

---

# Unsupervised Learning of Effective Actions in Robotics

---

Marko Zarić<sup>1</sup> Jakob Hollenstein<sup>1</sup> Justus Piater<sup>1,2</sup> Erwan Renaudo<sup>1</sup>

<sup>1</sup>Department of Computer Science, University of Innsbruck, Innsbruck, Austria

<sup>2</sup>Digital Science Center, University of Innsbruck, Innsbruck, Austria

firstname.lastname@uibk.ac.at

## Abstract

Learning actions that are relevant to decision-making and can be executed effectively is a key problem in autonomous robotics. Current state-of-the-art action representations in robotics lack proper effect-driven learning of the robot’s actions. Although successful in solving manipulation tasks, deep learning methods also lack this ability, in addition to their high cost in terms of memory or training data. In this paper, we propose an unsupervised algorithm to discretize a continuous motion space and generate “action prototypes”, each producing different effects in the environment. After an exploration phase, the algorithm automatically builds a representation of the effects and groups motions into action prototypes, where motions more likely to produce an effect are represented more than those that lead to negligible changes. We evaluate our method on a simulated stair-climbing reinforcement learning task, and the preliminary results show that our effect driven discretization outperforms uniformly and randomly sampled discretizations in convergence speed and maximum reward.

## 1 Introduction

How to autonomously learn effective actions in robotics is a crucial question to enable robots to tackle diverse tasks in realistic environments with low supervision. The importance of action is highlighted in cognitive science, where recent developments propose to redefine cognition as “embodied action” [3]. On the other hand, implicit models of action are recurring in autonomous robotics, necessary for various systems in the robot: symbolic actions represented as operators allow long-term action planning [14, chap 14], continuous actions are encoded as motion primitives or policies in order to be executed in the environment [14, chap 15]. Advanced perception capabilities like affordance detection process sensor data in regards to the actions the robot is able to perform in the environment [13, 20].

However, in robotics, this diversity of action models at several levels of abstraction leads to partial and heterogeneous representations, a problem that is identified by [21]. They propose a tentative definition of action in robotics in order to formalize the design and learning of action representation. Supported by results from other fields of cognitive science, they particularly emphasize the key importance of the concept of *effect* produced by motor behaviors, in order to define an action. They also examine how researchers approach the question of action representations in robotics and highlight several open challenges to be addressed. In this work, we focus on addressing one of them: “Intensifying effect-centricity and effect grounding” (in action representation). More specifically, we are interested in providing discrete actions often referred to as *action symbols* at the decision

level that are *grounded in the actual physical effects produced by the interactions of the robot with its environment*.

For that, we design an algorithm to generate what we call “action prototypes”: each prototype is a set of motion parameters that let the robot produce a specific type of effect. Generating these prototypes involves a) finding out which types of effects are possible in a given environment through exploration, b) building the class of equivalent effects, and c) finding a set of representative motion parameter sets that allows achieving such effects reliably. Our work contributes by formalizing and implementing an effect-centric action space discretization algorithm<sup>1</sup>. Additionally, we create a toy environment in Gazebo with a Gym-Wrapper called “Up The Stairs”<sup>2</sup> with a continuous motion and observation space well suited for action space discretization evaluations. Finally, we perform a comparative evaluation of the proposed effect-centric discretization algorithm and trivial discretization approaches on the “Up The Stairs” environment.

The rest of this paper is organized as follows: Section 2 reviews existing approaches that learn action prototypes, Section 3 describes our multi-step approach relying on unsupervised methods to build effect and action representations. In Section 4, we describe the environment in which we evaluate this method as well as its relevance for decision-making in robotics in a reinforcement learning (RL) problem. Finally, we examine the implications and potential limitations of the method in Section 5.

