Transfer of Knowledge for a Climbing Virtual Human: a Reinforcement Learning Approach

Benoît Libeau, Alain Micaelli and Olivier Sigaud

***Abstract*— In the reinforcement learning literature, transfer is the capability to reuse on a new problem what has been learnt from previous experiences on similar problems. Adapting transfer properties for robotics is a useful challenge because it can reduce the time spent in the first exploration phase on a new problem. In this paper we present a transfer framework adapted to the case of a climbing Virtual Human (VH). We show that our VH learns faster to climb a wall after having learnt on a different previous wall.**

I. INTRODUCTION

Complex mechanical systems such as humanoid robots or Virtual Humans (VHs) raise so many control challenges that the robotics practitioners cannot hope to program all the aspects of their control by hand. As a result, a paradigmatic shift towards new control approaches is necessary.

Among these new approaches, learning is central. A lot of effort is spent on imitation learning [1], on supervised learning of robot models for control [2], [3] and on Reinforcement Learning (RL) [4]. In the case of RL, many technical developments are of interest for robotics research. Among them, *transfer* is the capability to reuse on a new problem what has been learnt from experience on a previous problem. Transfer capabilities are important in the context of robotics applications because the initial exploration phase of the standard RL process is either very expensive or even impractical with a robot. A solution could then be to perform training in a simplified or simulated context and then transferring the corresponding knowledge to the actual robotics context before a final tuning phase, instead of learning from scratch with the robot.

More generally, the central challenges of using RL techniques in a robotics context come from the fact that these techniques are generally designed in the context of small and discrete problems whereas most control problems are large and continuous. Thus one has to adapt RL techniques to large and continuous control problems.

In this paper, we will present a case study where we adapt a RL transfer framework to the case of a complex VH learning to climb successive walls. In this context, the transfer challenge we address is the following: we want our VH to learn to climb efficiently on a first wall so that, when confronted afterwards with different walls, it can reuse what it has learnt and learn faster to climb them efficiently. We will focus on the design of a problem representation that results in such transfer properties.

Our paper is organized as follows. After a short presentation of transfer in RL, we describe our case study, explaining how we adapt our VH simulation to the RL and transfer framework. We present our experimental results and discuss the transfer capabilities of our agent, as well as the realism of our working hypotheses.

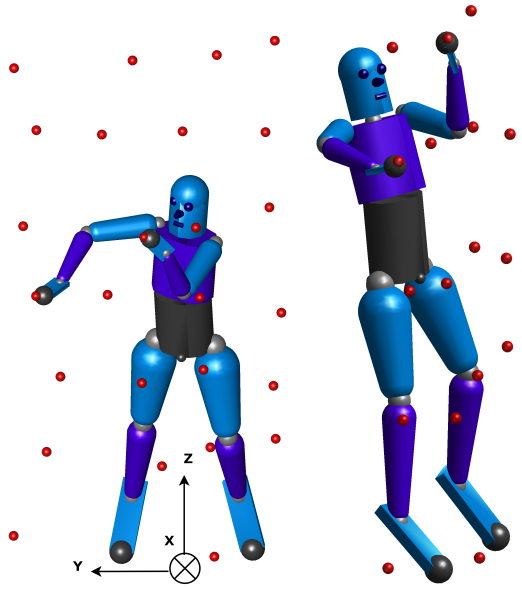


Fig. 1. Views of our 32 degrees of freedom virtual human and of the holds

II. BACKGROUND

A. Transfer in Reinforcement Learning

Reinforcement Learning [5] is a framework where an agent acquires knowledge of a Markov Decision Process (MDP) through trial and errors. In Q-learning [6], the learnt value function is stored for all state-action pairs in the so-called Q-table. In Dyna-Q [7], the algorithm we actually use, the convergence of the value function is sped up by the use of a learnt model of the environment.

