Motion Planner Augmented Action Spaces for Reinforcement Learning

Jun Yamada*, Gautam Salhotra†, Youngwoon Lee*, Max Pflueger†, Karl Pertsch*,
Peter Englert†, Gaurav S. Sukhatme†, Joseph J. Lim*

*Cognitive Learning for Vision and Robotics Lab
†Robotic Embedded Systems Laboratory
Department of Computer Science
University of Southern California, Los Angeles

***Abstract*—Deep reinforcement learning (RL) agents are able to learn contact-rich manipulation tasks by maximizing a reward signal, but require large amounts of experience, especially in environments with many obstacles that complicate efficient exploration. In contrast, motion planners use explicit models of agent and environment to plan collision-free paths to faraway goals, but suffer from model-inaccuracies in contact-rich tasks. In this work, we propose to combine the benefits of both approaches by formulating a novel action space for continuous robotic control tasks that equips RL agents with long-horizon planning capabilities. Using this action space we train model-free RL agents that *learn* to decide when to make use of the motion planner purely from reward signals. On multiple simulated object manipulation tasks, we show that our motion-planner augmented action space increases learning efficiency and facilitates exploration in environments with many obstacles.**

I. Introduction

In recent years, deep reinforcement learning (RL) has shown promising results in continuous control problems [24, 7, 15, 18]. Driven by rewards, robotic agents were able to learn tasks such as grasping [17, 9, 5, 16] and peg insertion [5]. However, the prior works mostly have been demonstrated in controlled and clean environments, whereas a real-world environment is often cluttered with many objects and obstacles unrelated to the task, which makes exploration by RL agents challenging and data inefficient. This problem is exacerbated in situations with sparse rewards where feedback is scarce and considerable exploration is required *before* a learning signal is received.

Motion planning (MP) is an alternative to solve robot tasks in complex environments, which has been widely studied in the robotics literature [1, 12, 10, 3]. These methods find a collision-free path between two robot states by searching over a graph of feasible transitions in the robot's configuration space, providing asymptotic completeness and optimality guarantees. In contrast to model-free RL approaches, MP algorithms require explicit models of agent and environment and struggle on tasks that involve rich interactions with objects or other agents, where it is challenging to obtain accurate models [20].

In this work, we propose to combine the strengths of both motion planning and RL by augmenting the action space of an RL agent with the capabilities of a motion planner. A model-free RL method [7] can be used to learn an RL policy that controls the robot by predicting state changes in joint space [6]. While prior works with joint-space control [19] constrain the action space to be small to prevent collision and reduce controller error, we propose to allow a much wider action range with the help of a motion planner that plans collision-free paths to faraway goals.

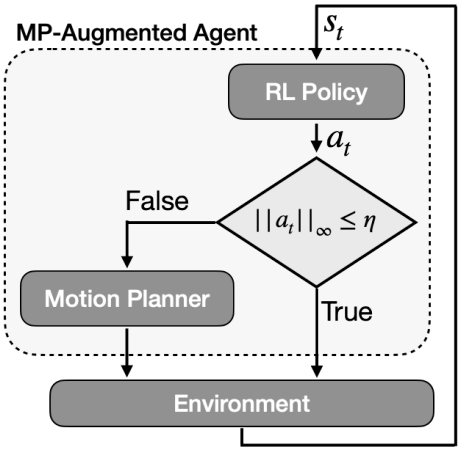


Fig. 1: Our framework composed of a motion planner and RL policy. Given a state s_t, the RL policy outputs an action a_t, which represents changes in the joint state. If the action is large (i.e., $||a_t||_\infty > \eta$ where η is a threshold), thus, likely to have a collision, the action a_t is realized using the motion planner to achieve collision-free execution. Otherwise, the action is simply fed to the environment.

Our framework has multiple benefits: (1) the proposed framework can add motion planning capabilities to *any* RL agent as we do not require changes to the agent's architecture or training algorithm. (2) The agent itself can decide when to use which control approach: by predicting small target displacements, the agent can fall back to conventional action space, for example, when complex object manipulation is required and motion planner solutions would be inaccurate. In other situations, the agent can predict large displacements, using the motion planner, for efficient exploration. Crucially,

this decision does not need to be pre-determined, but can instead be learned by the agent through maximization of the task reward. This allows the agent to adapt the use of the motion planner to its own changing capabilities over the course of training.

