GRIMGEP: Learning Progress for Robust Goal Sampling in Visual Deep Reinforcement Learning

Grgur Kovač, Adrien Laversanne-Finot, Pierre-Yves Oudeyer

GRIMGEP: Learning Progress for Robust Goal Sampling in Visual Deep Reinforcement Learning

Grgur Kovač
Flowers Team
INRIA(FR)
grgur.kovac@inria.fr

Adrien Laversanne-Finot
Flowers Team
INRIA(FR)
adrien.laversanne-finot@inria.fr

Pierre-Yves Oudeyer
Flowers Team
INRIA(FR)
pierre-yves.oudeyer@inria.fr

Abstract: Autonomous agents using novelty based goal exploration are often efficient in environments that require exploration. However, they get attracted to various forms of distracting unlearnable regions. To solve this problem, Absolute Learning Progress (ALP) has been used in reinforcement learning agents with predefined goal features and access to expert knowledge. This work extends those concepts to unsupervised image-based goal exploration. We present the GRIMGEP framework: it provides a learned robust goal sampling prior that can be used on top of current state-of-the-art novelty seeking goal exploration approaches, enabling them to ignore noisy distracting regions while searching for novelty in the learnable regions. It clusters the goal space and estimates ALP for each cluster. These ALP estimates can then be used to detect the distracting regions, and build a prior that enables further goal sampling mechanisms to ignore them. We construct an image based environment with distractors, on which we show that wrapping current state-of-the-art goal exploration algorithms with our framework allows them to concentrate on interesting regions of the environment and drastically improve performances. The source code is available at https://sites.google.com/view/grimgep[1].

Keywords: Goal exploration, Curiosity, Learning Progress, Intrinsic Motivation, Goal-conditioned Deep Reinforcement Learning

1 Introduction

Although recent work in reinforcement learning has shown that robots can learn complex individual skills such as grasping [2], locomotion [3, 4], and manipulation tasks [5], designing reinforcement learning algorithms that perform well in sparse reward scenarios is still an open challenge of artificial intelligence. Standard reinforcement learning algorithms struggle in the sparse reward scenario because they rely on simple exploration behavior such as random actions.

As a result, learning complex tasks often requires manually collecting examples [6, 7] or running learning algorithms over a long period of time which may not be possible in real life scenarios. Designing better exploration schemes would help agents autonomously discover interesting features that can then be used to learn the long term objective. Developing efficient exploration algorithms would thus help create a more autonomous learning agent.

Several approaches have been considered in order to improve the exploration performances of reinforcement learning algorithms. One approach is to reward the agent for discovering novel observations in the form of an intrinsic reward that is added to the original reward of the environment [8].

[1]The source code is a modified version of the official code from [1] https://github.com/vitchyr/rlkit

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

This intrinsic reward often models the "surprisal" of observations encountered by the agent and is often computed as the error of some predictive model.

While this approach has seen good results in many environments [9], even allowing an agent to solve tasks purely driven by intrinsic rewards, it falls short in simple environments, in the presence of an action-induced distracting region [9, 10]. This problem is often referred to as the noisy TV problem as an agent that aims at maximizing surprise would get irresistibly attracted by a TV outputting random images as a result of actions done by the agent.

One way to solve this problem is to use a curiosity bonus or a goal sampling mechanism based on the progress of the agent such as the learning progress [11, 12]. This general idea can be derived into many different exploration algorithms [13, 14, 15]. For example, SAGG-RIAC [13] gradually clusters the environment into sub-regions of different learning progress and samples goals in the more promising regions (e.g. the regions with high learning progress). This approach has been shown to enable learning of inverse models in high-dimensional robots [13]. In other approaches, the goal space is separated into different regions corresponding to different high level goals before exploration, from which goals can then be sampled based on the associated curiosity measure [14, 15, 16].

However, those approaches have been mostly applied to state-based environments where the environments provide high level observations such as position or velocity. In the case of image-based environments, those cannot be applied directly due to the high dimensionality of the observations. Adapting approaches based on the learning progress to image based environment is thus an open problem.

In this paper we provide a solution to action-induced distractors (the noisy TV problem) for image based environments for goal exploration algorithms. The general idea is to cluster the environment into different clusters from which we can estimate the associated learning progress. From a high level perspective our method can be viewed as a way to learn a prior over possible goals. The goal of this prior is to learn which parts of the environment should be explored. This prior can then be combined with the goal sampling method of any goal exploration algorithm in order to guide exploration. We experimentally demonstrate that our framework improves the overall performances of various goal exploration algorithms and autonomously focuses the exploration on interesting or learnable regions on the environment.

Contributions The main contribution of this work is to study how learning progress can be estimated in image-based environments and how it can be used to guide exploration of goal exploration algorithms in Deep RL. Precisely, we make the following contributions:

- We design a new framework that allows estimation of learning progress in image-based environments for goal sampling in goal-conditioned Deep RL.
- We instantiate this framework and combine it with existing goal exploration algorithms, and show that it improves the performances of these algorithms in the presence of distractors.
- We design a new novelty seeking CountBased IMGEP.
- We provide an easily customizable image-based environment constructed for studying noisy distractors in goal driven exploration tasks.

