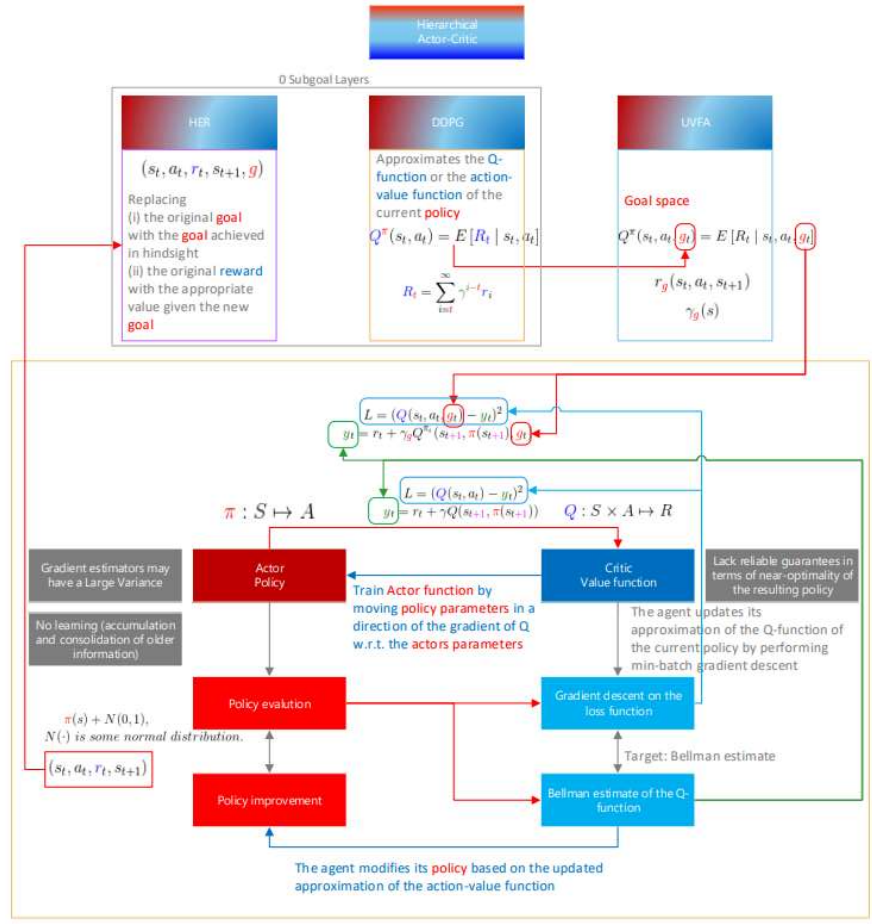
Hierarchical Actor-Critic

Hierarchical Actor-Critic (https://blogs.cuit.columbia.edu/zp2130/files/2019/02/Hierarchical-Actor-Critic.pdf)

Download Hierarchical_Actor-Critic Flowchart (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/Hierarchical_Actor-Critic.pdf)



Terminology

	Artificial intelligence	Optimization/decision/control
a	Agent	Controller or decision maker
b	Action	Control
c	Environment	System
d	Reward of a stage	(Opposite of) Cost of a stage
e	Stage value	(Opposite of) Cost of a state
f	Value (or state-value) function	(Opposite of) Cost function
g	Maximizing the value function	Minimizing the cost function
h	Action (or state-action) value	Q-factor of a state-control pair
i	Planning	Solving a DP problem with a known mathematical model
j	Learning	Solving a DP problem in model-free fashion
k	Self-learning (or self-play in the context of games)	Solving a DP problem using policy iteration
l	Deep reinforcement learning	Approximate DP using value and/or policy approximation with deep neural networks
m	Prediction	Policy evaluation
n	Generalized policy iteration	Optimistic policy iteration
o	State abstraction	Aggregation
p	Episodic task or episode	Finite-step system trajectory
q	Continuing task	Infinite-step system trajectory
r	Afterstate	Post-decision state

Hierarchical Actor-Critic (HAC) helps agents learn tasks more quickly by enabling them to **break** problems down **into** short sequences of actions. They can divide the work of learning behaviors among multiple policies and explore the environment at a higher level.

In this paper, authors introduce a novel approach to hierarchical reinforcement learning called Hierarchical Actor-Critic(HAC). The algorithm enables agents to learn to divide **tasks involving continuous state and action spaces** into **simpler problems belonging to different time scales.**

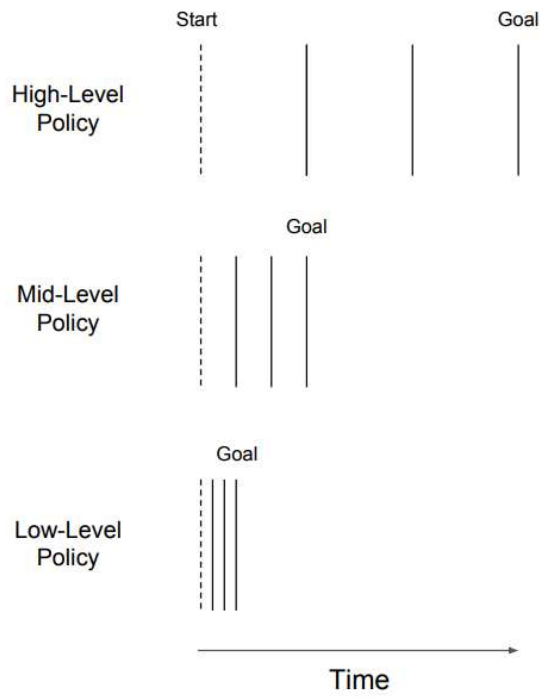


Figure 1. Example HAC Hierarchy. Agent in figure uses three policies to learn a behavior. Each policy specializes in breaking down problems into subproblems with finer time resolutions.

The low-level policy outputs actual agent actions so the **vertical lines** for the **low-level policy** can be interpreted as **subgoals** requiring **one action**.

In the figure, the **high-level policy** breaks down the end goal into three subgoals with relatively large time resolutions. The **mid-level policy** specializes in dividing each subgoal from the **high-level policy** into three subgoals belonging to shorter time scales. Finally, the **low-level policy** specializes in decomposing each subgoal from the mid-level policy into three agent actions which represent the smallest time resolution.

