SELF-IMPROVING LOGIC FROM EXPERIMENTAL OBSERVATIONS

October 1, 2024

ABSTRACT

Learning relevant, transferable representations of actions to drive model-based reinforcement learning processes stands as a major challenge in robotics on the path to general-purpose autonomous agents, equivalent in their reasoning power to Large Language Models. To this end, we introduce a novel framework which allows autonomous agents to learn how to represent their actions as high-dimensional rotations over the system's observations. We then show how such representations may be considered optimal under the assumption that actions are distance-preserving, and present how these representations of low-level actions can be composed to represent sequences of actions and allow for multi-scale hierarchical learning and long-horizon planning. We finally discussed schemes to compare such representations in order to allow for a better informed transfer of skills across tasks and better understand the agent's behaviour, before conducting experiments using a modified TD-MPC2 agent to better quantify $in\ concreto$ the limitations of our framework.

1 Introduction

Following recent advances in the field on Natural Language Processing, which allowed several Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020 (2)) or LLAMA2 (Touvron et al., 2023 (28)) to achieve near-human performance over a variety of language-based tasks, considerable effort was put into exploiting the implicit representations and world models contained within such models to build or enrich reinforcement learning datasets (Tiafas et al., 2024 (30), Zheng et al., 2024(35)) , to plan and direct low-level agents to help them tackle long-horizon or complex tasks (Zhang et al., 2023 (36), Sun et al., 2024 (25), Zhou et al., 2024 (37)), or to extract from low-dimensional observations adequate system state descriptors (Chen et al., 2023 (3)), for instance. However, such approaches face a dual challenge: while larger models are more precise and better able to adapt with minimal fine-tuning to a wide variety of downstream tasks; on the other hand, they are also expensive to run and use as part of reinforcement learning schemes, where hundreds of thousands or even millions of updates are often required to reach convergence on some tasks and where an agent is often required to plan or react in real-time.

In this paper, we propose an alternative to the general, humanly-understandable representations provided by LLMs in the context of predictive control and planning tasks. Instead of replicating the Transformer (Vaswani et al., 2017 (31)) architecture by learning sequences of words and their correspondence in a latent action skill space, we leverage experimental observations to first derive a set of useful primitive actions, and learn their representation as a group action over the agent's observation space. We then combine and refine these primitives into relevant longer sequences of actions in downstream learning to further improve and enrich the agent's skills and reasoning abilities with sequences of increasing complexity while preserving the predictive power of our action model.

In practice, we recast the problem of learning action representations and dynamics as an instance of the well-known Wahba problem (Wahba, G, 1965 (32)). This problem, originally set by the astrophysicist Grace Wahba, seeks to find an element of $SO(n)$ to optimally shift one set of coordinates to another, where each shift is defined using a noisy matching of basis vectors from each set. We in turn match the observed environment state before executing a given action to the observed state after performing this action. This allows us to derive the corresponding rotation as a representation of the action executed by the agent, to be used in downstream learning. Crucially, these representations only depend on d_Ω, the state observation space's dimension, as long as the same training environment is reused across tasks. They also require as little as $O(d_\Omega)$ observations to become uniquely optimal representations.

In our experiments, we implement our SILEO framework within a slightly modified TD-MPC2 model architecture to account for a finite set of actions instead of the high-dimensional hypercubes used in the original implementation. After a short pre-training designed to identify relevant actions through a clustering of the embeddings provided by the agent's policy, we calculate their relevant representations as rotations and reuse them as a set of primitive skills. These action primitives equate to a fixed embedding in the agent's initial action or command space. In downstream learning, we then use these representations as the dynamics model of a Model Predictive Control planning algorithm, and refine these representations with new observations while maximising the entropy of our action set. This is achieved by replacing overused skills with the most useful sequences of skills starting with them. To summarise our main contributions:

Action Representations We present a novel scheme to represent actions in model predictive control as learned rotations over either an observation space or a learned state latent space, alongside the conditions required for these representations to be near-optimal

Multi-scale Hierarchical Learning We propose a method to refine during downstream learning a given set of action representations into longer sequences, to be used as higher-level skills in planning

Transfer Learning We show that learned action representations may be reused across tasks and agents while mitigating training performance loss, while remaining useful to evaluate the relative complexity and similarity of one task compared to another

Long Horizon planning We put forward several ways to extend in the future the low-level action-based SILEO framework implemented in our experiments to subtasks. We then show how long-horizon planning may then be recast as a (multiscale) factoring problem.

2 Preliminaries

Problem formulation This work is built under the theoretical framework of Partially Observable Markov Decision Processes, whose resolution aims at optimising under a certain reward function the sequence of actions taken by an agent from a given state, under the assumption that the decision-making agent only has access to noisy observations of these states. Formally, a POMDP is a 7-uple $(S, A, R, T, \Omega, O, \gamma)$, where

- S is a set of states
- A is a set of actions
- $R : S \times A \to \mathbb{R}$ is the reward
- $T : S \times A \to S$ is a random variable representing state transitions
- Ω is a set of observations
- $O : S \to S$ is a random variable representing state observations
- $\gamma \in [0, 1)$ is the discount factor

As with all Markov Decision Processes, at each time step t the agent is expected to take an action $a \in A$ to move from a lower-reward state $s_t \in S$ to a hopefully higher-reward state s_{t+1}. However, without knowledge of the ground truth state at this time step the agent only has access to a random observation $\omega_t \in \Omega$, given by the value of $O(s_t)$. Moreover, the transition from an observed state and the action chosen to the ground truth state at the next time step is also random, represented by the transition random variable T: $\Omega \times A \to S$.

Assumptions In this work, we assume that both conditional random variables T and O follow a Gaussian probability distribution. This assumption allows us to merge both Gaussian noises as one, that is, to consider a unified random variable $\tau : S \times A \to S$ defined as $\tau = T \circ (O \otimes (Id_A))$, which as the composition of two Gaussian random variables also follows a Gaussian probability distribution. This variable directly maps a ground truth state and action to another ground truth state without having to consider individually the output of O and then T.

We further assume that the set of observations $O \subset \Omega$ can be embedded in a finite-dimensional real vector space, with dimension d. While this assumption is standard in a control setting, we further assume that this embedding is entirely located in the d-dimensional unit sphere S^{d-1}. While this may seem egregious, since any compact subset of $\mathbb{R}^{d-1}$ is homeomorphic to a subset of S^{d-1} (see Appendix 1.1 for a detailed proof), this condition can be met by any compact Hausdorff embedding as long as it is projected to a higher dimension for our computation.

Finally, we assume that notwithstanding a Gaussian noise given by τ, the effect of each action may be represented globally as an endomorphism of S^{d-1}, $id\ est$ a matrix from the orthogonal group $O(d)$ and even $SO(d)$. Under our previous assumptions, this is equivalent to assuming that the actions over the embedded observation space are both quick to execute and mostly distance-preserving (see Appendix 2 for a rigorous proof), which while reasonable in a control setting may not be applicable in a context where significant external forces (e.g gravity) or numerous collisions, in particular inelastic, are to be expected.

Wahba's problem In this work, action representations rely on a solution to Wahba's problem provided by Ruiter and Forbes in 2014 (22). This least-squares problem seeks to find an optimal rotation matrix between two matching sets of (weighted) noisy coordinate vectors. Here, we recast the problem using system observations before an action was performed as a coordinate set, and system observations after this action as the second, to deduce a representation of the action as the best rotation matrix to match these observations. Under our assumptions, notably since τ is Gaussian, we may then find optimal solutions to each instance of Wahba's problem using SVD-derived methods.

In more precise terms, given an action $a \in A$, and $N > 1$ d-dimensional observations $(\omega_k, \omega'_k)_{1 \leq k \leq n}$ such that for all k, there has been a time step when $\tau(\omega_k, a) = \omega'_k$, we seek to find $R_a \in SO(d)$ that minimises the following reconstruction cost:

$$J(R_a) = \frac{1}{2} \sum_{k=1}^{N} \|\omega'_k - R\omega_k\|^2$$

3 Approach

In this section, we introduce in detail our SILEO framework, and its applications to Reinforcement Learning tasks and foremost control; we present both the theoretical advantages and potential drawbacks of the presented framework for dynamics and model-based planning (section 3.1), multi-level hierarchical learning from the composition of primitive and composite actions (section 3.2), transfer learning and cross-task comparison using geodesic distance (section 3.3) and long horizon planning in controlled environments (section 3.4). Implementation details and results are further presented in section 4.