2 Related work

Goal exploration algorithms Intrinsically Motivated Goal Exploration Processes (IMGEPs) are a class of exploration algorithms that explore by repeatedly setting goals for themselves that they then try to achieve. They exist in two forms: population-based IMGEPs leveraging non-parametric models (e.g. [13, 15]), and goal-conditioned RL IMGEPs leveraging Deep RL techniques studied in this paper (e.g. [17, 1, 16]). Combined with an efficient curiosity mechanism for sampling goals, this approach has been shown to enable high-dimensional robots to learn very efficiently locomotion skills [13], manipulation of objects [18, 19, 17, 1] or tool use [15]. Methods using absolute learning progress [13] to drive goal sampling were shown to scale up to real world environments with potentially many forms of distractors, including action-induced distractors [15, 16] often modeled with the Noisy TV problem. However, these works relied on abstract hand-defined goal and state spaces (e.g. based on object positions and velocities). An exception is Laversanne-Finot et al. [20], which

used learning progress to sample goals in learned latent spaces, but this relied on population-based learners and required offline pre-training. Here, we adapt the learning progress approach to Deep RL goal-exploring agents that perceive their environments through pixels, and study their properties in this context.

Recently some approaches studied goal exploration in the image-based goal-conditioned Deep RL framework. For instance [17] learns a goal-conditioned policy on top of a learned embedding of the environment. Warde-Farley et al. [21] learns a goal conditioned policy by maximizing the mutual information between the goal state and the achieved state. While the goal policy achieves a good performance, the heuristic used for goal sampling is very simple and was not shown to scale to large environments with distractors. [1] improves upon [17] by designing a mechanism that incentives goal sampling to focus on exploring the frontier of the distribution of known goals, implementing a form of novelty search [22]. However, it can be naturally attracted by distractors that generate novel observations. Here, we experimentally identify this limit, and show that the learning-progress based goal sampling mechanism we introduce can be used as a prior on top of these algorithms to enable them to avoid irrelevant or unlearnable regions of the environment, while keeping their efficient dynamics in learnable regions.

Curiosity-driven Deep RL A common trend to improve exploration in the classical reinforcement learning setting with sparse external rewards has been to supplement the task reward with intrinsic reward [23, 8, 24, 25]. Approaches considering image-based low-level perception have used intrinsic rewards measuring various forms of novelty, based on counts [23] or prediction errors [8, 26]. While these approaches can give impressive results, they have fallen short on simple environments in the presence of a distractor that is partialy controlled by one of the agent's actions. [27] studied the use of learning progress as an intrinsic reward to enable Deep RL agents to be robust to distractors. However this approach relied on high-level disentangled state representations and did not include the notion of goal exploration.

Autonomous discovery of skills Another emerging field studies unsupervised discovery of diverse skill repertoires [28, 29]. As they have been relying on maximizing variation of novelty/diversity measure, they are also limited in environments with distractors, and do not consider goal exploration and goal-conditioned policies.

3 Approach

3.1 Problem definition

Open ended unsupervised goal based exploration

We study the problem of open ended unsupervised goal exploration. The objective is to maximise the performance over the goal space. This objective can be formalized with the following two equations:

$$\text{argmax}_\theta P_\theta \tag{1}$$

$$P_\theta := \int_{g \sim \mathcal{G}} f(g, \tau_{g,\pi_\theta}) \, dg \tag{2}$$

With P_θ representing the agent's performance, $\mathcal{G}$ being the goal space and $\tau_{\pi_\theta,g}$ being the trajectory resulting from following a policy π parameterized by θ while aiming for a goal g. f is the goal fulfillment function specifying to what extent was the goal g reached in episode $\tau_{\pi_\theta,g}$. This function is defined by the experimenter for the purpose of evaluation. For example, for an image-based navigation task, the goal space would consist of images and the function f could be the negative distance to the correct location associated with a selected set of testing goals (images) that will be used for evaluation. During training, we must autonomously discover which images can be used as feasible goals, and we do not have access to the function f nor the test set.

Intrinsically Motivated Goal Exploration Process (IMGEP)

One way to optimize this objective is to use an Intrinsically Motivated Goal Exploration Process (IMGEP). IMGEPs explore their environment by iteratively sampling goals that they then try to

achieve, leading to an efficient exploration in practice. We adopt the IMGEP formalization of [15] and extend it to the context of goal conditioned RL. An IMGEP is defined by a tuple:

$$\text{IMGEP} := (\mathcal{G}, \mathcal{S}, \pi_\theta, \mathcal{C}, \mathcal{O}, \gamma), \tag{3}$$

where $\mathcal{G}$ is the goal space and $\mathcal{S}$ is the state space. π_θ is a goal conditioned policy (for example a neural network) parameterized by θ. $\mathcal{C}$ is a cost function estimating the experimenter's f from (1). In an unsupervised setting, this function must be designed without access to f. $\mathcal{O}$ is an optimization scheme for optimizing θ according to $\mathcal{C}$. γ is a goal policy:

$$\gamma : \mathcal{H} \to p, p : \mathcal{G} \to [0, 1] \tag{4}$$

γ is a function that, using the history of training, creates a distribution for sampling the next goal. This function usually uses the history of observed states to estimate the utility, based on intrinsic rewards like novelty or learning progress, of sampling goals, and constructs a distribution that gives higher probabilities to goals providing higher intrinsic utility.

Biasing the IMGEPs goal sampling

In this work we still use IMGEPs to optimize the objective in equation (1), but we focus on the problem of robustifying their goal sampling by a prior. This prior should direct the IMGEP's goal sampling towards relevant regions of the goal space.

