Direct Value Learning: a Preference-based Approach to Reinforcement Learning

David Meunier[1], **Yutaka Deguchi**[2], **Riad Akrour**[1]
Einoshin Suzuki[2], **Marc Schoenauer**[1], **Michele Sebag**[1]

Abstract. Learning by imitation, among the most promising techniques for reinforcement learning in complex domains, critically depends on the human designer ability to provide sufficiently many demonstrations of satisfactory quality.

The approach presented in this paper, referred to as DIVA (Direct Value Learning for Reinforcement Learning), aims at addressing both above limitations by exploiting simple experiments. The approach stems from a straightforward remark: while it is rather easy to set a robot in a target situation, the quality of its situation will naturally deteriorate upon the action of naive controllers. The demonstration of such naive controllers can thus be used to learn directly a value function, through a preference learning approach. Under some conditions on the transition model, this value function enables to define an optimal controller.

The DIVA approach is experimentally demonstrated by teaching a robot to follow another robot. Importantly, the approach does not require any robotic simulator to be available, nor does it require any pattern-recognition primitive (e.g. seeing the other robot) to be provided.

1 Introduction

Since the early 2000s, significant advances in reinforcement learning (RL) have been made through using direct expert's input (inverse reinforcement learning [18], learning by imitation [10], learning by demonstration [16]), assuming the expert's ability to demonstrate quasi-optimal behaviors and to provide an informed representation.

In 2011, new RL settings based on preference learning and allegedly less demanding for the expert have been proposed (more in section 2).

In this paper, a new preference-based reinforcement learning approach called DIVA (*Direct Value Learning for Reinforcement Learning*) is proposed. DIVA aims at learning directly the value function from basic experiments. The approach is illustrated on the simple problem of teaching a robot to follow another robot. It is shown that DIVA yields a competent follower controller without requiring the primitive "I see the other robot" to be either provided, or explicitly learned.

2 State of the art

Reinforcement learning is most generally formalized as a Markov decision process. It involves a state space $\mathcal{S}$, an action space $\mathcal{A}$, and an upper bounded reward function r defined on the state space $r : \mathcal{S} \mapsto \mathbb{R}$. The model of the world is given by the transition function $p(s, a, s')$, expressing the probability of arriving in state s' on making action a in state s under the Markov property; in the deterministic case, the transition function $tr : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ gives the state $tr(s, a)$ of the agent upon making action a in state s. A policy $\pi : (\mathcal{S}, \mathcal{A}) \mapsto \mathbb{R}$ maps each state in $\mathcal{S}$ on some action in $\mathcal{A}$ with a given probability. The return of policy π is defined as the expectation of cumulative reward gathered along time when selecting the current action after π, where the initial state s_0 is drawn after some probability distribution q on $\mathcal{S}$. Denoting $a_h \sim \pi(s_h)$ the random variable action selected by π in state s_h, $s_{h+1} \sim p(s_h, a_h, s)$ the state of the agent at step $h + 1$ conditionally to being in state s_h and selecting action a_h at step h, and r_{h+1} the reward collected in s_{h+1}, then the policy return is

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \gamma^h r_h | s_0 \sim q \right]$$

where $\gamma < 1$ is a discount factor enforcing the boundedness of the return, and favoring the reaping of rewards as early as possibly in the agent lifetime.

The so-called value function $V_\pi(s)$ estimates the expectation of the cumulative reward gathered by policy π when starting in state s, recursively given as:

$$V_\pi(s) = r(s) + \gamma \sum_{a,s'} \pi(s, a) p(s, a, s') V_\pi(s')$$

Interestingly, from a value function V can be derived a greedy policy π_V, provided the transition function is known: when in state s, select the action a leading to the state with best expected value:

$$\pi_V(s) = \text{argmax}_{a \in \mathcal{A}} \left\{ \begin{array}{ll} p(s, a, s')V(s') & \text{probabilistic trans.} \\ V(tr(s, a)) & \text{deterministic trans.} \end{array} \right\} \quad (1)$$

By construction, π_{V_π} is bound to improve on π. By learning value function V^* as the maximum over all policies π of V_π

$$V^*(s) = max_\pi V_\pi(s)$$

one can thus derive the optimal policy $\pi^* = \pi_{V^*}$.