We evaluate our framework on simulated robot object manipulation tasks in the presence of multiple obstacles and sparse rewards that pose exploration challenges for RL agents. We show that our motion planner augmented agent can quickly learn to solve the tasks while model-free RL agents suffer from slow convergence.

In summary, the contributions of this paper are as follows:

- We propose a flexible framework for augmenting an RL agent with a motion planner to improve learning-efficiency for robot control.
- We validate the benefits of our method with motion planning augmented action spaces over model-free RL on simulated object manipulation tasks.

II. Related Work

There are several works that combine motion planners and RL. Angelov et al. [2] proposes to integrate motion planner options into hierarchical RL (HRL) to solve a compositional task given demonstrations. Stulp and Schaal [21] proposes a HRL framework in which low-level policies are dynamic motion primitives [8]. Lee et al. [13] uses a motion planner to move a robot arm to a target region and, when close to an object, switch to model-free RL for learning contact-rich manipulation. In contrast, our method does not require a library of primitive skills or demonstrations. It also does not require a pre-trained perception system to switch from motion planner to model-free RL. Instead, we propose to *learn* how to balance the use of the motion planner and model-free RL to solve a task.

III. Motion Planner Augmented RL

A. Preliminaries

A Markov decision process (MDP) is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho, \gamma)$ consisting of states, actions, reward $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, transition function $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, initial state distribution ρ, and discount factor $\gamma \in [0, 1]$. The agent's action distribution at time step t is represented by a policy $\pi_\theta(a_t|s_t)$ with state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$, where θ denotes the parameters of the policy. For learning continuous control policies we define the action space of the policy as the displacement in the agent's joint space $a_t = \Delta q_t$, where q_t represents a robot joint state.

Motion planning methods are mainly categorized into sampling-based motion planning and trajectory optimization. In this work, we use kinematic sampling-based motion planners. A motion planner computes a path from a given start state q_t to a given goal state g. We denote such a motion planning query as $\tau_{1:H} = \text{MP}(q_t, g)$ where the resulting path is represented as a dense sequence of H joint configurations $\tau_{1:H} = (q_{t+1}, \ldots, q_{t+H})$.

Algorithm 1: Motion Planner Augmented RL

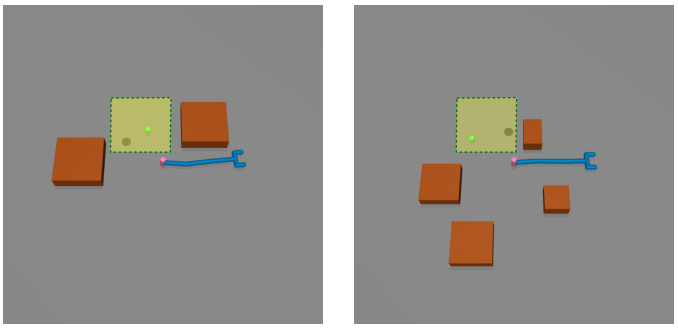
Inputs:
Motion planner MP
Transition model P and reward R
Augmented transition model $\tilde{P}$ and reward $\tilde{R}$
Action limit η
Initial state distribution ρ
Policy π_ϕ

for $i = 1, \ldots, N$ **do**
  $s_0 \sim \rho(s_0), t \leftarrow 0$
  while *episode not done* **do**
    $a_t \sim \pi_\phi(a_t|s_t)$
    if $||a_t||_\infty > \eta$ **then**
      # MP execution
      $g \leftarrow q_t + a_t$
      $H, \tau_{1:H} \leftarrow \text{MP}(q_t, g)$
      $s_{t+H} \leftarrow \tilde{P}(s_t, \tau_{1:H})$
      $r_t \leftarrow \tilde{R}(s_t, \tau_{1:H})$
    else
      # Primitive action execution
      $H \leftarrow 1$
      $s_{t+1} \leftarrow P(s_t, a_t)$
      $r_t \leftarrow R(s_t, a_t)$
    end
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+H})\}$
    $t \leftarrow t + H$
    update π_ϕ using model-free RL
  end
end

B. Motion Planner Augmented Reinforcement Learning

To utilize motion planning within the RL framework, we augment the continuous control policy's action space with the ability to make calls to a motion planner. The decision whether to call the motion planner or directly execute the predicted action is based on its magnitude $||a_t||_\infty$, i.e. the size of the predicted displacement, as illustrated in Fig. 1. If the target state difference is small, the action is directly executed using a feedback controller, as is common practice in model-free RL. For target state differences $||a_t||_\infty > \eta$ that pass a threshold hyperparameter η, a motion planner is called that computes a path $\tau_{1:H} = (q_{t+1}, \ldots, q_{t+H})$ with horizon H towards the goal defined as $g = q_t + a_t$, which is then executed on the robot with a feedback controller. Crucially, this design gives the policy the freedom to chose whether to call the motion planner or directly execute actions in the environment by predicting large or small target state differences respectively. The complete RL update loop with a motion-planner augmented agent is described in Algorithm 1.