3.1 Representing actions by their dynamics

In prior work, notably derived from SAC (Haarnoja et al., 2018 (6)) or DRQ (Kostrikov et al., 2020 (13)) agents such as PEARL (Rakelly et al., 2019 (20)) or TD-MPC (Hansen et al., 2022 (7)), actions are identified by their embeddings in a latent action space of arbitrary dimension, usually within a hypercube centered in 0. While a bijective interpretation function is used to translate these embeddings into concrete actions realised by an agent's actuators, there is no global equivalent at the reasoning level to be used by the agent. The latent action space is only used at a local level by neural networks tasked with predicting the next latent state or an expected reward when given as input an observation and an action for instance.

Unlike with words or pictures in the case of Transformer-derived agents however, there is intuitively a topological group structure that can be derived from this set of low-level actions in a control setting. Indeed, though this group and its structure may be hard to assess directly, most actions taken by an agent in control benchmarks such as RLBench (James et al., 2019 (12)), ALFREd (Shridhar et al., 2020 (24)), Meta-World (Yu et al., 2019 (34)) or the DeepMind Control Suite (Tassa et al., 2018 (27)) are invertible, often by simply reversing the command sequence at the actuator's level. Moreover, most actions can be continuously composed with one another without breaking the rules of the training environment, and the bijection between latent action space and actuator command is designed to be continuous and even an homeomorphism. However, this group structure is computationally intractable to determine, since even if we consider only a set of low-level actions as generators, simply chaining two actions one after the other requires us to consider the result in a space larger than the initial actuator command space to account for the fact that the resulting action is no longer a low-level actuator command, which leads recursively to an indefinitely large action space.

It is therefore natural to try and devise a way to consider these actions taken from an inaccessible group as group actions, more precisely to consider instead of the group itself a representation of this topological group on the observation space. Let us recall that given a group G, a representation (ρ, V) of this group is comprised of a vector space V and a morphism $\rho : G \to Aut(V)\ GL(V)$. Assuming that actions $a \in A$ over the observation space Ω from our PO-MDP have this topological group structure, a faithful (that is, injective) representation would allow us to identify these actions with elements of $Aut(\Omega)$. Under our assumptions stated in the preliminaries, $Aut(\Omega)$ would be $O(d)$, and our actions may even be represented by $SO(d)$ as shown in Appendix 1, where $d = dim(\Omega)$.

In order for an agent to learn this faithful representation from its experience interacting with its environment, we can leverage our framework by recasting the problem as an instance of Wahba's problem and treating each action as a shift with Gaussian noise from one system of coordinates in the observation space Ω to another. Since under our assumptions a given action from the latent action space acts on the observation space using an element of $SO(d)$, an agent can observe its effects on the system through a replay buffer or with online experiments and deduce, for instance using Singular Value Decomposition-derived methods(22), a solution to this instance of Wahba's problem with as little as $O(d)$ observations if they are sampled independently at random. Indeed, with high probability $O(d)$ observations form a basis of the latent d-dimensional observation space Ω, forming the d-dimensional basis required by the method presented in by de Ruiter et al., 2013 (22) to find the unique optimal rotation on S^d that represents our given action.

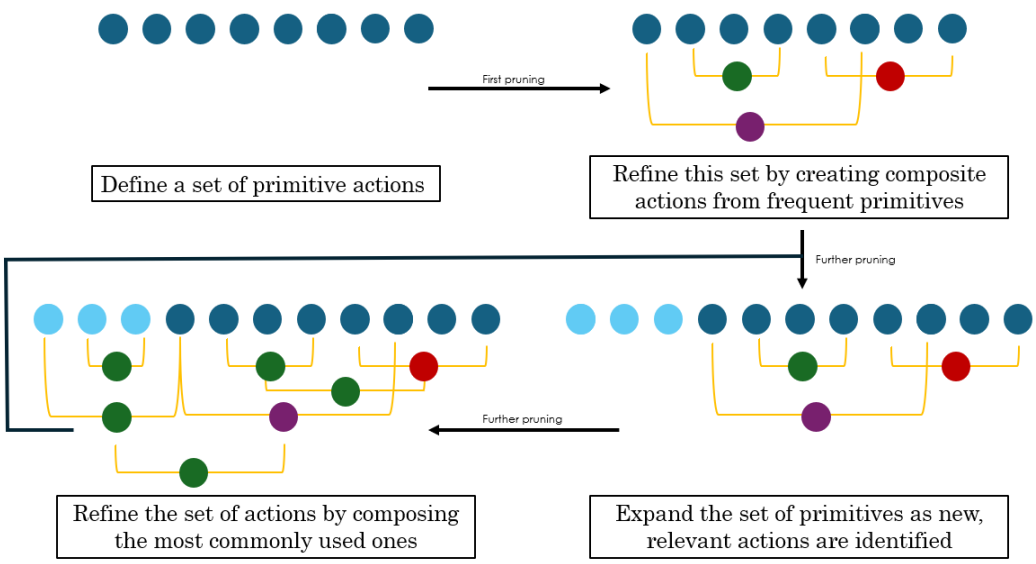

Figure 1: Repeated pruning to refine an action set

There remains the problem however of identifying relevant actions to which this method could be applied. While some RL agents such as SPiRL (Pertsch et al., 2020 (18)) or ReSkill (Rana et al., 2022 (21)) are trained to optimise the use of a fixed set of skills, these are expensive to develop since they usually require large datasets modeled after expert demonstrations. Other self-supervised agents use a continuous hypercube or hyperball as a latent action space instead of a discrete set of embeddings. In our implementation, this space needed to be discretised and clustered to provide a generating set of actions. Since in many self-supervised exploration implementations (such as TDMPC2 (Hansen et al., 2024 (8)) or Plan2Explore (Sekar et al., 2020 (23)) actions are at least initially uniformly distributed in the absence of any prior, a pre-training phase is therefore required for task-relevant low-energy clusters to emerge. Indeed, with uniformly distributed actions, the only meaningful choice for a set of primitive actions would be the center of the ball or the hypercube, oftentimes the null action, while local minima would be partitions in equivalent volumes of the entire space. In our experiments, we used a K-Medoids algorithm after a pre-training phase to choose the most statistically relevant actions for the agent to use for the rest of its training.

3.2 Self-improving logic

One of the main benefits of our SILEO framework lies in the ability to consider the global dynamics of actions instead of considering these dynamics on a per-state basis. This allows us to efficiently generalise but also refine and build new skills using global operations instead of considering trajectories built from pointwise dynamics. In our framework, composing one action with another is therefore as simple as multiplying the two rotation matrices that represent them. This enables second-order logic operations for an agent, which not only learns to manipulate actions but also discovers how they are related as either "parents" or "continuations" during its training. Finally, we draw inspiration from Natural Language Processing and notably tokenisation processes used by Large Language Models, to define entropy maximisation (following Wen et al., 2024 (33)) as a relevant criterion for refining the skillset learned by the agent.

Following Sutton et al., 1999 (26), a policy $\pi : \Omega \to A$ may be trained to predict the best action to take following a given observed state $\omega \in \Omega$. However, this policy is biased by design, since its intent is to *continuously* assign to latent states an action to take, which may lead to imbalance or approximations if some actions are much more frequently found than others in a given dataset regardless of their relevance to a given task. Instead, we propose to balance our actions and maximise the entropy of our action set by splitting frequently found actions into composite ones of greater duration, and to complement the policy with reward estimation in order to differentiate skills born from the same primitive action. The algorithm and its detailed explanation are found on the next page.

Note: In our experiments, $CLUSTER$ was performed using K-medoids, and $EXTRACT$ extracted the transitions observed after the agent performed the action $Centroid$. $BUILDACTION$ is the constructor of an Action class, which notably stores a rotation calculated from the transitions stored in $CentroidBuffer$. This class also has a $GetOccurrences$ method to fetch the number of (recent) occurrences stored with each instance, and a $ComposeAction(a, b)$ method, which yields the representation of $b \circ a$ from Actions a and b.