HAC builds off three techniques from the reinforcement learning literature:

(i) the Deep Deterministic **Policy Gradient** (DD**PG**) learning algorithm (Lillicrap et al., 2015)

(ii) Universal **Value Function** Approximators (U**VF**A) (Schaul et al., 2015)

(iii) **Hindsight** Experience Replay (HER) (Andrychowicz et al., 2017)

DDPG: an actor-critic algorithm

DD**PG** serves as the key learning infrastructure within Hierarchical Actor-Critic. DD**PG** is an **actor–critic** algorithm and thus uses two neural networks to enable agents to learn from experience. The **actor network** learns a deterministic policy that maps from states to actions

$$\pi : S \mapsto A$$

The **critic network** approximates the Q-function or the action-value function of the current policy

$$Q^{\pi}(s_t, a_t) = E\left[R_t \mid s_t, a_t\right]$$

where

$$R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$$

R_t: the discounted sum of future **rewards**

Thus, the **critic network** maps from (state, action) pairs to expected long-term reward:

$$Q : S \times A \mapsto R$$

Policy evaluation

The agent first interacts with the environment for a period of time using a noisy policy $\pi(s) + N(0,1), N(\cdot)\ is\ some\ normal\ distribution.$ The transitions experienced are stored as (s_t, a_t, r_t, s_{t+1}) **(s_t, a_t, r_t, s_{t+1})** tuples in a **replay buffer.**

The agent then updates its approximation of the **Q-function** of the current policy by performing min-batch gradient descent on the **loss function**:

