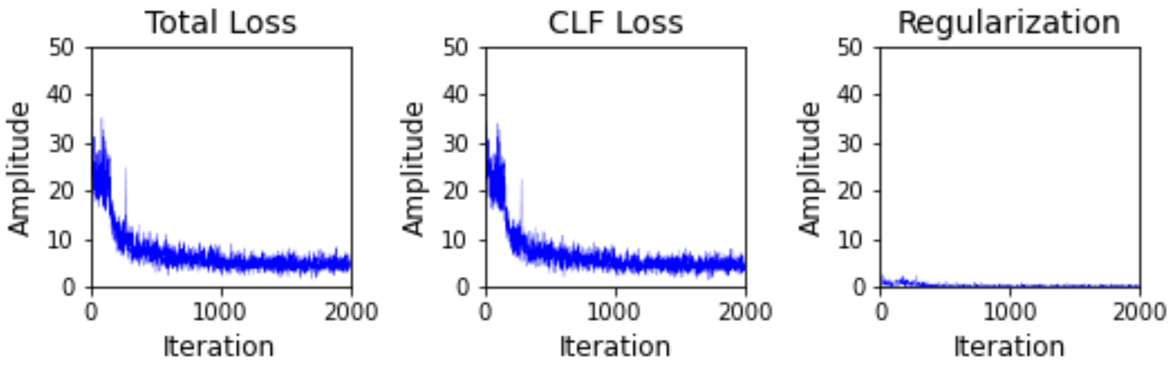
- Constraint-agnostic inferred utility over-emphasizes constrained features.

one hard constraint



Infer constraints, given proxy utility

- $L(\boldsymbol{\theta}) = \sum_{(\mathbf{q},f)\in B} \mathrm{KL}(P_{\boldsymbol{\theta}}(\mathbf{q}) \,||\, f) + \alpha\, \mathrm{Reg}(\boldsymbol{\theta})$
  - Gradient descent on neural network parameters ($\boldsymbol{\theta}$)



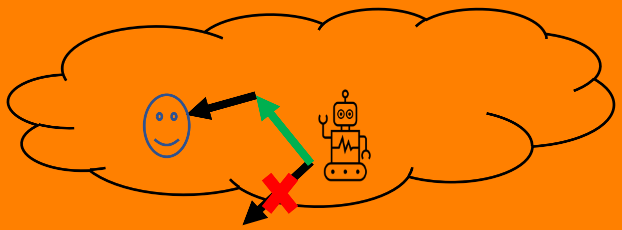
  - Feasible designs but classified infeasible: 4.3%
  - Infeasible designs but classified feasible: 0%
- Predict top designs by: $\arg\max_{q} u_{\boldsymbol{w}}(q) \cdot h_{\boldsymbol{\theta}}(q)$
  - Predicted top-5 designs: [133 23 45 114 173]
  - Real top-5 designs: [133 23 45 114 173]

Future Works

- Infer the utility and constraint simultaneously
  - Alternating gradient descent: $v_{\boldsymbol{w},\boldsymbol{\theta}}(q) = u_{\boldsymbol{w}}(q) \cdot h_{\boldsymbol{\theta}}(q)$
- Active learning: how to select the most informative queries to present to the human designer
  - Information gain
  - What is the analog metric?

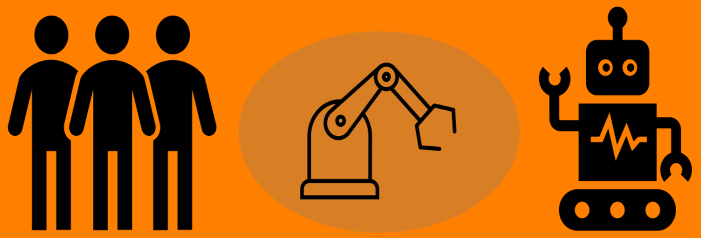
Key Takeaways

Supervisory Control



- Discounted reach-avoid Bellman equation enables reinforcement learning to solve HJ PDE
- We treat the policy as untrusted oracles and employ a shielding scheme → Learned policy is the best-effort reach-avoid policy

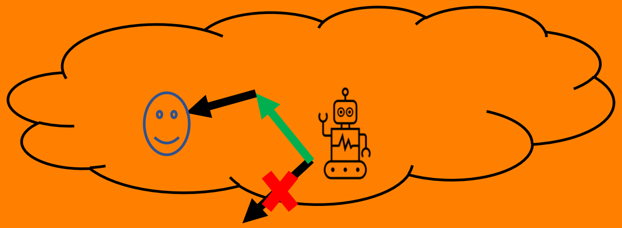
Inverse Specification



- Separating constraints from components of objective function makes the problem easier
- We can infer constraints by querying human a pair of designs and receiving ranking feedback

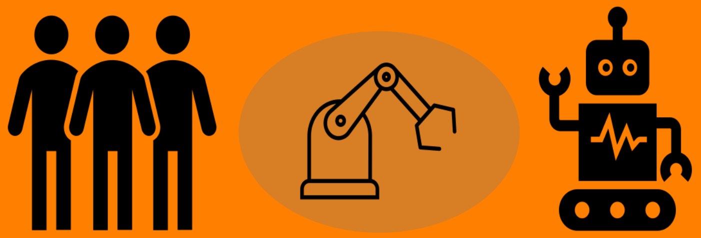
Future Works

Supervisory Control



- How to separate the action policy and reach-avoid value function?
- How to use RL to solve zero-sum reach-avoid differential game?
- How to use shielding scheme to improve PAC-Bayes bound on novel environments?

Inverse Specification



- How to infer utility and constraint together?
- How to select the most informative queries to present to the human designer?

Reference

- Supervisory Control: Shielding

  - J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. “Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints”. Proceedings of the 18th Inter-national Conference on Hybrid Systems: Computation and Control. HSCC ’15. Seattle, Washington: Association for Computing Machinery, 2015, pp. 11–20
  - J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, et al. “Bridging Hamilton-Jacobi safety analysis and reinforcement learning”.2019 International Conference on Robotics and Automation (ICRA). IEEE. 2019,pp. 8550–8556
  - O. Bastani. “Safe Reinforcement Learning with Nonlinear Dynamics via Model Predictive Shielding”. 2020. arXiv: 1905.10691[cs.LG].

- Inverse Specification

  - B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum Entropy Inverse Reinforcement Learning,” in Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI, 2008.
  - P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” 2017. arXiv: 1706.03741 [stat.ML].
  - D. R. Scobee and S. S. Sastry, “Maximum likelihood constraint inference for inverse reinforcement learning,” in International Conference on Learning Representations, 2020.
