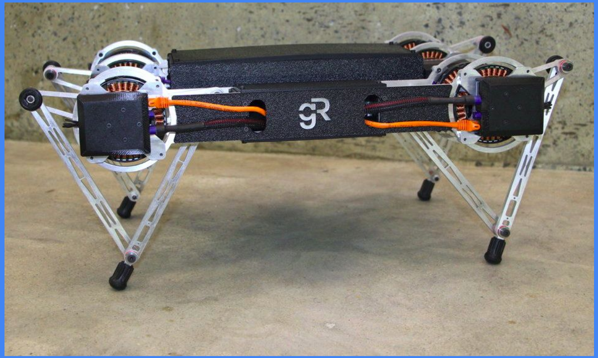
Few Shot Learning for Robot Motion

Intelligent Robotics Seminar 06.01.2020
University of Hamburg
Lisa Mickel

Content

- Introduction
- Reinforcement learning
- Approach 1: Model free maximum entropy
- Approach 2: Model based
- Results: Simulation and real-life
- Comparison & conclusion

[1]


[2]

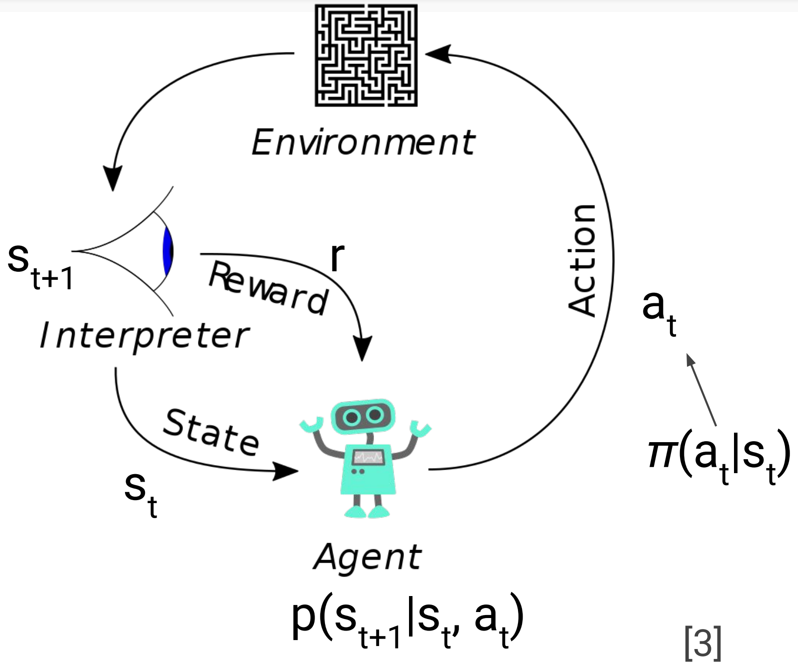
Introduction

- State-of-the-art locomotion controllers: explicit planning of motions → precise knowledge of robot dynamics

- Here: reinforcement learning on Minitaur robot
- Requirements:
  - Stable gait on various terrains
  - Sample efficient
  - Safe learning

Reinforcement Learning

- Markov Decision Process (MDP):
  - State and action space
  - Transition probability
  - Policy
  - Reward function
- Model free: learn policy $\boldsymbol{\pi}(a_t|s_t)$
- Model based: learn transition probability $p(s_{t+1}|s_t, a_t)$



[3]

Approach 1: *Learning to Walk via Deep Reinforcement Learning*

Haarnoja , Ha , Zhou , Tan, Tucker, Levine

Google Brain, University of California, Berkeley

Jun 2019

- Model free algorithms often limited to simulation
- Extension of maximum entropy learning

A1: Maximum Entropy Learning

- Entropy = measure for variance
- Encourage exploration by including entropy of policy

$$\sum_{t=0}^{T} \mathbb{E}_{\tau \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) - \alpha_t \log \pi_t(\mathbf{a}_t|\mathbf{s}_t) \right]$$

  - Hyperparameter α = temperature
  - Training results dependent on its value

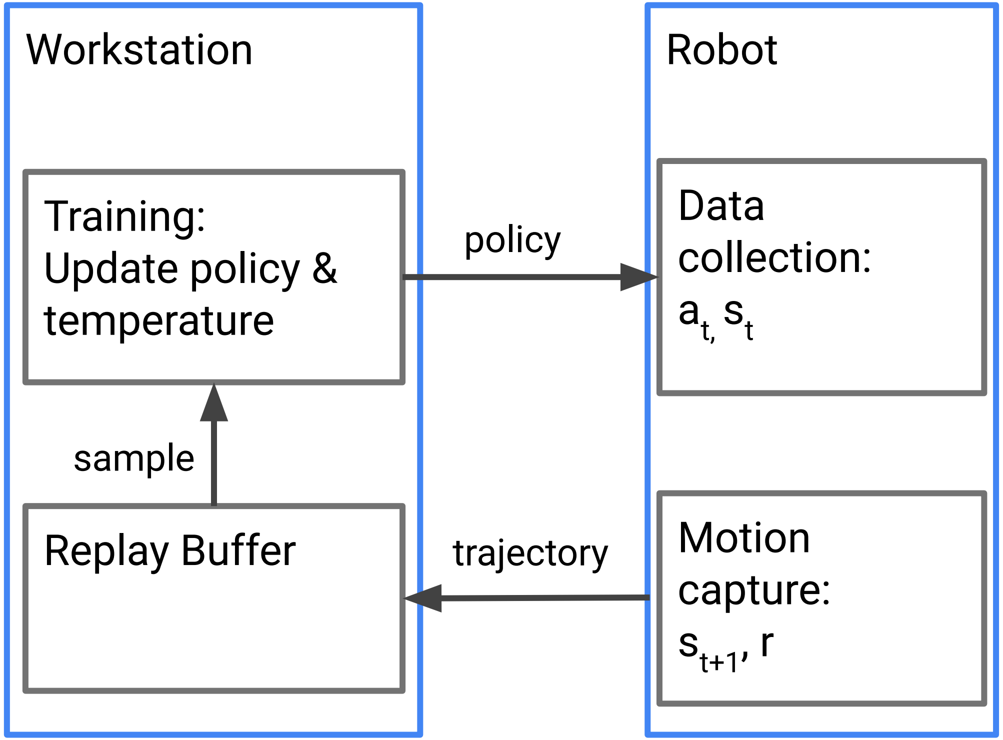
- New approach: learn temperature
  - Add constraint: Minimum expected entropy H of policy *π*

$$\mathbb{E}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) + \alpha_t \left(-\log\left(\pi_t(\mathbf{a}_t|\mathbf{s}_t)\right) - \mathcal{H} \right) \right]$$