$$L = (Q(s_t, a_t) - y_t)^2 y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$$

the target y_t is the **Bellman estimate** of the **Q-function**

Policy improvement

The agent modifies its **policy** based on the updated **approximation** of the **action-value function**. The actor function is trained by moving its **parameters** in the direction of the **gradient of Q** w.r.t. the **actors parameters.**

UVFA

Goal:

$$Q^{\pi}(s_t, a_t, g_t) = E\left[R_t \mid s_t, a_t, g_t\right]$$
$$r_g(s_t, a_t, s_{t+1})$$
$$\gamma_g(s)$$

HER

Even though an agent may have failed to achieve its **given goal** in an episode, the agent did learn a sequence of actions to **achieve a different objective** in **hindsight –** the state in which the agent **finished**.

Learning how to **achieve different goals** in the goal space should help the agent better determine how to achieve the **original goal**.

Creating a separate copy of the transitions:

$$(s_t, a_t, r_t, s_{t+1}, g)$$

that occurred in an episode and replacing

(i) the original **goal** with the **goal** achieved in hindsight

(ii) the original **reward** with the appropriate **value** given the new **goal**.

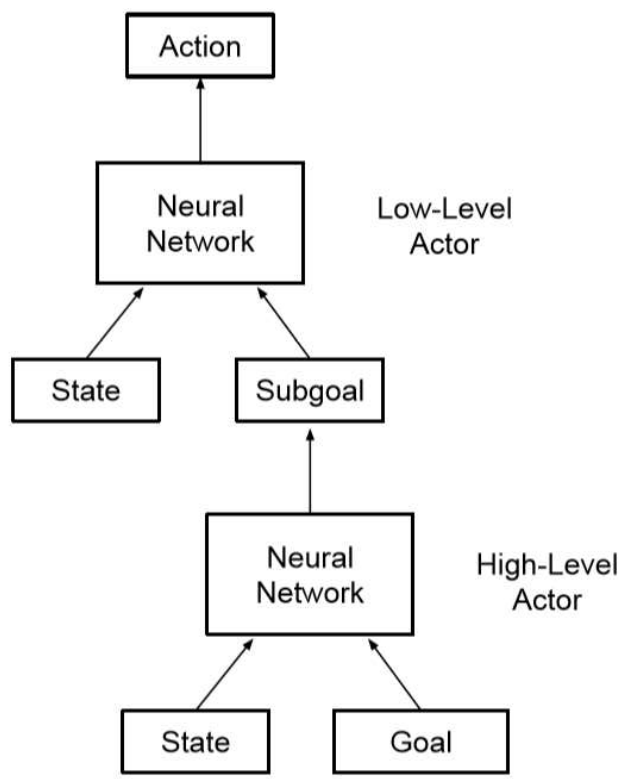


Figure 2. Hierarchical policy with 1 subgoal layer

The hierarchical **policy** is composed of multiple **goal-based policies** or **actor** networks.

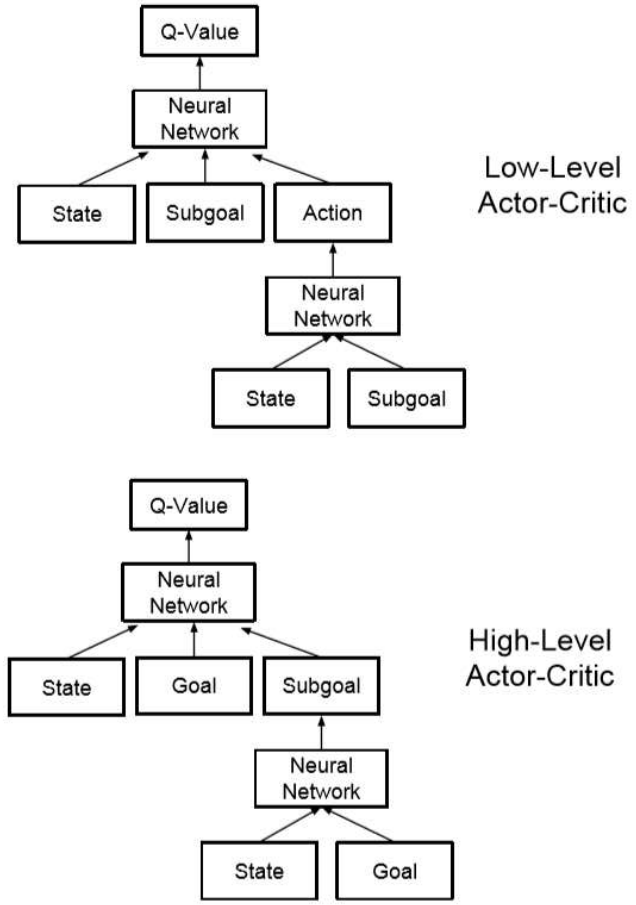
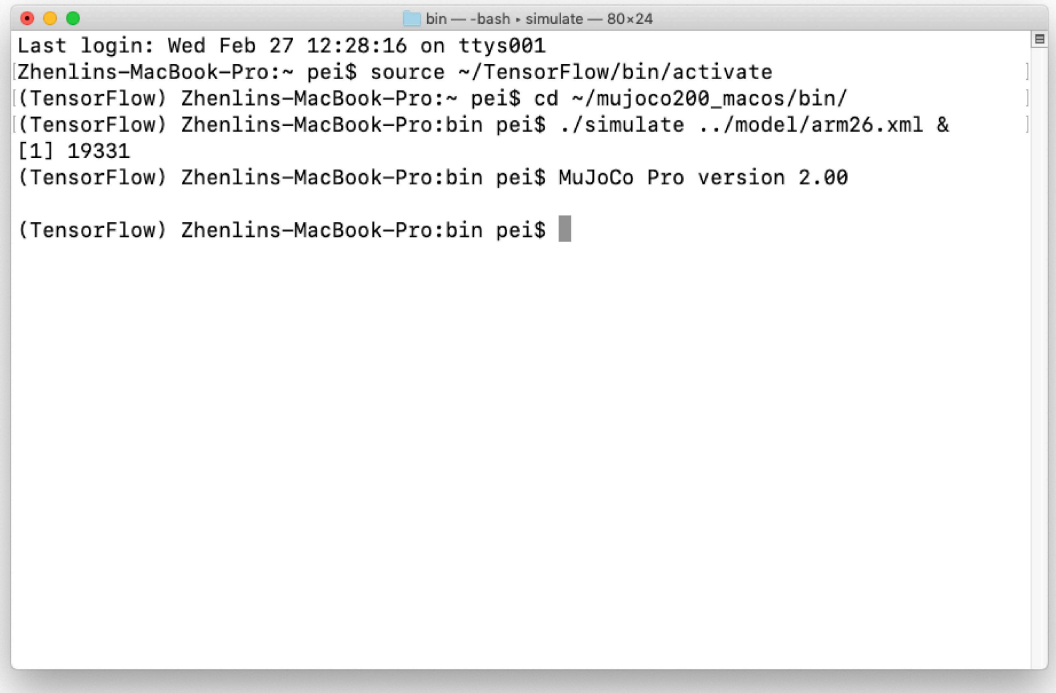
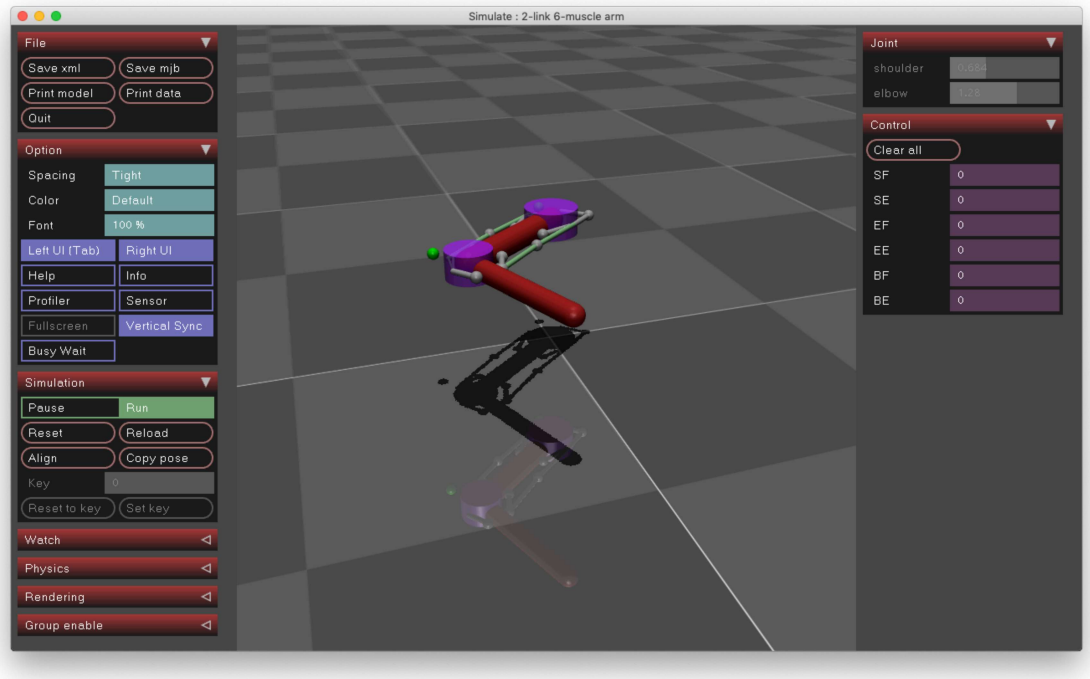