Algorithm 1 Pruning algorithm

Require: a buffer B of successive observations, a (sometimes empty) set of processed actions L, a policy $\pi : \Omega \rightarrow A$, a $criterion$ to add actions to L, a $threshold$ to start searching for composite actions.
Initialise $Actions$ as an empty list.
for $\omega \in B$ **do**
  Append $\pi(\omega)$ to $Actions$ ▷ Actions stored alongside the states in the buffer can also be used
end for
$Assignments, Centroids \leftarrow$ CLUSTER($Actions$)
for $Centroid$ in $Centroids$ **do**
  if $Centroid$ verifies $criterion$ **then** ▷ We used frequency and distance to the closest action as criteria
    $CentroidBuffer \leftarrow$ EXTRACT(Centroid, Assignments, Buffer)
    Append BUILDACTION($Centroid, CentroidBuffer$) to L
  end if
end for
Initialise $Occurrences$ as an empty list.
for $a \in L$ **do**
  Append GETOCCURRENCES(a) to $Occurrences$
end for
while $\exists a \in L$ s.t. $Occurrences(a) > threshold$ **do** ▷ e.g > mean + 2 standard deviations of Occurrences
  for b in L **do**
    $c \leftarrow$ COMPOSEACTION(a, b)
    if c verifies $criterion$ **then**
      Append c to L
      Append GETOCCURRENCES(c) to $Occurrences$
      $Occurrences(a) \leftarrow Occurrences(a) - Occurrences(c)$
    end if
  end for
end while

We call this algorithm a skills pruning algorithm, since it essentially cuts skills that overshadow others to allow for smaller, more balanced branches to regrow in their stead. Its goal is to produce skills of varying complexity but roughly equivalent in terms of frequency. It uses as roots the skills already learned by the agent, and considers for frequent skills which other skills, composite or not, would be the most likely continuations to this skill. It then creates new skills from the composition of our root skill and the most rewarding continuations, for them to be considered alongside others in downstream learning. A policy becomes mostly irrelevant in the setting of an expanding skills graph, although it does help restraining the search scope in the resulting graph; what matters however is consistent reward estimation across different depths of prediction resulting from the chaining of long or short skills in planning.

The resulting set of learned skills or actions may be represented as a graph, using as roots one of the primitive actions. While initially the graph is a disjoint set of small trees, after further pruning connected components merge and form a single connected graph. This graph may be analysed using different primitive skills as roots to explore the "transitions", that is the inheritance links from a parent skill/node to a more complex descendant, using methods such as CEM planning (Huang et al., 2021 (11)) or PETS (Chua et al., 2018 (4)), to ensure the most relevant skill is chosen by the agent at a given state and time step. This method allows for an agent to learn how to balance its different logic and planning scales, since under a given planning horizon of n actions, it can learn to use actions of varying length or complexity, where the choice of more complex skills translates greater mastery of and confidence in an action sequence.

This algorithm remains sensitive on the time of pruning however; indeed, agents which are more proficient than others at a given task usually have better defined clusters of actions and thus less action primitives and more composite skills. The repeated pruning of a graph during training however allows an agent to adapt to its most recent experience by integrating new and often more complex composite skills to its repertoire, reducing the total number of hyperparameters in the algorithm to 3: pruning frequency, accepted deviation from mean or median and the size of the buffer to be considered while pruning. These ensure the long-term viability of the hierarchical learning undertaken by the agent to ensure relevant composite skills are learned even among changing environmental conditions.

3.3 Transfer learning and task distance

In the context of transfer learning, one of the main hurdles encountered in PO-MDPs is that policies are often non-transferable across tasks. Indeed, since they map latent states to the estimated best action to take if the agent founds

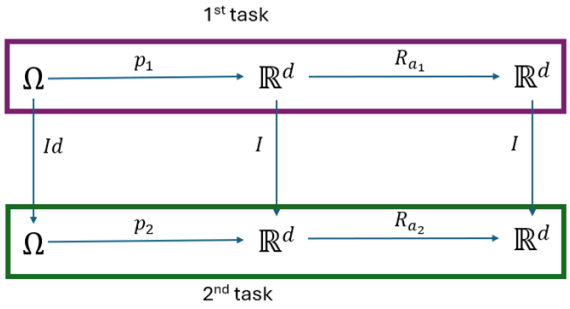

Figure 2: Matching different state embeddings

itself in one of these states to complete a task, they are highly dependent on both the encoding algorithm used to transform observations into latent states and the task itself. Crucially, even though "task-agnostic" dynamics models may be jointly trained or pre-trained to supplement an agent's decision-making process, they often remain dependent on learned embedded states even though an optimal embedding for a task may be detrimental to another.

In our framework, action dynamics may be learned directly from experimental observations, independently of any latent state embedding scheme that may be used for reward estimation or policies. Beyond the computational advantage that comes with using matrices of smaller dimension, using our framework with state observations allows the agent to learn a truly task-agnostic dynamics model, that may be reused across tasks without requiring any adaptation, as long as the actuator and its observation space remain unchanged. Moreover, the topological group structure chosen, which considers actions as a subgroup of $SO(d)$, allows us to consider a wide array of metrics to analyse the learning process of an agent. A variety of distance functions may for instance be used to compare two given actions and better understand the set of skills learned by the agent. These distances notably include the canonical geodesic distance, as defined below:

$$\text{If } A, B \in SO(d), A^T B \text{ too, hence } \exists (\lambda_i)_{1 \leq i \leq d} \in]-\pi; \pi] \text{ s.t. } A^T B \sim diag((e^{-i\lambda_i})_{1 \leq i \leq d})$$
$$\text{We define } d(A, B) = \sqrt{\sum_{i=1}^{d} \lambda_i^2}$$

This geodesic distance is by definition invariant under orthonormal base changes, which allows it to remain relevant even when comparing actions learned using different latent spaces instead of the agent's observation space. Indeed, while embedding functions are usually not isometric, which makes these actions not directly comparable, it is still possible to approximate one basis of latent states by another. More precisely, given two tasks and $p_1, p_2 : \Omega -> \mathbb{R}^d$ the embedding functions used by the agents for these tasks, we seek $I \in O(\mathbb{R}^d)$ (if applicable, $p_1^{-1}(p_2)$) that makes the diagram in Figure 2 commutative.

Given R_{a_1}, R_{a_2} the rotation matrices associated with two distinct actions a_1 and a_2 used by the agent to solve tasks 1 and 2 respectively, it is therefore possible (even if using latent spaces instead of observation spaces) to directly compute (or approximate, see appendix 1.5.2 for the full process) a task-agnostic distance function between these matrices as a proxy for the distance between these actions. This distance in turn allows us to determine a lowest-energy matching between the sets of relevant actions learned by the agent to solve both tasks and to define the energy of such a matching as the distance between two tasks. Moreover, while such a distance could already be computed from the embeddings of the discrete set of primitive actions chosen for both tasks, the process we described extends to composite actions or skills, and also takes as a basis of comparison the empirical dynamics of the commands used by an agent instead of their semi-arbitrary latent representations.

3.4 Guided exploration and long-horizon planning

Using rotations as representations for an agent's actions also allows it, on slightly modified observation spaces, to plan a trajectory without needing external rewards or policies. Indeed, given a starting region $O \in \Omega$ and a target region $C \in \Omega$, and a matching of $d = dim(\Omega)$ points $(o_i, c_i)_{0<i<d-1}$ (both independent and with unit norm) from both regions, we may consider as for low-level actions a rotation matrix $S \in SO(d)$ which seeks to verify for all i, $S(p(o_i)) = p(c_i)$, where p is an homeomorphism between $\mathbb{R}^d$ and $\mathbb{R}^{d+1}$. We can therefore deduce a rotation matrix to represent a given task, and even any sub-tasks which we would deem relevant from the state transitions desired.

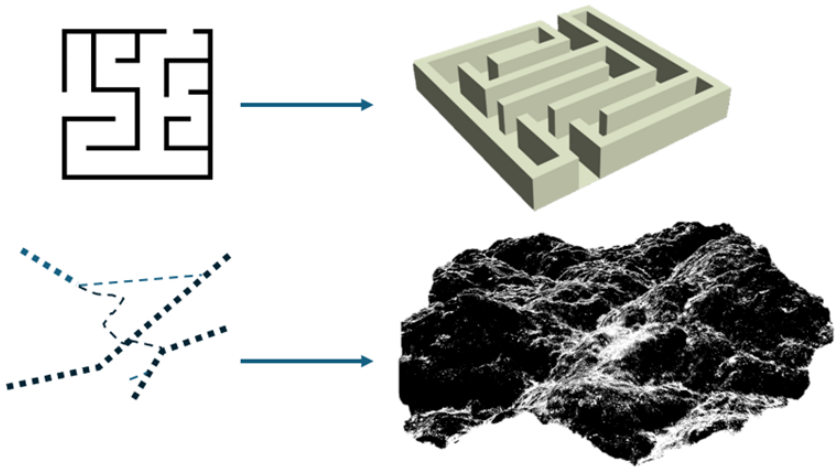

Figure 3: Using known geometry or empirical trajectories to warp the observation space

Within our framework, assuming that some actions have been learned, the process of finding a solution to a task or a sub-task becomes equivalent to finding that of an approximate factoring problem, where a planner would attempt to reconstruct the task's rotation matrix using the subtasks or low-level actions at its disposal, represented by their rotations. Such a problem is NP-hard in the general case, as if even if we consider simultaneously reducible rotations as actions, solving this problem becomes equivalent to solving a multidimensional knapsack problem as presented in Püchinger et al., 2010 (19) by using the eigenvalues' rotation angles as weights. A simpler version of this problem limiting itself to reconstructing a task from a set of sub-tasks may be solved however in a small enough time to be viable. Such an approach would be an alternative to LLM guidance as defined in Pallagani et al., 2024 (16) and used recently in Plan-Seq-Learn (Dalal et al., 2024 (5)) for instance, where in order to complete a task a Large Language Model sets the order of sub-tasks to complete while a low-level agent independently learns how to execute each sub-task.