## 2 Related Work

In robot action learning, a large part of the research effort is dedicated to learning (encoding of) trajectories useful to solve a task, as shown by the several surveys on robot manipulation [4, 7]. Standard representations include motion primitives, as often done in learning by demonstration, and policies learned with reinforcement learning. To the best of our knowledge, the explicit learning of actions based on the effect they produce is not often explicitly addressed [21]. The current state-of-the-art approach to learning motions in robotics is to train an end-to-end deep neural network through reinforcement learning [12, 18, 6]. However, limitations include the number of parameters, training data, and time required for the learning process to converge [8].

A promising approach proposed by [19] uses a hierarchical reinforcement learning approach to learn both a goal abstraction and an agent’s policy for a given task. They show that the goal representation (i.e., the policy effects) learned by their algorithm is necessary to solve tasks in complex continuous environments with sparse rewards. However, they learn to solve a task in a RL setting, which means their representations are tied to a task and biased by the reward function.

The dependency on a reward function is lifted when explicitly considering the effects: one can find effect-based action learning in the affordance learning literature in robotics. The focus is on learning the relationship between environmental features, robot behavior, and their corresponding effects [13]. [2] use a Bayesian representation of the causal relation between the environment, the robot actions, and the effects. They start with pre-coded elementary actions and handcrafted effect detectors and can find affordances based on collected data but do not let the robot discover effects. [1] propose an affordance equivalence operator based on the produced effect. Using a Bayesian Network, they model relations between object, action, and additionally, the actor from experimental data. These models allow to find equivalent elements, especially equivalent action to obtain the same effect, but do not modify the set of available actions.

A step towards learning a relevant effect representation is made by [17] in the context of social affordances. Despite using pre-coded actions, they compute effect codes based on the continuous variations of the features in effect space. This representation produces effect equivalence classes based on the mean and standard deviation of effects produced by one action. No supervision is required, and effect classes are thus autonomously discov-

---

<sup>1</sup><https://github.com/marko-zaric/action-prototype-gen.git>

<sup>2</sup><https://github.com/marko-zaric/up-the-stairs.git>

---

**Algorithm 1** Motion sampling

---

**Require:**  $\mathcal{O} = \times_{j=1}^n \mathcal{O}_j$  and  $\mathcal{M} = \times_{j=1}^n \mathcal{M}_j$      $\triangleright$  Motion space  $\mathcal{M}$  and Observation space  $\mathcal{O}$   
**Return:**  $\Omega$      $\triangleright$  Collection of (motion, effect) tuples  
 $i \leftarrow 0$   
**while**  $N \geq i$  **do**     $\triangleright$  Reset environment at each sample  
  Initialize  $s$   
  Sample  $m_i \sim \mathcal{U}(\mathcal{M})$   
  Perform motion  $m_i$  and observe  $s'$   
   $e_i = s' - s$   
  Store  $(m_i, e_i)$  to  $\Omega$   
   $i \leftarrow i + 1$   
**end while**

---

ered. The limitation here is that actions being hard-coded, some have multiple outcomes, contrary to the idea that the effect defines the action.

### 3 Methods

Solving the underlying problem for continuous systems is more challenging due to the infinite number of possible actions requiring parametric functions to describe action distributions. The primary aim of action space discretization is to maintain the simplicity of discrete actions in continuous control problems [15]. To tackle these tasks in a sample efficient manner, we consider a developmental scenario where the robot can explore interacting with the environment. The agent can issue commands in continuous motion space, and the goal is to find discrete action prototypes in a sample-efficient way where each prototype performs a reliable effect in the environment. The exploration phase is divided into three main stages: motion sampling, effect region clustering, and action prototype generation. All three stages operate unsupervised, resulting in ready-to-use action prototypes for a downstream discrete RL algorithm.