Figure 3. Actor-Critic networks for hierarchical policy with 1 subgoal layer.

Experiments

Mujoco physics engine(Todorov et al., 2012)





This Github repository (https://github.com/andrew-j-levy/Hierarchical-Actor-Critc-HAC-) contains the code to implement the Hierarchical Actor-Critic (HAC) algorithm. HAC helps agents learn tasks more quickly by enabling them to break problems down into short sequences of actions. The paper describing the algorithm is available here (https://openreview.net/pdf?id=ryzECoAcY7).

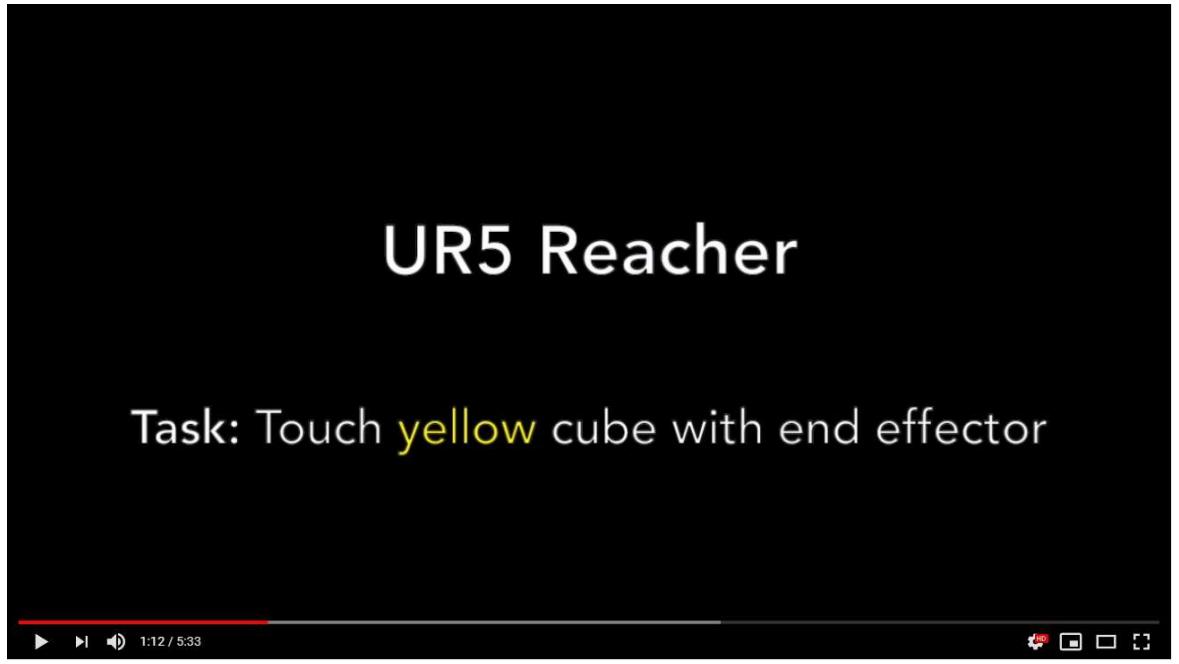
Hierarchical Actor-Critc (HAC)

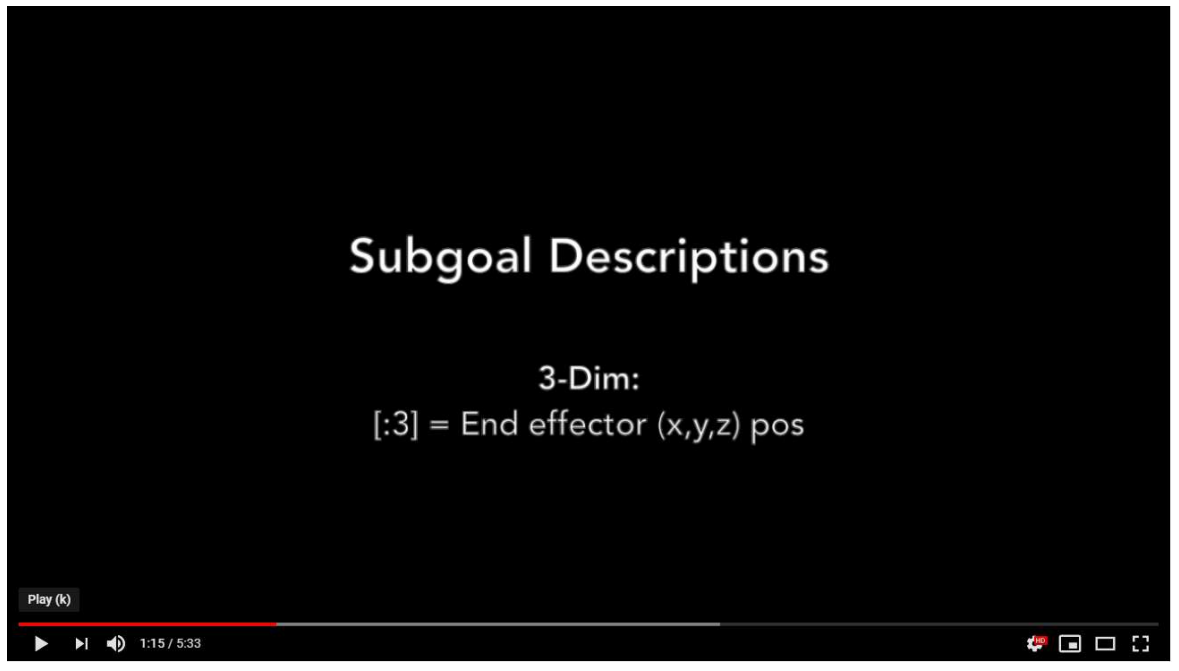
Code to implement the *Hierarchical Actor-Critic (HAC)* algorithm Github HAC (https://github.com/p9i/Hierarchical-Actor-Critc-HAC-)