We formalize this as the prior policy:

$$\delta_\psi : \mathcal{H} \to prior,\ prior : \mathcal{G} \to [0, 1] \tag{5}$$

where G is the goal space, $\mathcal{H}$ is the training history, and ψ are parameters of the prior policy. The prior policy, using the history of training, creates a bias distribution that can be combined with the distribution created by γ.

We define a biased IMGEP as:

$$\mathcal{I}_\psi := (\mathcal{G}, \mathcal{S}, \pi_\theta, \mathcal{C}, \mathcal{O}, \gamma \otimes \delta_\psi) \tag{6}$$

where ψ are the parameters of the prior policy and $\otimes$ is some operator combining two distributions (ex. multiplication). The biased version of the IMGEP differs from the original IMGEP in that the goal policy is combined with the prior policy.

Our objective is to find the prior policy parameters ψ that will, when combined with the goal policy γ, result in the best performance of this biased version of IMGEP. Formally, our objective is:

$$\text{argmax}_\psi\ P_{\mathcal{I}_\psi}. \tag{7}$$

3.2 The GRIMGEP framework

To optimize the objective in eq. 7, we present the **G**oal **R**egions guided **I**ntrinsically **M**otivated **G**oal **E**xploration **P**rocess (GRIMGEP) framework. The main idea is to take any IMGEP and improve it by constraining it to sample goals only in the interesting region of the goal space. GRIMGEP does this by clustering the goal space, estimating absolute learning progress (ALP) for each cluster, sampling a cluster according to these ALP estimates, and using the underlying IMGEP to explore in the sampled cluster. The general idea behind this scheme is that clusters corresponding to distracting regions will have small ALP and will therefore be sampled rarely.

The GRIMGEP framework consists of the following four components which are depicted and explained in figure 1:

I. **Clustering component:** The job of the clustering component is to separate the goal space into clusters. Ideally, some clusters should contain only the goals from the disracting regions, and other clusters only goals from the interesting regions. This component can be implemented in many different ways. One possible solution is to construct a *clustering* latent space, and perform the clustering on this latent space using some clustering algorithm.

   In our experiments, we use a GMM-based clustering algorithm, inspired by [30], on the latent space of a β-VAE (for implementation details see Appendix D).

II. **Intrinsic Reward Estimation component:** The goal of this component is to assign a value (or intrinsic reward) to each cluster. A suitable intrinsic reward is Absolute Learning Progress (ALP). ALP is very effective because it is low for both the unlearnable and the already learned regions, and high for the task that is currently being learned [11, 31].

In our concrete manifestation we estimate ALP as follows. The Intrinsic Reward Estimation component remembers all the goals that were sampled, the epoch number when they were sampled, and their resulting last states. At the beginning of every epoch, we calculate the performances by delegating this task to the underlying IMGEP[2]. We use the clustering algorithm to obtain the *cluster_ids* of all goals that were sampled. We construct a history of performances for each cluster. We do so by averaging the performances of the goals from the same cluster that were sampled in the same epoch. We split those histories into the first and the second halves. Then, we estimate ALP as the absolute difference of the means of those two halves. This calculation can be seen in the algorithm 1, and in equation (8) where l is the length of the cluster's history, and h is the cluster's history. One element of h is the average performance of goals (belonging to the same cluster) sampled in the same epoch.

$$ALP = |\frac{1}{l/2}\sum_{i=0}^{l/2} h_i - \frac{1}{l/2}\sum_{i=l/2}^{l} h_i| \tag{8}$$

III. **Prior Construction component:** The task of this component is to create the prior distribution that will be the only thing passed down to the underlying IMGEP. This distribution should give higher probabilities to goals from clusters with higher intrinsic rewards (ALPs) computed in the intrinsic reward estimation component.

In our experiments, we construct the prior as a masking distribution. It is constructed as follows: First, a cluster is sampled from the ALP estimates using a Multi-Armed Bandit. The goal sampling prior is then simply a uniform distribution over goals belonging to the sampled cluster. Such a prior ensures that the underlying IMGEP explores in the region sampled by the bandit. This procedure can be repeated multiple times in order to sample multiple goals in the same epoch. Further implementation details are explained in Appendix D.

IV. **The Underlying IMGEP:** This component can be any IMGEP as defined by equation (3).

We experimented with a few different algorithms as the underlying IMGEP as detailed in section 4. All IMGEPs sample goals from a replay buffer according to some distribution p_{imgep}. When used inside the GRIMGEP framework, that distribution is combined with the prior as in equation 9 where R is the replay buffer.

$$p(g) = \frac{p_{prior}(g)p_{imgep}(g)}{\sum_{g \in R} p_{prior}(g)p_{imgep}(g)} \tag{9}$$

We use p everywhere where the underlying IMGEP would normaly use p_{imgep}. In the underlying IMGEPs that we study that means p will be used for three purposes: 1) sampling goals for exploration, 2) sampling replacement goals for HER, and 3) since all of our underlying IMGEPS train their own VAE, biasing the training of this VAE.

The concrete manifestation is also depicted in algorithm 2. First, we do a warmup phase to fill the replay buffer by doing random actions. Then, for the first $start\_prior$ epochs, we do not use any kind of intrinsic rewards but just resample goals uniformly from the replay buffer. After this, we start using the GRIMGEP framework for goal sampling. In practice, we make use of parallel computing by sampling ten goals per epoch. For each of those goals, we sample a new cluster, construct a new prior and then sample a new goal using the probability distribution obtained by combining the prior and the goal sampling probability distribution of the IMGEP.