The efficiency of the RL process depends much on the representation of the states and actions, that defines for example the combinatorial size of the problem. Some representations of the states and actions can have good generalisation properties, so one can reuse the knowledge acquired by solving

Benoît Libeau and Alain Micaelli are with Laboratoire d'Intégration des Systèmes et des Technologies in Commissariat à l'Énergie Atomique, 18 route du Panorama, BP6, Fontenay Aux Roses, F92265, France benoit.libeau@ensta.org, alain.micaelli@cea.fr

O. Sigaud is with Institut des Systèmes Intelligents et de Robotique, Université Pierre et Marie Curie - Paris 6, CNRS UMR 7222, 4 place Jussieu, F75252 Paris Cedex 05, France olivier.sigaud@upmc.fr

one MDP, when facing another similar but different problem. This is known as *transfer*.

Konidaris proposed in [8] a framework for transfer in RL. He considers a sequence of task instances generated by an underlying parametrized environment model. In our case, the task instances will be different climbing walls. All the instances have their own state and action spaces, their own transition and reward functions (this specific information form the *problem-space* of the instance) but they have enough similarity (they share an *agent-space*) so that it is possible for an agent to use the knowledge acquired during one instance to solve another one.

In operational examples of transfer [9], [10], agent-spaces contain "perceptions" of the states and actions by the agent. These perceptions are subjective to the agent, so they do not depend on the problem instance the agent faces. Perceptions are descriptors of visited states and actions in the agent-space, but they can also be seen as standard states or actions and can be used in a regular RL process. Our goal is to learn such a *portable value function* that takes its input in an agent-space and can be used on different instances.

III. MATERIAL AND METHODS

A. Simulation Framework

Our agent is a VH: a tree of physics-simulated bodies [11] that can model a human or a humanoid robot. Our VH has 32 degrees of freedom, it is 1.70m high and the size of the bodies, as well as the nature and limits of the articular links between these bodies, are taken from anthropometric tables [12], [13].

Previous works on climbing robots often used dedicated grasping effectors [14], [15] and considered climbing as a problem of gait [16]. Bretl [17], [18] also addressed the technological challenge of physical robots but, using a wall with non-evenly-placed holds, introduced the problem of climbing as a planning problem similar to the one we address. Works on VH include several studies of sport movements [19], [20], but to our knowledge no specific work on wall-climbing.

The root body of the VH seen as a tree of bodies is the torso. We note $e_0 \in \mathbb{R}^3$ the position of this body in the scene referential (for simplicity, we do not consider the orientations of the bodies). The articular configuration space of the VH is described by $G = [e_0, q] \in \mathbb{R}^{35}$ where $q \in \mathbb{R}^{32}$ is the vector of its articular coordinates. Inside the configuration space, the reachable space $\mathcal{R}$ is the set of postures that comply with our VH constraints.

The problem we want to solve is planning like a human climber would do before starting on a new wall, with only a model of holds positions on the wall and a geometrical model of the VH. The environment consists of the climbing wall. Actually, we only model *holds*, that are small regions of space (spheres with a 4cm radius) that the VH can "grasp". The VH has "grasping spheres" at the end of its terminal effectors (hands and feet) and we say it *grasps* a hold when the grasping sphere intersects the hold sphere.

We generate random climbing walls in the following process. First, we create a grid of 4 (horizontally) by 8 (vertically) rectangular tiles with dimensions 0.3m $\times$ 0.5m. The holds are placed on the vertices of the tiles. Then, we translate each hold with a random vector in $\mathbb{R}^2$ generated according to a centered normal law with standard deviation $\sigma = 0.08$. The referential of the scene is defined as follows: all holds are in one plane defining the x-axis. The y-axis goes horizontally to the left. The z-axis goes vertically to the top of the wall.

Our operational space is defined by the positions of the grasping spheres and the root body[1]. It is described by $E = (e_0, e_1, e_2, e_3, e_4) \in \mathbb{R}^{15}$ where each e_i, for $1 \leq i \leq 4$ is a vector of the physical space $\mathbb{R}^3$, which gives the position of the end effector i in the referential of the scene.

As a high-level description of operational space, we call a *4 holds configuration* (or *4-configuration*) a mapping of the four effectors to distinct holds on the wall. It can be represented by a quadruple containing an identifying number for each hold. If $\mathcal{E}_{holds}$ is the set of all holds, a 4-configuration is $C^4 = (h_0, h_1, h_2, h_3)$ where each $h_i \in \mathcal{E}_{holds}$ is the number of the hold grasped by the end effector number i.