https://github.com/andrew-j-levy/Hierarchical-Actor-Critc-HAC- (https://github.com/andrew-j-levy/Hierarchical-Actor-Critc-HAC-)

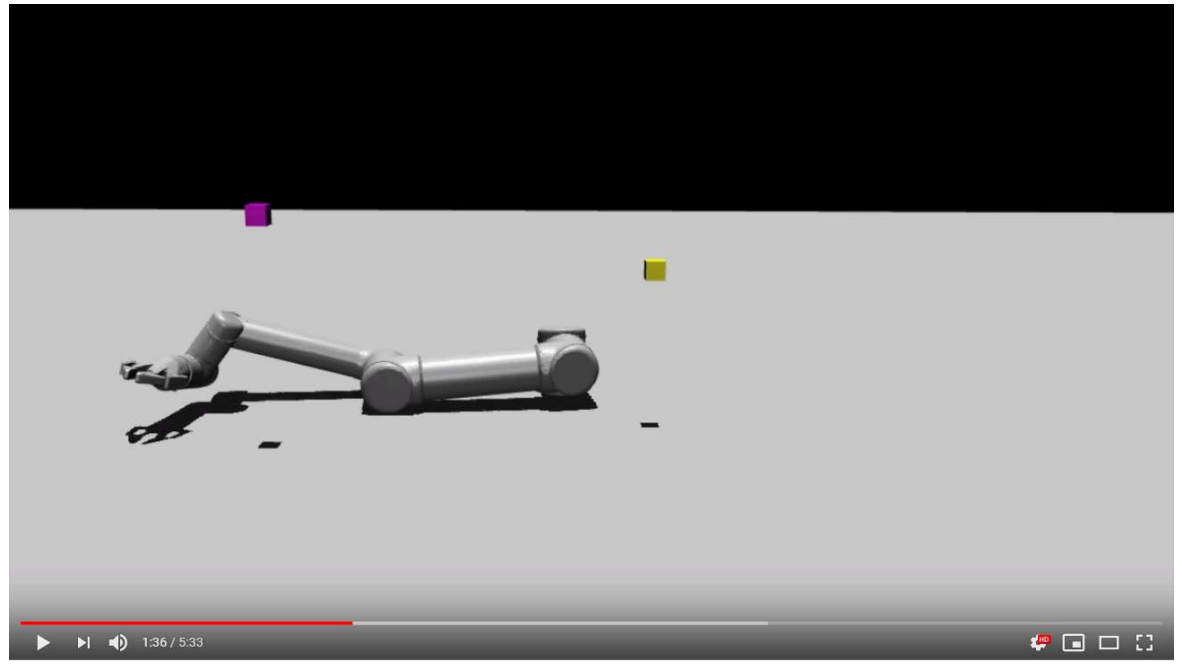
Learning Multi-level Hierarchies with Hindsight (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/Learning-Multi-level-Hierarchies-with-Hindsight.pdf)

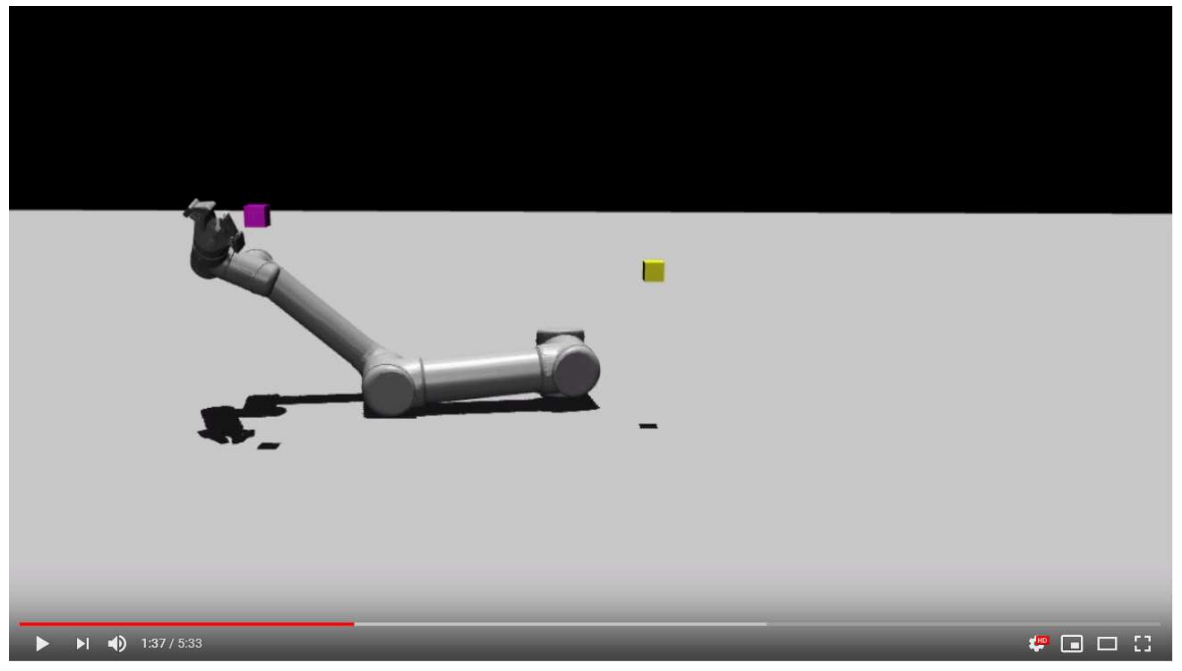
https://openreview.net/pdf?id=ryzECoAcY7 (https://openreview.net/pdf?id=ryzECoAcY7)