4 Experiments

In this section, we aim to answer the following questions:

[2]In this work, all of the the studied underlying IMGEPs train their own VAE and compute the performance as the L2 distance in the latent space. The performances need to be recomputed because this VAE is trained online.

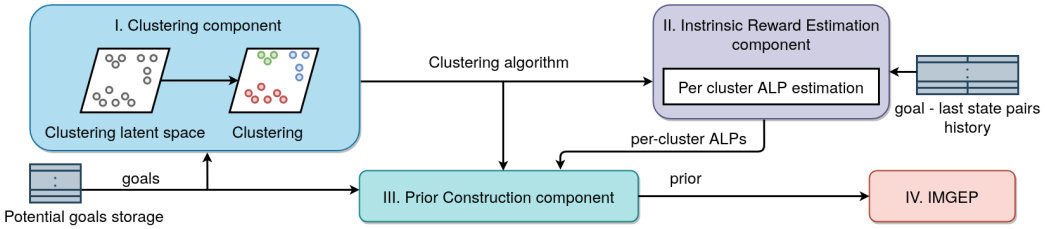


Figure 1: Goal sampling procedure in the GRIMGEP framework. 1) First, the **Clustering component** clusters the goal space. 2) The absolute learning progress (ALP) of each cluster is then computed by the **Intrinsic Reward Estimation component** using the history of attempted goals and the corresponding outcomes. 3) In the **Prior Construction component** a cluster is sampled using the ALP estimates. The goal sampling prior is then constructed as the masking distribution assigning a uniform probability over goals inside the sampled cluster and 0 probability to goals outside the cluster. 4) A goal is then sampled, inside **The Underlying IMGEP**, from the distribution formed by combining the goal prior and the underlying IMGEP goal sampling function according to Eq. 9. Details on each components are given in Sec. 3.2

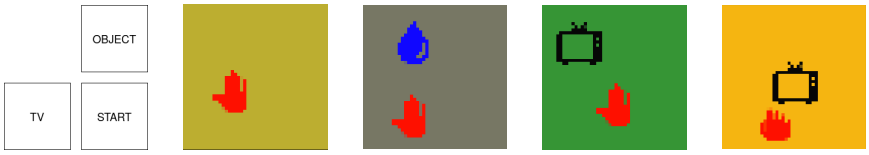
- How do current approaches behave in the presence of action-induced distractors?
- How does the GRIMGEP framework, when combined with existing goal exploration algorithms, improve their performances?
- What role do the LP-based intrinsic rewards play in the final performance of our manifestation of the GRIMGEP framework?

4.1 Experimental method

We study these questions in the *PlaygroundRGB* environment. It consists of three rooms through which the agent can move. The agent controls the position of the gripper arm through continuous actions and can close/open the arm. It observes the environment as an image of a top down view of the current room.

The available rooms are depicted in Fig. 2 and the topology of the environment in Fig. 2a. The agent always starts in the Start room. All the possible goals inside this room are very easy and require only moving the gripper to the correct location. The Object room represents the *interesting* part of the environment as it contains a movable object and is the only non-distracting part of the environment. The TV room plays the role of an action induced noisy distractor. This room contains a TV that can be turned on by closing the gripper. When the TV is turned on the location of the TV and the background color are randomized. Further details on the environment are given in Appendix C.

Since goals from the Start room are easy, we expect any goal exploration algorithm to learn goals inside this room. However, only algorithms exploring well should master goals inside the Object room. This is why, for evaluation, we construct a static test set of 25 goals from the Object room. Goals are completed if in the last state both objects are in the correct location. The performance of the agent is the average success over this evaluation set.

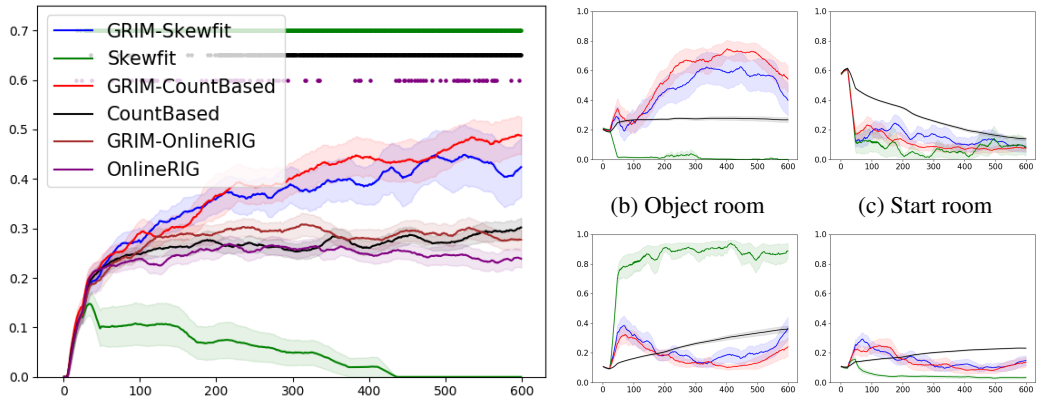


(a) topology (b) Start room (c) Object room (d) TV room off (e) TV room on

Figure 2: Different rooms of the PlaygroundRGB environment

4.2 Baselines

In this work, as the underlying IMGEPs, we study two novelty seeking approaches (Skewfit [1] and *CountBased*) and one approach that doesn't have exploration bonuses (OnlineRIG [17, 1]). Skewfit