We say that a hold configuration is *C-reachable* (to prevent confusion with other notions of reachability) if there exists a posture in $\mathcal{R}$ where the four end effectors are grasping the corresponding holds on the wall. Eventually, we can rephrase our problem: we want to find a sequence of C-reachable 4-configurations that will drive the VH to the top of the wall.

We use the rule that the climber should have at least three contacts to the wall at any time, so it moves one hand or foot at a time. Following this rule, Bretl's robots had quasi-static movements. We simulate only kinematic and no dynamic features: the dynamic problem of realising elementary reaching movements or displacement of the center of mass of the VH can be studied separately [11].

B. Inverse Kinematics

The point of Inverse Kinematics (IK) is to find an articular configuration corresponding to a configuration specified in the operational space. Let e_i^{des} be the position of the target hold of end effector i. We use a heuristic noted $H(C^4)$ to define the target position of the root body e_0^{des}. On the y-axis, the target is in the middle of the two feet holds. On the z-axis, the target is above the lowest foot hold translated by the height of the root body in a natural standing position.

Our IK algorithm builds a path from an initial articular configuration to another (previously unknown) articular configuration by descending the gradient of a cost function f that penalizes the distance between current and desired end effectors positions: $f(E) = \sum_{i=0}^{4} \frac{1}{2}\|e_i - e_i^{des}\|^2$. Considering the cost function as a function of the configuration space ($\hat{f} : G \mapsto \hat{f}(G) = f(E)$), at each iteration of the gradient descent algorithm (see Algo. 1.) the current point is displaced in the direction opposed to the gradient and characterized by a step α (line 8,9). However the direction of the descent is

[1]If we do not include the root body in the operational space, the random exploration in RL can lead the robot to unnatural postures (*e.g.* robot's back facing the wall), thus declaring some configurations not reachable when they obviously are.

modified to comply with the articular limits of the VH: the step of the descent is set to zero for articulations that would exceed their maximal extension (line 9).

Algorithm 1: Inverse Kinematics by Gradient Descent

```
1  G ← G_init ;
2  compute current (e_0, e_1, e_2, e_3, e_4) ;
3  t ← 1;
4  s ← termination test (e_0, e_1, e_2, e_3, e_4, t);
5  while s do
6      t ← t + 1;
7      V ← Compute gradient of f̂ in current point;
8      δG ← −αV;
9      G ← G+ modify descent direction (δG);
10     compute current (e_0, e_1, e_2, e_3, e_4);
11     s ← termination test (e_0, e_1, e_2, e_3, e_4, t);
12 if t > Δ_t then return G_init else return G ;
```

In order to determine when the final configuration is reached, we observe the current $e_i - e_i^{des}$ to see if all the hand spheres intersect the hold spheres (line 4). We also define a tolerance sphere for the position of the root body. When all the bodies are in the right areas, the gradient descent stops. By contrast, when the configuration is not C-reachable, the computation can continue forever, so we must stop it with some arbitrary criterion. We assume that all configurations that are not reached within a certain amount of time Δ_t are not C-reachable. Δ_t is fixed to 5.5 seconds of simulated time after studying the results of a large number of experiments.

Our IK algorithm is implemented by a function $G_{final} = \mathcal{IK}(G_{init}, C^4, e_0^{des})$. By keeping in memory all articular configurations experienced during the run, we can build a continuous path from G_{init} to G_{final} in $\mathcal{R}$, or rather a discrete sample of points along this path. Formally, an articular configuration G_t is said *G-reachable from* G_i if there is a continuous path going from G_i to G_t in $\mathcal{R}$:

$$\exists \mathcal{P} : [0,1] \to \mathcal{R}, \text{continuous} \,/\, \mathcal{P}(0) = G_i \text{ and } \mathcal{P}(1) = G_t$$

Experiments suggest that, for a big enough value of Δ_t, *any articular configuration in $\mathcal{R}$ is G-reachable from any other*.

