Curiosity Driven Exploration of Learned Disentangled Goal Spaces

Adrien Laversanne-Finot
Flowers Team
Inria and Ensta-ParisTech, France
adrien.laversanne-finot@inria.fr

Alexandre Péré
Flowers Team
Inria and Ensta-ParisTech, France
alexandre.pere@inria.fr

Pierre-Yves Oudeyer
Flowers Team
Inria and Ensta-ParisTech, France
pierre-yves.oudeyer@inria.fr

Abstract: Intrinsically motivated goal exploration processes enable agents to explore efficiently complex environments with high-dimensional continuous actions. They have been applied successfully to real world robots to discover repertoires of policies producing a wide diversity of effects. Often these algorithms relied on engineered goal spaces but it was recently shown that one can use deep representation learning algorithms to learn an adequate goal space in simple environments. In this paper we show that using a disentangled goal space (i.e. a representation where each latent variable is sensitive to a single degree of freedom) leads to better exploration performances than an entangled one. We further show that when the representation is disentangled, one can leverage it by sampling goals that maximize learning progress in a modular manner. Finally, we show that the measure of learning progress, used to drive curiosity-driven exploration, can be used simultaneously to discover abstract independently controllable features of the environment.

Keywords: Goal exploration, Intrinsic motivation, Independently controllable features

1 Introduction

A key challenge of lifelong learning is how embodied agents can discover the structure of their environment and learn what outcomes they can produce and control. Within a developmental perspective [1, 2], this entails two closely linked challenges. The first challenge is that of exploration: how can learners self-organize their own exploration curriculum to discover efficiently a maximally diverse set of outcomes they can produce. The second challenge is that of learning disentangled representations of the world out of low-level observations (e.g. pixel level), and in particular, discovering abstract high-level features that can be controlled independently.

Exploring to discover how to produce diverse sets of outcomes. Discovering autonomously a diversity of outcomes that can be produced on the environment through rolling out motor programs has been shown to be highly useful for embodied learners. This is key for acquiring world models and repertoires of parameterized skills [3, 4, 5], to efficiently bootstrap exploration for deep reinforcement learning problems with rare or deceptive rewards [6, 7], or to quickly repair strategies in case of damages [8]. However, this problem is particularly difficult in high-dimensional continuous action and state spaces encountered in robotics given the strong constraints on the number of samples that can be experimented. In many cases, naive random exploration of motor commands is highly inefficient due to high-dimensional action spaces or redundancies in the sensorimotor system [3]. More involved methods that value states or actions in terms of novelty, information gain, or prediction errors are limited the presence of "distractors" that cannot be controlled [9, 10, 11, 12, 13].

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

One approach that was shown to be efficient in this context is known as Intrinsically Motivated Goal Exploration Processes (IMGEPs) [3, 14, 15], an architecture closely related to Goal Babbling [16]. The general idea of IMGEPs is to equip the agent with a goal space, where each point is a vector of (target) features of behavioural outcomes. During exploration, the agent samples goals in this goal space according to a certain strategy. A powerful strategy for selecting goals is to maximize empirical competence progress using multi-armed bandits [3]. This enables to automate the formation of a learning curriculum where goals are progressively explored from simple to more complex, avoiding goals that are either too simple or too complex. This approach has been shown to enable high-dimensional robots to learn locomotion skills [3], manipulation of soft objects [16, 17] or tool use [15]. Related approaches were recently experimented in the context of Deep Reinforcement Learning [18, 19, 20].

Learning disentangled representations of goal spaces. Even if these approaches have been shown to be very powerful, one limit has been to rely on engineered representations of goal spaces. For example, experiments in [3, 7, 15, 20, 18] have leveraged the availability of goal spaces that directly encoded the position, speed or trajectories of objects/bodies. A major challenge is how to learn goal spaces in cases where only low-level perceptual measures are available to the learner (e.g. pixels). A first step in this direction was presented in [21], using deep networks and algorithms such as Variational AutoEncoders (VAEs) to learn goal spaces as a latent representation of the environment. In simple simulated visual scenes this approach was shown to be as efficient as using handcrafted goal features. But [21] did not study what was the impact of the quality of the learned representation. Moreover, when the environment contains several objects including a distractor object, an efficient exploration of the environment is possible only using modular curiosity-driven goal exploration processes, by focusing on objects that provide maximal learning progress, and avoiding distractor objects that are either trivial or not controllable [14, 15]. An open question is thus whether it is possible to learn goal spaces with adequate disentanglement properties and develop exploration algorithms that can leverage those learned disentangled properties from low-level perceptual measures.

Discovering high-level controllable features of the environment. Although methods to learn disentangled representation of the world exist [22, 23], they do not allow to distinguish features that are controllable by the learner from features describing external phenomena that are outside the control of the agent. However, identifying such independantly controllable features [24] is of paramount importance for agents to develop compact world models that generalize well, as well as to grow efficiently their repertoire of skills. One idea to address this challenge, initially explored in [25], is that learners may identify and characterize controllable sets of features as sensorimotor space manifolds where it is possible to learn how to control perceptual values with actions, i.e. where learning progress is possible. Unsupervised learning approaches could then build coarse categories distinguishing the body, controllable objects, other animate agents, and uncontrollable objects as entities with different learning progress profiles [25]. However, this work only considered identifying learnable and controllable manifolds among sets of *engineered* features.

In this paper, we explore the idea that a useful learned representation for efficient exploration with IMGEPs would be a factorized representation where each latent variable would be sensitive to changes made in a single true dregree of freedom of the environment, while being invariant to changes in other degrees of freedom [26]. Further on, we investigate how independently controllable features of the environment can be identified among these disentangled variables through interactions with the environement. We study this question using β-VAEs [22, 27] which is a natural extension of VAEs and have been shown to provide good disentanglement properties. We extend the experimental framework of [21], simulating a robot arm learning how it can produce outcomes in a scene with two objects, including a distractor. As a **first contribution**, this paper demonstrates that using a disentangled state representation is beneficial to exploration: using IMGEPS, the agents explores more states in fewer experiments than when the representation is entangled. Furthermore, disentangled representations learned by β-VAEs can be leveraged by modular curiosity-driven IMGEPs to explore as efficiently as using handcrafted low-dimensional scene features in experiments that include both controllable and distractor objects. On the contrary, we show that representations learned by VAEs are not sufficiently structured to enable a similarly efficient exploration. As a **second contribution**, this article shows that identifying abstract independently controllable features from low-level perception can emerge from a representation learning pipeline where learning disentangled features from passive observations (β-VAEs) is followed by curiosity-driven active exploration driven by the maximization of learning

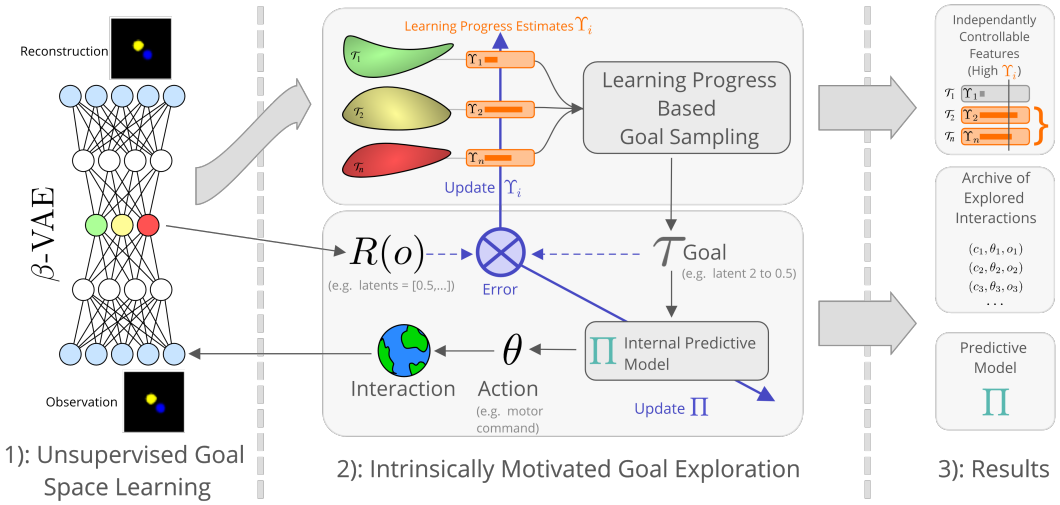


Figure 1: The IMGEP-MUGL approach.

progress. This second phase allows in particular to distinguish features related to controllable objects (disentangled features with high learning progress) from features related to distractors (disentangled features with low learning progress).