Formally, we extend the conventional RL formalism from Section III-A to accommodate augmentation with motion planners by assuming a generalized transition model on the augmented action space $s_{t+H} = \tilde{P}(s_t, \tau_{1:H})$ that also includes the single-action case as a one-step trajectory. Further, we as-



2-obstacle Pusher 4-obstacle Pusher

Fig. 2: *2-* and *4-obstacle Pusher* tasks. Target region for sampling object (green ball) and goal (black circle) location marked in the yellow region. The *4-obstacle Reacher* is identical to the *4-obstacle Pusher* environment without the object.

sume an augmented reward function $\tilde{R}(s_t, \tau_{1:H})$ that computes the sum of rewards along the path $\tau_{1:H}$. This augmented action space implicitly provides the robot with an option to call the motion planner by giving faraway goals. This can be seen as a semi-MDP [23] where the duration of an action varies depending on the length of the path H.

C. Motion Planner Implementation Details

Sampling-based motion planning methods only distinguish between configurations that are either valid or in collision with the environment. This property makes it difficult to apply them on manipulations tasks where contacts are required. A crucial step to still being able to use these algorithms on manipulation tasks is to separate the environment into manipulatable and static objects. We adapt the collision checker to ignore collisions between manipulatable objects and the robot. This allows the algorithms to compute paths that manipulate the objects of interests while staying away from static obstacles.

Motion planning is often computationally expensive and RL requires many iterations to learn a policy. In order to prevent slow training in the RL framework, we propose a planning module that, when called by the policy, first interpolates a path between the given start and goal state. If the interpolated path is not collision-free, then a motion planner is called, which can find a path even in the presence of many obstacles.

IV. Experiments

We design our experimental evaluation to answer the following questions: (1) Can a model-free RL agent augmented with motion planner capabilities solve complex manipulation tasks more efficiently than off-policy RL agents? (2) Is the motion-planner-augmented agent better able to explore the environment? To answer these question, we conduct experiments on three environments of increasing difficulty: in the *4-obstacle Reacher* the agent needs to reach a target position randomly sampled within a target region in the presence of four obstacles. In the *2-obstacle Pusher*, and *4-obstacle Pusher* (see Fig. 2) the agent needs to push an object into a goal position in the presence of two and four obstacles respectively. Positions of both object and goal get sampled inside the target region. The state space consists of the agent's joint angles as well as positions of goal, p_{goal}, and object, p_{obj} for the pushing tasks. All of our environments are simulated in the MuJoCo physics engine [25].

For our experiments, we use $\eta = 0.1$ that defines the threshold when the motion planner is called for action execution. The full action range is defined as $a \in [-2, 2]$. We use soft actor-critic (SAC) [7], the state-of-the-art model-free RL algorithm, as base agent for both the baseline and our method. For motion planning we use the RRT-connect planner [11] from the OMPL library [22]. The reward for the reaching task is defined as:

$$R_{\text{reach}} = -\mathbb{1}_{||p_{\text{eef}} - p_{\text{goal}}||_2 \geq 0.04},$$

where p_{eef} is the end-effector position and $\mathbb{1}$ is the indicator function. For the pushing tasks the reward is defined as:

$$R_{\text{push}} = -0.3||p_{\text{eef}} - p_{\text{goal}}||_2 - 0.9||p_{\text{eef}} - p_{\text{obj}}||_2 + 150 \cdot \mathbb{1}_{\text{success}},$$

where $\mathbb{1}_{\text{success}}$ indicates that the object is successfully pushed to the goal.