#### 3.1 Motion sampling

The initial setting for this method is a robot environment with a continuous state space  $\mathcal{O}$  defined as  $\mathcal{O} = \times_{i=1}^n \mathcal{O}_i$  where each  $\mathcal{O}_i$  is a bounded interval for the  $i$ th feature value ( $\times$  denotes the Cartesian Product). We consider a continuous motion space  $\mathcal{M} = \times_{i=1}^n \mathcal{M}_i$  where  $\mathcal{M}_i$  takes values in a bounded interval. These are the motor controls in robot actuators with their respective minimal and maximal value. We define an effect  $e_t$  at time step  $t$  as follows

$$e_t = s_t - s_{t-1} \quad \forall t : s_t \in \mathcal{O} \quad (1)$$

where  $\mathcal{O}$  is an observation space and  $s_t$  and  $s_{t-1}$  are two consecutive states linked by motion  $m_{t-1}$ . At this stage, the robot samples a random motion  $m_t$  uniformly from motion space  $\mathcal{M}$  at each step and performs it always starting from the environment's initial position. When performing a motion, our method focuses on the effect it produces in the environment and stores the resulting  $(m_{t-1}, e_t)$  tuples in  $\Omega$ . Pseudocode for this stage is given in Algorithm 1. Since the environment is reset after every collected sample, we drop the time index  $t$ .

#### 3.2 Effect region clustering

Effect region clustering is achieved by grouping the effects resulting from the sampled motions to generate effect classes  $\mathcal{C}_k$ , as described by Algorithm 2. A different method for grouping the samples into effect categories is selected depending on the specified number of relevant effect dimensions. If only one effect dimension is interesting for the task, simple histogram binning groups the samples into the respective categories. Otherwise, we opt for the K-Means algorithm ([9]) to cluster effects in multidimensional space. In order to find the best number of classes to represent the data, we perform multiple iterations of K-Means while increasing the number of generated centroids. The iteration that produces

---

**Algorithm 2** Effect region clustering

---

**Require:**  $N$  motion-effect tuples  $\Omega$ ,  $\psi$  max clusters**Return:**  $\mathcal{C}_1, \dots, \mathcal{C}_N$ ▷  $N$  distinct Effect Regions**if**  $\dim(e) > 1$  **then** $\theta \leftarrow 0$ 

▷ maximal silhouette score

 $\phi \leftarrow 2$ 

▷ best number of clusters

**for each**  $i \in [2, \dots, \psi]$  **do** $s \leftarrow$  silhouette score of KMEANS( $\{e \in \Omega\}$ , clusters =  $i$ )**if**  $\theta < s$  **then** $\theta \leftarrow s$  $\phi \leftarrow i$ **end if****end for** $\{\mathcal{C}_k : k \in [1, \dots, \phi]\} \leftarrow$  KMEANS( $\{e \in \Omega\}$ , clusters =  $\phi$ )**else**borders, bins  $\leftarrow$  Histogram( $\{e \in \Omega\}$ )Create  $\mathcal{C}_k$  for non-empty bins**for each**  $(e, m) \in \Omega$  **do**Append  $(e, m)$  to  $\mathcal{C}_k$  where  $e \in$  borders( $\mathcal{C}_k$ )**end for****end if**

---

clusters with the highest silhouette score are considered to be the current best representation of possible effects and used as  $\mathcal{C}_k$ . In each version, labels are amended for each sample generated at the previous stage, resulting in  $(m_{t-1}, e_t, k)$  tuples.

### 3.3 Action prototype generation

Each class  $\mathcal{C}_k$  constitutes a collection of similar effects, each with a respective motion that caused the effect. This one-to-one relationship allows us to group the sampled motions by their associated effect label  $e_k$ . We call this collection of motions “action”  $\mathcal{A}_k$ : all the motions that achieve the same class of effect.

In order to achieve  $e_k$ , any motion from action  $\mathcal{A}_k$  can be performed. However, instead of maintaining all these individual motions, we want to find a small number of  $p_k \in \mathcal{M}$  for each effect class  $\mathcal{C}_k$ . All  $p_k$  are action prototypes available at the discrete decision-making level of the robot.