2 Modular goal exploration with learned goal spaces

This section introduces *Intrinsically Motivated Goal Exploration Processes* with modular goal spaces as they are typically used in environments with handcrafted goal spaces. It then describes the architecture used in this article where the handcrafted goal space is replaced by a representation of the space that is learned before exploration and then used as a goal space for IMGEPs. The overall architecture is summarized in Figure 1.

2.1 Intrinsically motivated goal exploration processes with modular goal spaces

To fully understand the IMGEP approach, one must imagine the agent as performing a sequence of contextualized and parameterized experiments. The problem of exploration is defined using the following elements:

- A context space $\mathcal{C}$. The context c represents the initial state of the environment. It corresponds to parameters of the experiment that are not chosen by the agent.
- A parameterization space Θ. The parameterization θ corresponds to the parameters of the experiment that the agent can control at will (e.g. motor commands for a robot).
- An observation space $\mathcal{O}$. Here we consider an observation o to be a vector representing all the signals captured by the agent sensors during an experiment (e.g. raw images).
- An environment dynamic $D : \mathcal{C}, \Theta \rightarrow \mathcal{O}$ which maps parameters performed in a certain context, to observations (or outcomes). This dynamic is considered unknown.

For instance, a parameterization could be the weights of a closed-loop neural network controller for a robot manipulating a ball. A context could be the initial position of the ball and an observation could be the position of the ball at the end of a fixed duration experiment. The exploration problem can then be simply put as:

> Given a budget of n experiments to perform, how to gather tuples $\{(c_i, \theta_i, o_i)\}_{i=1...n}$ which maximize the diversity of the set of observations $\{o_i\}_{i=1...n}$.

One approach that was shown to produce good exploration performances is Intrinsically Motivated Goal Exploration Processes. This algorithmic architecture uses the following elements:

- A goal space $\mathcal{T}$. The elements $\tau \in \mathcal{T}$ represent the goals that the agent can set for himself. We also use the term *task* to refer to an element of $\mathcal{T}$.
- A goal sampling policy $\gamma : \mathcal{T} \mapsto [0, 1]$. This distribution allows the agent to choose a goal in the goal space. Depending on the exploration strategy being active or fixed, this distribution can evolve during exploration.
- A Meta-Policy mechanism (or Internal Predictive Model) $\Pi : \mathcal{T}, \mathcal{C} \mapsto \Theta$, which given a goal and a context, outputs a parameterization that is most likely to produce an observation *fulfilling* the goal, under the current knowledge.
- A cost function $C : \mathcal{T}, \mathcal{O} \mapsto \mathbb{R}$, internally used by the Meta-Policy. This cost function outputs the fitness of an observation for a given task τ.

When the environment is simple, such as for experiments presented in [21] where a robotic arm explore its possible interactions with a single object, the structure of the goal space is not critical. However, in more complex scenes with multiple objects (e.g. including tools or objects that cannot be controlled), it was shown in [14] that it is important to have a goal space which reflects the structure of the environment. In particular, having a modular goal space, i.e. of the form $\mathcal{T} = \bigoplus_{i=1}^{N} \mathcal{T}_i$, where the $\mathcal{T}_i$ are different modules representing the properties of various objects, leads to much better exploration performances. In that case a goal can correspond to achieving an observation where a given object is in a given position.

The algorithmic architecture works as follows: at each step, the exploration process samples a module, then samples a a goal in this module, observes the context, executes a meta-policy mechanism to guess the best policy parameters for this goal, which it then uses to perform the experiment. The observation is then compared to the goal, and used to update the meta-policy (leveraging the information for other goals) as well as the module sampling policy. Depending on the algorithmic instantiation of this architecture, different Meta-Policy mechanisms can be used [3, 14]. In any case, the Meta-Policy must be initialized using a buffer of experiments $\{c_i, \theta_i, o_i\}$ containing at least two different o_i. As such, a bootstrap of several *Random Parameterization Exploration* iterations is always performed at the beginning. This leads to Algorithmic Architecture 1 . The reader can refer to Appendix 6.1 for a detailed explanation of the Meta-Policy implementation.

Algorithmic Architecture 1: Curiosity Driven Modular Goal Exploration Strategy

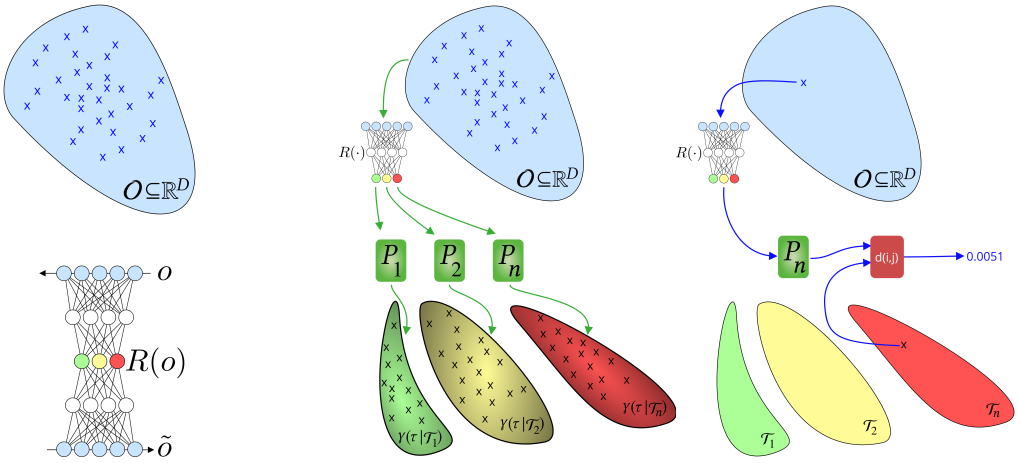
Input:
Goal modules (engineered or learned with MUGL): $\{R, P_i, \gamma(\cdot|i), C_i\}$, Meta-Policy Π, History $\mathcal{H}$

begin
  for *A fixed number of Bootstrapping iterations* **do**
    Observe context c
    Sample $\theta \sim \mathcal{U}(-1, 1)$
    Perform experiment and retrieve observation o
    Append (c, θ, o) to $\mathcal{H}$
  Initialize Meta-Policy Π with history $\mathcal{H}$
  Initialize module sampling probability $p = \mathcal{U}(n_{mod})$
  for *A fixed number of Exploration iterations* **do**
    Observe context c
    Sample a module $i \sim p$
    Sample a goal for module i, $\tau \sim \gamma(\cdot|i)$
    Compute θ using Meta-Policy Π on tuple (c, τ, i)
    Perform experiment and retrieve observation o
    Append (c, θ, o) to $\mathcal{H}$
    Update Meta-Policy Π with (c, θ, o)
    Update module sampling probability p according to Eq. (2)
return *The history* $\mathcal{H}$