In order to bias the factoring algorithm and help it find a solution more quickly, two complementary approaches might be needed. Firstly, in order to evaluate whether or not a given factoring is viable, a distance metric over the elements of $SO(d)$ that we consider would transform the factoring algorithm into a (reversed) hill-climber. While SO(d) is finite-dimensional and as such all of its distance metrics are equivalent (that is, they define the same metric space topology), the geodesic distance described in section 3.3 would be a particularly meaningful candidate distance metric, as minimising it is exactly minimising the difference between current and target singular values.

Second, in order to handle the cases where some states might be unreachable from others (due to the presence of walls for instance in a maze or due to the limitations of an actuator), a dilatation of the observation space that would significantly increase the distance between such states would allow a distance-based hill climber to avoid forbidden or impossible actions, while preserving the structure and task-agnostic nature of the observation space. An illustration of this process using a maze and a set of valid trajectories is shown above in Figure 3.

While in the case of the maze such a dilatation could be pre-computed, a solution analogous to a gravity field could also be used in cases where the underlying observation space is not easily understood or segmented. In essence, during a pre-training phase, each observation from the buffer would add weight to its position, adding in practice a coordinate in a supplemental dimension. The problem would then be recast with this additional dimension and after smoothing the differences to make this modified observation space a smooth manifold, forming a new observation space to be treated as any other would be otherwise using our framework during training.

Concrete implementation of these methods are however highly dependent on the resolution of a specific task, and as such have not been explored in our experiments; it is therefore left to future work to determine the viability of using our SILEO framework for factoring-based long-horizon planning.

4 Experiments

In this section, we present various experiments we conducted to evaluate the viability of the SILEO framework in a control setting while solving various tasks from the DMControl dataset. The purpose of these experiments was to assess the limitations of our hypotheses and to quantify the relative impact of each of our approximations of the initial agent's

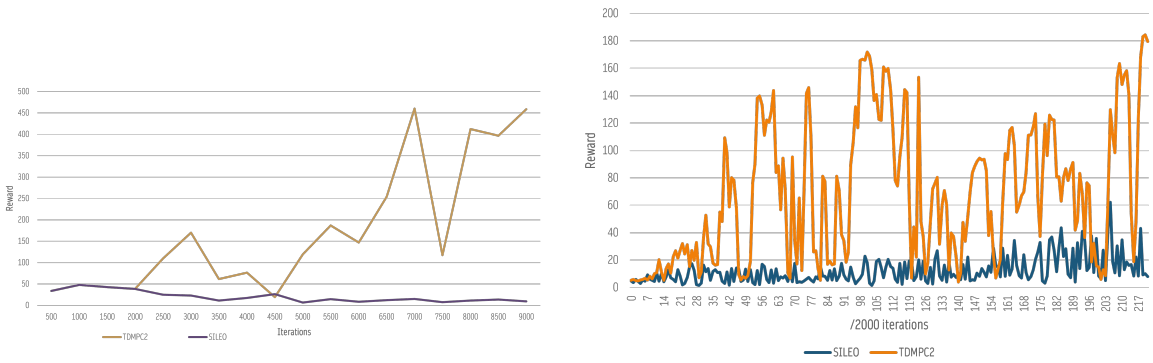

Figure 4: Walker-walk, dog-run baseline performance

behaviour. We used TD-MPC2 (Hansen et al., 2024 (8)) as the baseline agent to be modified to follow our framework, with the goal of assessing the viability of our hypotheses in a diverse set of tasks and better characterise the situations where using our SILEO framework could faithfully replicate the downstream performance of the agent.

4.1 walker-walk

Likely one of the "easiest" tasks of the dataset with a 25-dimensional state observation space and a 6-dimensional action space, the original TD-MPC2 agent converges in this task in less than 20 000 iterations, reaching a reward of 900-950 depending on the random seed, to be compared to an average reward of around 45 using random actions. However, this task primarily aims at teaching a simple two-legged "walker" how to fight against gravity to be able to walk without falling down. It is therefore a great example of tasks that cannot be solved using solely our framework, since the assumption that actions are (mostly) distance-preserving does not hold when fighting against a dominant external force with such a simple agent. Indeed, in our experiments the agent seemingly learns how to $fail$ the task, converging at a reward of around 10 which is significantly below the performance of random actions.

To validate the hypothesis that gravity was at fault, we used two different projections for the observations used in the pruning algorithm and the trainer to ensure the states used to deduce representations would have nothing to do with those used in dynamics prediction to update these representations. This allowed our agent to maintain performance slightly above the level of random actions for a short time after pre-training, before reverting to its previous behaviour after enough updates were made to erase the transitions used in the pruning algorithm from the agent's memory.

4.2 dog-run, dog-walk

Significantly more complex than the previous task, with a 223-dimensional state observation space and a 38-dimensional action space, the dog-run task represents a moderate challenge to the unmodified TD-MPC2 agent, which in 200 000 updates reaches an average reward just under 200, to compare to an average reward of 6 for random actions. Our framework initially improves the performance of the agent from the end of pre-training onwards, with an average performance at 50 000 steps moving from 6.0 to 14 and equivalent performance at 100 000 steps. However, due to the small (ranging from 16 to 512) set of actions chosen for pruning, and thus extremely coarse topology of our actions, downstream learning is noticeably slower. The limitations of our model also show here, since even though the observation space is much larger gravity is once again the main competing force working against our agent.

In order to evaluate the relative contributions of each of these factors, we leveraged vectorised environments and mean pooling to train the agent on 4 different environments, half of which planned their actions using our framework, the rest with the original agent's policy. All policy actions proposed by the latter 2 environments were recast as SILEO actions for model updates. The idea behind this approach is to compensate for the handicap of a finite set of actions by planning on both the original continuous setting and our modified discrete framework to add some observation noise, before updating the agent by matching policy actions their nearest actions in our small set of skills.

In these experiments, we also evaluated the contribution of composite actions, which are initially more vulnerable to unaccounted external forces. In both cases, the number of initial primitive actions was set to 16, with a limit on the total number of actions set to 512. We found that although slower and with far higher variance as the number of actions was reduced, this mixed approach both with and without composite actions initially kept up with the original agent at first and kept improving with more training steps, although at a slower rate ranging from a third to a tenth of the original.

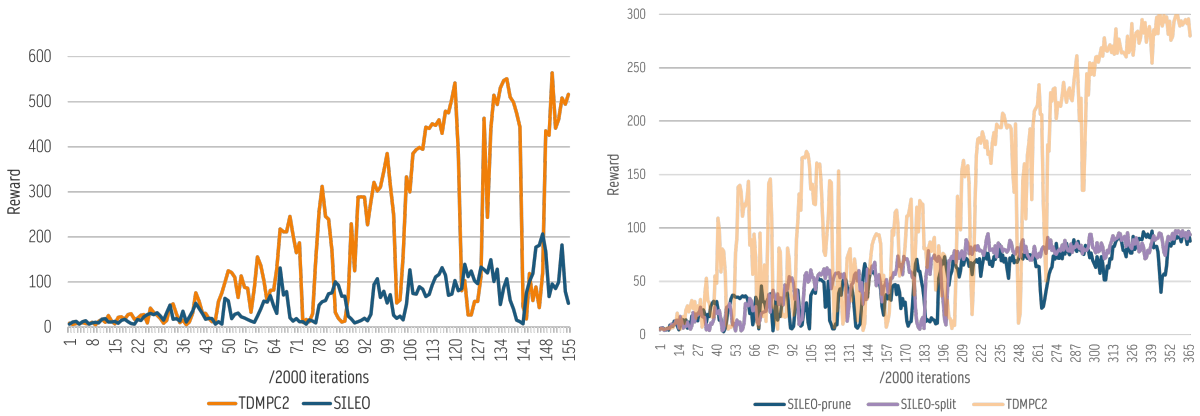

Figure 5: dog-walk, dog-run "boosted" agent's performance. Reward shown as a function of updates