We selected the RGNG (Robust growing neural gas) algorithm by [10] to generate the action prototypes. This algorithm utilizes a combination of topological learning and outlier resilience, which stays true to the underlying sample topology of the input data.

RGNG needs a maximally allowed number of nodes on initialization. We consider an effect-driven approach to calculate this number based on the underlying effect samples of class  $\mathcal{C}_k$ . For each class  $k$ , we determine the mean  $\mu_k^e$  and the standard deviation  $\sigma_k^e$  in effect space. After normalizing the mean and standard deviation across all classes, we calculate the number of prototypes  $\xi_k$  for each class according to Equation (2).

$$\xi_k = \left\lceil \frac{(1 - \text{cv}_k) \cdot \text{std}_k}{\min_k((1 - \text{cv}_k) \cdot \text{std}_k)} \right\rceil \quad (2)$$

$$\text{cv}_k = \frac{\sigma_k^e}{\mu_k^e} \quad (3)$$

where  $\text{cv}_k$  is the coefficient of variation (defined in Equation (3)). This is a statistical measure of dispersion relative to the mean. A high coefficient of variation means the action is less robust; therefore, the first term reduces the number of prototypes for unreliable actions. The second term reduces the number of prototypes for potentially reliable actions with little variability to avoid motion overlap. The division by the minimal value ensures

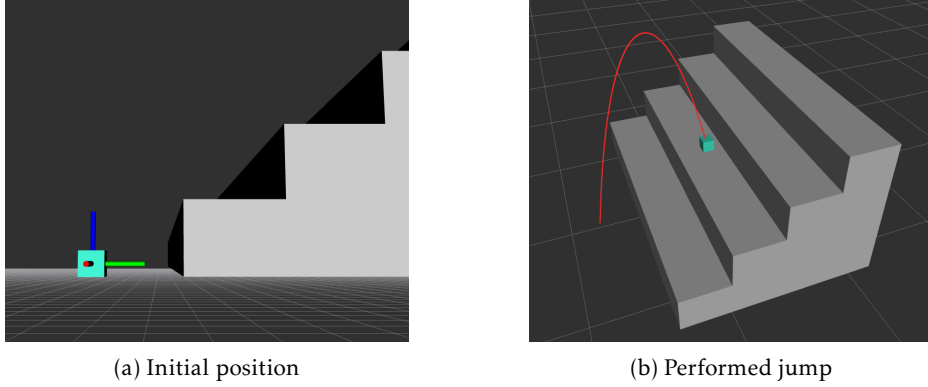


Figure 1: The simulated environment: (a) the robot (the turquoise cube) starts facing a stairway consisting of 4 steps in its initial position. It can apply a force in the y-z plane to its center of mass (red: x, green: y, blue: z) (b) the robot after performing a motion, in its final position. The red line is the performed trajectory.

that each encountered class gets at least one prototype. Finally, we run either RGNG or calculate the motion space mean  $\mu_k^m$  to generate the action prototypes  $p_k$  for each class  $\mathcal{C}_k$

$$p_k = \begin{cases} \mu_k^m & \text{if } \xi_k == 1 \\ \text{RGNG}(\xi_k) & \text{if } \xi_k > 1. \end{cases} \quad (4)$$

## 4 Results

We evaluate the algorithm in a simulated environment where a simple robot can move by “jumping”, i.e., applying a force to its center of mass. Figure 1 illustrates the setup. The “jumping” command is

$$m = \{\alpha, \mu\}$$

where  $\alpha = (0, \alpha, 0)$  is the direction vector of the force applied at the robot center of mass, and  $\mu$  is its amplitude. The robot receives

$$s = \{x, y, z, qx, qy, qz, qw, d_{obs}\}$$

as features, where  $x, y, z$  encodes its position in the world frame,  $qx, qy, qz, qw$  is the quaternion representing its orientation and  $d_{obs}$  the distance to the next obstacle on the x-axis.