In a modular architecture the goal sampling policy reads:

$$\gamma(\tau) = \gamma(\tau|i)p(i), \tag{1}$$

where $p(i)$ is the probability to sample the $\mathcal{T}_i$ module, and $\gamma(\tau|i)$ is the probability to sample the goal τ given that the module i was selected. The strength of the modular architecture is that modules can



(a) Representation learning on observations (Lines 2-5)

(b) Generation of projection operators and estimation of $\gamma(\tau|i)$ distributions (Lines 6-7)

(c) Cost function generation (Line 8)

Figure 2: The three main steps of the MUGL algorithm

be selected using a curiosity-driven *active* module sampling scheme. In this scheme, $\gamma(\tau|i)$ is fixed, and $p(i)$ is updated at time t according to:

$$p(i) := 0.9 \times \frac{\Upsilon_i(t)}{\sum_{k=1}^{N} \Upsilon_k(t)} + 0.1 \times \frac{1}{N}, \tag{2}$$

where $\Upsilon_i(t)$ is an *interest* measure based on the estimation of the average *improvement* of the precision of the meta-policy for fulfilling goals in $\mathcal{T}_i$, which is a form of learning progress called *competence progress* (see [3] and Appendix 6.1 for further details on the interest measure). The second term of Equation (2) forces the agent to explore a random module 10% of the time. The general idea is that monitoring the learning progress allows the agent to concentrate on objects which can be learned to control while ignoring objects that cannot.

2.2 Modular Unsupervised Goal-space Learning for IMGEP

In [21], an algorithm for *Unsupervised Goal-space Learning* (UGL) was proposed. The principle is to let the agent observe another agent producing a diversity of observations $\{o_i\}$. This set of observations is used to learn a low-dimensional representation which is then employed as a goal-space. In these experiments, there is always a single goal space corresponding to the learned representation of the environment. However, if one wishes to use the algorithm presented in the previous section, it is necessary to have different goal spaces: one for each module.

In order to use a Modular Goal Exploration strategy with a learned goal space, we propose Algorithm 2, which performs *Modular Unsupervised Goal-space Learning* (MUGL) and is represented in Figure 2. The idea is to learn a representation of the observations in the same way as UGL. The modules are then defined as subsets of latent variables. For example, a module could be made of the first and second latent variables. Accordingly, goals sampled by this module are defined as achieving a certain value for the first and second latent variables of the representation of an observation. The underlying rationale is that, if we manage to learn a *disentangled* representation of the observations, each latent variable would correspond to a single property of a single object. Thus, by forming modules containing only latent variables corresponding to the same object, the exploration algorithm may be able to explore the different objects separately.

After learning the representation, a specific criterion is used to decide how the latent variables should be grouped to form modules. In the particular case of VAEs and βVAEs, the grouping is made by projecting latent variables which have similar Kullback-Leibler on their respective subspace (see Appendix 6.1). Since representations learned with VAEs and βVAEs come with a prior over the latent variables, instead of estimating the modular goal-policies $\gamma(\tau|k)$, we used the Gaussian prior

Algorithm 2: Modular Unsupervised Goal-space Learning (MUGL)

Input:
Representation learning algorithm $\mathfrak{R}$ (e.g. VAE, βVAE), Kernel Density Estimator algorithm $\mathfrak{E}$

begin
  for *A fixed number of Observation iterations* n_r **do**
    Observe external agent produce observation o_i
    Append this sample to database $\mathcal{D}_o = \{o_i\}_{i=0,\dots,n_r}$
  Learn an embedding function $R : \mathcal{O} \mapsto \mathbb{R}^{n_d}$ using algorithm $\mathfrak{R}$ on data $\mathcal{D}_o$
  Generate an ensemble of projection operators $\{P_k\}$
  Estimate $\gamma(\tau|k)$ from $\{P_k R(o_i)\}_{i=0,\dots,n_r}$ using algorithm $\mathfrak{E}$
  Set the cost functions to be $C_k(\tau, o) = \|P_k R(o) - \tau\|$
return *The goal modules* $\{R, P_k, \gamma(\tau|k), C_k\}$.

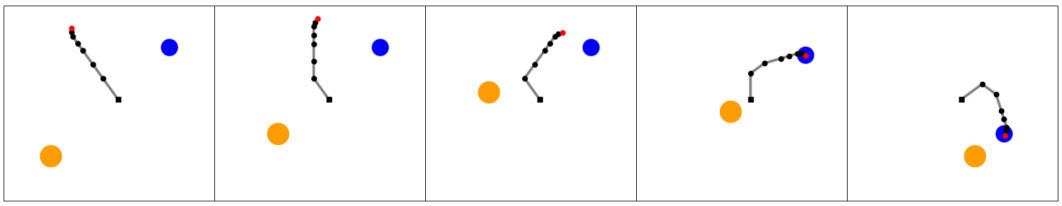


Figure 3: A roll-out of experiment in the *Arm-2-Balls* environment. The blue ball can be grasped and moved, while the orange one is a distractor that can not be handled, and follows a random walk.

assumed during training. Finally, the cost function $C_k(\tau, o)$ is defined, using the distance between the goal and the k-th projection of the latent representation of the observation.

The overall approach combining IMGEPs with learned modular goal spaces is summarized in Figure 1. Note that the algorithm proposed in [21] is a particular instance of this architecture with only one module containing all the latent variables. In this case there is no module sampling strategy, and only a goal sampling strategy. This specific case is here referred to as *Random Goal Exploration* (RGE).

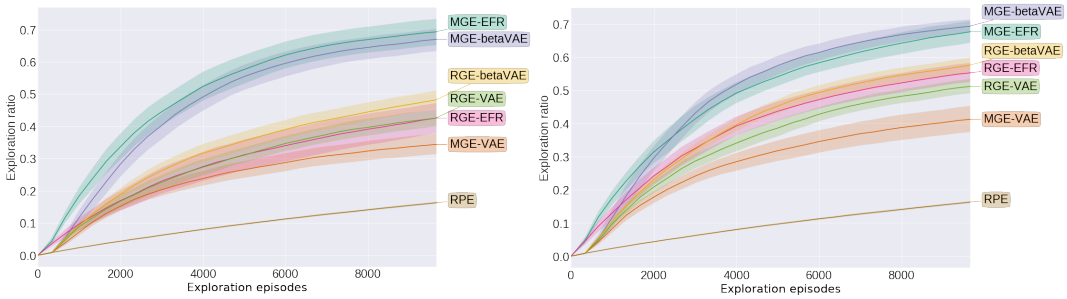
3 Experiments

We carried out experiments in a simulated environment to address the following questions[1]:

- To what extent is the structure of the learned representation important for the performance of IMGEP-UGL in terms of efficiently discovering a diversity of observations?
- Is it possible to leverage the structure of the representation with Modular Curiosity-Driven Goal Exploration algorithms?
- Can the learning progress measure of goal exploration be used to identify controllable abstract features of the environment?

Environment We experimented on the **Arm-2-Balls** environment, where a rotating 7-joints robotic arm evolves in an environment containing two balls of different sizes, as represented in Figure 3. One ball can be grasped and moved around in the scene by the robotic arm. The other ball acts as a distractor: it cannot be grasped nor moved by the robotic arm but follows a random walk. The agent perceives the scene as a 64×64 pixels image. For the representation learning phase, we generated a set of images where the positions of the two balls were uniformly distributed over $[-1, 1]^4$. These images were then used to learn a representation using a VAE or a βVAE. In order to assess the importance of the disentangled representation, we used the same disentangled/entangled representation for all the instantiations of the exploration algorithms. This allowed us to study the effect of disentangled representations by eliminating the variance due to the inherent difficulty of learning such representations.

[1] Code available at `https://github.com/flowersteam/Curiosity_Driven_Goal_Exploration`.



(a) Small exploration noise ($\sigma = 0.05$)

(b) Large exploration noise ($\sigma = 0.1$)

Figure 4: Exploration ratio during exploration for different exploration noises.