Under this hypothesis[2], we can build an operational definition of reachability: a 4-configuration is *C-reachable* if there exists an articular configuration that is G-reachable from the current configuration and that grasps the four holds.

C. Problem-Space Representation

The states of our RL process are *3 holds configurations*. A *3-configuration* is similar to a 4-configuration but with only three end effectors grasping a hold, the fourth being the *free* hand or foot. A 3-configuration vector is a 4-configuration vector with one zero element for the free hand or foot. For example, a 3-configuration with a free right hand is $C^3 = (0, h_1, h_2, h_3)$, with $h_i \in \mathcal{E}_{holds}$. The action we consider is the number of the hold that the free hand will reach. In state $s = C^3$, the action a_1 tries to reach the 4 holds configuration $C_1^4 = (a_1, h_1, h_2, h_3)$.

In order to define the transition and reward functions of our MDP, let us assume the agent is taking action a_1 in state s. The next 4-configration, if C-reachable, will be the one determined by the old and new holds: C_1^4 in our example. In order to chose the next state among the 4 different 3-configurations that correspond to C_1^4, the next free end effector is determined in a round robin (right hand, left hand, right foot, left foot, right hand…). The next state in our example would be $C_1^3 = (a_1, 0, h_2, h_3)$. After a successful transition, the agent receives a small negative reward (Algo. 2, line 19), except if its center of mass has passed the final height, which means that the episode ends with a positive reward (line 15). If C_1^4 is not C-reachable, the agent will stay in state s and receive a big negative reward (line 11).

In order to reduce the combinatorial size of the problem, not every 3- and 4-configurations are considered for the RL process: the configurations that are not geometrically meaningful are removed in two cases. First, if the maximum distance between all pairs of holds in the configuration is higher than the maximum extension of the VH (we took 2.8m as an upper bound of this value). Secondly, if the configurations is not "natural", based on the relative positions of the holds. We do not consider configurations with both feet above the hands or with crossed arms or legs. Computation over 20 random-generated walls show that only 7.3% of the possible 3-configurations and 1.4% of the 4-configurations pass the test.

Finally, during the RL process, we must compute the sequence of postures corresponding to the visited states and actions. The hold density on our wall is high enough so there is at least one 4-configuration that is C-reachable from any given 3-configuration. Thus, there are solutions to our problem, and the agent will eventually find one.

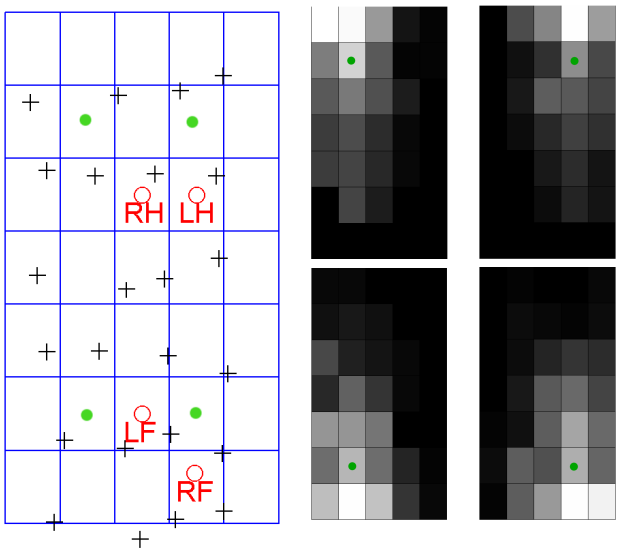
The sequence (G_i) of the articular configurations taken by the VH during the RL process, is defined by the following recurrence relation: $G_{i+1} = \mathcal{IK}(G_i, C_{i+1}^4, H(C_{i+1}^4))$.

We have a *cache* process that stores all computed reachable 4-configuration and the corresponding articular configuration, so we only have to run our IK algorithm once per 4-configuration.