(a) Success rates

(b) Object room

(c) Start room

(d) TV room ON

(e) TV room OFF

Figure 3: Comparison of algorithms alone and in combination with the GRIMGEP framework. Ten seeds were used and the dots depict statistically significant ($p < 0.05$, Welch's t-test) results when compared to the GRIM version. The shaded areas correspond to standard errors and the bold line to the mean smoothed over 25 epochs. The proportion of goals sampled in each room is represented on the right. They correspond to goals sampled for both exploration and for replacement in HER.

was shown to outperform a number of baselines for unsupervised exploration. It uses a VAE to model the uniform distribution of the goal space by giving higher priorities to novel looking data (see Appendix A.1 for more details). CountBased is a new novelty seeking approach and one of the contributions of this paper. CountBased estimates the visitation count of states using a downsampled version of the observation. It then prioritizes sampling of states with a low visitation count (see Appendix A.3 for more details). OnlineRIG [17], is a modified version of Skewfit that does not use the skewing mechanism for exploration. Instead, in OnlineRIG, goals are sampled uniformly from the replay buffer.

4.3 Results

How do current approaches behave in the presence of noisy distractors?

We can see, in Fig. 3, that Skewfit is heavily drawn to the noisy part of the TV room resulting in very low sampling of other parts of the environment, notably the Object room. As a result of not exploring the Object room enough, the final performance diminishes. CountBased also samples a lot of goals in the TV room. However, in comparison to Skewfit, the focus is separated between both the TV-on and TV-off goals (see Fig. 3d and Fig. 3e). OnlineRIG is also not able to achieve high performances. The reason is that, due to the lack of exploration incentive, it sampling goals mostly from the Start room (see Fig. 5c). A more detailed comparison between Skewfit and CountBased is given in Appendix A.4.

Overall, this experiment demonstrates that this environment requires exploration incentives but, that these incentives should not be novelty based.

How does the GRIMGEP framework change the behaviour of current approaches in the presence of noisy distractors?

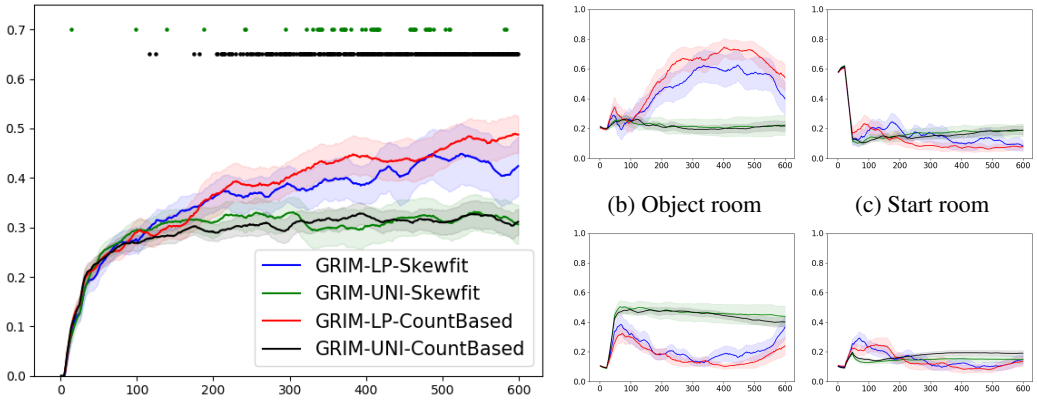
To answer this question we wrap the Skewfit and CountBased with the GRIMGEP framework. As can be seen in Fig. 3, both GRIM-Skewfit and GRIM-CountBased, focus more on the Object room and less on the TV room than their unwrapped counterparts. This results in faster learning and much better performances at the end of the training. When using OnlineRIG inside the GRIMGEP framework we can again observe a strong focus on the Object room (see Fig. 5b), but we can also see that this focus alone is not sufficient to greatly improve performance.

This leads us to the conclusion that GRIMGEP successfully does two things: 1) it detects the relevant part of the goal space, and 2) successfully uses the novelty seeking exploration in this relevant region (see Appendix A.5 for more details).

What role do the LP-based intrinsic rewards play in the final performance of our manifestation of the GRIMGEP framework?

In our particular manifestation of the GRIMGEP framework the Intrinsic Reward Estimation component estimates the learning progress for each cluster. The Prior Construction component then takes these LP estimates, samples a cluster and constructs a masking prior.

In the experiments in Figure 4 we test how cluster sampling based on LP (GRIM-LP-*imgep_name*) impacts the performances in comparison to uniform cluster sampling (GRIM-UNI-*imgep_name*). As shown in Fig 4a, GRIMGEPs that use LP outperform the GRIMGEPs that don't. Furthermore, we can see that, when sampling the clusters uniformly, the TV Room with the TV on is sampled more. This is caused by the TV room with the TV on being represented by a large number of clusters because of the distractor.



(a) Success rates

(b) Object room

(c) Start room

(d) TV room ON

(e) TV room OFF

Figure 4: Comparison of the GRIMGEP framework with (LP) and without (UNI) per cluster intrinsic rewards. It is visible that GRIMGEP works better when LP is used to select the most interesting cluster. The curves are drawn in the same way as those in figure 3. The dots depict statistically significant ($p < 0.05$, Welch's t-test) results when compared to the LP version (GRIM-LP-*imgep_name*).