A1: System Setup

- On robot:
  - Execute policy
  - Measure robot state
  - Compute reward signal

- Workstation
  - Train with sample from buffer
  - Update policy (neural network) parameters and temperature



Approach 2: *Data Efficient Reinforcement Learning for Legged Robots*

Yang, Caluwaerts, Iscen, Zhang, Tan, Sindhwani

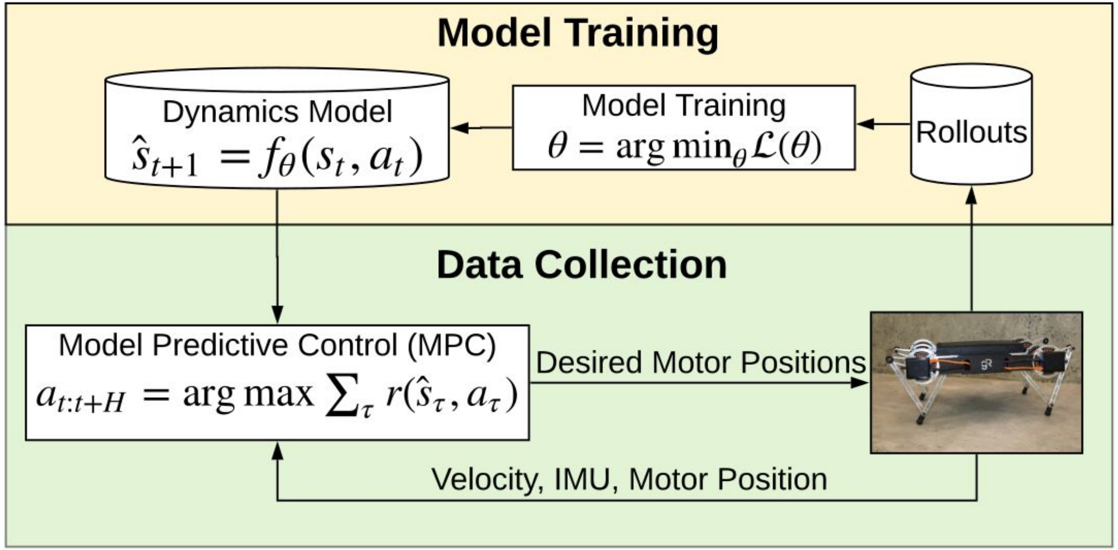
Robotics at Google
United States

Oct 2019

- Model based few shot RL algorithm

A2: System Setup

- MPC: plan action based on dynamics model → execute plan
- Current robot state as feedback, periodically replan
- Periodic retraining with all trajectories

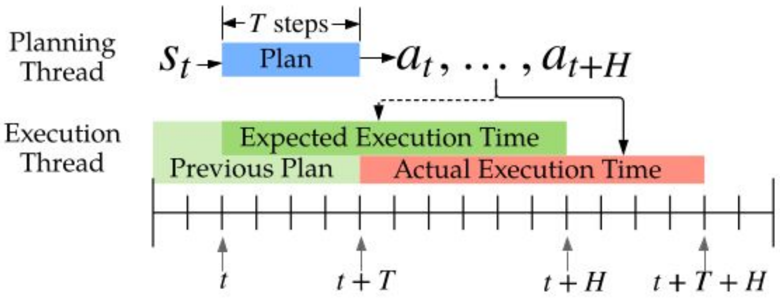


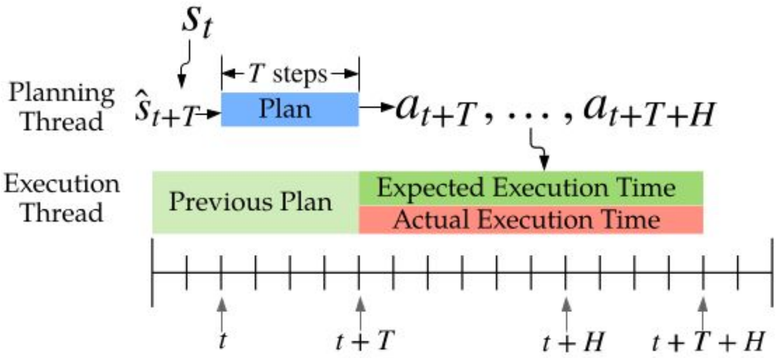
[A2]

A2: Planning

- Control frequency > planning frequency
  - Simultaneous planning and execution of actions
  - Planning horizon: 450 ms (=75 control steps), replan every 72 ms

- Planning latency
  → plan based on future robot state (asynchronous control)





[A2]

A2: Training

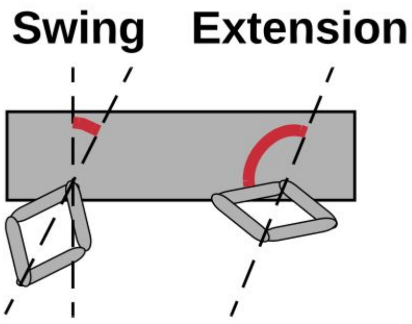
- Dynamics model: Neural network
- Long term accuracy of dynamics model: multi-step loss function
- Predict n states and average over single step error → accumulation of error