D. Agent-Space Representation and Transfer Algorithm

The representation of states and actions we presented in the previous section is the problem-space of our transfer experiment. It is sufficient to solve one instance of the problem but it is specific to this instance and the acquired knowledge cannot be used on another instance. We now present the agent-space of our problem, where states and actions are defined relatively to the agent. The main idea of this new representation is to project the holds on a "moving grid", composed of 35 (5×7) rectangular tiles with size (0.3m × 0.5m) (see Fig. 2(a)). The number of tiles is chosen to

[2]If this hypothesis is wrong, it only means we will miss some solutions of our problem. We are safe from trying to reach 4-configurations that are not C-reachable, which would have catastrophic results in a dynamic simulation.



(a) (b)

Fig. 2. (a) Projection on the grid of the virtual human of Fig. 1 (front view). Black crosses represent holds. Red circles show the tiles that contain the end effectors. The green dots are the P_i points that are projected on the free hands or feet. (b) Projection on the grid of all state-action pairs with free right hand (top left), free left hand (top right) and free feet (bottom left and right). For each map, maximal values are in white and minimal values in black. The green dots are the relevant P_i points.

keep the same combinatorial size as the previous problem and their size is chosen so that the grid covers $\mathcal{R}$.

The holds are now represented by their positions on the "grid", which change with the movements of the VH. In all other respects, the definition of a state as a vector of three holds and a free hand, as well as the definition of the action as the hold to reach remains the same.

In order to build this new representation, we project the holds on the grid as follows. We consider the four excentered P_i points (see Fig. 2(a), for example P_1 for the right hand is the top left-hand one) and we project the grid so that the P_i point lies on the position of the corresponding free hand or foot, before limb movement. We then simply identify a hold by the number of the tile where it lies at a given time. Conversely, when we need to project the grid on the holds (chose one hold corresponding to a given tile), we take the hold that is the nearest from the center of the tile.

We could use our new state and action representation in a classic RL algorithm, but then we would lose the possibility to use an efficient (*i.e.* exact, though incomplete) model. The transition function is more complicated in the grid representation than in the wall-based one: during a movement of the VH, the descriptors of the holds under the grasping hands or feet change, because the free hand and the P_i point change. Moreover, the size of the tiles implies that a state may refer to several hold configurations, so we cannot keep a repertoire of known postures and have to compute a new articular configuration corresponding to known states when we meet these states again.

In order to have both the computational efficiency of the wall-based representation and the generalisation properties through the grid representation, we use both representations and solve the problem with two parallel RL processes.

Algorithm 2: Parallel Reinforcement Learning Processes in Problem-Space and Agent-Space

1 put VH in initial 4-configuration C^4_{init}: $G \leftarrow G_{init}$;
2 $free\_hand \leftarrow 1$;
3 next state s' is initial state $s \leftarrow s_{init}$;
4 **while** *true* **do**
5 $s \leftarrow s'$;
6 chose action a in s;
7 compute new 4-configuration C^4;
8 $G_{old} \leftarrow G$;
9 $G = \mathcal{IK}(G, C^4, H(C^4))$;
10 **if** $G == G_{old}$ **then**
11 $r \leftarrow -1000$;
12 next state s' is current state: $s' = s$;
13 **else**
14 **if** VH *passed final height* **then**
15 $r \leftarrow 200000$;
16 next state s' is initial state: $s' \leftarrow s_{init}$;
17 $free\_hand \leftarrow 1$;
18 **else**
19 $r \leftarrow -1$;
20 $s' \leftarrow next\_state(s, a, free\_hand)$;
21 $free\_hand \leftarrow free\_hand + 1 \mod 4$;
22 Update(Q_{wall},s,a,s',r);
23 Update(Q_{grid},s,a,s',r);
24 $Model(s, a) = (s', r)$;
25 **for** $i = 1$ *to* 50 **do**
26 chose random action a_1 previously taken in s_1;
27 $(s_2, r_1) = Model(s_1, a_1)$;
28 Update(Q_{wall},s_1,a_1,s_2,r_1);
29 Update(Q_{grid},s_1,a_1,s_2,r_1);

We now consider states as couples $s = (s^{wall}, s^{grid})$ where s^{wall} is the descriptor of the state in problem-space, and s^{grid} is its translation on the grid. We use the same couples for actions. Our agent uses two different Q-tables. Q_{wall} uses the wall-based representation of states to learn in the problem-space whereas Q_{grid} uses the grid representation to learn the portable action value function. Both Q-tables are updated according to the standard Q-learning equation $Q(s,a) \leftarrow Q_(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$, with $\gamma = 0.9$ and $\alpha = 0.1$ (Algo. 2, lines 22,23).