The interested reader is referred to [19, 20] for a comprehensive presentation of the main approaches to Reinforcement Learning, namely value iteration and policy iteration algorithms, building a sequence of value functions V^i and policies π_i converging to V^* and π^*. The bottleneck of estimating the optimal value function is that all states must be visited sufficiently often, and all actions must be triggered in any state, in order to enforce the convergence of V^i and $\pi(V^i)$ toward V^* and π^*. For this reason, RL algorithms hardly scale up when the size of the state and action spaces is large, all the more so as the state description must encapsulate every information relevant to action selection in order to enforce the Markov property.

New RL approaches devised to alleviate this bottleneck and based on preference learning have been proposed in 2011 [11, 2]. [11] is concerned with the design of the reward function in order to facilitate RL; for instance in the medical protocol application domains,

[1] TAO, CNRS-INRIA-LRI, Université Paris-Sud
[2] Dept. Informatics, ISEE, Kyushu University

how to associate a numerical negative reward to the patient's death ? The authors thus extend the classification-oriented approach first proposed by [17] as follows. In a given state s, an action a is assessed by executing the current policy until reaching a terminal state (rollout). On the basis of these assessments, pairs of actions can be ranked with regard to state s and policy π ($a <_{s,\pi} a'$). These ranking constraints are exploited within a learning-to-rank algorithm (e.g. RankSVM [13]), yielding a ranking hypothesis $h_{s,\pi} : \mathcal{A} \mapsto \mathbb{R}$. The claim is that such action ranking hypotheses are more flexible than classification hypotheses, aimed at discriminating "the" best actions from the others conditioned by the current state. In summary, the ranking hypothesis depends on the policy and the current state, and operates on the action space.

Quite the contrary, in [2] the ranking hypothesis operates on the policy space. The motivating application is swarm robotics, facing two severe issues. Firstly, swarm robotics is hardly compatible with generative model-based RL approaches; simulator-based approaches suffer from the supra-linear computational complexity of simulations w.r.t. the number of robots in the swarm (besides the simulation noise). Secondly, swarm robotics hinders the inverse reinforcement learning approach [1, 15], using the expert demonstrations to learn a reward function. In most cases the swarm expert cannot describe (let alone demonstrate) the individual robot behavior leading to the desired swarm behavior (known as the inverse macro-micro problem [7]). The proposed approach, called PPL (*Preference-based Policy Learning*) proceeds along an interactive optimization setting: the robot(s) demonstrates a behavior, which is ranked by the expert comparatively to the previous best demonstration. The ranking constraints are exploited through a learning-to-rank algorithm, yielding a ranking hypothesis on the policy demonstrations and thus on the policy space Π ($h : \Pi \mapsto \mathbb{R}$). This ranking hypothesis is used as policy return estimate, casting RL as an optimization problem (find $\pi^*_h =$ argmax $h(\pi)$). Policy π^*_h is demonstrated to the expert, who ranks it compared to the previous best demonstration, and the process is iterated. Note that PPL thus faces the same difficulty as interactive optimization at large [9, 22]: if the expert is presented with too constrained a sample of demonstrations, she does not have a chance to teach her preferences to the system. PPL is thus extended to integrate an active learning criterion, yielding the APRIL (*Active Preference learning-based Reinforcement Learning*) algorithm [3].

3 Overview of DIVA

This section introduces and formalizes the principle of DIVA, and discusses its strengths and limitations w.r.t. the state of the art.