Figure 2 shows the k-means clustering result on the collected data in effect space visualized in motion space. The method is able to find meaningful classes on the y,z space. Other dimensions of the effect space are random in this setting since the control action is applied in the y-z plane. Classes  $\mathcal{C}_1$  and  $\mathcal{C}_3$  correspond to no strict change in the environment where the robot does not reach the first step. In classes  $\mathcal{C}_2, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6$ , significant changes in the position of the robot are produced. Each subsequent class (in order  $\mathcal{C}_5, \mathcal{C}_2, \mathcal{C}_6, \mathcal{C}_4$ ) represents the robot making it one step higher. The empty area in the top left of Figure 2 corresponds to overshooting off the stairs, which we excluded in the final clustering.

Figure 3 shows the action prototype found by our method/other methods. Our method manages to find distinct prototypes in significant effect areas. RGNG without the metric in Equation (2) finds distinct prototypes in significant effect areas but creates too many prototypes in “low effect” and “low variation in effect” areas because all areas are deemed as equally important. The uniformly random prototype selection fails to capture a large portion of the high-effect areas and cannot follow the underlying topology in the effect space.

To measure the quality of the generated prototypes, we constructed a reinforcement learning problem in the aforementioned environment with 15 steps instead of four. We created

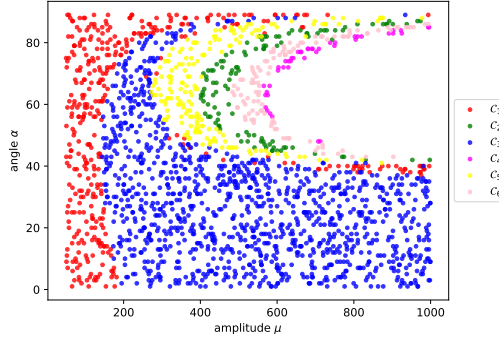


Figure 2: The visualization of the k-Means effect region clustering in y and z space in action space shows coherent classes even though the clustering was performed in effect space, which reaffirms these two spaces’ correlation. Classes  $\mathcal{C}_1$  and  $\mathcal{C}_3$  correspond to no strict change in the environment where the robot does not reach the first step. The other regions in the top half represent a one-, two-, three-, or four-step height gain (from left to right).

a custom Gymnasium [16] environment wrapper for our simulation with reward function  $r_t$  (from now on called “Up the stairs” environment). This reward function rewards jumping one step at a time and punishes falling proportional to the number of steps fallen.

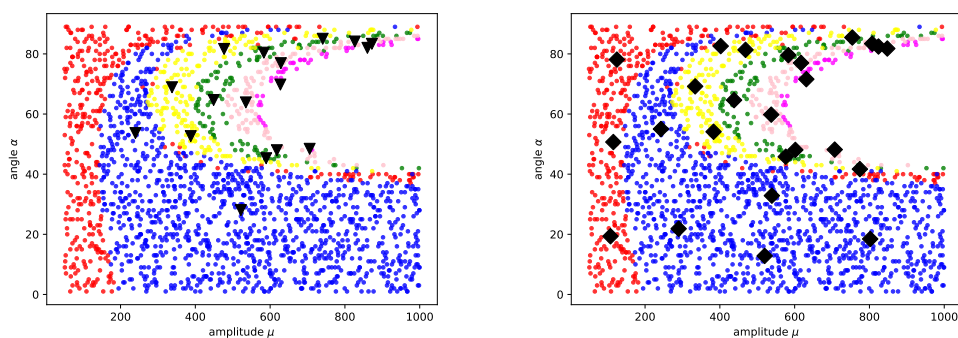
$$r_t = \begin{cases} 1 & \text{if } (s_{t+1}^z - s_t^z) > 0 \\ -(s_{t+1}^z - s_t^z)/0.3 & \text{if } (s_{t+1}^z - s_t^z) \leq 0. \end{cases} \quad (5)$$