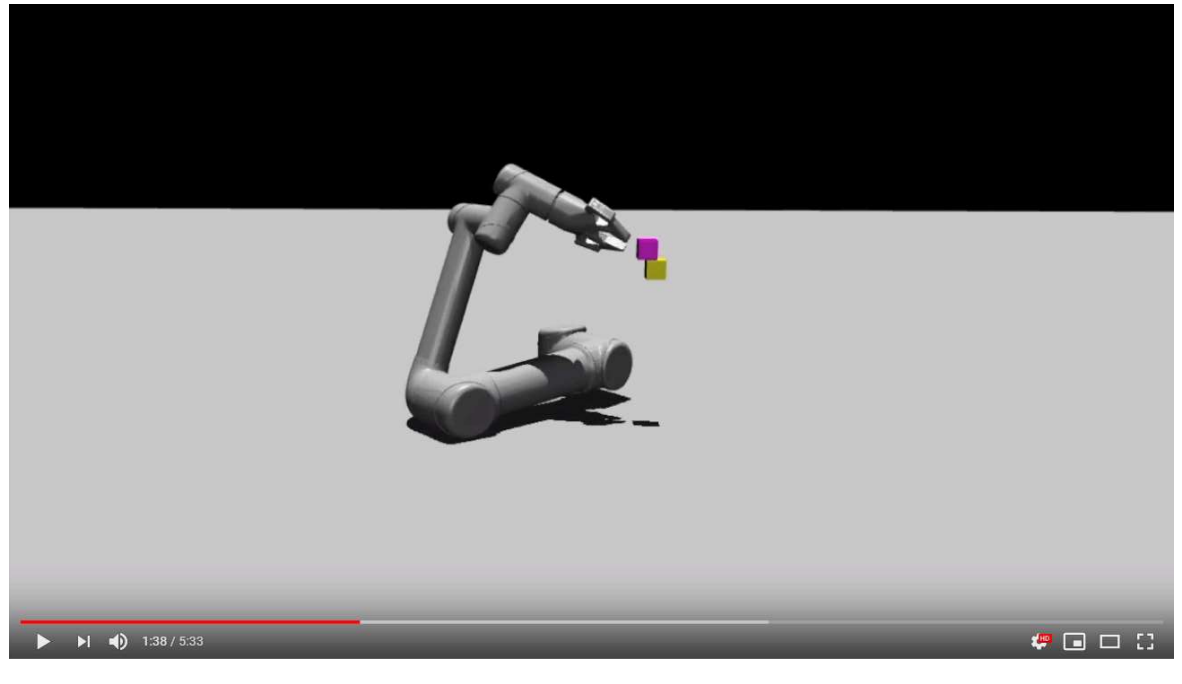

UR5 Reacher
Task: Touch yellow cube with end effector
1:12 / 5:33

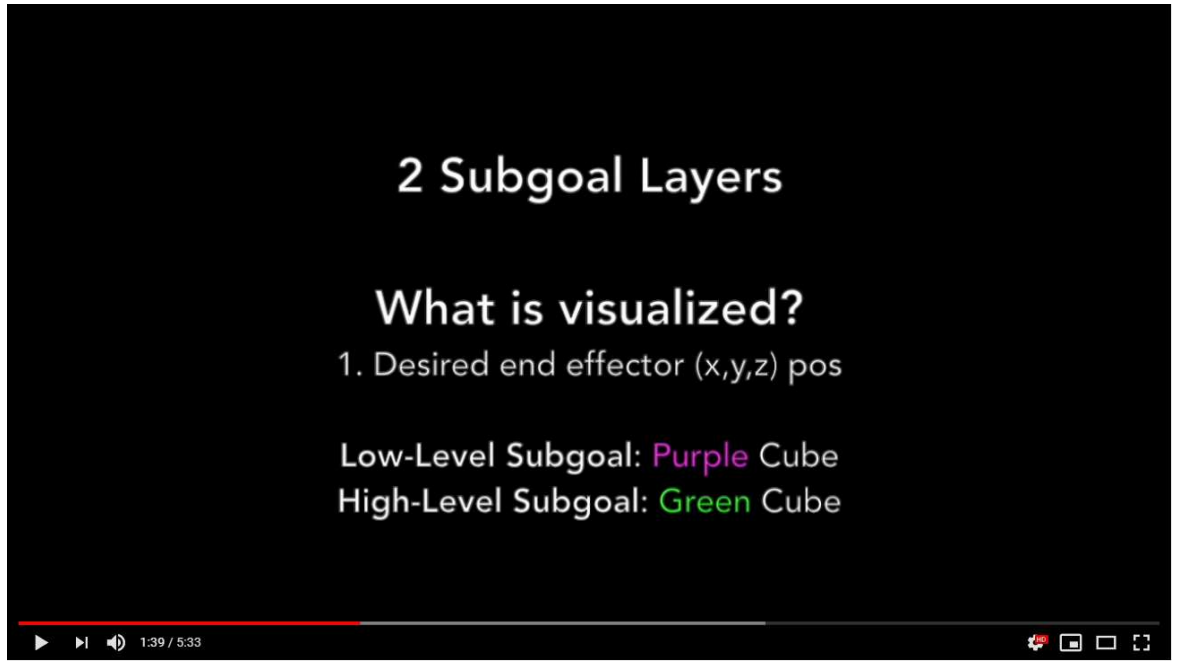

Subgoal Descriptions
3-Dim:
[:3] = End effector (x,y,z) pos
Play (k)
1:15 / 5:33

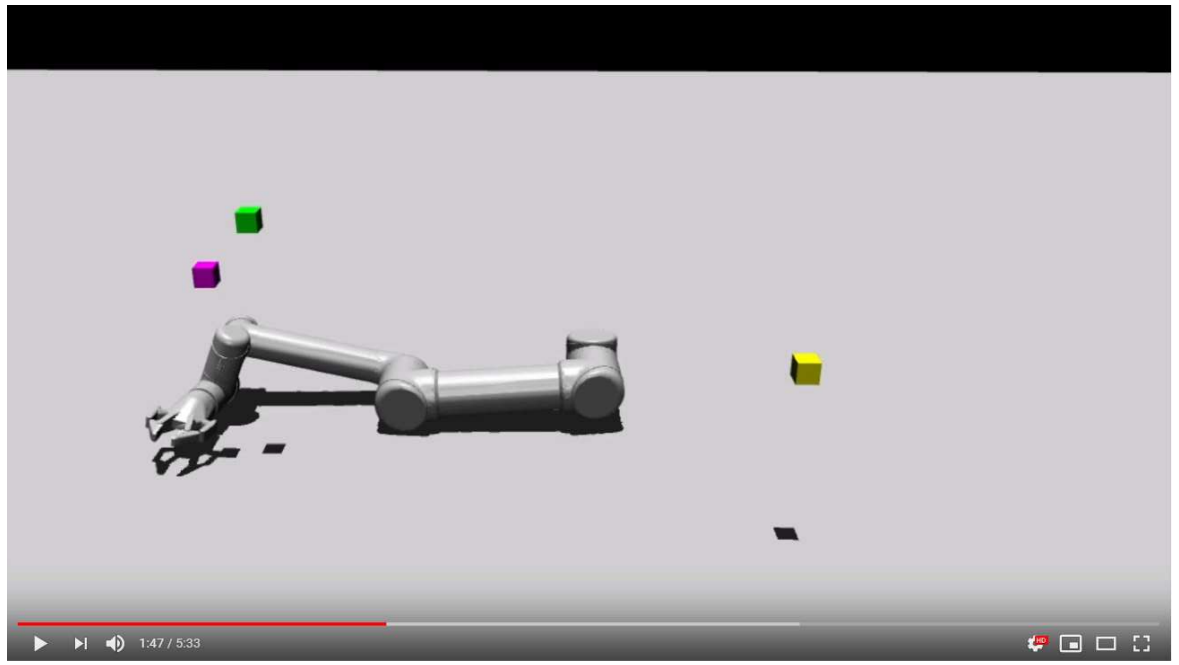

1 Subgoal Layer
What is visualized?
1. Desired end effector (x,y,z) pos
Low-Level Subgoal: Purple Cube
Play (k)
1:19 / 5:33