Baselines Results obtained using IMGEPs with learned goal spaces are compared to two baselines:

- The first baseline is the naive approach of **Random Parameter Exploration (RPE)**, where exploration is performed by uniformly sampling parameterizations θ. In the case of hard exploration problems, this strategy is regarded as a low performing one, since no previous information is leveraged to choose the next parameterization. This strategy gives a lower bound on the expected performances of exploration algorithms.
- The second baseline is **Modular Goal Exploration with Engineered Features Representation (MGE-EFR)**: it corresponds to a modular IMGEP in which the goal space is handcrafted and corresponds to the true degrees of freedom of the environment. In the *Arm-2-Balls* environment it corresponds to the positions of the two balls, given as a point in $[-1, 1]^4$. Since essentially all the information is available to the agent under a highly semantic form, it is expected to give an upper bound on the performances of the exploration algorithms. We performed experiments with both one module (**RGE-EFR**) and two modules (one for the ball and one for the distractor) (**MGE-EFR**).

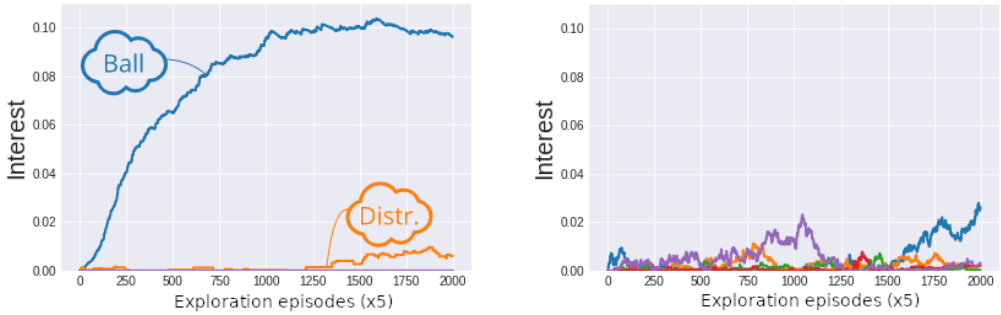
4 Results

To assess the performances of the **MGE** algorithm on learned goal spaces, we experimented with two different representations coming from two learning algorithms: β**-VAE** (disentangled) and **VAE** (entangled). In each case, we ran 20 trials of 10,000 episodes each, for both the **RGE** and **MGE** exploration algorithms. One episode is defined as one experimentation/roll-out of a parameter θ.

Exploration performances The exploration performance of all the algorithms was measured according to the number of cells reached by the ball in a discretized grid of 900 cells (30 cells for each dimension of the ball that can be moved; the distractor is not accounted for in the exploration evaluation). Not all cells can be reached given that the arm is rotating and is of unit length: the maximum ratio between the number of reached cells and all the cells is approximately $\pi/4 \approx 0.8$.

In Figure 4, we can see the evolution of the ratio of the number of cells visited with respect to all the cells through exploration. First, one can see that all the algorithms have much better performances than the naive **RPE**, both in term of speed of exploration and final performance. Secondly, for both **RGE** and **MGE** with learned goal spaces, using a disentangled representation is beneficial. One can also see that when the representation used as a goal space is disentangled, the modular architecture (**MGE-**β**VAE**) performs much better than the flat architecture (**RGE-**β**VAE**), with performances that match the modular architecture with engineered features (**MGE-EFR**). However, when the representation is entangled, using a modular architecture is actually detrimental to the performances since each module encodes then only partially for the ball position. Figure 4 also shows that the **MGE** architectures with a disentangled representation performs particularly well even if the exploration noise is low whereas the **RGE** architectures or **MGE** architectures with an entangled representation relies on a large exploration noise to produce a large variety of observations. We cross-refer to Appendix 6.7 for examples of exploration curves together with exploration scatters.

Benefits of disentanglement and modules



(a) Disentangled representation (βVAE)

(b) Entangled representation (VAE)

Figure 5: Interest evolution for each module during exploration. In the case of a disentangled representation the algorithm shows interest only for the module which correspond to latent variables encoding for the position of the ball (which is unknown by the agent, which does not distinguish between the ball and the distractor).

The evolution of the interest of the different modules through exploration is represented in Figure 5a. First, in the disentangled case, one can see that the interest is high only for the modules corresponding to the latent variables encoding for the ball position.[2] This is natural since these latent variables are the only ones that can be learned to control with motor commands. In the entangled case, the interest of each module follows a random trajectory, with no module standing out with a particular interest. This effect can be understood as follows: the entanglement introduces spurious correlations between the observations and the tasks in *every* module, which bring the interest measures to follow random fluctuations based on the collected observations. These correlations, in turn, lead the agent to sample more frequently policies that in fact did not have any impact on the observation, making the overall performance worse (see Appendix 6.1 and 6.6 for details).

Independently Controllable Features

As explained above and illustrated in Figure 5a, when the representation is disentangled, the MGE algorithm is able to monitor the learnability of certain modules (possibly individual latent features, see 6.5), and leverage it to focus exploration on goals with high learning progress. This is illustrated on the interest curves by the clear difference in interest between modules where learning progress happens and those where it does not. It happens that modules that produce high learning progress correspond precisely to modules that can be controlled. As such, as a side benefit of using modular goal exploration algorithms, the agent discovers in an unsupervised manner which are the features of the environment that can be controlled (and in turn explores them more). This knowledge could then be used by another algorithm whose performance depends on its ability to know which are the independantly controllable features of the environment.

5 Conclusion

In this paper we studied the role of the structure of learned goal space representations in IMGEPs. More specifically, we have shown that when the representation possesses good disentanglement properties, they can be leveraged by a curiosity-driven modular goal exploration architecture and lead to highly efficient exploration. In particular, this enables exploration performances as good as when using engineered features. In addition, the monitoring of learning progress enables the agent to discover which latent features can be controlled by its actions, and focus its exploration by setting goals in their corresponding subspace.

The perspectives of this work are twofold. First it would be interesting to show how the initial representation learning step could be performed online. Secondly, beyond using learning progress to discover controllable features during exploration, it would be interesting to re-use this knowledge to acquire more abstract representations and skills.

[2]The semantic mapping between latent variables and external objects is made by the experimenter.

Acknowledgments

We would like to thank Olivier Sigaud for helpful comments on an earlier version of this article. Experiments presented in this paper were carried out using the PlaFRIM experimental testbed.

References

[1] G. Baldassarre and M. Mirolli. *Intrinsically Motivated Learning in Natural and Artificial Systems*, volume 9783642323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-32374-4. doi:10.1007/978-3-642-32375-1.

[2] A. Cangelosi and M. Schlesinger. From Babies to Robots: The Contribution of Developmental Robotics to Developmental Psychology. *Child Development Perspectives*, feb 2018. ISSN 17508592. doi:10.1111/cdep.12282.

[3] 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.

[4] B. Da Silva, G. Konidaris, and A. Barto. Active learning of parameterized skills. In *International Conference on Machine Learning*, pages 1737–1745, 2014.

[5] T. Hester and P. Stone. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247:170–186, 2017.

[6] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*, 2017.

[7] C. Colas, O. Sigaud, and P.-Y. Oudeyer. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[8] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015.

[9] 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.

[10] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, 2017.

[11] A. G. Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[12] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*, 2017.

[13] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 2011. ISSN 10636560. doi:10.1162/EVCO_a_00025.

[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.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[16] 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. ISSN 19430604. doi:10.1109/TAMD.2010.2062511.

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

[18] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight Experience Replay. In *Nips*, jul 2017. URL http://arxiv.org/abs/1707.01495.

[19] J. Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313, 2013.