5 Conclusion

We studied the problem of unsupervised goal exploration in the presence of action-induced distractors. We have shown that current approaches, that aim at maximizing novelty, fail in such environments. On the other hand, the GRIMGEP framework is able to avoid those regions and drive exploration towards interesting regions through estimation of learning progress. The GRIMGEP framework acts as a goal sampling prior that can be combined with any goal exploration algorithm. We have shown that, in the presence of distractors, combining the GRIMGEP framework with state-of-the-art exploration algorithms allows them to ignore distracting regions while focusing on the interesting regions, and ultimately improves their performances.

A natural extension of this work would be to use a more general clustering mechanism. For example, one can imagine using the latent space of a contrastive network, where states close in time would be close in the latent space, to tackle visually richer environments [10, 32].

Acknowledgments

This work was partially supported by Ubisoft, Bordeaux. Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr/).

References

[1] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[3] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[4] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10, 2020.

[5] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[6] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132, 2020.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[9] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[10] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[11] J. Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991.

[12] O. Pierre-Yves, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[13] A. Baranes and P. Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013. ISSN 09218890. doi:10.1016/j.robot.2012.05.008.

[14] S. Forestier and P. Y. Oudeyer. Modular active curiosity-driven discovery of tool use. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:3965–3972, 2016. ISSN 21530866. doi:10.1109/IROS.2016.7759584.

[15] S. Forestier, Y. Mollard, and P. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*, abs/1708.02190, 2017. URL `http://arxiv.org/abs/1708.02190`.

[16] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340, 2019.

[17] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[18] M. Rolf, J. J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

[19] S. M. Nguyen and P.-Y. Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, 2014.

[20] A. Laversanne-Finot, A. Péré, and P.-Y. Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

[21] D. Warde-Farley, T. V. de Wiele, T. D. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *CoRR*, abs/1811.11359, 2018. URL http://arxiv.org/abs/1811.11359.

[22] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[23] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[24] Y. Burda, H. Edwards, D. Pathak, A. J. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *CoRR*, abs/1808.04355, 2018. URL http://arxiv.org/abs/1808.04355.

[25] R. Raileanu and T. Rocktäschel. RIDE: rewarding impact-driven exploration for procedurally-generated environments. *CoRR*, abs/2002.12292, 2020. URL https://arxiv.org/abs/2002.12292.

[26] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[27] K. Kim, M. Sano, J. De Freitas, N. Haber, and D. Yamins. Active world model learning with progress curiosity. *arXiv preprint arXiv:2007.07853*, 2020.

[28] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[29] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 10519–10531, 2019.

[30] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853, 2020.

[31] F. Kaplan and P.-Y. Oudeyer. Maximizing learning progress: an internal reward system for development. In *Embodied artificial intelligence*, pages 259–270. Springer, 2004.

[32] S. Venkattaramanujam, E. Crawford, T. Doan, and D. Precup. Self-supervised learning of distance functions for goal-conditioned reinforcement learning. *arXiv preprint arXiv:1907.02998*, 2019.

[33] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[34] C. E. Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

[35] H. Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52:345–370, 02 1987. doi:10.1007/BF02294361.

A Studied underlying IMGEPs

A.1 Skewfit algorithm

Skew-fit[1] is an iterative algorithm that aims to train a generative model modeling the uniform distribution over the feasible goal space. To ensure this feasibility, Skew-fit constructs a dataset for training the generative model only from the already observed states. They can do this because they, like us, assume the goal space to be equivalent to the state space. After each epoch of the generative model training, a new Skewed dataset is constructed. It is constructed by setting each state's weight as inversely proportional to the current generative model's probability of that state. This skewed dataset is then used to train the generative model in the next epoch. This dataset's entropy increases in each epoch, so the model converges to modeling a uniform distribution.

This algorithm is then applied to unsupervised goal driven exploration. The generative model is a VAE and for its training dataset the replay buffer is used. They propose two mechanisms for goal sampling. Those mechanisms are: sampling directly from the current VAE or sampling from the skewed replay buffer. In this work we focus on the latter tough both mechanisms could be used with the GRIMGEP fremework. They train a SAC agent with the reward being the current VAE's negative L2 latent distance between the goal and the state. In every epoch, goals are sampled, episodes run, new data added to the replay buffer, and both the VAE and the agent are trained. It should be noted that for training the SAC agent, data is sampled uniformly from the replay buffer, and for training the VAE and sampling HER replacement goals, a skewed replay buffer is used. They denote this algorithm Skewfit + RIG though for simplicity we denote it Skewfit in this work.

A.2 Skewfit problem formalization

Pong et al. [1] formalize the problem as minimizing $\mathcal{I}(S;G) = \mathcal{H}(G|S) - \mathcal{H}(G)$. This can be understood as training the agent ($\mathcal{H}(G|S)$) on as diverse goals as possible ($-\mathcal{H}(G)$). This formalization, though sufficient for the environments they study, is not sufficient for environment containing noisy distracting regions. Such regions are a source of entropy and therefore maximizing $\mathcal{H}(G)$ will lead the algorithm to focus on such regions. In short, the drawback of this formalization that it focuses on novelty while our formalization, in equations (1) and (2), focuses on learnability.