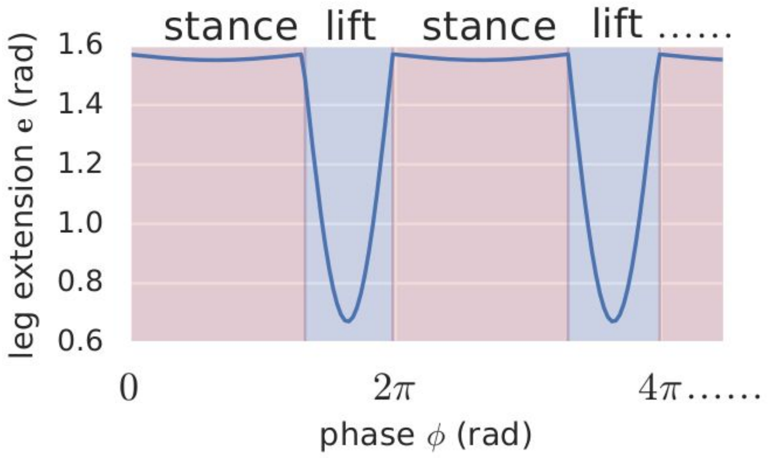
$$\mathcal{L}_{\text{multi-step}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{s}_{t:t+n}, \boldsymbol{a}_{t:t+n-1}) \in \mathcal{D}} \frac{1}{n} \sum_{\tau=1}^{n} \left\| (\boldsymbol{s}_{t+\tau} - \boldsymbol{s}_{t+\tau-1}) - f_\theta(\hat{\boldsymbol{s}}_{t+\tau-1}, \boldsymbol{a}_{t+\tau-1}) \right\|_2^2$$

A2 Trajectory Generators

- Smooth robot motion
  → Trajectory generators (TGs)

- Periodically lift legs
- 4 independent phases
  → Freely modulate leg movements independently

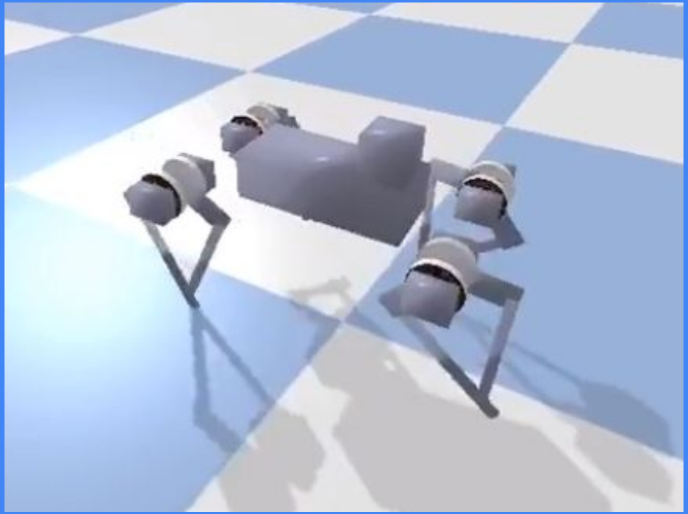




[A2]

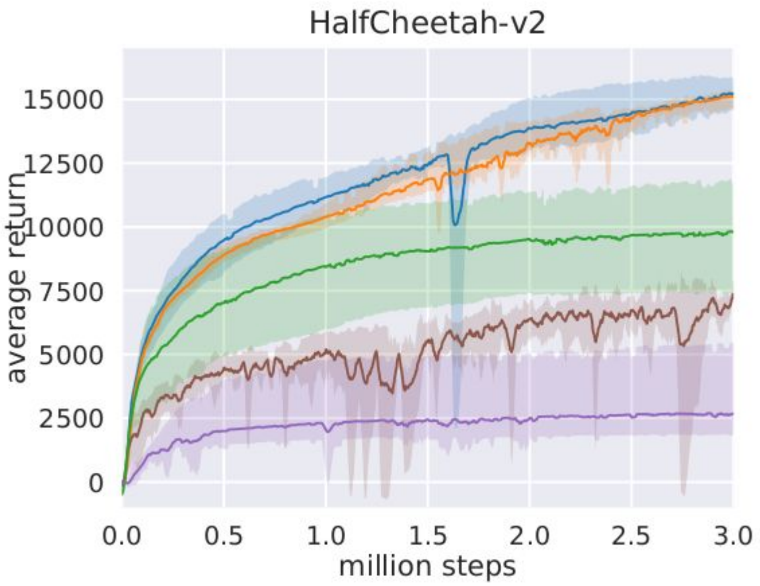
Results: Simulation

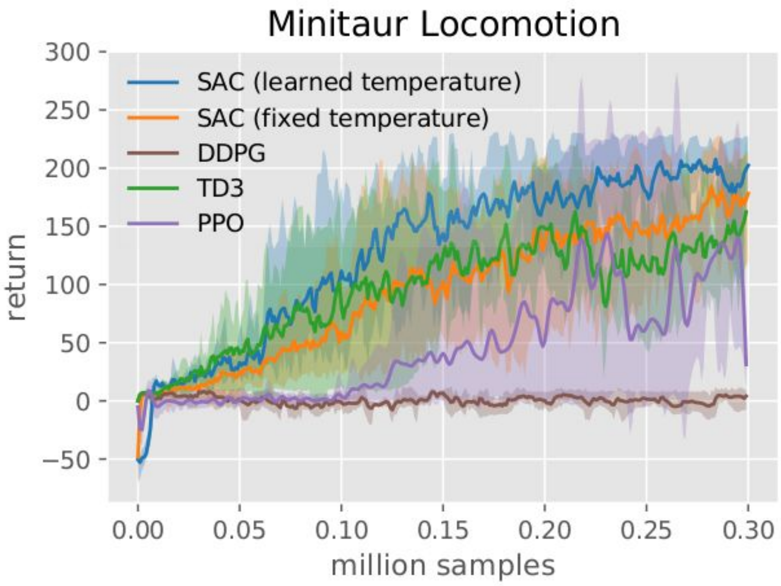
- Goal:
  - **A1:** Walk straight
  - **A2:** Walk forward matching speed profile