3.1 Principle

DIVA is rooted in Murphy's law (*Anything that can possibly go wrong, does*). Formally, it posits that when the agent happens to be in some good situation, its situation tends to deteriorate under *most* policies. Let us illustrate this idea on the simple problem of having a robot following another robot in an open environment. Assume that the follower robot is initially situated behind the leader robot (Fig. 1, left, depicts the follower state, given as its camera image). Assume that both follower and leader robots are equipped with the same simple *Go Ahead* controller (same actuator value on the left and right wheel of both robots). Almost surely, each robot trajectory will deviate from the straight line, due to e.g. the imperfect calibration of the wheel actuators or different sliding frictions on the ground. Almost surely, the two robot trajectories will be deviated in a different way. Therefore, the follower will at some point lose track of the leader (Fig. 1, right, depicts the follower state after circa 52.7 (+-20.6) time steps, that is, XX seconds).

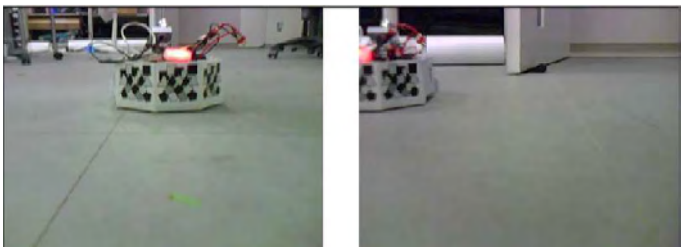

Figure 1: Left: The follower robot is initially aligned behind the leader robot. Right: Both leader and follower robots are operated by the same *Go ahead* controller. Due to mechanical drift, the follower sooner or later loses track of the leader.

The intuition can be summarized as: the follower state was never as good as in the initial time step; it becomes worse and worse along time.

3.2 Formalization

The above remarks enable to define a ranking hypothesis on the state space, as follows. Let us consider K trajectories of the robot follower noted $S_1 \dots S_K$, where each trajectory S_i is defined as a sequence of states $s_t^{(i)}, t = 0 \dots T_i$. A value function is sought as a function V mapping the set of states $\mathcal{S}$ onto $\mathbb{R}$, satisfying constraints $V(s_t^{(i)}) > V(s_{t+1}^{(i)})$ for all $i = 1 \dots K$ and $t = 0 \dots T_i - 1$.

Formally, it is assumed in the following that the state space $\mathcal{S}$ is embedded in $\mathbb{R}^d$. A linear value function $\widehat{V^*} : \mathcal{S} \mapsto \mathbb{R}$ is defined as

$$\widehat{V^*}(s) = \langle \widehat{w}^*, s \rangle$$

where $\widehat{w}^* \in \mathbb{R}^d$ is given after the standard learning-to-rank regularized formulation [4]:

$$\begin{array}{c} \widehat{w}^* = \text{ arg min } \frac{1}{2}||w||_2^2 + C \sum_{i=1}^k \sum_{t<t' \leq T_i} \xi_{t,t'}^{(i)} \\ s.t. \, \forall \, 1 \leq i \leq K, \, 0 \leq t < t' \leq T_i \\ \langle w, s_t^{(i)} \rangle \geq \langle w, s_{t'}^{(i)} \rangle + 1 - \xi_{t,t'}^{(i)}; \qquad \xi_{t,t'}^{(i)} \geq 0 \end{array} \tag{2}$$

This quadratic optimization under constraints problem can be solved with affordable empirical complexity [14]. After Eq. (1) and provided that the transition model is known, value function $\widehat{V^*}$ derives a policy $\widehat{\pi}^*$. Further, by construction any value function derived by monotonous transformation of $\widehat{V^*}$ induces the same policy $\widehat{\pi}^*$.