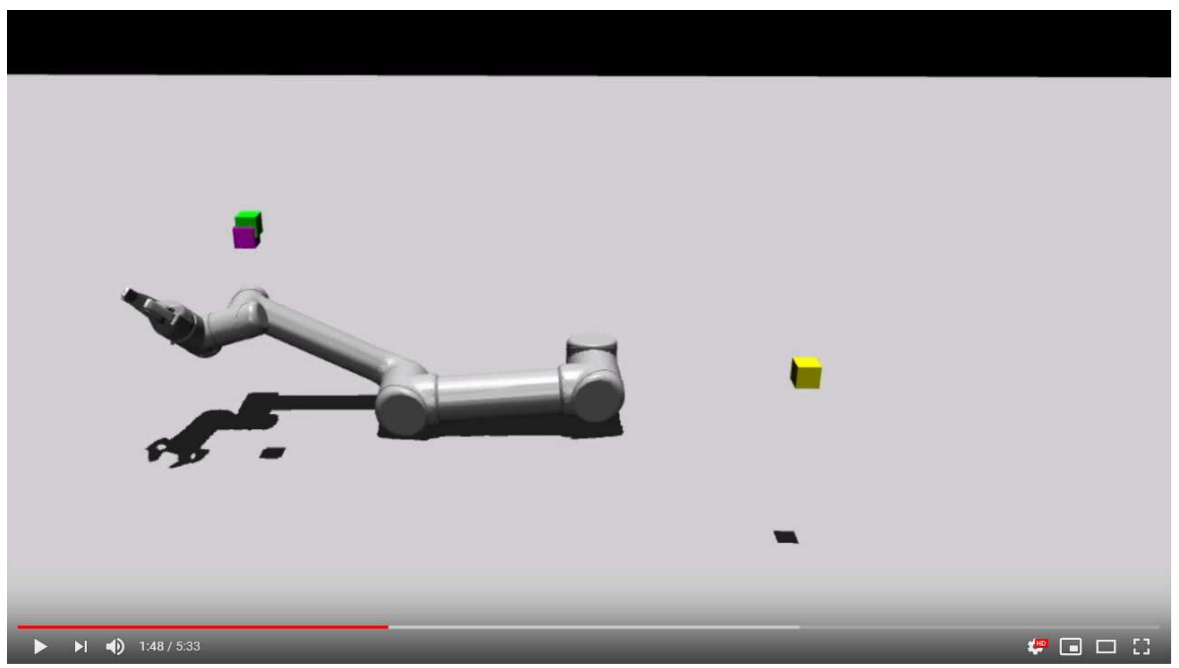

1:36 / 5:33

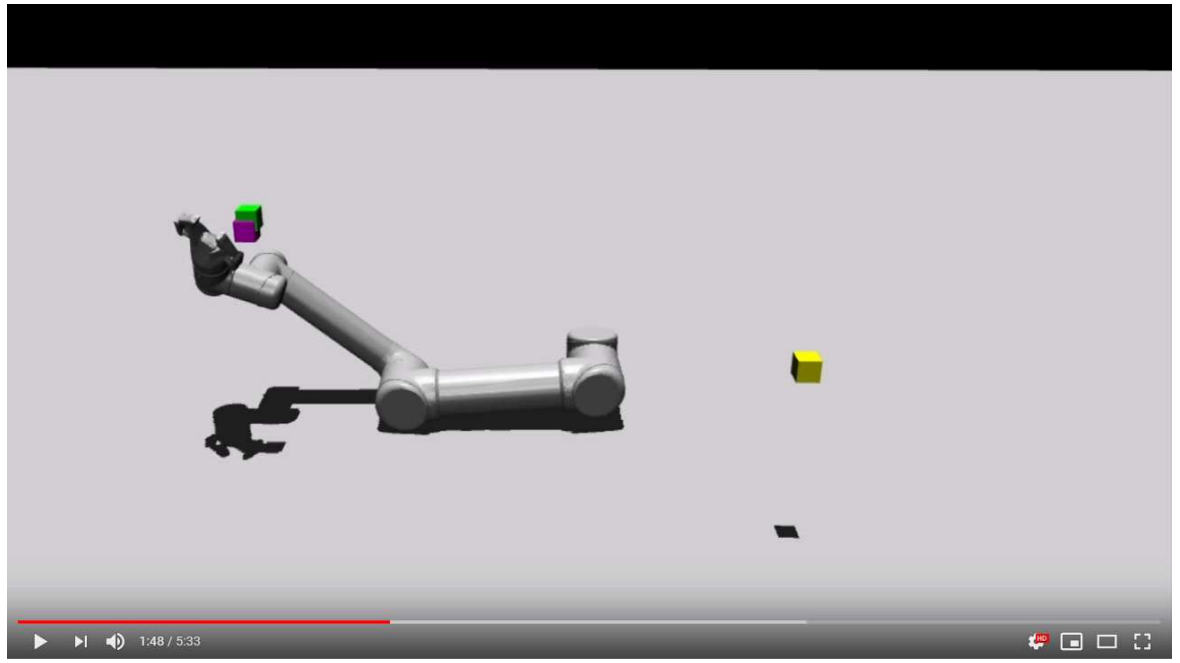

1:37 / 5:33

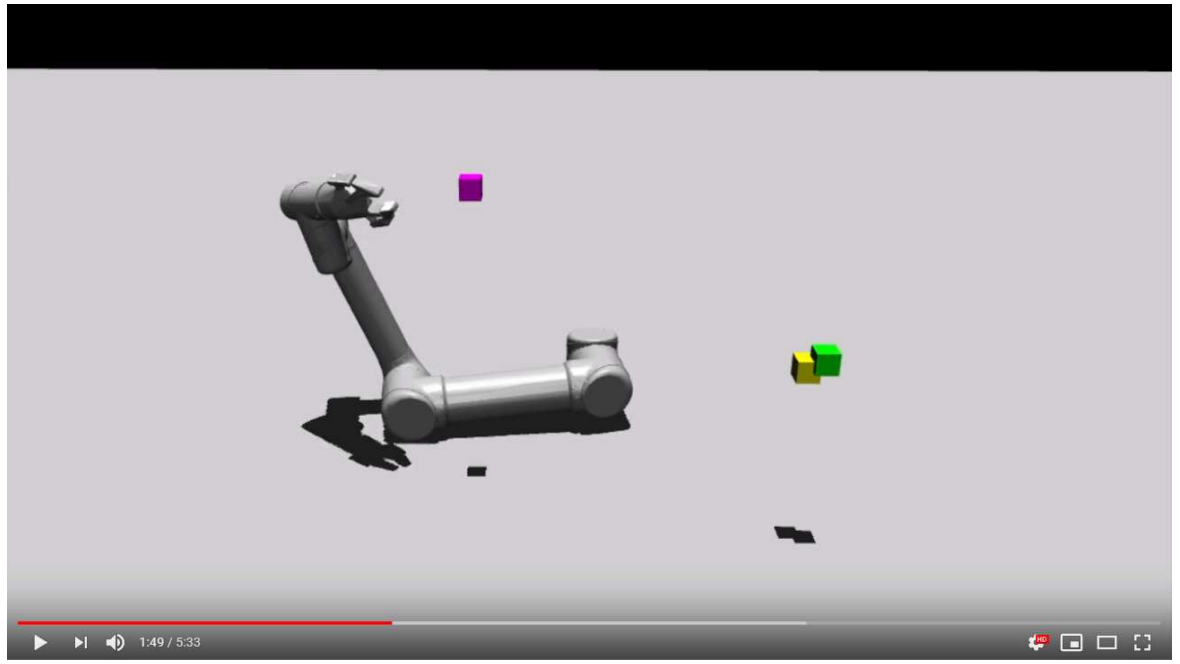

1:38 / 5:33

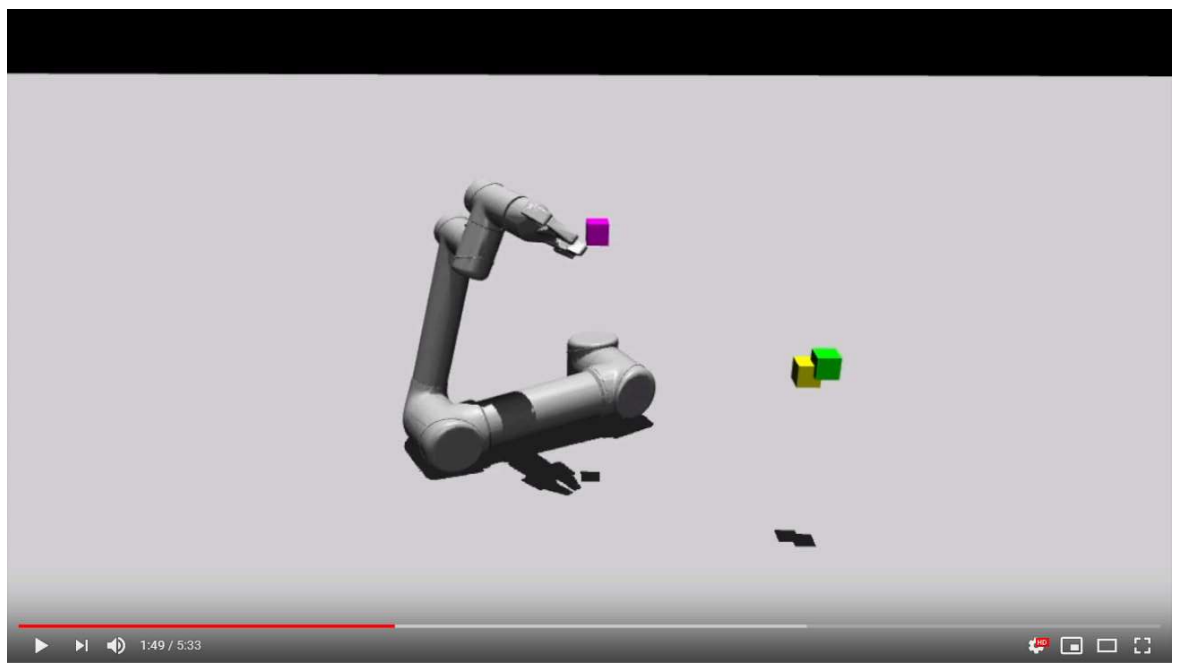

2 Subgoal Layers
What is visualized?
1. Desired end effector (x,y,z) pos
Low-Level Subgoal: Purple Cube
High-Level Subgoal: Green Cube
1:39 / 5:33