[A1]

A1: Performance



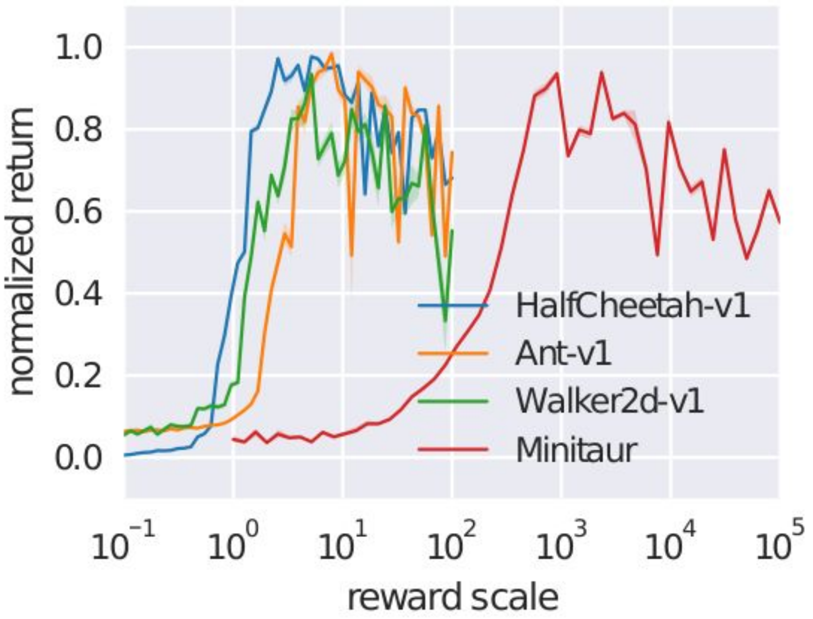


[A1]

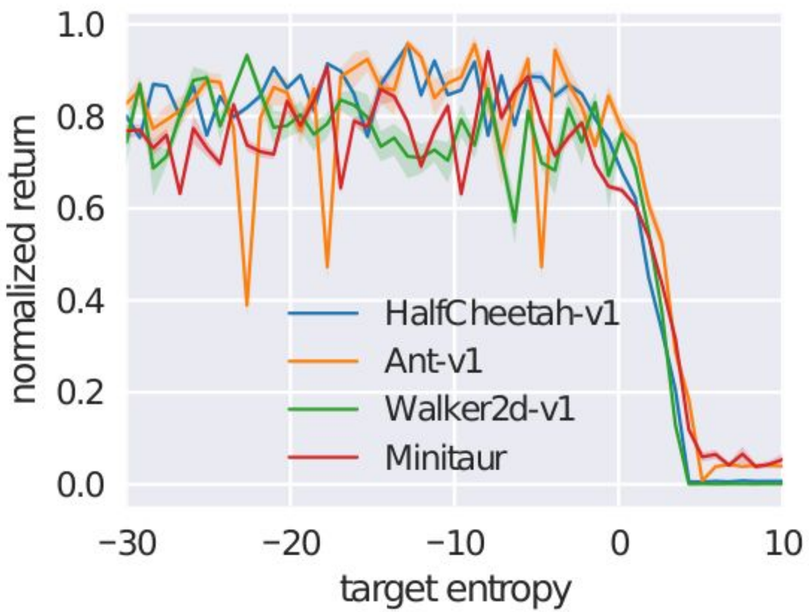
- Several benchmark tests
- Compare to standard algorithms → A1 matches best performance
- Best on minitaur robot