3.3 Discussion

Among the main inspirations of the DIVA approach is TD-Gammon [21]. TD-Gammon, the first backgammon program to reach a champion level in the 80s, exploits games generated from self-play to train a value function along a temporal difference algorithm. The value function is likewise trained from a set of games, or trajectories S_i described as a sequence of positions $s_t^{(i)}, t = 0 \dots T_i$. The difference is as follows. Firstly, TD-Gammon only imposes the value for the initial and the final positions, with $V(s_0^{(i)}) = 1/2$ (the initial position is neutral), and $V(s_{T_i}^{(i)}) = 1$ (respectively 0) if the first player wins (resp., loses) the game. Secondly, the learning problem

is regularized through a total variation minimization (minimizing $\sum_i \sum_{t=1}^{T_i-1}(V(s_t^{(i)}) - V(s_{t+1}^{(i)}))^2$).

A key difference between DIVA and TD-Gammon regards the availability of a simulator. If a generative model (a simulator) is available for the problem domain, then indeed RL can rely on cheap and abundant training data. In robotics however, simulator-based training is prone to the so-called reality gap [8]. Generally speaking, the robotic framework is hardly compatible with data-intensive learning approaches, due to the poor accuracy of simulator-based data on the one hand, and the cost (experimenter time and robot fatigue) of in-situ experiments on the other hand.

A consequence is that TD-Gammon can exploit high quality simulated data whereas DIVA relies on a limited amount of data. Further, these data are experimentally acquired and should require little or no expertise from the human experimenter. Along the same line, TD-Gammon only prescribes the values attached to the initial and final state of each trajectory; the bulk of learning relies on the regularization term. Quite the contrary, DIVA exploits short trajectories and sets comparison constraints on the values attached to each time step (besides using regularization too). Finally, TD-Gammon requires all winning/losing terminal states to be associated the same value; there is no such constraint in DIVA, as states from different trajectories are not comparable.

Both approaches suffer from a same limitation: policy π_V is computed from value function V iff the transition function is known or well approximated (Eq. (1)). Further, finding the optimal action requires one to solve an optimization problem in each time step, which usually requires the action space to be small. It will be seen (section 4.2) that a cheap estimate of the transition function can alleviate the above limitations in the robotic framework in some cases. An additional limitation of DIVA is that the value function V is learned from a set of trajectories obtained from a "naive" controller, which does not necessarily reflect the target (test) distribution, thus raising a transfer learning problem [6]. This issue will be discussed in section 5.

4 Experimental validation

A proof of principle of DIVA is given in a robotic framework (section 4.1), based on a delayed transition model estimate (section 4.2) and a continuity assumption (section 4.3). The experimental setting and goals of experiments are described in section 4.4 and results are reported and discussed in section 4.5.

4.1 Framework

The robotic platform is a Pandaboard, driven by the dual-core ARM Cortex-A9 OMAP4430, with each core running at 1 GHz, and equipped with 1 GB DDR2 RAM. The robot is equipped with a USB camera with resolution (320×240), and color depth of monochrome 8bit. All experiments are done in-situ, taking place in a real laboratory full of tables, chairs, feet and various (moving) obstacles. Although no model of the world (transition function or simulator) is available, a cheap estimate thereof can be defined (see below).

The primary description of the robot state is given by its camera image (in $\mathbb{R}^{76800}$). A pre-processing step aimed at dimensionality reduction and inspired from the SIFT (scale-invariant feature transform) descriptors is used and operated on the Pandaboard using the Linux OS. More precisely, SURF (speeded up robust feature) descriptors are used, claimed to be several times faster and more robust against different image transformations than SIFT [5]. Finally, each block of $d \times d$ contiguous pixels in the initial image is represented by an integer, the number of SURFs occurring in the block ($d = 1, 2, 4, 16$ in the experiments, with $d = 4$ the best empirical setting, and the only considered in the remainder of the paper). The state space $\mathcal{S}$ is finally included in $\mathbb{R}^{4800}$.

4.2 Delayed transition model estimate

In the following, the action space $\mathcal{A}$ includes three actions: *Ahead*, *Right* and *Left*. The extension to richer and more gradual action spaces is left for further work. As already mentioned, the definition of a controller from a value function requires a transition model to be available (Eq. (1)). Two working assumptions are done to overcome the lack of an accurate transition model for the considered open world.