The simulated transitions, that must speed up the convergence, are computed with the wall-based problem model and then are transferred to Q_{grid} (lines 24-29).

Our action selection algorithm is inspired by the classical ϵ-greedy method (see Algo. 3). Most of the time, we want to use the knowledge of Q_{wall} (lines 2-5) because RL on the wall-based problem more easily leads to an optimal policy. We then take the best action in the sense of Q_{wall} and call

Algorithm 3: Action Selection

1 agent is in state s ;
2 $a_1 \leftarrow$ best possible action in state s for Q_{wall};
3 $r_1 \leftarrow$ pseudo-random number in $[0, 1]$;
4 **if** $r_1 > \epsilon$ *and* a_1 *has already been taken in* s*;*
5 **then** chosen actions is a_1;
6 **else**
7 $\quad a'_1 \leftarrow$ best possible action in state s for Q_{grid};
8 $\quad r_2 \leftarrow$ pseudo-random number in $[0, 1]$;
9 $\quad$ **if** $r_2 > \epsilon$ *and* a'_1 *has already been taken in* s*;*
10 $\quad$ **then** chosen actions is a'_1;
11 $\quad$ **else**
12 $\quad\quad$ chose random possible action;

this choice *w-greedy*.

When the best action for Q_{wall} has never been taken in the considered state, we prefer to chose the best action according to Q_{grid}. This happens for example when the agent starts on a new wall after training a previous wall. We call the choice of the best action (lines 7-10) of Q_{grid} *g-greedy*.

When neither of the best actions for Q_{wall} and Q_{grid} were tried in the past, we chose a random action. We also want to take a random action from time to time because only exploration can ensure us convergence to the optimal policy. This choice is described as *random* (line 12).

IV. EXPERIMENTAL RESULTS

We evaluate the transfer capability of our agent by the following experiment. The VH first learns on a "training" wall for 8000 iterations. Then it is tested for "transfer" on another wall for another 8000 iterations, keeping only its learnt Q_{grid} because the learnt Q_{wall} cannot be reused. 8000 iteration is far too short to hope we can converge to an optimal policy: we hardly explore 0.5% of the action-state pairs. Performing 8000 iterations lasts around 30 hours on our machine, with an AMD Opteron 64 processor (2.6GHz) with 4Go of memory. Our simulation environment with our VH, as well as our inverse kinematics algorithm and our RL framework are implemented in Matlab. Unless stated otherwise, the figures presented in this sections are averages over 6 experiments for training and 6 for transfer.

There are several phases during the learning of the agent on the training wall, as best seen on Fig. 3, showing the cumulative number of actions chosen following the three selection regimes. First, the agent performs a random exploration of the states and actions. After about 1000 iterations, the agent starts using some knowledge from the grid and takes some *g-greedy* actions. Then, gradually, while the agent explores more and more states, a bigger number of *w-greedy* actions are taken. Those *w-greedy* actions are really exploitation of the acquired knowledge because their success rate (defined as the proportion of actions leading to reachable 4-configurations) is one (see Table I). Fig. 4 shows that the performance converges to about 30 moves, whereas the optimal policy always going up would take around 10 actions.

TABLE I

SUCCESS RATE AND VERTICAL DISPLACEMENT OF e_0 (cm) FOR SINGLE SUCCESSFUL ACTION: AVERAGES OVER THE 2,000 FIRST ITERATIONS (BEGINNING) AND OVER THE WHOLE EXPERIMENTS (OVERALL)

	beginning		**overall**	
	training	transfer	training	transfer
vert. displacement of e_0				
w-greedy	7.3	12.0	9.4	10.6
g-greedy	0.4	1.8	4.3	3.4
random	2.0	2.0	2.0	2.0
success rate				
w-greedy	100%	100%	100%	100%
g-greedy	46%	45%	60%	57%
random	43%	44%	43%	44%