A1: Influence of hyperparameter on performance

- SAC: Temperature = inverse reward scale



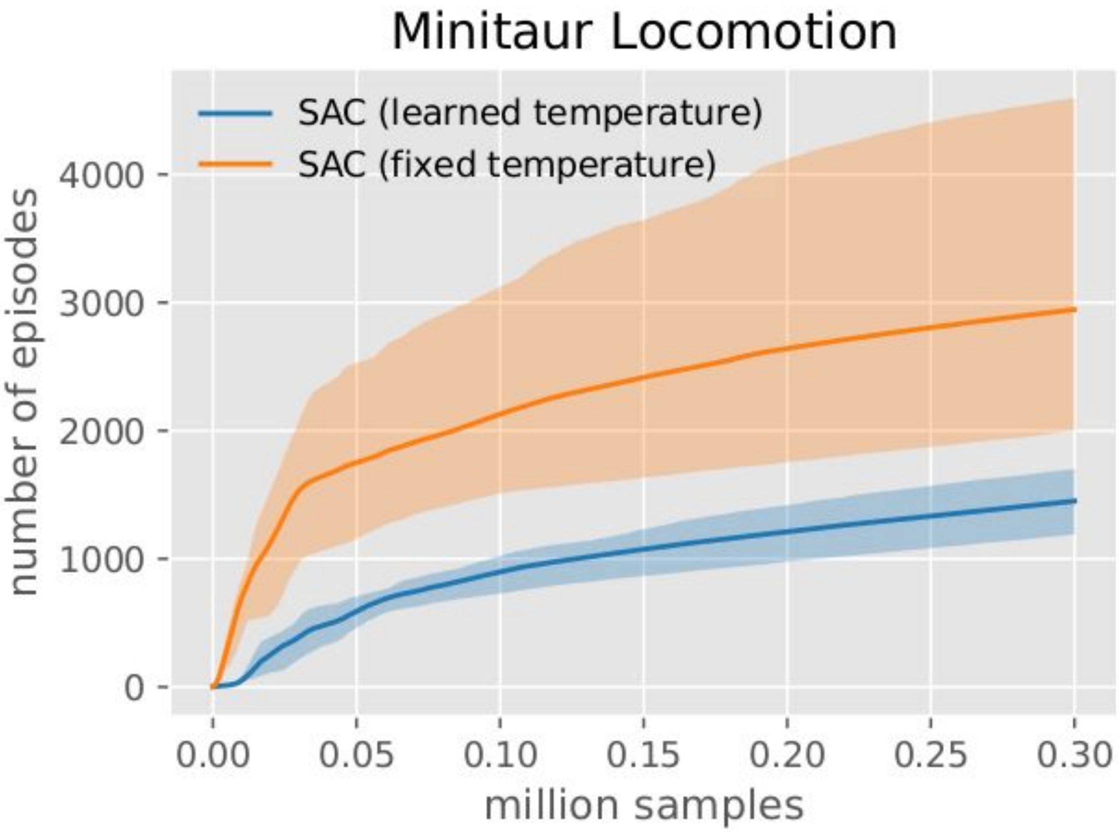
- A1: Minimum expected entropy



[A1]

A1: Data Efficiency

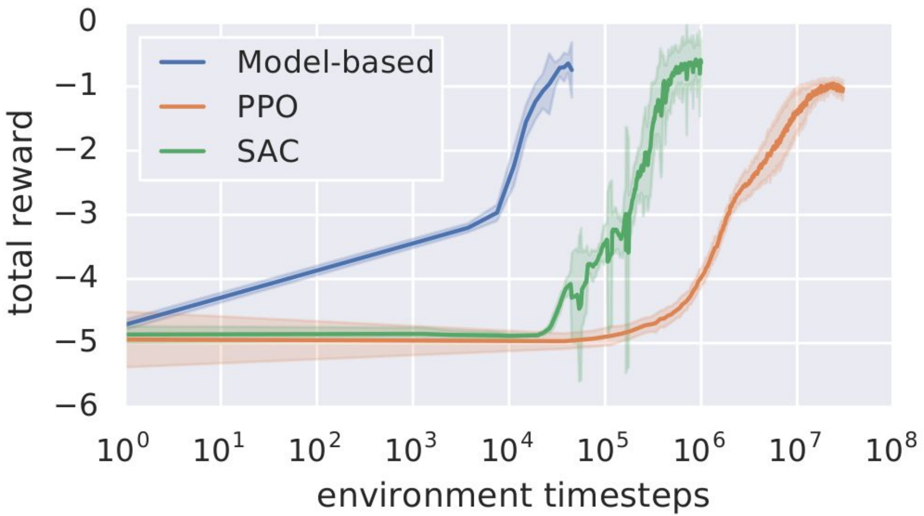
- Compare to standard SAC
- Two measures for data efficiency:
  - number of control steps
  - number of episodes



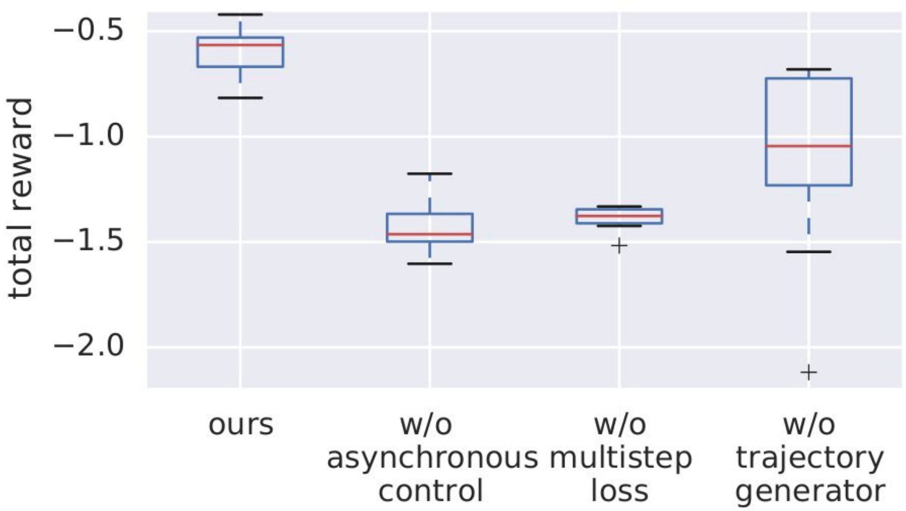
[A1]