Firstly, a delayed transition model estimate is defined as follows. Let s_{t-1} and s_t denote the states at time $t-1$ and t respectively, and assume that the action selected at time $t-1$ is *Ahead*. Then, the idea is that if the robot had turned right at time $t-1$, it would have seen approximately the same image as in s_t, but translated on the left; additionally, some new information would have been recorded on the rightmost camera pixels while the information on the leftmost pixels would have been lost. In other words, given $s_t = tr(s_{t-1}, \textit{Ahead})$ one can compute an approximation of $tr(s_{t-1}, \textit{Right})$, by a circular shift of s_t. Let state s_t be described as a pixel matrix $s_t[i,j]$ where $i = 1 \ldots n_w,\ j = 1 \ldots n_h$ and n_w (respectively n_h) is the width (resp. height) of the camera image. Then,

$$tr(s_{t-1}, \textit{Right}) \approx \{s_t[i + \Delta \pmod{n_w}, j]\}$$

where Δ is a constant translation width (64 pixels in the experiments; this will be relaxed in further work, section 5). Note that the unknown rightmost pixels in $tr(s_{t-1}, \textit{Right})$ are arbitrarily replaced by the leftmost pixels in s_t; the impact of this approximation will be discussed further.

More generally, given $s_t = tr(s_{t-1}, a)$, a delayed transition model estimate is given by

$$\widehat{tr}(s_{t-1}, a' | a, s_t = tr(s_{t-1}, a)) = \{s_t[i + \ell(a, a') \pmod{n_w}, j]\}$$

where the translation width $\ell(a, a')$ is Δ (resp. $-\Delta$) if a= *Ahead*, a' =*Right* (resp. *Left*), and completed by consistency (Table 3).

a	a'	$\ell(a,a')$
$Ahead$	$Right$	Δ
$Left$	$Ahead$	Δ
$Left$	$Right$	2Δ

a	a'	$\ell(a,a')$
$Ahead$	$Left$	$-\Delta$
$Right$	$Ahead$	$-\Delta$
$Right$	$Left$	-2Δ

(3)

This estimate is delayed as it is only available at time t, since $\widehat{tr}(s_{t-1}, a' | a, s_t = tr(s_{t-1}, a))$ is computed from s_t. To some extent, this estimate can cope with partially observable environments (another robot of the swarm might arrive in sight of the current robot at t, while it was not seen at time $t-1$). On the other hand, the estimation error is known to be concentrated in the peripheral pixels.

4.3 Continuity assumption

Given a value function $\widehat{V^*}$, the action a_{t-1} which has been selected at $t-1$ and the current state s_t, the delayed transition model estimate defines *what would have been the best action* a^*_{t-1} that should have been selected at time $t-1$ instead of a_{t-1}, after Eq. (1):

$$a^*_{t-1} = argmax_{\ a \in \mathcal{A}} \left\{ \widehat{V^*}(\widehat{tr}(s_{t-1}, a | a_{t-1}, s_t)) \right\} \qquad (4)$$

Figure 2: A lesion study: recording irrelevant states at the beginning of the follower trajectory (the white board on the left and the author feet on the right).

The continuity assumption posits that the environment changes gracefully, implying that the action which was the most appropriate at time $t-1$ is still appropriate at time t. Along this line, the controller defined from $\widehat{V^*}$ and noted $\widehat{\pi}^*$ selects at time t action a^*_{t-1} as defined by Eq. (4). A further assumption behind the definition of $\widehat{\pi}^*$ is that the noise of the delayed transition model estimate is moderate with respect to $\widehat{V^*}$. In other words, it is assumed that the peripheral pixels (unknown in truth and arbitrarily filled from s_t) are not key to value function $\widehat{V^*}$ (the corresponding weights have low absolute value).