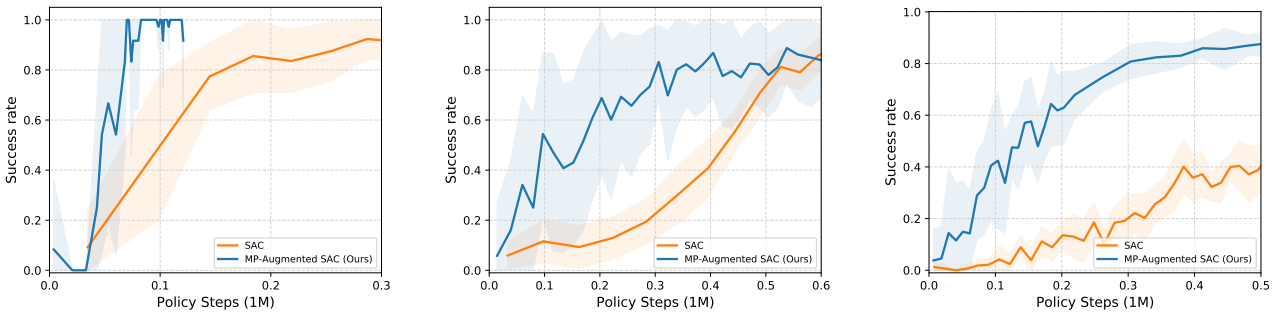
A. Efficient RL with Motion Planner Augmented Action Spaces

We compare the success rate of our method to that of the SAC baseline over the course of training in Fig. 3. We show that the motion-planner augmented policy is able to learn the tasks more efficiently than the baseline. This improvement in learning efficiency is especially large in the most challenging task, *4-obstacle Pusher*, which requires the policy to learn object manipulation in the presence of many obstacles.

We show a qualitative example of the learned behavior of our agent on the *4-obstacle Pusher* task in Fig. 4. By setting wide target states it learns to use the motion planner for the initial segments of the task that require to maneuver the agent towards the target object without colliding with obstacles. The motion planner is able to plan collision-free tasks and successfully reaches the target object. The remainder of the task requires more intricate control for pushing the object to the goal, a task that the motion planner is not well-suited to solve. Instead, the motion planner resorts to conventional, model-free RL by predicting small state difference actions, thereby successfully pushing the object to the goal location. We emphasize that this switching behavior is purely learned from rewards without pre-specified heuristics for when to use which control approach. This shows that the agent can learn to adjust the usage of the motion planner to it's own capabilities and the requirements of the task.

B. Improved Exploration through Motion Planning

To better understand *why* augmenting RL agents with motion planning capability improves training efficiency, we compare the exploration behavior of the motion-planner augmented agent and the SAC agent in Fig. 5. The SAC agent initially explores only in close proximity to its initial position as it struggles to find valid trajectories towards the goal region in between the obstacles. In contrast, the motion-planner



(a) 4-obstacle Reacher (b) 2-obstacle Pusher (b) 4-obstacle Pusher

Fig. 3: Success rates of our motion planner augmented SAC (orange) and baseline SAC (blue) averaged over 4 seeds. Our approach can leverage the motion planner to converges within a smaller number of policy interactions than the baseline. Note that the number of policy interactions is not equivalent to the number of environment steps as the execution of motion planner trajectories involves multiple primitive actions. Both SAC and ours are trained for the same environment steps.

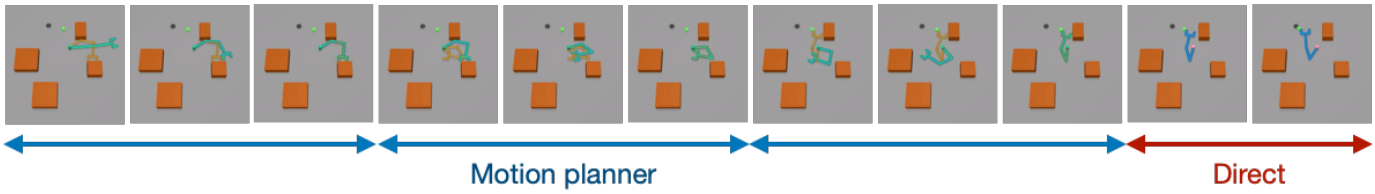


Fig. 4: Successful episode of the motion planner augmented agent on the *4-obstacle Pusher* task. In the initial phases of the episode, that require reaching towards the object, the agent relies on the motion planner and sets wide target state differences (target states overlayed in brown). The motion planner is able to reach the posed target states on collision-free paths. Once the object is reached and more intricate manipulation is required, the agent switches to normal, model-free RL and directly executes actions without using the motion planner. The agent learns this switching behavior directly from rewards without the need to define pre-specified heuristics for when to use which control method.