A.3 CountBased approach

This approach is very similar to Skewfit, but it has a different intrinsic reward. This intrinsic reward is computed using a count-based novelty measure. As shown in equation 10, we downsize an image (to 10x10) and quantize each channel into 3 different values and then count how many times the representation was observed. The goal's intrinsic reward is then computed by exponentiating the count by an hyperparmeter $\alpha \in [-1, 0]$, inspired by [1]. Also as in Skewfit, we use this reward to construct the distribution to prioritize data for training the VAE, sampling exploration goals and sampling HER replacement goals. When using this approach as part of the GRIMGEP framework we multiply this distribution with the prior as in equation 9.

$$R_{CB}(image) = count(quantize(downsize(image)))^{\alpha} \tag{10}$$

A.4 Analysis of Skewfit and CountBased differences

In figure 3 we can see that Skewfit and CountBased behave differently despite them both being a form of novelty search. If we look at Skewfit we can observe a drop of focus on the Start, the Object room, and the TV-off part of the TV room. Our explanation for this is that Skewfit is more greedy. In other words, it *marks* the TV-on TV room goals as *novel* and all the other goals as *known*. To be more precise, one background color might look very novel for Skewfit, so its training will oversample this background color resulting in another color looking very novel in the next iteration. It will then proceed to *jump* between different background colors. This results in shifting focus from all the regions not *marked* as *novel* (the Start room is then sampled more because of the quantity of such goals in the buffer, see. Appendix A.5).

For CountBased, on the other hand, there is a drop of focus only on the Start room. There is no drop of focus on the Object room and a rise of focus on both TV-off and TV-on goals from the TV room.

We believe that, this is because CountBased orders the rooms by their novelty. TV room being the most novel (TV-on goals being more novel than TV-off goals), then Object room, and Start room being the least novel. This results in shifting the focus only from the Start room.

A.5 About the α hyperparameter

It is relevant to note that Skewfit has a regularization hyperparameter α which interpolates between an uniform and a skewed distribution (-1 being completely skewed and 0 being uniform). In their work they experiment with 4 different values of this parameter (-0.25, -0.5, -0.75, -1.0)[1]. In all our experiments when Skewfit is used inside the GRIMGEP framework (GRIM-Skewfit in fig. 3, GRIM-LP-Skewfit and GRIM-UNI-Skewfit in fig. 4) we use $\alpha = -0.75$. However, when the raw Skewfit is used (Skewfit in fig. 3) we use $\alpha = -0.25$. This makes the raw Skewfit less prone to strive for novelty and therefore less prone to being distracted by the noisy regions of our environment (we have also observed this in our experiments). We can see that, even with this advantage, it is still drawn to the distracting TV room. That demonstrates another benefit of the GRIMGEP framework. Using Skewfit inside the GRIMGEP framework enables us to use more aggressive exploration bonuses ($\alpha = -0.75$) without training becoming unstable.

B Hyperparametres

For the underlying IMGEPs, all the hyperparameters, including the ones for the training VAE, are the same as in Pong et al. [1] in the "Visual Door" experiments, except the α hyperparameter (see Appendix A.5).

The clustering VAE hyperparmetres are shown in table 1.

representation size	3
batch_size	128
beta	1
lr	0.001
Encoder	
kernel sizes	[5, 3]
num of channels	[4, 4]
strides	[3, 2]
Decoder	
kernel sizes	[3, 3]
num of channels	[4, 4]
strides	[2, 2]

Table 1: The Clustering VAE hyperparmaeters. A reduced version of the training VAE used in the underlying IMGEPs.

Other hyperparmaeters are shown in table 2.

T	5
episode length	50
room size	1.5x1.5

Table 2: Other hyperparmaeters.

C PlaygroundRGB environment

As stated, the PlaygroundRGB consists of three rooms through which the agent can move. The action space is three dimensional ($a \in \mathbb{R}^3$). The first two dimensions correspond to moving the gripper, and the last one to opening and closing it. The gripper is open if the last dimension is positive. The dimensions of each room are 1.5x1.5. The agent moves to another room by going to middle 0.5 of the wall separating those two rooms.

The TV room contains both an easy task and an action induced noisy distractor. When the gripper is open (the TV is off) the background color and the TV location do not change and are always the same. This is an easy task where the agent just has to learn to move the gripper without closing it.

When the gripper is closed the TV is on. Every timestep with the closed gripper a new TV location is randomized and the background color changed to the color selected at the beginning of the rollout. The TV-on background color is selected at the beginning of each rollout from as set of five possible colors. Meaning, during one rollout there is one TV-on color and during the whole training there are a total of five possible TV-on colors.

For evaluation, the 25 test goals were constructed by selecting 5 possible locations (center, NW, NE, SW, SE) and doing the cartesian product of this locations for both objects. We evaluate the location of an object as correct if its L-∞ distance from the goal's location is less then 0.2.

D Details on the GRIMGEP components

I. **Clustering component:** To create the clustering latent space we use a β-VAE[33]. It is relevant that this representation focuses on the features which are relevant for separating the regions and not on the specific details relevant for training the agent. For this purpose, we reduce the size of the VAE and its latent space. For hyperparameters see Appendix B. We train this VAE online after each epoch on the data uniformly sampled from the replay buffer.

   On this clustering latent space we train ten different GMM[34] models each having a different number of clusters (1, 3, ..., 19). Then we choose the best GMM by its AIC[35] score. This mechanism was inspired by [30]. The best GMM is set as the clustering function that is the output of this component. Each epoch the process is repeated and a new GMM selected.