4.4 Experimental setting

The controller goal is to enable the follower robot to follow the leader robot.

Eleven trajectories are recorded. Each trajectory is initialized with the follower positioned behind the leader at various locations in the lab, and both robots operated with the constant policy $\pi_0(s) =$ *Ahead* for 128 time steps. The i-th trajectory thus records the follower state $s_t^{(i)}, t = 1 \ldots 128$. A computational time step amounts to circa .5 second of real-time (two frames per second), due to the on-board computation of the SURF descriptors. The value function $\widehat{V^*}$ is learned from these trajectories as in section 3.2.

The primary goal of experiments is to study the performance of controller $\widehat{\pi}^*$, in terms of the average time the follower can actually follow the leader. Further, in order to allow for larger time horizons, the controller run on the leader robot is an obstacle avoidance (Braitenberg) controller, enabling the leader to run for a couple of hundred time steps before being stopped. Note that modifying the leader policy amounts to considering a different environment, making the goal more challenging as the test setting differs from the training one: the leader can turn abruptly upon seeing an obstacle, whereas it turns only very gradually in the training trajectories.

The second goal of experiments is to assess the robustness of the approach with respect to noise. A lesion study is conducted by perturbing the initial states in the training trajectories, e.g. recording the images seen by the follower (e.g. the walls or the experimenter) before the follower actually starts to follow the leader (Fig 2). The value function learned from the perturbed trajectories and the associated controller are referred to as NOISY-DIVA.

The DIVA approach and the merits of a rank-based approach are also assessed by comparison with a simple regression based approach, referred to as REG-DIVA, where value function $\widehat{V^*}$ is regressed from the training set $\mathcal{E} = \{(s_t^{(i)}, -t), i = 1 \ldots 11, t = 1 \ldots 128\}$.

Finally, the assumption that peripheral pixels are not relevant to value function $\widehat{V^*}$ (section 4.3) is also examined, depicting the average weight of the value function for each (block of) pixel(s) in the image.

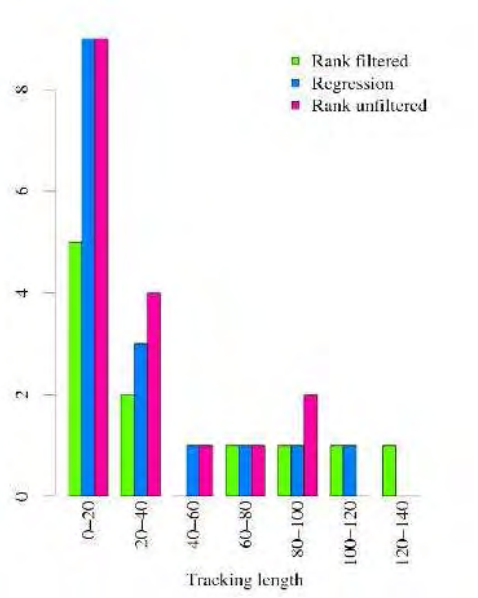

Figure 3: Comparative results of DIVA, NOISY-DIVA and REG-DIVA: Histogram of the number of consecutive time steps the follower keeps on following a Braitenberg-operated leader, over 5 runs. One time step corresponds to ca half a second, due to the on-board computation of the SURF descriptors (2 frames per second).

4.5 Results

The performance of the DIVA controller is assessed by learning $\widehat{V^*}$ from all 11 training trajectoriesin DIVA NOISY-DIVA and REG-DIVA modes and running the associated $\widehat{\pi}^*$ controllers. Every controller $\widehat{\pi}^*$ is launched on the follower robot for five runs. In each run, the follower is initially positioned behind the Braitenberg-operated leader, at various locations in the lab (same initial locations for DIVA, NOISY-DIVA and REG-DIVA settings). In each run, the follower is manually repositioned behind the leader when it loses the track.