We used the SAC and the DQN implementation of Stable-Baselines3 [11] as reinforcement learning agents in our evaluation. As a baseline for the achievable performance on this task, we trained the SAC agent [5] with 475146 parameters on the continuous motion space for 30000 steps. Additionally, we trained three Deep Q-learning agents with 5520 parameters each on our effect-based, uniform, and random prototypes. Figure 4 shows the mean reward with its respective standard deviation for each agent evaluated five times for each of the four seeds after every 500 timesteps. Each of the DQN agents collected data for 3000 steps into the replay buffer before starting to learn. The blue vertical dashed line in Figure 4 marks the point of learning start for the uniform and random prototypes. The red vertical dashed line is the sum of the 3000 DQN exploration steps and the 2000 motion samples collected before prototype generation.

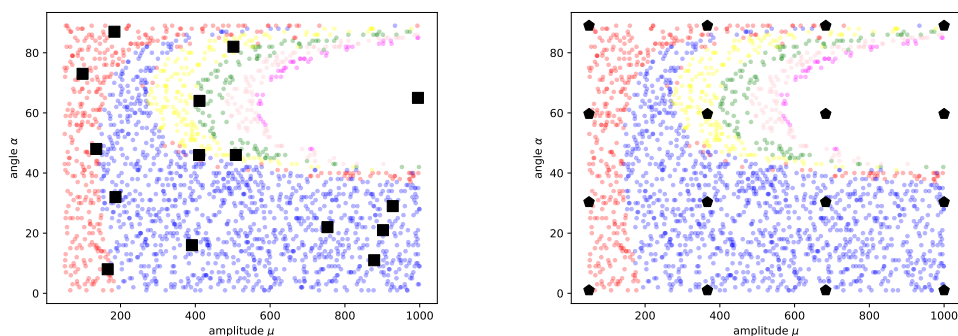
Our method exhibits robustness similar to SAC regarding standard deviation, which the other methods fail to achieve. While all methods fall short of the performance achievable in the continuous setting with SAC, our method achieves the highest mean return out of the three discrete methods. The effect-based method holds advantages over both alternative discretization methods: Random prototypes, compared to effect-based prototypes, rise to their maximum performance similarly fast but fail to reach the same average reward. Uniform prototypes are close in average reward to our method but converge way slower. Finally, although the performance of our method could be better compared to SAC, it achieves it with an 85 times smaller number of parameters. Some runs of our method achieve the maximal performance, but more investigation is needed to understand the reason for the lower average.

## 5 Discussion

Our custom “Up the stairs” environment shows promising first results in the search for automatic effect-based action discovery. A definite upside is that with a disproportionately smaller network size compared to the baseline SAC (factor 85), our method achieved competitive results with its preemptive effect-based prototype generation. One current limitation is the need for discrete effect spaces for clear effect class separation. Discrete



(a) Effect Region Clustering + RGNG with variation dependent prototype quantity (Ours) (b) Effect Region Clustering + RGNG and fixed number of prototypes for all classes