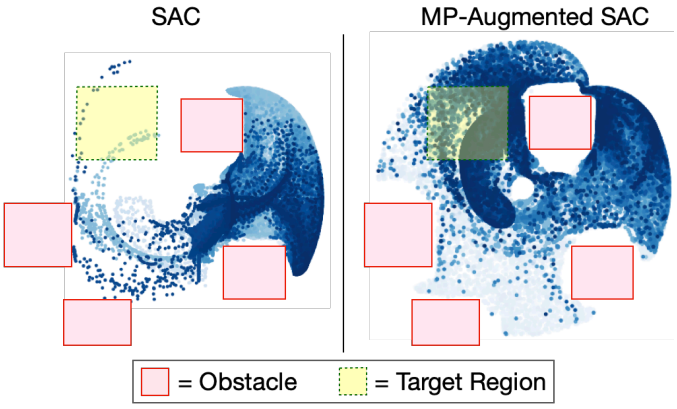


Fig. 5: End-effector positions of SAC (**left**) and motion planner augmented SAC (**right**) after the first 100k training steps in *4-obstacle Pusher*. The usage of the motion planner allows the motion planner augmented agent to explore the environment more widely early on in training.

augmented agent can explore a wide range of target positions by using the motion planner to find collision-free trajectories to faraway goal states. This allows the agent to much quicker learn the task, especially in the presence of many obstacles.

V. Conclusion

In this work, we propose a flexible framework that combines the benefits of motion planning and reinforcement learning for sample-efficient learning of continuous robot control tasks. Specifically, we design a single RL policy with an action space augmented by a motion planner. The policy learns when to use the motion planner and a primitive action directly through reward maximization without the need for pre-specified heuristics. The experimental results show that our approach improves the training efficiency over a conventional, model-free RL baseline, especially in environments that require object manipulation in the presence of many obstacles.

Encouraged by the results presented in this paper, we plan to apply this method to more complex robot manipulation tasks such as peg-in-hole [5] and furniture assembly [14] with realistic robot arms, both in simulation and in the real world. We hypothesize that in these tasks efficient exploration of the environment is a major bottleneck to policy learning and assume that augmenting agents with motion planning capabilities can help to address this problem. We also want to explore more advanced planning techniques [4] that plan paths on constrained configuration spaces that allow to describe more complex behavior.

VI. Acknowledgement

We thank our colleagues from the Cognitive Learning for Vision and Robotics Lab (CLVR) and Robotic Embedded System Lab (RESL) for the valuable discussions that considerably assisted the research.

References

[1] Nancy M Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of the International Conference on Robotics and Automation*, 1996.

[2] Daniel Angelov, Yordan Hristov, Michael Burke, and Subramanian Ramamoorthy. Composing diverse policies for temporally extended tasks. *IEEE Robotics and Automation Letters*, 5(2):2658–2665, 2020.

[3] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[4] Peter Englert, Isabel M. Rayas Fernández, Ragesh K. Ramachandran, and Gaurav S. Sukhatme. Sampling-based motion planning on manifold sequences. arXiv:2006.02027, 2020.

[5] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. SURREAL: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

[6] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the International Conference on Robotics and Automation*, 2017.

[7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[8] A. Ijspeert, J. Nakanishi, P Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, (25):328–373, 2013.

[9] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[10] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[11] James J Kuffner Jr and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2000.

[12] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[13] Michelle A. Lee, Carlos Florensa, Jonathan Tremblay, Nathan Ratliff, Animesh Garg, Fabio Ramos, and Dieter Fox. Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning, 2020.

[14] Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. *arXiv preprint arXiv:1911.07246*, 2019.

[15] Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S. Hu, and Joseph J. Lim. Composing complex skills by learning transition policies. In *Proceedings of International Conference on Learning Representations*, 2019.

[16] Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International Conference on Learning Representations*, 2020.

[17] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. In *International Symposium on Experimental Robotics*, 2016.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[19] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *arXiv preprint arXiv:1906.08880*, 2019.

[20] Mark Moll, Lydia Kavraki, Jan Rosell, et al. Randomized physics-based motion planning for grasping in cluttered and uncertain environments. *IEEE Robotics and Automation Letters*, 3(2):712–719, 2017.

[21] F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *International Conference on Humanoid Robots*, 2011.

[22] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[23] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181 – 211, 1999.

[24] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[25] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.