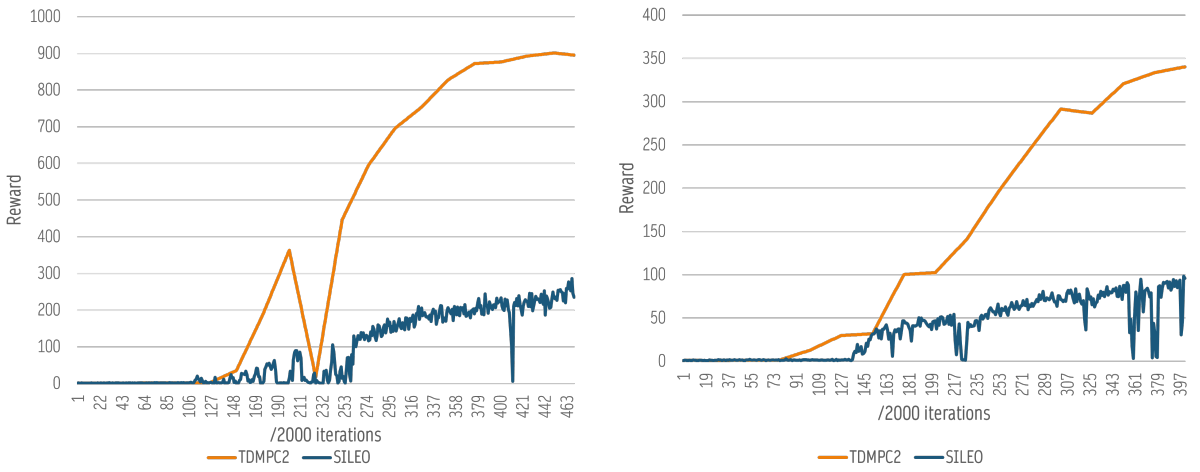

Figure 6: humanoid-walk, humanoid-run "boosted" agent's performance. Reward shown as a function of updates

4.3 humanoid-run, humanoid-walk

Humanoid-run is one of the most difficult tasks in the dataset for TD-MPC2, whereas humanoid-walk was comparatively simple to solve. These tasks have an observation space of dimension 67 and an action space of dimension 21. Although similar to walker-walk as they feature bipedal agents, these tasks require a much finer control of humanoid anatomy. In practice, the significantly higher number of joints and poses makes this task as much, of an exercise in maintaining rigidity and balance than in learning to use two legs, which makes our framework better adapted to the task. The initial agent converges on the run task at a reward of 600 after 8 million iterations, and on walk reaches 900 in 2 million; with less dominant gravity, our agent was found to approximate this performance level much more faithfully than in dog tasks in both cases, despite a number of actions limited to 512.

4.4 Knowledge transfer, distances

Finally, we explored a concrete implementation of an action transfer and distance measure between tasks based on the geodesic distances between the primitives selected to perform these tasks. In order to provide a unified measure, we applied a variant of the Gale-Shapley algorithm between two sets of actions, seeking to assign each action of the current task an action of the reference task according to their relative distance. Once the coupling had been achieved, we calculated its energy before substituting the actions of the current task with their equivalent in the previous task to evaluate the impact on training. We then measured each agent's performance after coupling (at 100 and 200 000 steps respectively for dog and humanoid tasks), and measured the task distance between dog-run and dog-walk as well as between humanoid-run and humanoid-walk.

Table 1: run vs. walk

Agent	Action distance	dimension	Reward decrease
dog	27.9 ± 2.1	223	-6.3 (-14.6%)
humanoid	16.4 ± 0.9	67	-2.5 (-11.3%)

5 Limitations

As discussed earlier in this paper and illustrated by our experiments, the use of orthogonal special group elements to represent robotic commands has its limitations. Some of these are imputable to the limitations of the theoretical framework itself and to certain assumptions that simply do not hold in a concrete setting for solving certain tasks, while others probably have more to do with implementation choices that favoured speed of execution and ease of interpretation over accurate modelling that would conform in every aspect to the requirements of our theory.

Firstly, the choice of the class of functions, or rather the group of linear operators chosen as the representation space is relatively restrictive; indeed, as presented in Appendix 1, it is based on the dual assumption that actions are quick to execute and preserve distances between states, in other words that the environment and the agent are essentially rigid and devoid of obstacles. In addition, the computations required, which are based on the assignment of transitions associated to these actions, did not easily lend themselves to the use of a continuous space in the image of the topological group $SO(n)$, which forced us to use only a finite sample of actions drawn from this group and probably seriously limited the learning speed in our experiments.

Future work could seek to overcome these problems by defining in advance relevant elements of the group as primitive actions, before generating from these well-chosen elements a subgroup of $SO(n)$ or any other group chosen to represent the dynamics of these actions. This approach would be more rigid as it would restrict itself in advance to certain primitives. Indeed, while infinitesimal generators could be used, these would be drawn from the Lie group of the function class used for representation and therefore deal with a class of operators that is much more diverse and difficult to understand. However, it would provide a partial solution to the discretisation problem, and would be more compatible with an end-to-end deep learning approach instead of relying on algebraic operations that are disconnected from the other parameters of the agent's systems.

Another solution to this problem would be to allow the agent to learn the representation itself, i.e. a continuous morphism defined over a continuous space of actions and with values in $SO(d)$ or any other relevant group of functions acting on the space of observations. Although this approach would be more complex to implement, we did not pursue it in this paper primarily because it would have been more difficult to construct composite actions insofar as the control space generally has no group structure. That being said, using a much higher-dimensional space which would extend the original action/command space together with an encoder/decoder that would map this larger space to the rotations associated with action sequences of varying length could allow a viable continuous representation to be learned.

Another difficult hypothesis to verify in our experiments was the reversibility of actions. Indeed, while this is a prerequisite for preserving distances, it is partially inadequate in the implementation chosen, which only retains transitions 'forwards' in time and does not attempt to establish a bidirectional model (as proposed for instance by Lai et al., 2020, (15), Hu et al., 2023 (10) or Höftmann et al, 2023 (9)), even though learned rotations lend themselves just as much if not better to bidirectional planning (as proposed by G. Baldassarre in 2003 (1)) compared to unidirectional models.

6 Conclusion

In this paper, we presented the theoretical justification and ran experiments to build and refine a framework enabling RL agents to develop Self-Improving Logic from Experimental Observations, thus proposing a novel method to build and refine representations of actions as group actions. We therefore allow autonomous agents engaged in a reinforcement learning procedure to derive, from the dynamics of the actions they perform, a dynamic representation that may be intuitively understood by human beings while remaining useful to the optimisation of the agent's decision-making process. While the initial results are encouraging as they were obtained on tasks deliberately chosen to be difficult to solve in order to test the limits of our framework, the proposed implementation still suffered from limitations that impacted its overall performance. There also remains several open questions both in terms of evaluation and concrete implementation of certain ideas or natural extensions of this paper, which would need to be answered by future works in order to confirm that this new perspective is viable and could improve in the long run both the performance and our understanding of RL agents and their reasoning capabilities.

References

[1] Baldassarre, G. (2003). Forward and Bidirectional Planning Based on Reinforcement Learning and Neural Networks in a Simulated Robot. In *Butz, M.V., Sigaud, O., Gérard, P. (eds) Anticipatory Behavior in Adaptive Learning Systems. Lecture Notes in Computer Science*, vol 2684. Springer, Berlin, Heidelberg.

[2] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma-teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei. "Language Models are Few-Shot Learners." *ArXiv preprint abs/2005.14165* (2020): n. pag.

[3] Chen, Siwei, Anxing Xiao and David Hsu. "LLM-State: Open World State Representation for Long-horizon Task Planning with Large Language Model." *Arxiv abs/2311.17406* (2023).

[4] Chua, Kurtland, Roberto Calandra, Rowan Thomas McAllister and Sergey Levine. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models." In *Neural Information Processing Systems* (2018).

[5] Dalal, Murtaza, Tarun Chiruvolu, Devendra Singh Chaplot and Ruslan Salakhutdinov. "Plan-Seq-Learn: Language Model Guided RL for Solving Long Horizon Robotics Tasks." *ArXiv preprint abs/2405.01534* (2024)

[6] Haarnoja, Tuomas, Aurick Zhou, P. Abbeel and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." *ArXiv preprint abs/1801.01290* (2018): n. pag.

[7] Hansen, Nicklas, Xiaolong Wang and Hao Su. "Temporal Difference Learning for Model Predictive Control." In *International Conference on Machine Learning* (2022).