A2: Performance

- Comparison to model free algorithms

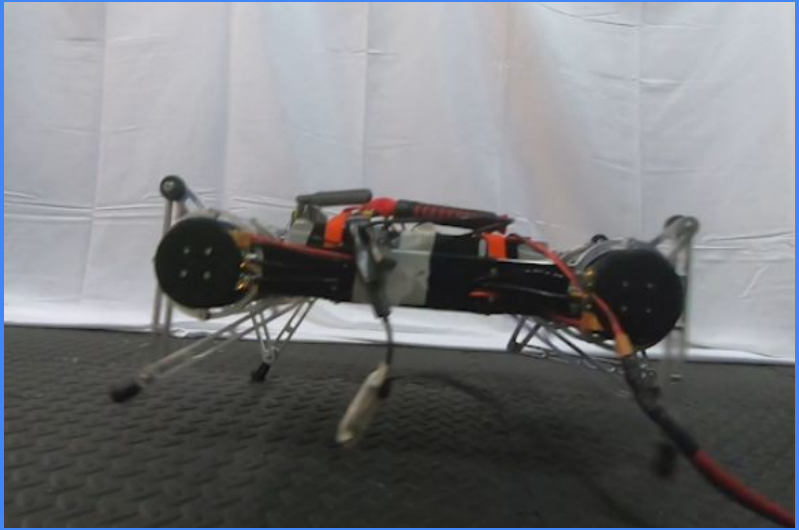


- Influence of algorithm components on performance



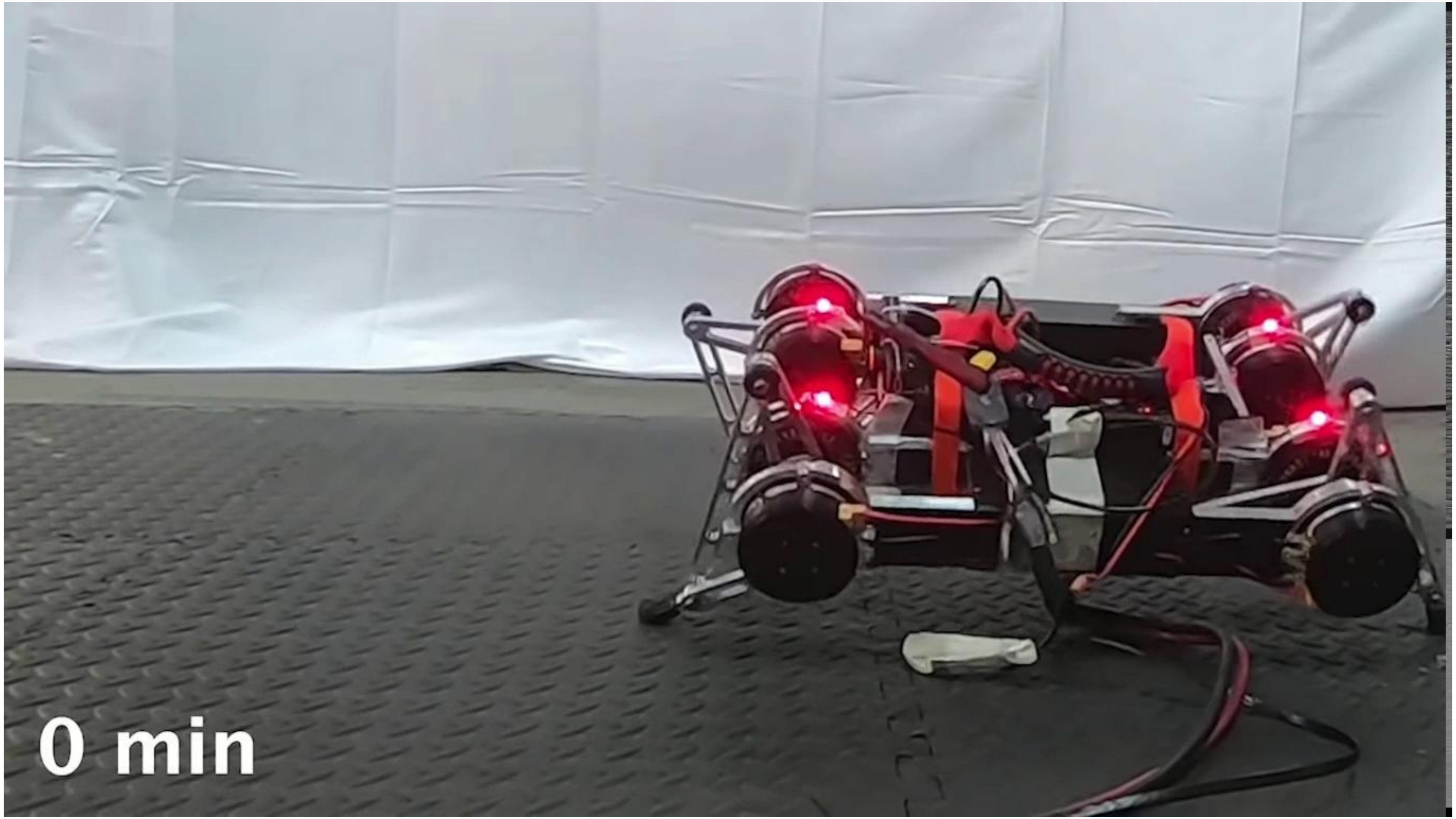
[A2]

Results: Real-Life



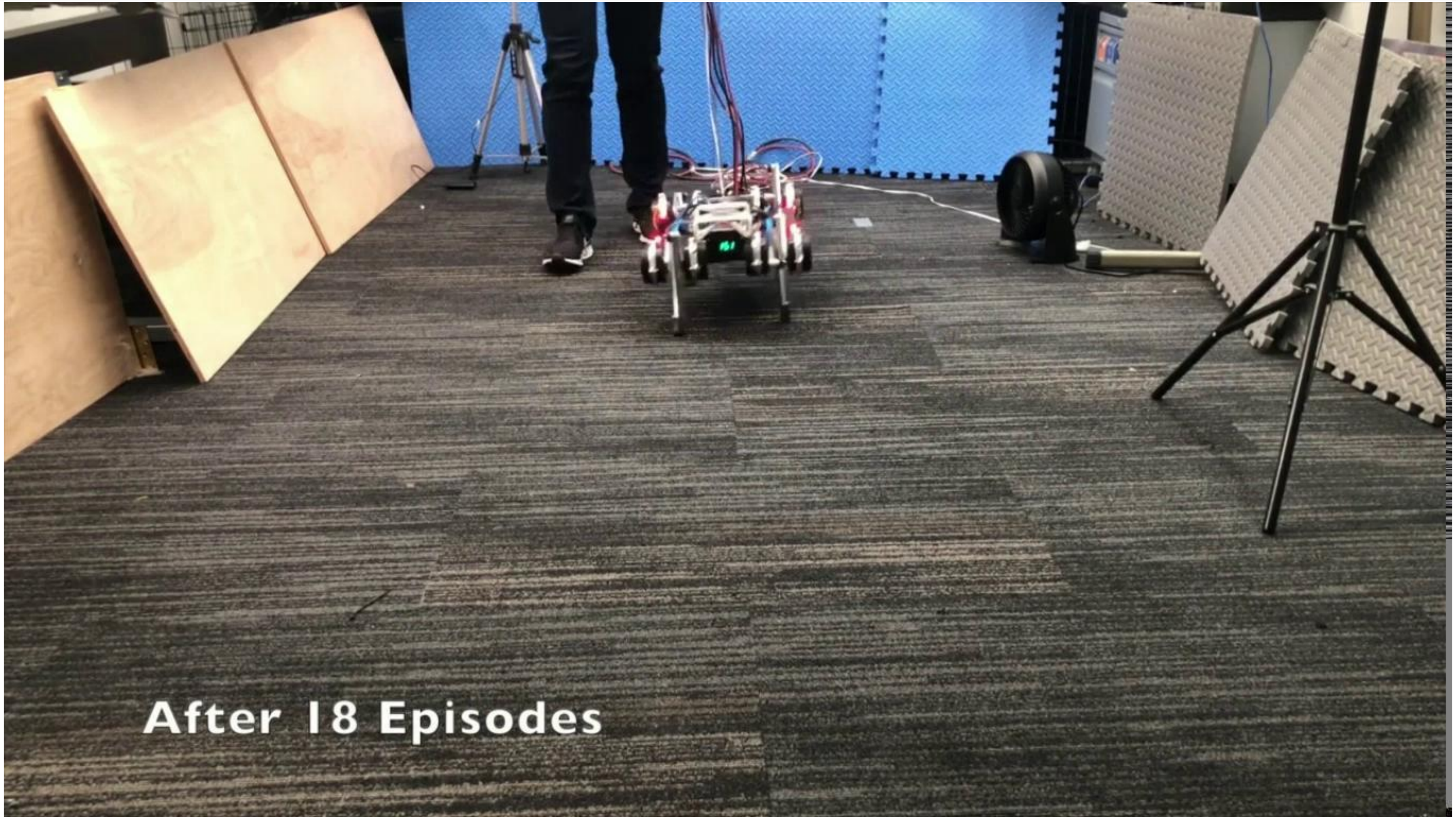
[A1]