[20] C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

[21] A. Péré, S. Forestier, O. Sigaud, and P.-Y. Oudeyer. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. In *ICLR*, pages 1–26, 2018. URL http://arxiv.org/abs/1803.00781.

[22] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner. Early Visual Concept Learning with Unsupervised Deep Learning. *arXiv preprint arXiv:1606.05579*, jun 2016. URL http://arxiv.org/abs/1606.05579.

[23] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arXiv preprint arXiv:1606.03657*, 2016.

[24] V. Thomas, E. Bengio, W. Fedus, J. Pondard, P. Beaudoin, H. Larochelle, J. Pineau, D. Precup, and Y. Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1708.01289*, 2017. URL http://acsweb.ucsd.edu/{~}wfedus/pdf/ICF{_}NIPS{_}2017{_}workshop.pdf.

[25] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, apr 2007. ISSN 1089778X. doi:10.1109/TEVC.2006.890271.

[26] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. ISSN 01628828. doi:10.1109/TPAMI.2013.50.

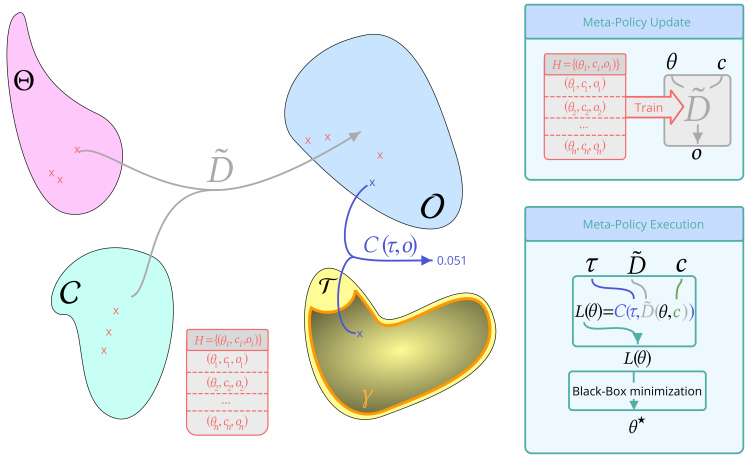
[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

[28] F. C. Y. Benureau and P.-Y. Oudeyer. Behavioral Diversity Generation in Autonomous Exploration through Reuse of Past Experience. *Frontiers in Robotics and AI*, 3(March), 2016. ISSN 2296-9144. doi:10.3389/frobt.2016.00008.

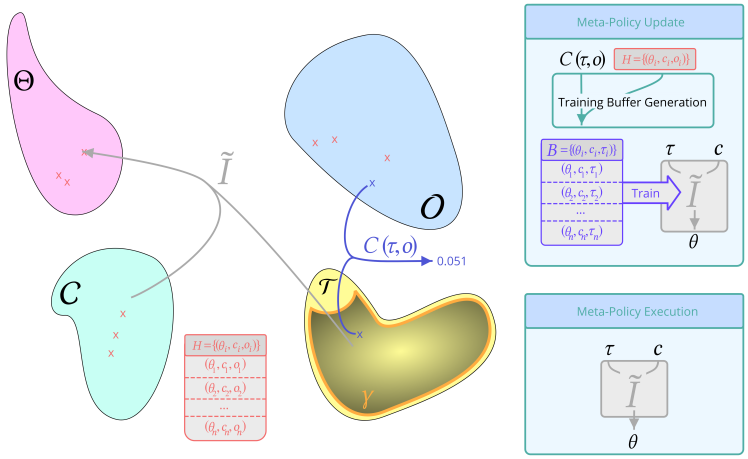
[29] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-Shot Visual Imitation. In *ICLR*, pages 1–12, 2018. URL http://arxiv.org/abs/1804.08606.

[30] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. In *Nips*, 2017.

[31] D. P. Kingma and J. L. Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.



(a) Direct-Model Meta-Policy



(b) Inverse-Model Meta-Policy

Figure 6: The two different approaches to construct a meta-policy mechanism.

6 Appendices

6.1 Intrinsically Motivated Goal Exploration Processes

In this part, we give further explanations on Intrinsically Motivated Goal Exploration Processes.

Meta-Policy Mechanism This mechanism allows, given a context c and a goal τ, to find the parameters θ that are most likely to produce an observation o *fulfilling* the task τ. The notion of an observation o *fulfilling* a task τ is quantified using a cost function $C : \mathcal{T} \times \mathcal{O} \mapsto \mathbb{R}$. The cost function can be seen as representing the fitness of the observation o regarding the task τ.

The meta-policy can be constructed in two different ways which are depicted in Figure 6:

- **Direct-Model Meta-Policy:** In this case, an approximate phenomenon dynamic model $\tilde{D}$ is learned using a regressor (e.g. LWR). The model is then updated regularly by performing a training step with the newly acquired data. At execution time, for a given goal τ, a loss function is defined over the parameterization space through $L(\theta) = C(\tau, \tilde{D}(\theta, c))$. A black-box optimization algorithm, such as L-BFGS, is then used to optimize this function and find the optimal set of parameters θ (see [3, 14, 28] for examples of such meta-policy implementations in the IMGEP framework).
- **Inverse-Model Meta-Policy:** Here, an inverse model $\tilde{I} : \mathcal{T} \times \mathcal{C} \mapsto \Theta$ is learned from the history $\mathcal{H}$ which contains all the previous experiments in the form of tuples (c_i, θ_i, o_i). To

do so, every experiments observations o_i must be turned into a task τ_i. The inverse model can then be learned using usual regression techniques from the set $\{(\tau_i, c_i, \theta_i)\}$.

In our case, we took the approach of using an Inverse-Model based Meta-Policy. We draw the attention of the reader on the following implementation details:

- Depending on the case, multiple observations, and consequently multiple parameters can optimally solve a task, while a combination of them cannot. This is known as the redundancy problem in robotics and special approaches must be used to handle it when learning inverse models, in particular within the IMGEP framework [3]. This has also been tackled under the terminology of multi-modality in [29]. To solve this problem, we used a κ-nn regressor with $\kappa = 1$.
- Turning observations $\{o_i\}$ into goals $\{\tau_i\}$ may prove difficult in some cases. Indeed, it may happen that a given observation does not solve optimally any task in the goal space, or that it solves optimally multiple tasks. In our case, we assumed that the learned encoder is a one-to-one map from observation space to goal space and thus, that every observation solves optimally a unique task in each module. Hence, tasks were associated to observations using the encoder R: $\tau_i := R(o_i)$.
- Since the different modules are associated to projection operators, each produced observation o optimally solve one task for each module. Indeed, if we consider projections on the canonical axis of the latent space, o will solve one task for each module, corresponding to each component of $R(o)$. This mechanism allows to leverage information of every single observation, for all goal-space modules. For this reason, one κ-nearest-neighbor model was used for each module of the goal space. At each exploration iteration all the modules are updated using their associated projection operators on the embedding of the outcome.

Our particular implementation of the Meta-Policy is outlined in Algorithm 3. The Meta-Policy is instantiated with one database per goal module. Each database store the representations of the observations projected on its associated subspace together with the associated contexts and parameterizations. Given that the meta policy is implemented with a nearest neighbor regressor, training the meta policy simply amounts to updating all the databases. Note that, as stated above, even though at each step the goal is sampled in only one module, the observation obtained after an exploration iteration is used to update all databases.