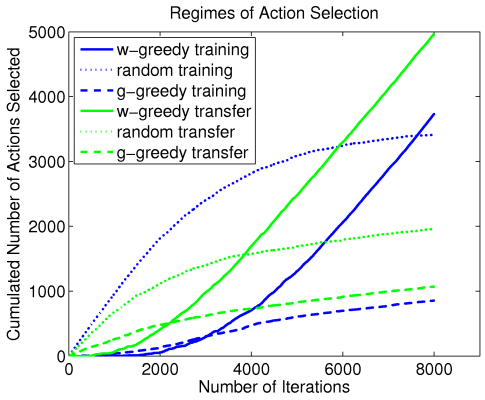


Fig. 3. Regimes of action selection

The knowledge acquired in the agent-space is represented as the values in Q_{grid}. High values code for state-action pairs leading the VH to the top of the wall. To visualize a policy, we first restrain to the states s_i^{rh} with a free right hand. Each action corresponds to the selection of a tile on the grid. For a tile on the grid a_j, we consider all the states (s_i^{rh}) where this action is possible and we sum the $Q_{grid}(s_i^{rh}, a_j)$ to produce the value of this action across all states. We then have a map that shows the tiles corresponding to good free right hand actions. We can also produce similar maps with the other end effectors as shown in Fig. 2(b). Tiles in black show actions that failed or were never performed, mostly because of our geometrical constraints. The preferred hand actions (tiles in white) are above the relevant P_i points (starting point of the limb movement), producing a movement upward.

Eventually, Fig. 4 shows that, during the first exploration phase, the episodes are significantly shorter in transfer than in training. As seen on Table I, the *g-greedy* actions in the beginning of the transfer experiments, when they mainly rely on the knowledge acquired during the training session, are more efficient (in terms of vertical displacement) than during training. Another way to visualize the transfer capabilities of the agent is on Fig. 3: the agent reaches the exploitation phase, when it uses only *w-greedy* actions, sooner in transfer

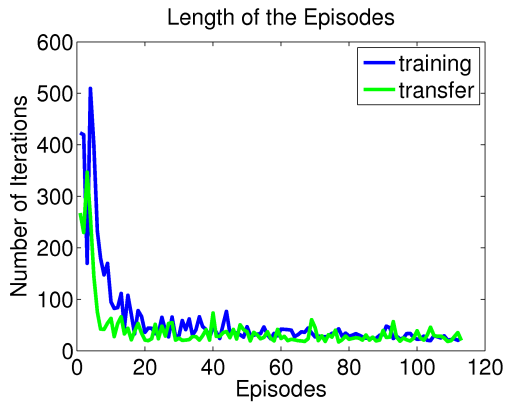


Fig. 4. Episode lengths during the experiments: transfer experiments converge faster

than in training. In the long run, the performance of the agent in transfer and in training are comparable.

V. DISCUSSION

Konidaris' framework does not give a unique method to compute a portable value function. It could be done, like in [10], by supervised learning from the pairs $((s_i, a_i), Q(s_i, a_i))$ obtained after convergence of the action-value function on the problem state. [9] obtains a policy by RL on a problem-space, then translates this policy in an agent-space and then back to another problem-space. We chose to learn the portable value function online, by a RL process on the agent-space that is parallel to the RL process on problem-space. This method ensures that, in the long run on any instance, our agent is efficient on the wall-based problem. Our approach with parallel RL processes also endows our agent with transfer capabilities. The agent uses knowledge of the agent-space obtained in a previous problem (contained in Q_{grid}) to reduce the time spent in the first exploration phase on a new problem, without overspecialising, *i.e.* loss of performance on the long run. This is a useful result because most classical RL techniques fail to improve the first exploration phase in a goal-directed problem. A similar result is obtained in [10] with "shaping".

Eventually, our case study is a very simplified formulation of the real-world human climbing problem. But in order to create dynamic movements, kinetic aspects have to be computed somehow: we now have a kinematic plan, a sequence of postures to be reached, that just needs to be implemented dynamically. Anyways, our approximations and simplifications enabled us to obtain encouraging results.