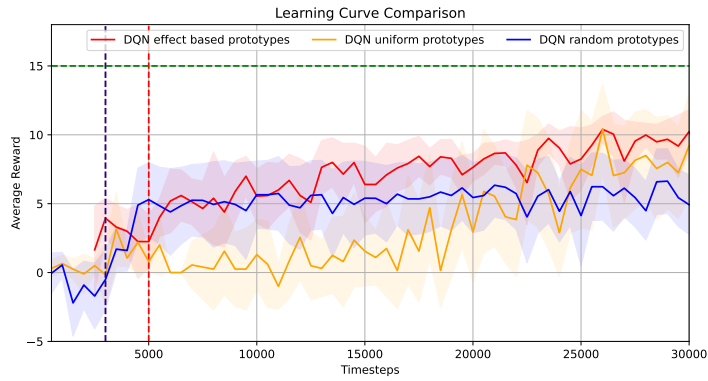


(c) Random prototypes (d) Uniform grid prototypes

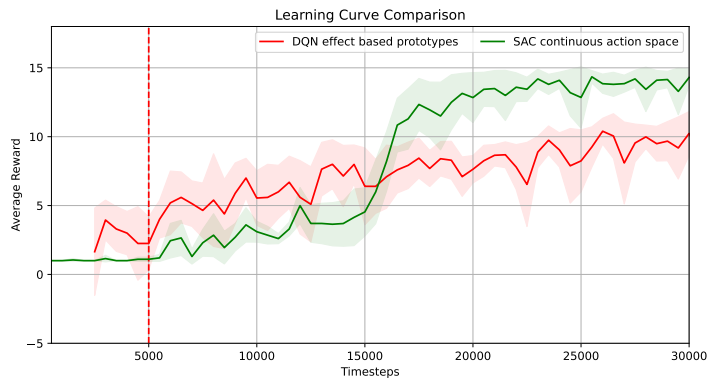
Figure 3: Action prototypes (visualized as regular polygons) found by (a) Effect Region Clustering with RGNG (Robust Growing Neural Gas Algorithm) and variation metric (Equation 2) for selecting the prototype quantity per class, (b) Effect Region Clustering with five prototypes per cluster RGNG, (c) random prototypes and (d) uniform grid prototypes. Methods (a) and (b) identify key change areas using Effect Region Clustering to generate prototypes across various effect motions. Method (b) generates excessive prototypes in stable areas, undermining the advantage of effect classes. Random and uniform methods yield many irrelevant prototypes. The colors of effect classes serve only as visual cues and do not affect methods (c) and (d).

effect spaces manifest through fixed boundary conditions in the environment (i.e., immovable stairs), which cause the effect class discovery to form clear boundaries as in Figure 2. There are no emergent effect category borders in purely continuous effect spaces (i.e., free space navigation). In future work, we want to investigate if effect-based discretization still yields benefits over the simpler uniform prototypes. Appendix B provides a visual aid for understanding the difference between discrete and continuous effect spaces.

When selecting the set of features that the algorithm should consider when evaluating the effects, there is a trade-off between performance and generalization. With each added available feature, the dimensionality of the clustering input space increases. If a lot of the added features are random, previously separate effect categories can merge into one. A visual representation of that can be found in Appendix C. If there is previous knowledge of insignificant effect features, removing them from consideration when clustering leads to a more robust effect class discovery. In future work, we want to investigate unsupervised options for measuring the randomness of each effect space feature so that the sweet spot of performance and generalization is discovered automatically.



(a)



(b)

Figure 4: During the “Up The Stairs” task learning, our effect-centric method, adjusted for 2000 samples for prototype generation, achieved faster convergence and higher maximal reward than random or uniform prototype generation methods. The SAC baseline, representing solvability in a continuous action space, has its maximal reward marked by a green dashed line at 15 in the graph (a). Graph (b) compares the learning curves of the SAC baseline, our effect-based discretization with DQN.

The action prototype selection method is critical in enabling the robot’s initial performance. Real-world settings and environments often do not exhibit evenly distributed high-effect regions. Our metric provides a reasonable estimate of how many prototypes each category should hold. After this initial decision, it is crucial to spread the action prototypes far from each other inside an effect category to create a wide range of different-looking motions in the robot’s repertoire.

## 6 Conclusion

We presented a new effect-centric approach to learn a discrete action set that can be used by decision-making components of robots grounded in their actual effects in the environment. This approach only makes assumptions about the state dimensions where effects should be measured and does not rely on the design of a reward function. Our method is constructed task-agnostic and shows promising preliminary results: in our toy environment “Up The Stairs” effect-based action space discretization outperforms uniformly and randomly sampled action space discretization in convergence speed and maximum reward.