Algorithm 3: Meta-Policy (simple implementation using a nearest-neighbor model)

Require: Goal modules: $\{R, P_k, \gamma(\tau|k), C_k\}_{k\in\{1,..,n_{mod}\}}$
Function `Initialize_Meta-Policy(`$\mathcal{H}$`):`
  for $k \in \{1, .., n_{mod}\}$ **do**
    database$_k \leftarrow$ VoidDatabase
    for $(c, \theta, o) \in \mathcal{H}$ **do**
      Add $(c, \theta, P_k R(o))$ to database$_k$
Function `Update_Meta-Policy(`c, θ, o`):`
  for $k \in \{1, .., n_{mod}\}$ **do**
    Add $(c, \theta, P_k R(o))$ to database$_k$
Function `Infer_parameterization(`c, τ, k`):`
  $\theta \leftarrow$ NearestNeighbor(database$_k, c, \tau$)
  return θ

Active module sampling based on Interest measure Recalling from the paper, at each iteration, the probability of sampling a specific module $\mathcal{T}_i$ is given by:

$$\gamma(i) := 0.9 \times \frac{\Upsilon_i(t)}{\sum_{k=1}^{N} \Upsilon_k(t)} + 0.1 \times \frac{1}{N},$$

where $\Upsilon_i(t)$ represents the interest of the $\mathcal{T}_i$ module after t iterations. Let $\mathcal{H}_t^{(i)} = \{(\tau_k, \theta_k, P_i R(o_k))\}_{\tau_k \in \mathcal{T}_i}$ be the history of experiments obtained when the goal was sampled in

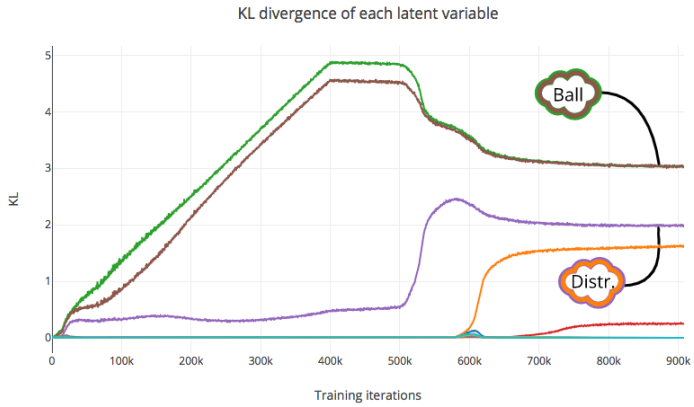


Figure 7: Kullback-Leibler divergence of each latent variable over training.

module $\mathcal{T}_i$. The progress in module i at exploration step t is defined as:

$$\delta_t^{(i)} = C_i(\tau_t, P_i R(o')) - C_i(\tau_t, P_i R(o_t)), \tag{3}$$

where o_t and τ_t are respectively the observation and goal for the current exploration step and o' is the observation associated to the experiment in $\mathcal{H}_t^{(i)}$ for which the goal τ' is the closest to τ_t. The interest of a module is designed to track the progress. Specifically, the interest of each module is updated according to:

$$\Upsilon_i(t) = \frac{n-1}{n}\Upsilon_i(t-1) + \frac{1}{n}\delta_t, \tag{4}$$

where $n = 1000$ is a decay rate that ensures that if no progress is made the interest of the module will go to zero over time. One can refer to [14] for details on this approach.

Projection criterion for VAE and βVAE An important aspect of the MUGL algorithm is the choice of the projection operators $\{P_k\}$. In this work, the representation learning algorithms are VAE and βVAE. In this case, two projection schemes can be considered:

- **Projection on all canonical axis:** n_d projection operators, each projecting the latent point on a single latent axis.
- **Projection on 2D planes sorted by $\mathbb{D}_{KL}$:** $\frac{n_d}{2}$ projection operators, each projecting on a 2D plane aligned with latent axis. The grouping of dimensions as 2D planes is performed by sorting the dimensions by increasing $\mathbb{D}_{KL}$, i.e. the divergence is computed for each dimension, by projecting the latent representation on the dimension and measuring its divergence with the unit gaussian prior. Latent dimensions are then grouped two by two according to their $\mathbb{D}_{KL}$ value.

In this work we mainly considered the second grouping scheme. The first grouping scheme could be considered to discover which features can be controlled. Of course in practice one often does not know in advance how many latent variables should be grouped together and it can be necessary to consider more advanced grouping schemes. In practice it is often the case that latent variables which correspond to the same objects have similar KL divergence value (see Figure 7 for an example of a training curve and appendix 6.2 for an explanation of this phenomenon). As such it could be envisioned to group latent variables which have similar KL divergence together.

6.2 Deep Representation Learning Algorithms

In this section we summarize the theoretical arguments behind Variational AutoEncoder (VAE) and βVAE.

Variational Auto-Encoders (VAEs) Let $\mathbf{x} \in \mathcal{X}$ be a set of observations. If we assume that the observed data are realizations of a random variable, we can hypothesize that they are conditioned by a random vector of independent factors $\mathbf{z}$, i.e. that $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{x}, \mathbf{z})$, where $p(\mathbf{z})$ is a *prior*

distribution over z and $p_\theta(\mathbf{x}, \mathbf{z})$ is a *conditional distribution*. In this setting, given a i.i.d dataset $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, learning the model amount to searching the parameters θ that maximizes the dataset likelihood:

$$\log \mathfrak{L}(\mathcal{D}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^i) \tag{5}$$

In most cases, the maximum likelihood estimation is computationally unfeasible. To overcome this problem, we can introduce an arbitrary distribution $q_\phi(\mathbf{z}|\mathbf{x})$. It is then easy to show that,

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})]. \tag{6}$$

is a lower bound of $p_\theta(\mathbf{x})$. The approach taken by VAEs is to *learn* the parameters of both conditional distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ that maximizes $\mathcal{L}(\mathbf{x}; \theta, \phi)$ over the dataset

β Variational Auto-Encoders (βVAEs) In essence, a VAE can be understood as an AutoEncoder with stochastic units ($q_\phi(\mathbf{z}|\mathbf{x})$ plays the role of an encoder while $p_\theta(\mathbf{x}|\mathbf{z})$ plays the role of the decoder), together with a regularization term given by the KL divergence between the approximation of the posterior and the prior.

Ideally, in order to be more easily interpretable, we would like to have a disentangled representation, i.e. a representation where a single latent is sensitive to changes in only one generative factor while being invariant to changes in other factors. When the prior distribution $p(\mathbf{z})$ is an isotropic unit Gaussian distribution ($p(\mathbf{z}) = \mathcal{N}(0, I)$) the role of the regularization term can be understood as a pressure that encourages the VAE to learn independent latent factors $\mathbf{z}$. As such, it was suggested in [22, 27] that modifying the training objective to:

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta\mathbb{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})], \tag{7}$$

where β is an additional parameter, will allow one to control the degree of applied pressure to learn independent generating factors by tuning the parameter β. In particular values of β higher than 1 should lead to representations with better disentanglement properties.