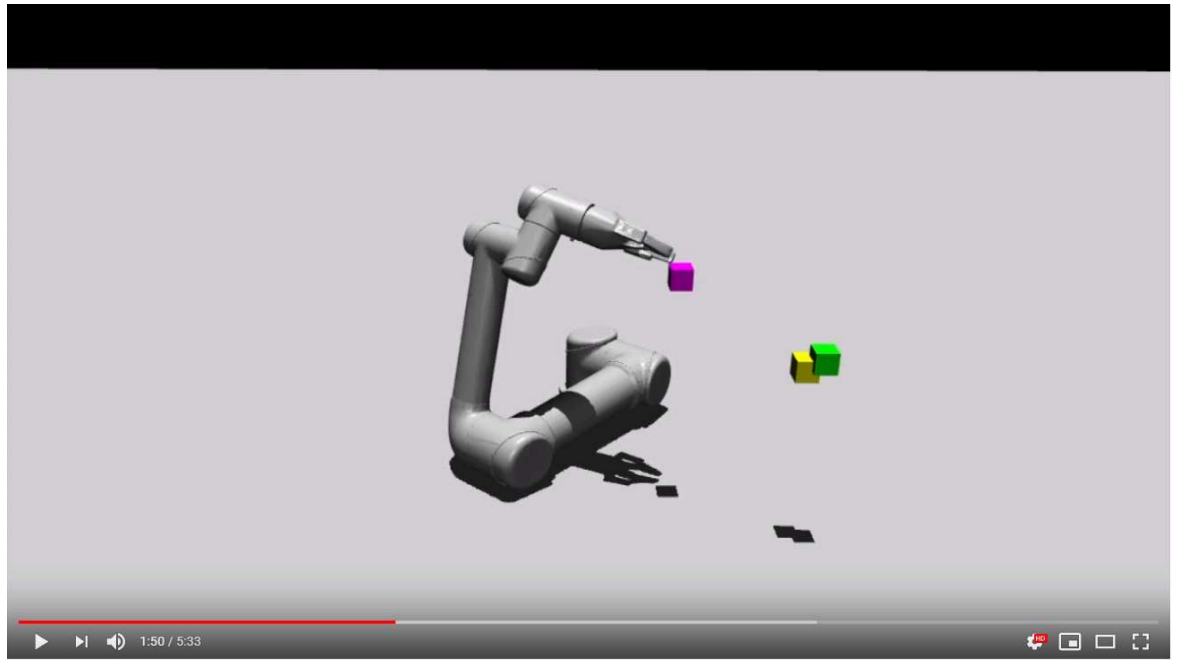

1:47 / 5:33

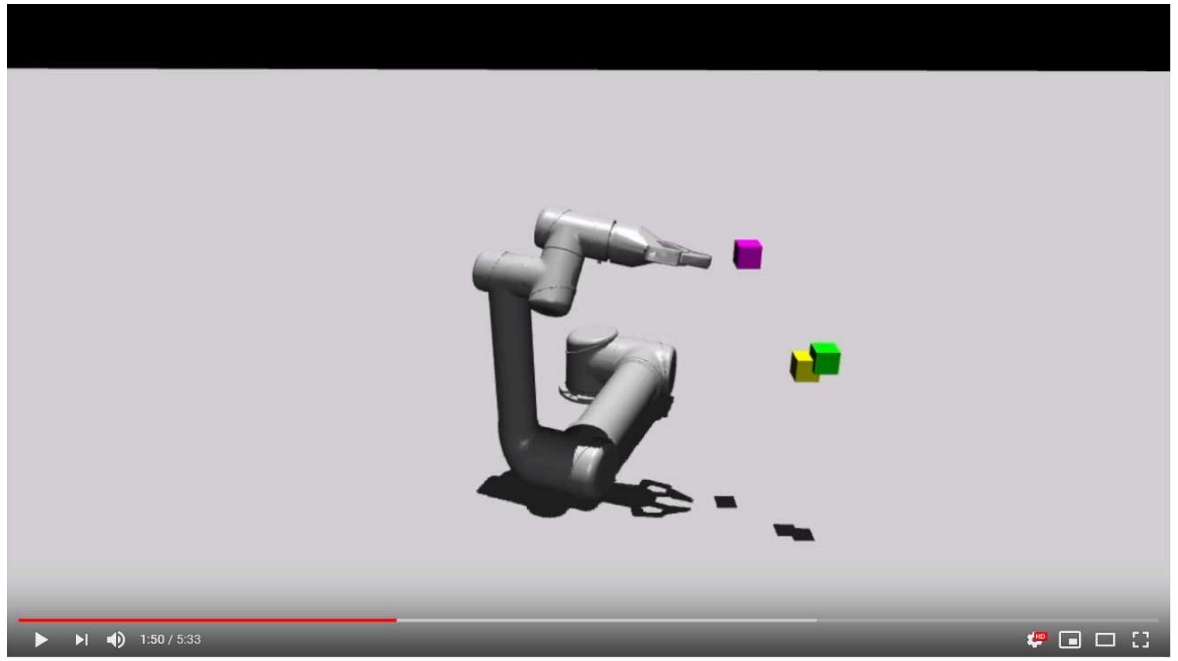

1:48 / 5:33

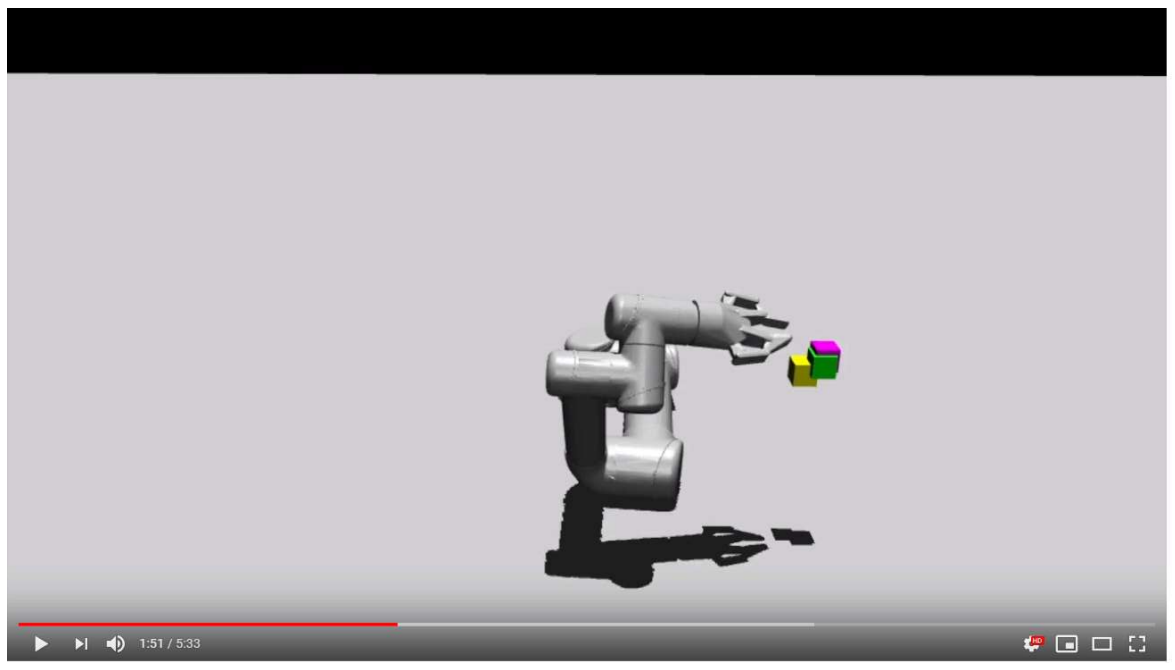

1:48 / 5:33


1:49 / 5:33


1:49 / 5:33







Hierarchical Actor-Critic Video Presentation

edit (https://blogs.cuit.columbia.edu/zp2130/wp-admin/post.php?post=4729&action=edit)

Author: Z Pei (https://blogs.cuit.columbia.edu/zp2130/author/zp2130/) on February 24, 2019

Categories: Actor-Critic Algorithms (https://blogs.cuit.columbia.edu/zp2130/category/actor-critic-algorithms/), AI (https://blogs.cuit.columbia.edu/zp2130/category/ai/), Algorithm (https://blogs.cuit.columbia.edu/zp2130/category/algorithm/), Machine Learning (https://blogs.cuit.columbia.edu/zp2130/category/machine-learning/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/category/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/category/rl/)

Tags: actor-critic (https://blogs.cuit.columbia.edu/zp2130/tag/actor-critic/), AI (https://blogs.cuit.columbia.edu/zp2130/tag/ai/), Algorithm (https://blogs.cuit.columbia.edu/zp2130/tag/algorithm/), Hierarchical RL (https://blogs.cuit.columbia.edu/zp2130/tag/hierarchical-rl/), Machine Learning (https://blogs.cuit.columbia.edu/zp2130/tag/machine-learning/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/tag/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/tag/rl/)

Other posts

RL Other Useful Reference (https://blogs.cuit.columbia.edu/zp2130/rl_other_useful_reference/) «» Hierarchical Policy Gradient Algorithms (https://blogs.cuit.columbia.edu/zp2130/hierarchical_policy_gradient_algorithms/)

Last posts

- Symbolic Netlist to Innovus-friendly Netlist (https://blogs.cuit.columbia.edu/zp2130/symbolic_netlist_to_innovus-friendly_netlist/)
- Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms (https://blogs.cuit.columbia.edu/zp2130/finite-sample_convergence_rates_for_q-learning_and_indirect_algorithms/)
- Solving H-horizon, Stationary Markov Decision Problems In Time Proportional To Log(H) (https://blogs.cuit.columbia.edu/zp2130/paul_tseng_1990/)
- Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear (Sometimes Sublinear) Run Time (https://blogs.cuit.columbia.edu/zp2130/randomized_linear_programming_solves_the_discounted_markov_decision_problem_in_nearly-linear_sometimes_sublinear_run_time/)
- KL Divergence (https://blogs.cuit.columbia.edu/zp2130/kl_divergence/)
- The Asymptotic Convergence-Rate of Q-learning (https://blogs.cuit.columbia.edu/zp2130/the_asymptotic_convergence-rate_of_q-learning/)
- Hierarchical Apprenticeship Learning, with Application to Quadruped Locomotion (https://blogs.cuit.columbia.edu/zp2130/hierarchical_apprenticeship_learning_with_application_to_quadruped_locomotion/)
- Policy Gradient Methods (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_methods/)
- Actor-Critic Algorithms for Hierarchical Markov Decision Processes (https://blogs.cuit.columbia.edu/zp2130/actor-critic_algorithms_for_hierarchical_markov_decision_processes/)
- Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation (https://blogs.cuit.columbia.edu/zp2130/hierarchical_deep_reinforcement_learning_integrating_temporal_abstraction_and_intrinsic_motivation/)