[8] Hansen, Nicklas, Hao Su and Xiaolong Wang. "TD-MPC2: Scalable, Robust World Models for Continuous Control." *ArXiv preprint abs/2310.16828* (2023)

[9] Höftmann, Marc, Jan Robine and Stefan Harmeling. "Backward Learning for Goal-Conditioned Policies." *ArXiv abs/2312.05044 (2023)*

[10] Hu, Xiaobo, Youfang Lin, Yue Liu, Jinwen Wang, Shuo Wang, Hehe Fan and Kai Lv. "A Reliable Representation with Bidirectional Transition Model for Visual Reinforcement Learning Generalization." *ArXiv abs/2312.01915* (2023)

[11] Huang, Kevin, Sahin Lale, Ugo Rosolia, Yuanyuan Shi and Anima Anandkumar. "CEM-GD: Cross-Entropy Method with Gradient Descent Planner for Model-Based Reinforcement Learning." *ArXiv preprint abs/2112.07746* (2021)

[12] James, Stephen, Z. Ma, David Rovick Arrojo and Andrew J. Davison. "RLBench: The Robot Learning Benchmark and Learning Environment." In *IEEE Robotics and Automation Letters 5* (2019): 3019-3026.

[13] Kostrikov, Ilya, Denis Yarats and Rob Fergus. "Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels." *ArXiv preprint abs/2004.13649* (2020)

[14] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[15] Lai, Hang, Jian Shen, Weinan Zhang and Yong Yu. "Bidirectional Model-based Policy Optimization." *ArXiv abs/2007.01995* (2020)

[16] Pallagani, Vishal, Kaushik Roy, Bharath Muppasani, F. Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, L. Horesh and Amit Sheth. "On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS)." In *International Conference on Automated Planning and Scheduling* (2024).

[17] Papadopoulo, Théodore, Lourakis, Manolis. Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications. In *Lect Notes Comput Sci. 1842* (2000).

[18] Pertsch, Karl, Youngwoon Lee and Joseph J. Lim. "Accelerating Reinforcement Learning with Learned Skill Priors." In *Conference on Robot Learning* (2020).

[19] Jakob Puchinger, Günther R. Raidl, Ulrich Pferschy. The Multidimensional Knapsack Problem: Structure and Algorithms. In *INFORMS Journal on Computing*, 2010, 22 (2), pp.250- 265. 10.1287/ijoc.1090.0344. hal-01224914

[20] Rakelly, Kate, Aurick Zhou, Deirdre Quillen, Chelsea Finn and Sergey Levine. "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables." In *International Conference on Machine Learning* (2019).

[21] Rana, Krishan, Ming Xu, Brendan Tidd, Michael Milford and Niko Sunderhauf. "Residual Skill Policies: Learning an Adaptable Skill-based Action Space for Reinforcement Learning for Robotics." In *Conference on Robot Learning* (2022).

[22] Ruiter, Anton de and James Richard Forbes. "On the Solution of Wahba's Problem on SO(n)." (2014).

[23] Sekar, Ramanan, Oleh Rybkin, Kostas Daniilidis, P. Abbeel, Danijar Hafner and Deepak Pathak. "Planning to Explore via Self-Supervised World Models." In *International Conference on Machine Learning* (2020).

[24] Shridhar, Mohit, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer and Dieter Fox. "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks." In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019): 10737-10746.

[25] Sun, Chuanneng, Songjun Huang and Dario Pompili. "LLM-based Multi-Agent Reinforcement Learning: Current and Future Directions." *ArXiv abs/2405.11106 (2024)*.

[26] Sutton, Richard S., David A. McAllester, Satinder Singh and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation." In *Neural Information Processing Systems* (1999).

[27] Tassa, Yuval, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap and Martin A. Riedmiller. "DeepMind Control Suite." *ArXiv preprint abs/1801.00690* (2018)

[28] Touvron, Hugo, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov and Thomas Scialom. "Llama 2: Open Foundation and Fine-Tuned Chat Models." *ArXiv preprint abs/2307.09288* (2023)

[29] L.N. Trefethen et D. Bau, In *Numerical Linear Algebra, Other Titles in Applied Mathematics*, Society for Industrial and Applied Mathematics (SIAM), 3600 Market Street, Floor 6, Philadelphia, PA 19104, (1997)

[30] Tziafas, Georgios and Hamidreza Kasaei. "Lifelong Robot Library Learning: Bootstrapping Composable and Generalizable Skills for Embodied Control with Language Models." In *2024 IEEE International Conference on Robotics and Automation* (ICRA) (2024): 515-522.

[31] Vaswani, Ashish, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. "Attention is All you Need." In *Neural Information Processing Systems* (2017).

[32] Wahba,G., "A Least-Squares Estimate of Satellite Attitude," *SIAM Review,Vol.7,No.3*,1965,p.409.

[33] Wen, Muning, Junwei Liao, Cheng Deng, Jun Wang, Weinan Zhang and Ying Wen. "Entropy-Regularized Token-Level Policy Optimization for Language Agent Reinforcement." *Arxiv preprint: abs/2402.06700* (2024).

[34] Yu, Tianhe, Deirdre Quillen, Zhanpeng He, Ryan C. Julian, Karol Hausman, Chelsea Finn and Sergey Levine. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning." *ArXiv preprint abs/1910.10897* (2019):

[35] Zeng, Yuwei, Yao Mu and Lin Shao. "Learning Reward for Robot Skills Using Large Language Models via Self-Alignment." *ArXiv preprint abs/2405.07162* (2024)

[36] Zhang, Jesse, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun and Joseph J. Lim. "Bootstrap Your Own Skills: Learning to Solve New Tasks with Large Language Model Guidance." *ArXiv preprint abs/2310.10021* (2023)

[37] Zhou, Zhehua, Jiayang Song, Kunpeng Yao, Zhan Shu and Lei Ma. "ISR-LLM: Iterative Self-Refined Large Language Model for Long-Horizon Sequential Task Planning." In *2024 IEEE International Conference on Robotics and Automation* (ICRA) (2024): 2081-2088.

A Appendix

A.1 Homeomorphism between a compact $K \subset \mathbb{R}^d$ and $K' \subset S^d$

While the following is an elementary geometric result derived from the homeorphism between a d+1-dimensional Riemannian sphere without a pole and a d-dimensional plane, we use this appendix to present different homeomorphisms that may be used for experiments, and their advantages and drawbacks in a control setting. Here the compact subset K on the plane is implicitly either the observation space or a learned latent space, both of which are assumed compact either as a consequence of the scope of each training environment or as the output of a multi-layer perceptron with linear activation for instance.

Theorem 1 *For any compact $K \subset \mathbb{R}^d$, there exists $K' \subset S^d$ such that K and K' are homeomorphic. Moreover, there exists a C^∞ diffeomorphism between K and K'.*

A.1.1 Proof

We use a stereographic projection and its inverse, defined as follows:

$$\pi : \mathbf{x} = (x_1, ..., x_d) \to \frac{2}{||\mathbf{x}||^2+1}.(x_1, ..., x_d, \frac{||\mathbf{x}||^2-1}{2})$$

$$\pi^{-1} : \mathbf{y} = (y_1, ..., y_{d+1}) \to \frac{1}{1-y_{d+1}}.(y_1, ..., y_d)$$

Since both π and π^{-1} are continuous and invert one another, we have that K and K' defined as $\pi(K)$ are homeomorphic.

While π is trivially C^∞, being essentially a multiplication by a rational fractional without real poles, π^{-1} does possess a real pole.

However, since K as a compact subspace of $\mathbb{R}^d$ which is Hausdorff, K is bounded; in particular, $K \neq \mathbb{R}^d$.

As $\mathbb{R}^d$ is homeomorphic to $S^d \backslash (0, ..., 0, 1)$ using π, there exists $\mathbf{z} \in S^d \backslash (0, ..., 0, 1)$ such that $z \notin K' = \pi(K)$.

Let us define P as one of the rotation matrices in $SO(d+1)$ such that $P((0, ..., 0, 1)) = \mathbf{z}$.

Since P is rigid we have that $P\pi$ and $\pi^{-1}P^{-1}$ define a C^∞-diffeomorphism between K and PK'.

A.1.2 Remarks

While stereographic projections are conformal, that is, preserve angles, they do not preserve distances; while the section of S^{d+1} cut by the plane $x_{d+1} = 1/2$ in our example remains unchanged, the section of the plane formed by the open disc within this circle is the image of the part of the sphere that lies "below" the plane. To be precise, $\pi^{-1}(S^{d+1} \bigcap x_{d+1} < 1/2) = B(\frac{1}{\sqrt{2}})$, yielding significant distance distortions. It is even a theorem; due to a sphere having positive Gaussian curvature compared to a plane's null Gaussian curvature, no homeomorphism between the two can be simultaneously isometric and conformal. While their preserving of angles make stereographic projections more useful for our framework, since $SO(d)$ is parametrised by sets of d rotation angles, in a RL setting with sensitive gradients isometric projections might yield better results.