A1: Training Video



[v1]

A2: Training Video

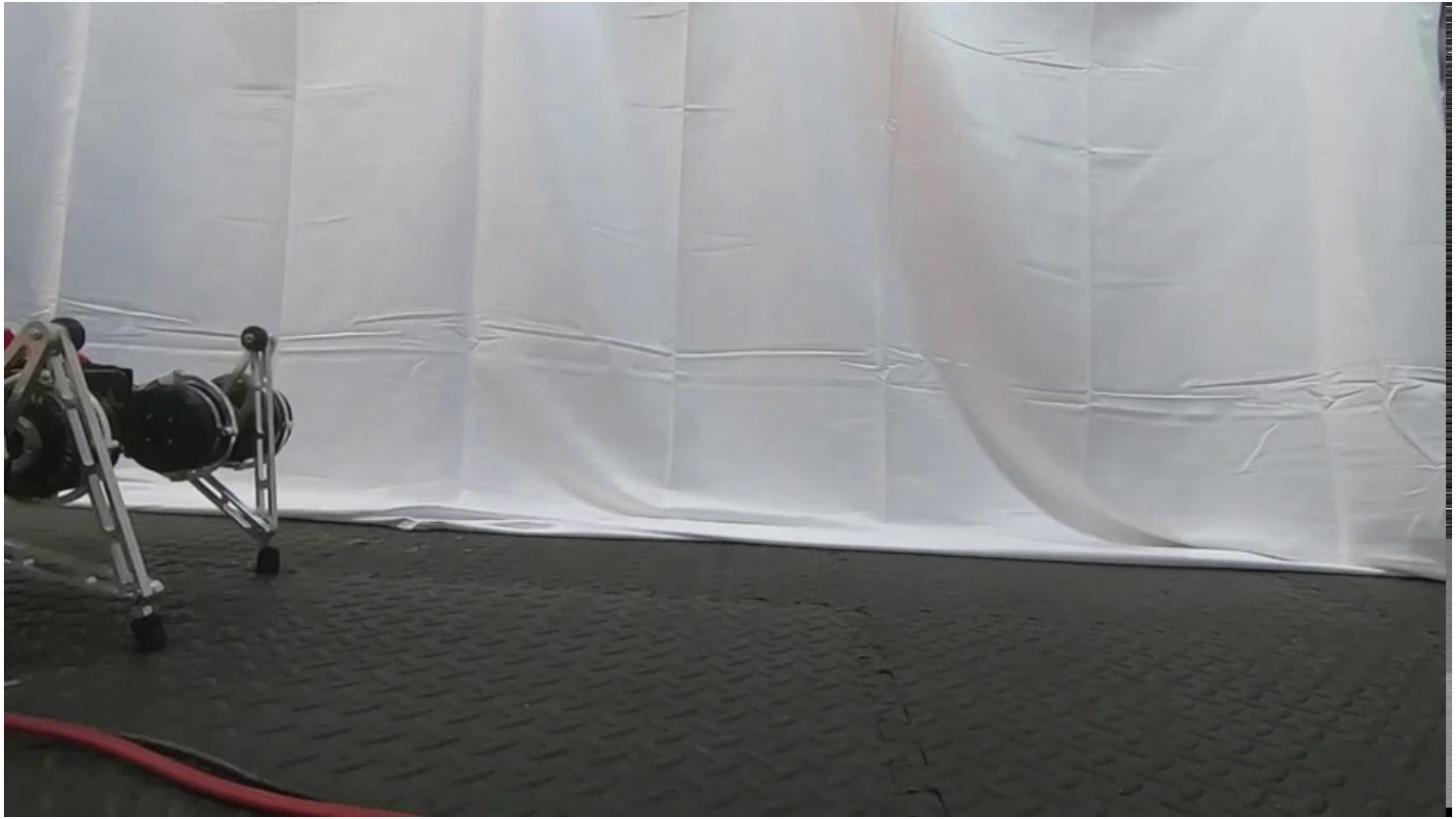


[v2]