One of the drawbacks of βVAE is that for large values of β the reconstruction cost is often dominated by the KL divergence term. This leads to poor reconstructed samples where the model ignores some of the factors of variation altogether. In order to tackle this issue, it was further suggested in [30] to modify the training objective to be:

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta|\mathbb{D}_{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] - C|, \tag{8}$$

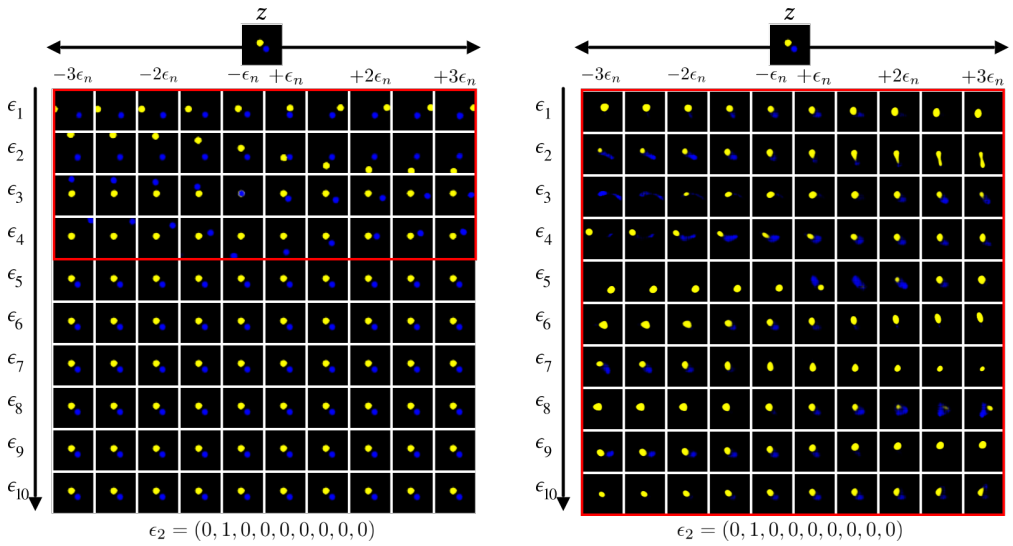
where C is a new parameter that defines the capacity of the VAE. The value of C determines the capacity of the network to encode information in the latent variables. For low values of the capacity the network will mostly reconstruct properties which have a high reconstruction cost whereas high capacity ensures that the network can have a low reconstruction error. By optimizing the training objective (8) with a gradually increased capacity the network will start to encode features with high reconstruction cost and then progressively encode more factors of variations whilst retaining disentangling in previously learned factors. At the end of the training one should thus obtain a representation with good disentanglement properties where each factor of variation is encoded into a unique latent variable.

In our experiments we used the training objective of Eq. (8) as detailed in Sec. 6.4.

6.3 Disentanglement properties

We compared the disentanglement properties of two representations. One with the procedure outlined in Sec. 6.2 with $\beta = 150$ and a capacity linearly increased to 12 over the course of the training. The other representation was a vanilla VAE with $\beta = 1$. In order to assess the disentanglement properties of the two representations we performed a latent traversal study. The results of which are displayed in Figure 8.

It was experimentally observed that the positions of the two balls were indeed disentangled in most cases when the representation was obtained using a βVAE even though the data used for the training was generated using independent samples for the position of the two balls. As explained in the previous section, this effect can be understood as follows: since the two balls do not have the same



(a) Disentangled latent representation learned by β**VAE** (b) Entangled latent representation learned with **VAE**

Figure 8: (a) Latent traversal study for a disentangled representation (βVAE). Each row represents a latent variable and rows are ordered by KL divergence (lowest at the bottom). Each row represents the reconstruction obtained from the traversal of each latent variable over three standard variation around the unit Gaussian prior mean while keeping the other latent variables to the value obtained by running inference on an image of the dataset. From the picture it is clear that the first two latent variables encode the x and y position of the Ball and that the third and fourth latent variables encode the x and y position of the Distractor. At the end of the training the remaining latent variables have converged to the unit Gaussian prior. (b) Similar analysis for an entangled representation (VAE). No latent variable encode for a single factor of variation.

reconstruction cost, the VAE tends to reconstruct the object with the highest reconstruction cost first (in this case the largest ball), and when the capacity reaches the adequate value, it starts reconstructing the other ball [30]. It follows that the latent variables encoding for the position of the two balls are often disentangled.

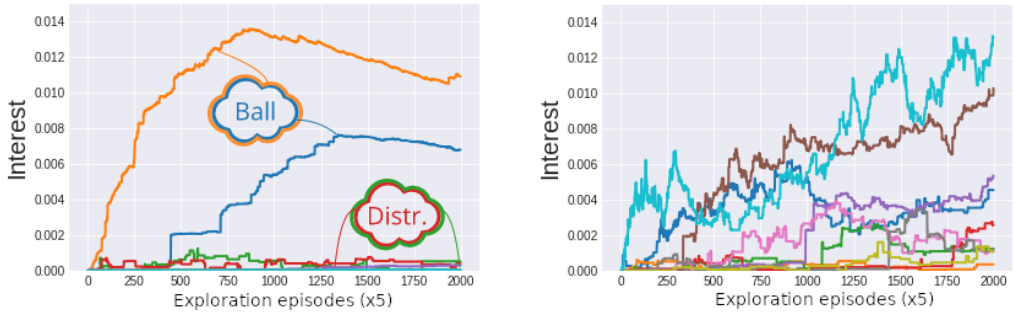
6.4 Details of Neural Architectures and training

Model Architecture The encoder for the VAEs consisted of 4 convolutional layers, each with 32 channels, 4x4 kernels, and a stride of 2. This was followed by 2 fully connected layers, each of 256 units. The latent distribution consisted of one fully connected layer of 20 units parametrizing the mean and log standard deviation of 10 Gaussian random variables. The decoder architecture was the transpose of the encoder, with the output parametrizing Bernoulli distributions over the pixels. ReLu were used as activation functions. This architecture is based on the one proposed in [22].

Training details For the training of the disentangled representation we followed the procedure outlined in Sec. 6.2. The value of β was 150 and the capacity was linearly increased from 0 to 12 over the course of 400,000 training iterations. The optimizer used was Adam [31] with a learning rate of $5e^{-5}$ and batch size of 64. The overall training of the representation took 1M training iterations. For the training of the entangled representation the same procedure was followed except that β was set to 1 and that the capacity was set to 0.

6.5 Interest curves for Projection on all canonical axis

In the main text of the paper we discussed the case of 5 modules. In general one can imagine having one modules per latent variable. In this case the agent would learn to discover and control each of the latent variables separately.



(a) Disentangled representation (βVAE) (b) Entangled representation (VAE)

Figure 9: Interest curves for Projection on all canonical axis

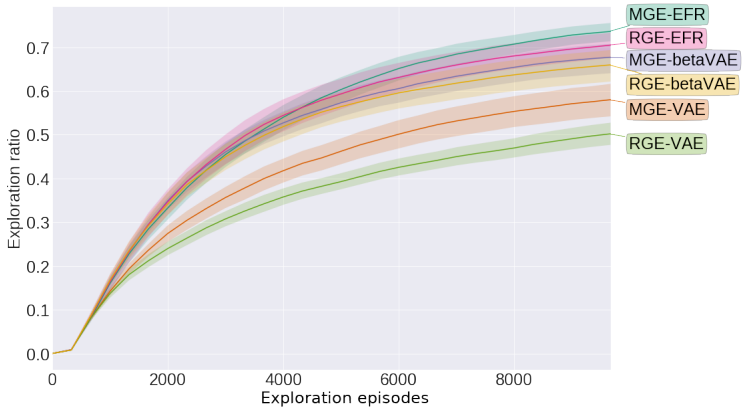


Figure 10: Exploration ratio through epochs for all the exploration algorithms in the *Arm-2-Balls* environment with a distractor that does not move.

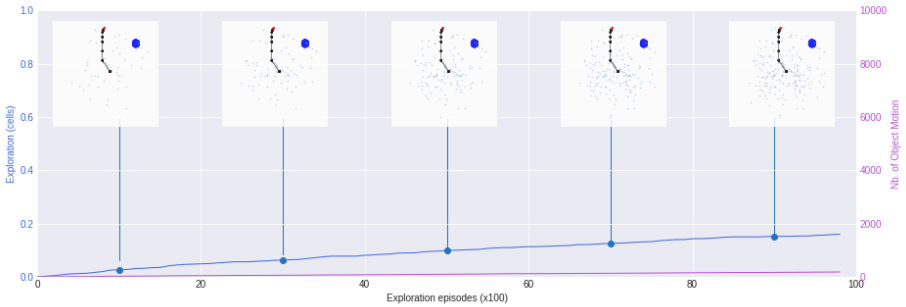
In Figure 9 is represented the interest curves when there are 10 modules, one for each latent variable. When the representation is disentangled (βVAE), the interest is high only for modules which encode for some degrees of freedom of the ball. On the other hand, when the representation is entangled, the interest follows some kind of random walk for all modules. This is due to the fact that all the modules encode for both the ball and the distractor position which introduces some noise in the prediction of each module.