One such projection would be the "wrapping" of K, which is in essence the inverse of an isometric projection of the sphere's surface. On S^d, such an isometric wrapping may be achieved using a normal geodesic coordinate system (with the associated geodesic distance) around a reference point. Taking as reference point $\mathbf{x_0} = (0, ..., 0, 1)$, and d normal geodesics intersecting at $\mathbf{x_0}$ for instance, $B(0, \pi)$ in $\mathbb{R}^d$ is equivalent to $S^d \backslash (-\mathbf{x_0})$ by equating normal geodesic coordinates centered on $\mathbf{x_0}$ and Cartesian coordinates centered on 0. If $K \subset \mathbb{R}^d$ is a set of possible observation and $sup_{\mathbf{x} \in K} ||\mathbf{x}||_\infty < M$, a simple scaling operation would likewise allow to project isometrically this d-dimensional observation set K on the d-dimensional sphere. However, beyond being more expensive to compute due to the back-and-forth conversion between Cartesian and geodesic coordinates, rotations using such isometric projections would be less meaningful due to the distortion of angles. As an example, on the 2-dimensional sphere this process already yields equilateral triangles whose sum of angles is equal to 270°.

A.2 Distance-preserving agent actions over a normed linear observation space are arbitrarily close to elements of $SO(d)$

While this result is at first glance counter-intuitive, since our actions are not assumed to be linear, their being both distance-preserving and acting over a normed subspace of a linear vector space is enough to prove our statement.

While in real-world experiments the assumption that actions preserve distances is likely to never be verified, the right balance between external forces, high-dimensional observation spaces and small time steps allows for this to represent a surprisingly relevant approximation.

Theorem 2 *Given $\epsilon > 0$, $n \in \mathbb{N}$, d the Euclidean distance over $(\mathbb{R}^n, (.|.))$, $\Omega \subset \mathbb{S}^n - 1$ a set of observations, A a topological group acting on Ω such that for all $a \in A$ and $\omega_1, \omega_2 \in \Omega$, $d(a \cdot \omega_1, a \cdot \omega_2) = d(\omega_1, \omega_2)$, there exists $r : A \to SO(n)$ such that $max_{\omega \in \Omega, a \in A}\, d(r(a)(\omega), a \cdot \omega) < \epsilon$*

In the following proof, we consider $a \in A$ an action. We will prove in order the two following statements:

1. $r_a : \omega \to a \cdot w$ is linear

2. Either r_a is in $SO(n)$, or it can be ϵ-approximated by $R_a \in SO(n)$.

A.2.1 Proof of linearity

Let us assume for convenience $\Omega = \mathbb{S}^{d-1}$; if the observation space is a subset of $\mathbb{S}^n - 1$ the result will still be verified by restriction.
Let $B = (e_1, ..., e_n)$ be an orthonormal basis of Ω, $(r_a(\epsilon_1), ..., r_a(\epsilon_n))$ a basis of $r_a(\Omega)$. Since r_a preserves distances, we have simultaneously if $i, j \in [|1; n|]$ and for all $1 \leq k \leq n$:

$$d(r_a(e_i + e_j), r_a(\epsilon_k)) = d(e_i + e_j, \epsilon_k) = d(e_i, \epsilon_k) + d(e_j, \epsilon_k),$$
$$d(r_a(e_i), r_a(\epsilon_k)) + d(r_a(e_j), r_a(\epsilon_k)) = d(e_i, \epsilon_k) + d(e_j, \epsilon_k).$$

Hence, $d(r_a(e_i + e_j), r_a(\epsilon_k))^2 = d(r_a(e_i), r_a(\epsilon_k))^2 + d(r_a(e_j), r_a(\epsilon_k))^2 = d(r_a(e_i) + r_a(e_j), r_a(\epsilon_k))^2$ Since these are $n = dim(\Omega)$ independent linear equations, we have shown:

$$(\mathbf{1}) : r_a(e_i + e_j) = r_a(e_i) + r_a(e_j) \text{ for all } 1 \leq i, j \leq n.$$

Let us recall that $\Omega \subset S^{d-1}$. Considering the absence of information on r_a beyond the sphere, we may extend it in a rigid continuous way to $\mathbb{R}^{d*}$ by defining

$$(\mathbf{2}) : r_a(x) = ||x||.r_a(\tfrac{x}{||x||}) \text{ for all } x \in R^{d*}.$$

Indeed, as r_a is assumed to be isometric, we already knew that

$$(\mathbf{3}) : r_a(0) = 0.$$

Using (1) and (3), we can already show $(\mathbf{4}) : r_a(e_i) + r_a(-e_i) = 0$.
If we further combine (1), (2), (3) with (4), for all $\lambda, \mu \in \mathbb{R}$, $1 \leq i, j \leq n$, we can write $r_a(\lambda.e_i + \mu.e_j) = r_a(\lambda.e_i) + r_a(\mu.e_j)$ and finally $r_a(\lambda.e_i + \mu.e_j) = \lambda.r_a(e_i) + \mu.r_a(e_j)$, thus showing the linearity of r_a.

■

A.2.2 Approximation in SO(n)

We have shown that for any "action" $a \in A$, its group action on the observation space Ω could be extended into a linear isometric map of $\mathbb{R}^n$. As is, we already have $r_a \in O(n)$, therefore either $r_a \in SO(n)$ or $r_a \in Sym(\mathbb{R}^n)$.

In the latter case, as the determinant is a continuous function one cannot expect to find an element of $SO(n)$ in an arbitrary close neighbourhood of r_a. This continuous nature also implies, by the intermediate value theorem and since A is a topological group, that the set of all $a \in A$ such that $det(r_a) = -1$ is disconnected from its complementary. The same theorem also implies that actions in the same connected component as the null action act through elements of $SO(n)$ on the observation space.

However, let us recall that we are considering agent actions, that is, actions executable at a time step by a given RL agent. This implies notably that the topological group A is either connected or very close to being connected. Indeed, we may intuitively consider that any command given to an actuator may even be pathwise connected to the null command. The actions considered in our framework however also encompass to a certain extent environmental perturbations, represented by our composite random variable τ in our problem formulation. As such, discrepancies may emerge and it cannot be ruled out that due to threshold effects for instance A might contain several connected components.

A solution to this problem is to consider smaller time steps in our Partially Observable Markov Decision Process, since intuitively disconnected components primarily emerge from the unforeseen perturbations and not from the actuator command. With infinetisimally small time steps, any environmental transformations may be approximated by a reversible succession of equilibrium states; as such, any actions come closer to pure commands, which themselves come relatively closer to the null command, since it is the only action that has no execution time.

We can translate this intuition formally as follows:

Let V_0 be an open neighborhood of the null action in the command space such that $A \cap V_0$ is connected; such a neighborhood exists by continuity of the determinant, and is non-trivial using the previous reversibility argument.

For any $t > 0$, let us denote A_t the set of actions that can be completed in less than t (seconds for instance).

Then, $(A_t)_{t\in\mathbb{R}^+}$ is a filtration of A, with $A_0 = \{a_0\}$ the null action.

However, $a_0 \in V_0$ and $A \cap V_0 \subset A$.

Therefore, there exists $t_1, t_2 > 0$ such that $A_{t_1} \subset A \cap V_0 \subset A_{t_2} \subset A$.

■

We have shown that with small enough time steps, all actions that can be completed by the agent act on the observation space through elements of $SO(n)$; while such small time increments might be hard to achieve $in\ concreto$, this concludes our proof that all agent actions can be approximated by (a composition of) elements of $SO(n)$.

A.3 Action dynamics as an instance of Wahba's problem

As shown previously, with a small enough time increment all actions which can be completed by an agent act on the observation space via elements of $SO(n)$, and this property in fact extends to all actions that are part of the same connected component as the null action. In this section, we show that the corresponding elements of SO(n) can be found by solving an instance of Whaba's problem, and that the resulting rotation matrices are optimal to represent a given action's dynamics.

A.3.1 Problem formulation

To find what rotation matrix corresponds to a given action, we aim to solve the following problem (using $e_n = (0, ..., 0, 1) \in \mathbb{R}^n$):

Problem 1 *(Wahba's Problem): Given $n \geq 2$, an action $a \in A$, $p : \mathbb{R}^{n-1} \to S^{n-1}\backslash\{e_{n+1}\}$ an homeomorphism and a matching of $N \geq 2$ state transitions $((p(s_{i,k}), p(s_{f,k})))_{1\leq k\leq N} \in S^{n-1}$ observed by effect of a, we aim to find a rotation matrix $C \in SO(n)$ such that the following reconstruction cost is minimised:*

$$J(R_a) = \frac{1}{N}\sum_{k=1}^{N} \|p(s_{f,k}) - Cp(s_{i_k})\|^2$$

It is clear that under our assumptions, and using r_a as the notation for the rotation matrix which represents the action of a $\in A$ on the observation space, that r_a is a global minimiser of J. One sufficient condition for it to be the unique solution is shown in On the Solution of Wahba's Problem on SO(n) (de Ruiter and Forbes, 2013 (22)):

If $B^{\mathbb{T}} = \sum_{k=1}^{N} s_{i,k}s_{f,k}^T$, the SVD-derived solution to $\mathbf{Problem\ 1}$ is unique if $rank(B) = n - 1$.