## Acknowledgements

This research was funded in whole or in part by the Austrian Science Fund (FWF) [ELSA, DOI: 10.55776/I5755]. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

## References

- [1] Mihai Andries, Ricardo Omar Chavez-Garcia, Raja Chatila, Alessandro Giusti, and Luca Maria Gambardella. Affordance equivalences in robotics: A formalism. *Frontiers in Neurobotics*, 12, 2018. ISSN 1662-5218. doi: 10.3389/fnbot.2018.00026. URL <https://www.frontiersin.org/articles/10.3389/fnbot.2018.00026>.
- [2] R. Omar Chavez-Garcia, Pierre Luce-Vayrac, and Raja Chatila. Discovering affordances through perception and manipulation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3959–3964, 2016. doi: 10.1109/IROS.2016.7759583.
- [3] Andreas K Engel, Alexander Maye, Martin Kurthen, and Peter König. Where’s the action? the pragmatic turn in cognitive science. *Trends in cognitive sciences*, 17(5): 202–209, 2013.
- [4] Andrej Gams, Tadej Petrič, Bojan Nemec, and Aleš Ude. Manipulation learning on humanoid robots. *Current Robotics Reports*, 3(3):97–109, 2022.
- [5] 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*, pages 1861–1870. PMLR, 2018.
- [6] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7), 2023. ISSN 1424-8220. doi: 10.3390/s23073762. URL <https://www.mdpi.com/1424-8220/23/7/3762>.
- [7] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: challenges, representations, and algorithms. *J. Mach. Learn. Res.*, 22 (1), jan 2021. ISSN 1532-4435.
- [8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [9] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [10] A Kai Qin and Ponnuthurai N Suganthan. Robust growing neural gas algorithm with application in cluster analysis. *Neural networks*, 17(8-9):1135–1148, 2004.
- [11] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- [12] Muhammed Saeed, Mohammed Nagdi, Benjamin Rosman, and Hiba H. S. M. Ali. Deep reinforcement learning for robotic hand manipulation. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–5, 2021. doi: 10.1109/ICCCEEE49695.2021.9429619.
- [13] Erol Sahin, Maya Cakmak, Mehmet Dogar, Emre Ugur, and Göktürk Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior - ADAPT BEHAV*, 15:447–472, 12 2007. doi: 10.1177/1059712307084689.

- [14] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Handbooks. Springer, 2016. ISBN 978-3-319-32550-7. doi: 10.1007/978-3-319-32552-1. URL <https://doi.org/10.1007/978-3-319-32552-1>.
- [15] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *Proceedings of the aaii conference on artificial intelligence*, volume 34, pages 5981–5988, 2020.
- [16] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL <https://zenodo.org/record/8127025>.
- [17] Kadir Firat Uyanik, Yigit Caliskan, Asil Kaan Bozcuoglu, Onur Yuruten, Sinan Kalkan, and Erol Sahin. Learning social affordances and using them for planning. In *CogSci 2013*, Berlin, Germany, July 31–August 2 2013.
- [18] Tianying Wang, En Yen Puang, Marcus Lee, Yan Wu, and Wei Jing. End-to-end reinforcement learning of robotic manipulation with robust keypoints representation, 2022.
- [19] Mehdi Zadem, Sergio Mover, and Sao Mai Nguyen. Goal Space Abstraction in Hierarchical Reinforcement Learning via Set-Based Reachability Analysis. In *2023 IEEE International Conference on Development and Learning (ICDL)*, pages 423–428, Macau, China, November 2023. IEEE. doi: 10.1109/ICDL55364.2023.10364473. URL <https://inria.hal.science/hal-04401878>.
- [20] Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017. doi: 10.1177/1059712317726357. URL <https://doi.org/10.1177/1059712317726357>.
- [21] Philipp Zech, Erwan Renaudo, Simon Haller, Xiang Zhang, and Justus Piater. Action representations in robotics: A taxonomy and systematic classification. *The International Journal of Robotics Research*, 38(5):518–562, 2019. doi: 10.1177/0278364919835020. URL <https://doi.org/10.1177/0278364919835020>.