The geometrical constraints on the holds configurations and our heuristic to place the root body of the VH are inspired by human wall-climber behaviours and they give rather human-like results. However, some hand actions reach holds that are below shoulder level. This does not produce a vertical movement to the top and leads to postures that would be difficult to maintain in a dynamic context: it takes less effort for human climbers to hang from holds than to support their weight with their arms. In order to solve these problems, one may improve the realism of our results by considering dynamical aspects and costs based on effort.

VI. CONCLUSIONS

In this paper, we described our use of a transfer framework in RL. We solved a planning problem for a VH performing kinematics movements on a climbing wall with two different RL representations. A first representation used absolute positions of the holds and the second one a subjective representation from the point of view of the agent. We presented an original RL algorithm that enabled us to use both action and state representations in parallel, so as to give transfer capabilities to our agent. We have shown that our agent can solve efficiently problems in training and in transfer. In transfer, our RL method reduces the time spent for the first exploration phase and does not affect the performance in the long run.

VII. ACKNOWLEDGMENTS

The authors would like to thank Cyrille Collette (CEA/LIST) for his help in the use of the VH and ENSTA ParisTech for granting access to their machines.

REFERENCES

[1] Calinon, S., Guenter, F., Billard, A.: Goal-directed imitation in a humanoid robot. Int. Conf. on Robotics and Automation (2005)
[2] Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal control for redundantly actuated arms. Proc. Tenth Int. Conf. on the Simulation of Adaptive Behavior (2008)
[3] Nguyen-Tuong, D., Peters, J., Seeger, M., Schoelkopf, B.: Learning inverse dynamics: a comparison. Proc. of the Euro. Symp. on Artificial Neural Networks (2007)
[4] Peters, J., Schaal, S.: Reinforcement learning for operational space control. Int. Conf. on Robotics and Automation (2007)
[5] Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. The MIT Press (1998)
[6] Watkins, C., Dayan, P.: Q-learning. Machine Learning **8**(3-4) (1992) 279–292
[7] Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. Int. Conf. on Machine Learning (1990)
[8] Konidaris, G.: A framework for transfer in reinforcement learning. Int. Conf. on Machine Learning (2006)
[9] Stolle, M., Atkenson, C.: Knowledge transfer using local features. 2007 (Proc. of the IEEE Symp. on Approx. Dyn. Prog. and RL)
[10] Konidaris, G., Barto, A.: Autonomous shaping: Knowledge transfer in reinforcement learning. ICML (2006)
[11] Collette, C., Micaelli, A., Andriot, C., Lemerle, P.: Dynamic balance control of humanoids for multiple grasp and non coplanar frictional contacts. Int. Symp. on Visual Computing (2007) 734–744
[12] Dempster, W., Gaughran, G.: Properties of body segments based on size and weight. American Journal of Anatomy (120) (1967) 33–45
[13] Hanavan, E.: Mathematical model of the human body. Wright-Paterson Air Force Base (1964)
[14] Sitti, M., Fearing, R.: Synthetic gecko foot-hair micro/nano-structures for future wall-climbing robots. ICRA (2003)
[15] Yano, T., Numao, S., Kitamura, Y.: Developpment of a self-contained wall climbing robot with scanning type suction cups. Intl. Conf. on Intelligent Robots and Systems (1998)
[16] Nagabuko, A., Hirose, S.: Walking and running of the quadruped wall-climbing robot. IEEE Int. Conf. on Rob. and Aut. **2** (1994) 1005–1012
[17] Bretl, T., Rock, S., Latombe, J.C., Kennedy, B., Aghazarian, H.: Free-climbing with a multi-use robot. Int. Symp. Exp. Rob. (2004)
[18] Bretl, T., Lall, S., Latombe, J.C., Rock, S.: Multi-step motion planning for free-climbing robots. WAFR (2004)
[19] Liu, C., Popovic, Z.: Synthesis of complex dynamic character motion from simple animation. ACM Transactions on Graphics (2002)
[20] Treuille, A., Lee, Y., Popovic, Z.: Near-optimal character animation with continuous control. ACM Transactions on Graphics (2007)