II. **Intrinsic Reward Estimation component:** This component's pseudocode is shown in the algorithm 1.

Data: $history$ - history of sampled goals and resulting last states, cl - clustering function
Result: $ALP_c, \forall c \in clusters$ - per cluster intrinsic rewards

```
// Initialize the history of each cluster as an empty list.
h_c = [ ], ∀c ∈ {1..n_clusters};
// Performance recomputation
for epoch_rollouts ∈ history do
    p_c = [ ], ∀c ∈ {1..n_clusters};
    for rollout ∈ epoch_rollouts do
        c = cl(rollout.goal);
        perf = recompute_reward(rollout.goal, rollout.last_state);
        append(perf, p_c);
    end
    for c ∈ n_clusters do
        append(mean(p_c), h_c);
    end
end
// LP estimation
for c ∈ clusters do
    ALP_c = estimate_ALP(h_c) ;                    // by equation 8
end
```

Algorithm 1: Intrinsic reward estimation component

III. **Prior Construction component:**

Our prior takes the form of a masking distribution. It is constructed in the following steps:

1. Using cluster LPs, sample a cluster according to the probability defined in equation 11 where C is the number of clusters and T is a hyperparameter. This is a Multi-Armed Bandit with a 20% uniform prior. The hyperparmaeter T enables us to set how much priority we want to give to the high LP clusters, which is usefull if we have a lot of clusters with small LPs.

$$p(c) = \frac{4}{5}\frac{LP_c^T}{\sum_{i=1}^{C} LP_c^T} + \frac{1}{5}\frac{1}{C} \tag{11}$$

2. Construct the prior according to equation 12 where c is the sampled cluster, n_c is the number of goals from the buffer that are in cluster c, and cl is the clustering function. This is essentially a masking distribution giving uniform probabilities for goals inside the sampled cluster and zero for the rest. Another way of looking at this prior is saying what we sample a cluster and allow the underlying IMGEP to sample the goals only from this cluster.

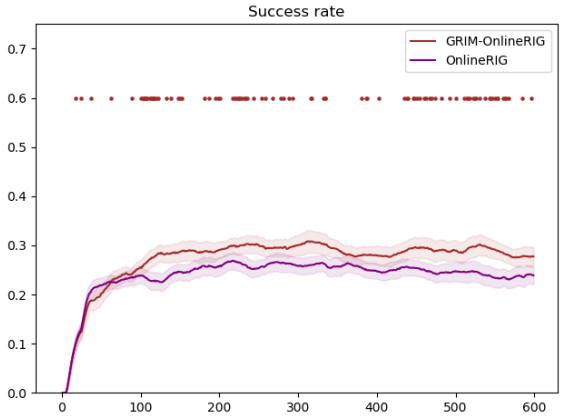
$$prior(g) = \begin{cases} \frac{1}{n_c}, cl(g) == c \\ 0, else \end{cases} \tag{12}$$

Result: trained goal conditioned policy

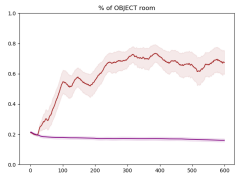
```
envs.reset();
// random rollouts to fill goal buffer
for t ∈ {0, 1, .., n_warmup} do
    trajs = envs.random_rollout();
    replay_buffer.add(trajs);
end

// exploration
for i ∈ {0, 1, .., N_epochs} do
    envs.reset();
    if i >start_prior then
        cluster_LPs = estimate_LPs(history) ;                    // by algorithm 1
    end
    for j ∈ {1, ..., num_parallel_goals} do
        if i >start_prior then
            cluster = sample_cluster(cluster_LPs) ;               // using eq. 11
            prior = construct_prior(cluster) ;                    // using eq. 12
            goal = IMGEP.sample_goal(prior, IMGEP.replay_buffer) ;  // using eq. 9
        else
            goal = sample_goal_uniformly(IMGEP.replay_buffer);
        end
        trajectory = IMGEP.rollout(goal);
        IMGEP.replay_buffer.add(trajectory);
        history.add(goal, trajectory[-1]);
    end
    IMGEP.train(replay_buffer) ;    // Updates the policy and the training VAE
    train_clustering_VAE(IMGEP.replay_buffer);
    cl=fit_new_clustering_function();
end
```

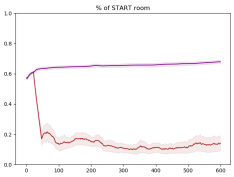
Algorithm 2: A pseudocode explaining the concrete manifestation of the GRIMGEP framework used in this work.



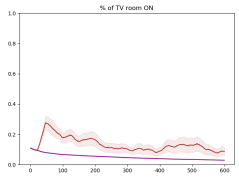
(a) Success rates



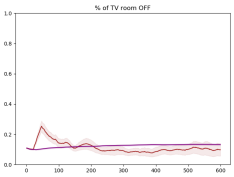
(b) Object room



(c) Start room



(d) TV room ON



(e) TV room OFF

Figure 5: Comparison of Online-RIG alone and in combination with the GRIMGEP framework. Ten seeds were used and the dots depict statistically significant ($p < 0.05$, Welch's t-test) results. The shaded areas correspond to standard errors and the bold line to the mean (smoothed over 25 epochs). The goal sampling percentages are the calculated on the goals sampled for both exploration and for replacement in HER (all approaches use HER and choose the replacement goals in the exact same way as for exploration).