6.6 Effect of noise in the distractor

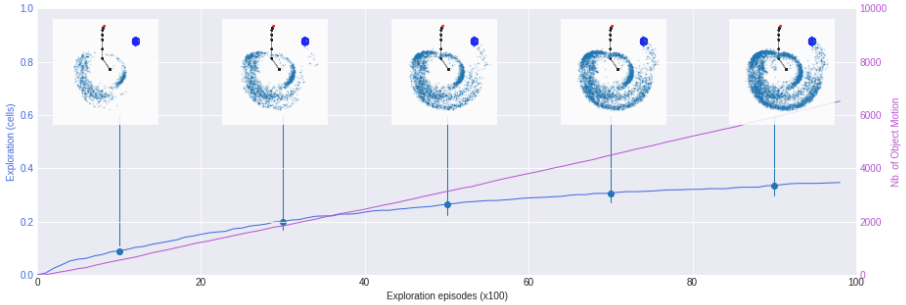
We also experimented with different noise level in the displacement of the distractor. As expected, when the noise level is low, the distractor does not move very far from its initial position and no longer acts as a distractor. In this case there is no advantage of using a modular algorithm as illustrated by Figure 10. However, it is still beneficial to have a disentangled representation since it helps in learning good inverse models.

6.7 Exploration Curves

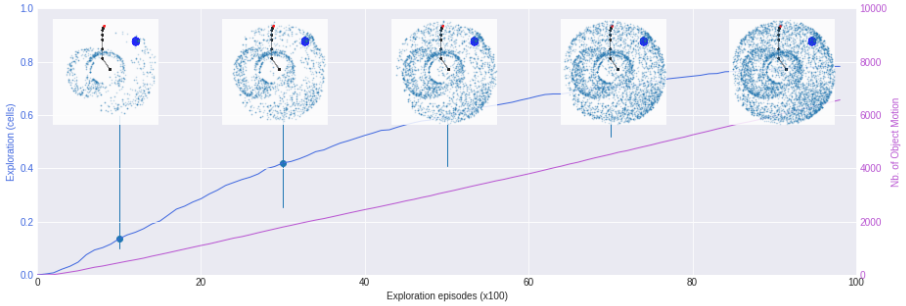
Examples of exploration curves obtained with all the exploration algorithms discussed in this paper (Figure 11 for algorithms with engineered features representation and Figure 12 for algorithms with learned goal spaces). It is clear that the random parameterization exploration algorithm fails to produce a wide variety of observations. Although the random goal exploration algorithms perform much better than the random parameterization algorithm, they tend to produce observations that are cluttered in a small region of the space. On the other hand the observations obtained with modular



(a) Random Parameterization Exploration



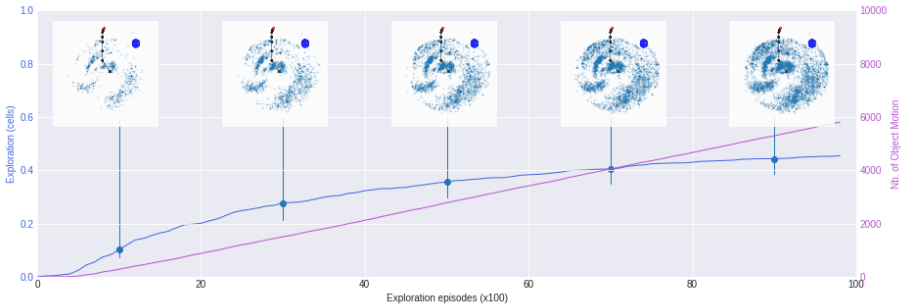
(b) Random Goal Exploration with Engineered Features Representation (RGE-EFR)



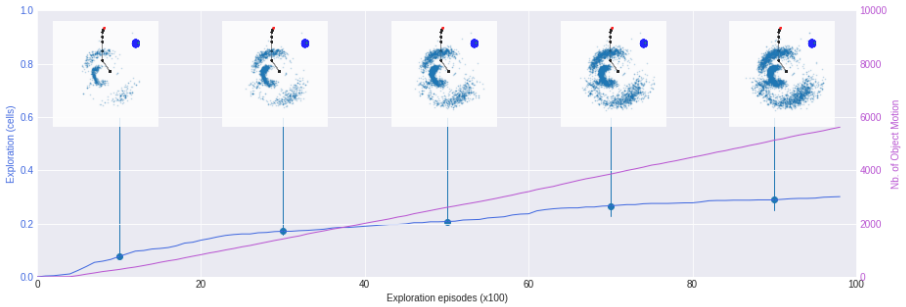
(c) Modular Goal Exploration with Engineered Features Representation (MGE-EFR)

Figure 11: Examples of achieved observations together with the ratio of covered cells in the *Arm-2-Balls* environment for **RPE**, **MGE-EFR** and **RGE-EFR** exploration algorithms. The number of times the ball was effectively handled is also represented.

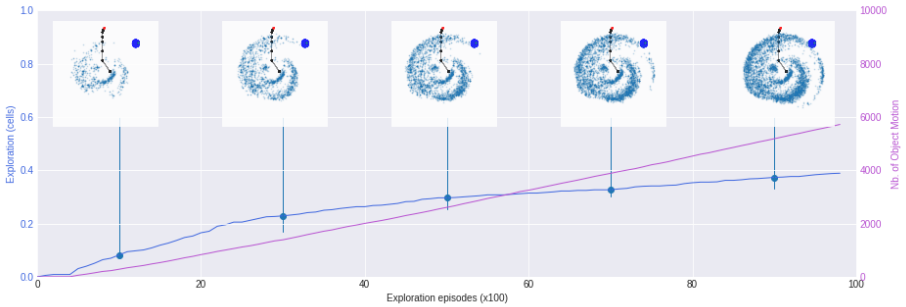
goal exploration algorithms are scattered over all the accessible space, with the exception of the case where the goal space is entangled (VAE).



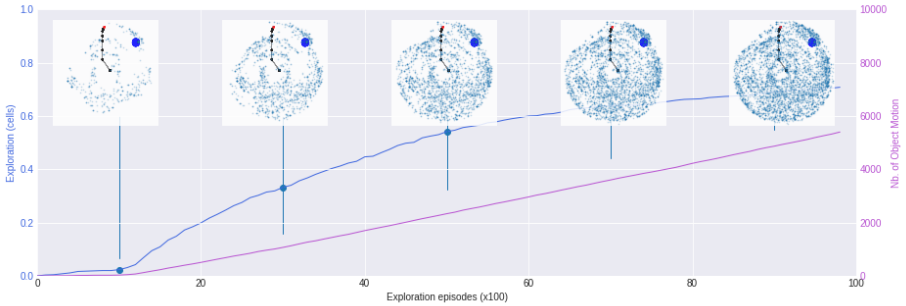
(a) Random Goal Exploration with an entangled representation (VAE) as a goal space (RGE-VAE)



(b) Modular Goal Exploration with an entangled representation (VAE) as a goal space (MGE-VAE)



(c) Random Goal Exploration with a disentangled representation (βVAE) as a goal space (RGE-βVAE)



(d) Modular Goal Exploration with a disentangled representation (βVAE) as a goal space (MGE-βVAE)

Figure 12: Examples of achieved observations together with the ratio of covered cells in the *Arm-2-Balls* environment for **MGE** and **RGE** exploration algorithms using learned goal spaces (**VAE** and β**VAE**). The number of times the ball was effectively handled is also represented.