A.3.2 Proof that r_a is found

Within our framework, observations in the $(n-1)$-dimensional observation space are sampled independently at random from a randomly initialised buffer; as such, if $N = \Omega_{time\to\infty}(n)$, it is expected that the rank of $((s_{i,k})_{1\leq k\leq N\in R^{n-1}}$ would be $n - 1$, that is that the set of initial states would contain a free family.
Furthermore, since a diffeomorphism p was used to project R^{n-1} on $S^{n-1}\backslash e_n$, the rank of the family is conserved by the projection.
We can conclude that $rank(B) = n - 1$ for $N = \Omega_{time\to\infty}(n)$.
As such, after enough training steps the solution to $\mathbf{Problem\ 1}$ given by the SVD method described in (de Ruiter and Forbes, 2013 (22)) is unique.
Therefore, as a global minimiser r_a is the unique solution that is found using this method.

■

A.3.3 Optimality of the resulting dynamics model

The entire purpose of our framework is to enable a RL agent to learn and refine action representations through their experimental dynamics. In order to restrain action dynamics to elements of SO(n) however, we made the essential hypothesis that actions would be distance-preserving. While already difficult to satisfy on the original observation space, by projecting this space on the n-1-dimensional sphere we introduced another distortion, at least of either angles or distances.

If a distance-preserving projection is used, modelling the dynamics on the sphere becomes equivalent to modelling them on the original space; in that case, since under our hypothesis the group action associated with an action $a \in A$ is identifiable with an element of $SO(n)$, and since as shown before the uniqueness of the solution depends only on the rank of the set of observations used, this representation is optimal. Indeed, the purpose of any dynamics model is to learn the group action associated with each of the possible actions in the set A, and under our assumptions solving a particular instance of Wahba's problem accomplishes this task.

The most relevant and convenient projections however are stereographic projections. Indeed, since they preserve angles, a rotation matrix learned on the n-1-dimensional sphere may be restricted to the original observation space and be interpreted with minimal calculations. Furthermore, while they distort the distances between observation vectors, these distortions are locally negligible since these projections are C^∞. We may therefore reuse the small time increments argument used in Appendix 2 to restrict the actions considered to those that can be completed by the agent in a small enough time frame. Intuitively, these actions have smaller eigenvalues, which confines the image of any given point by this action to a small neighbourhood around it and prevents significant distance distortions. Thus, even in the case of stereographic projections, it remains possible under this condition to approximate the distance-preserving requirement and to achieve a near-optimal dynamics model under our framework.

A.4 Complexity considerations

A first criterion to take into account before considering the theoretical performance of our approach is the complexity of the operations to be carried out. As it stands, each action in our method requires two components: a vector giving its position in the command space and a rotation translating the dynamics observed by the agent, giving a minimum memory complexity in $O(a * d^2)$ floating point numbers if the control space is of dimension a and the observation space is of dimension d. In our implementation, we also used a pool of $2.d$ state transitions used to periodically recalculate the rotation, which was not necessary as this update could be done without keeping the transitions in memory, and did not change the total memory complexity. With a finite number of actions $a\ priori$ independent of a or d, this theoretical complexity remains unchanged, although in our experiments an agent was able to build up to 11,000 actions after pruning, including 800 primitives for a space of dimension 223, suggesting a real memory cost of $O(d^3)$ or even $O(d^4)$ floats.

From a computational point of view, evaluating the dynamic model comes down to a simple matrix product, resulting in a cost of $O(d^2)$ multiplications. This is the main computational advantage of our method, since it allows us to estimate the dynamics of an agent with its environment with a quadratic cost with respect to the complexity of the environment in question and not in $O(||(a + d) * d||^3)$ for example by using a perceptron with two hidden layers as in TD-MPC2. The cost of updating a rotation matrix after observing a state transition is also relatively low. Indeed, the cost of the singular value decomposition of a matrix compiling transitions as used in our implementation, which corresponds to the creation of an action, is in $O(d^3)$ operations, although very fast in practice when optimised by the Pytorch library. For an optimal theoretical complexity, assuming that each rotation has singular values of multiplicity at most 1, for example due to rounding errors or experimental noise in the observations, the singular value decomposition operation is differentiable (cf.Papadopoulo and Lourakis, 2000 (17)), reducing the cost of an update to $O(d^2)$ operations described in the article to be added to the calculation of the same complexity of $y^t z$ which is the matrix to which to apply this method if $y \to z$ is the observed transition. Updating the action used after an iteration therefore costs $O(d^2)$ in theory; we have chosen an implementation with a cost of $O(d^3)$ in practice, however, due to the many non-optimised operations in Python that are required by the method proposed by Papadopoulo and Lourakis.

As for the functions used at greater or lesser intervals during training, their complexity is more significant. The pruning algorithm used stores a defined number of transitions during pre-training before any rotation is calculated, as a basis for the K-Medoids algorithm used to build a first set of transitions.

Insofar as a larger number of stored transitions increases the representativeness of our action set, an order of magnitude of $O(d^2)$ transitions was used, for a total memory footprint of $O(d^3)$ floating point numbers. The other arrays used, for example the one constructed from this one to store the vectors in the control space representing the observed actions, have a lower or similar memory cost. Assuming a maximum number of repetitions T, the K-Medoids algorithm used (a

variant of Lloyd's heuristic) thus has a complexity in $O(d^5.T)$, insofar as the method is applied to the $O(d^2)$ commands given at each transition. The number of primitive actions retained, although small in general, can go up to $O(d)$ in practice. By retaining the dominant terms, the part of the pruning algorithm dedicated to the construction of new primitives therefore has a memory footprint of $O(d^3)$ floating point numbers and costs $O(d^5T)$ elementary operations in our implementation. The second part of the pruning algorithm, corresponding to the creation of composite actions, incurs a much lower cost. In fact, the number of occurrences of a given action can be stored and updated with it, allowing us to identify those that are over-represented in a time and memory space that is linear with the number of actions, of the order of d or so. Taking this order of magnitude and noting c_R,c_π the time complexity for the calculation of reward and strategy respectively ($O(d^3)$ operations in the original TD-MPC2 paper), identifying the most relevant sequences of each action has a cost of $O(d^4)$ operations or even $O(d^5)$ in the case of a very long list of actions. In the most general case, this cost is of $O(max(c_R, c_\pi).d^2)$.

In the infrequent case when the number of composite actions created would be linear with the number of actions (which would cause the action list to grow exponentially if this situation was permanent...), we can refine this worst-case complexity estimate into $O(max(c_R, c_\pi, d^3).d^2)$ by integrating the cost of the singular value decomposition used to create a new action. In terms of memory, the prevailing cost is that of storing the new actions, costing on the order of $O(d^2)$ floating point numbers or $O(d^3)$ in the case of a linear number of actions created during an iteration of the algorithm.

Finally, the variant of the Gale-Shapley algorithm used to calculate the distance between two tasks and transfer actions from one to the other has a time complexity in $O(d^4)$ operations assuming a very large number of actions of the order of $O(d^2)$ for each task, a cost dominated by the calculation of the distances required to establish the "preferences". Indeed, calculating the geodesic distance between two rotations of of dimensions dxd requires the calculation of the eigenvalues of each $R_a^T R_{a'}$ for a, a' used respectively to solve the first and second task considered.

Since orthogonal matrices are in particular normal, the QR algorithm used by standard libraries such as numpy or pytorch can converge in $O(d^3)$ operations (cf. Trefethen, and Bau, 1997 (29)), resulting in a total cost of $O(d^5)$ operations or even $O(d^7)$ if each task involves a very large number of actions, of the order of d^2. In the case when the matrices have been established from latent states and not from observations, the cost of calculating the rotation representing the transition from one encoding of observations to another is added; considering $O(d)$ test observations and noting c_e the time complexity of the encoder used, calculating the transitions therefore costs $O(c_e.d)$ operations, to which are added $O(d^3)$ operations to calculate the corresponding rotation. While these operations are costly in terms of time, with the order of $O(max(c_e.d, d^7))$ operations in the worst case, in terms of memory they only cost $O(d^2)$ floating-point numbers since for each task only a small number of d x d matrices are stored.