Training Results

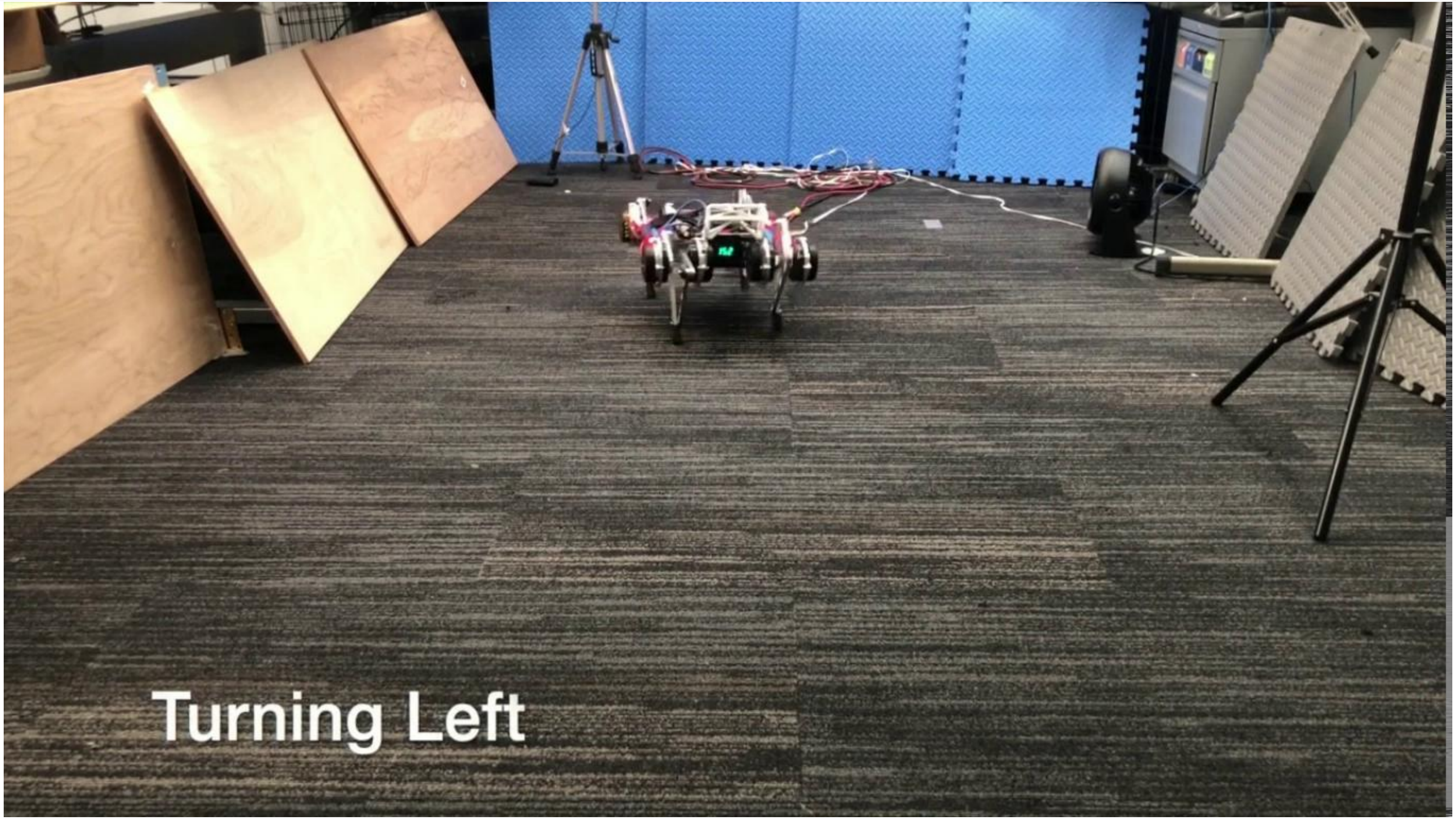
Approach	**A1**	**A2**
Walking speed	0.32 m/s (0.8 body lengths/s)	0.66 m/s (1.6 body lengths/s)
Steps	160 000	45 000
Episodes	400	36

A1: Generalization



[v3]

A2: Generalization



[v2]

Comparison

Approach	**A1**	**A2**
Gait	Learns sinusoidal pattern, different front and hind leg frequency	Adapts sinusoidal pattern of TGs Higher walking speed
Data efficiency	Better than standard SAC	Better than A1
Hyperparameters	Minimum expected entropy	Planning algorithm, multi step loss (simulation)
Gait generalizability	Slope, step, obstacle	Slope New tasks
Range of applicability	Various robots	Problem specific Adaptability?

Conclusion and Outlook

- Two data efficient reinforcement algorithms that successfully train real-life minitaur robot to walk

- Future work:
    - Additional sensors → more complex behaviours
    - Safety measures → larger robots

Thank you for your attention!

References

A1: Tuomas Haarnoja,, Sehoon Ha , Aurick Zhou , Jie Tan, George Tucker, Sergey Levine; *Learning to Walk via Deep Reinforcement Learning*; arXiv:1812.11103v3 [cs.LG]; Jun 2019

A2: Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, Vikas Sindhwani; *Data Efficient Reinforcement Learning for Legged Robots*; arXiv:1907.03613v2; Oct 2019

Other:

- https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287
- https://en.wikipedia.org/wiki/Reinforcement_learning
- https://spinningup.openai.com/en/latest/algorithms/sac.html
- https://en.wikipedia.org/wiki/Markov_decision_process

Image Sources

[1] https://newatlas.com/anymal-quadruped-robot-eth-zurich/52097/

[2] https://www.hackster.io/news/meet-ghost-minitaur-a-quadruped-robot-that-climbs-fences-and-opens-doors-bfec23debdf4

[3] https://en.wikipedia.org/wiki/Reinforcement_learning

Video links

[v1] https://www.youtube.com/watch?time_continue=4&v=FmMPHL3TcrE&feature=emb_logo

[v2] https://www.youtube.com/watch?v=oB9IXKmdGhc&feature=youtu.be

[v3] https://www.youtube.com/watch?v=KOObeIjzXTY&feature=emb_logo