The histogram over the 5 runs of the number of consecutive time steps while the follower does follow the leader is displayed in Fig. 3, reporting the respective performances of DIVA, NOISY-DIVA and REG-DIVA. As was expected, the best results are obtained for the noiseless rank-based DIVA approach, followed by the noisy rank-based NOISY-DIVA approach. The fact that some tracking sequences are very short is explained as the leader meets very soon an obstacle and turns fast, making the follower lose the track. Overall, the follower stays on the leader track for over 60 time steps in about 40% of the experiments while the track was lost after 52.7 (+-20.6) time steps in the training experiments.

The performance of the DIVA controller is also assessed on the training trajectories along a leave-one-out procedure, learning 11 value functions $\widehat{V^{*(i)}}$ from all trajectories but the i-th one. $\widehat{V^{*(i)}}$ is computed on the remaining i-th trajectory (Fig. 4). As expected, the value of $\widehat{V^{*(i)}}$ decreases along time. Interestingly, DIVA and NOISY-DIVA exhibit similar behaviors, rapidly decreasing as t increases although the value remains high for some runs; after visual inspection, the drift occurs late for these runs. The behavior of REG-DIVA is more erratic, which is explained as the regression setting is over constrained regarding the quality of the experiments, requiring all states $s_t^{(i)}$ to be associated to the same value $-t$.

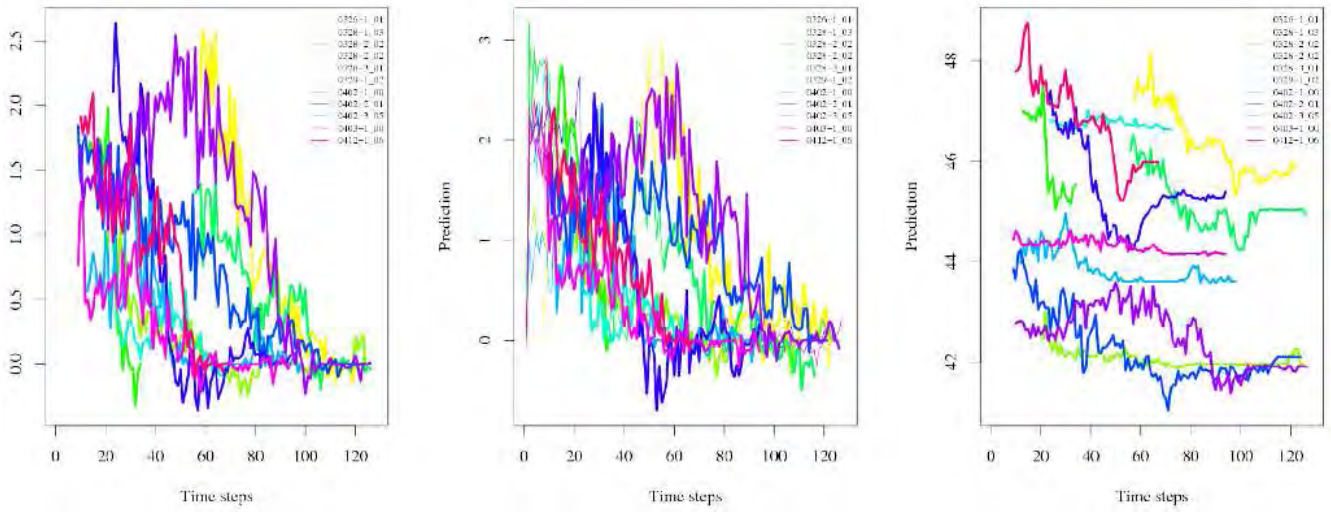

Figure 4: Comparative behavior of the value function $\widehat{V^*}_i$ along time. A leave-one-out procedure is used: the value function is learned from all but one trajectory, and its value on the remaining trajectory is reported. The value function learned by DIVA, NOISY-DIVA and REG-DIVA are reported respectively on the left, middle and right.

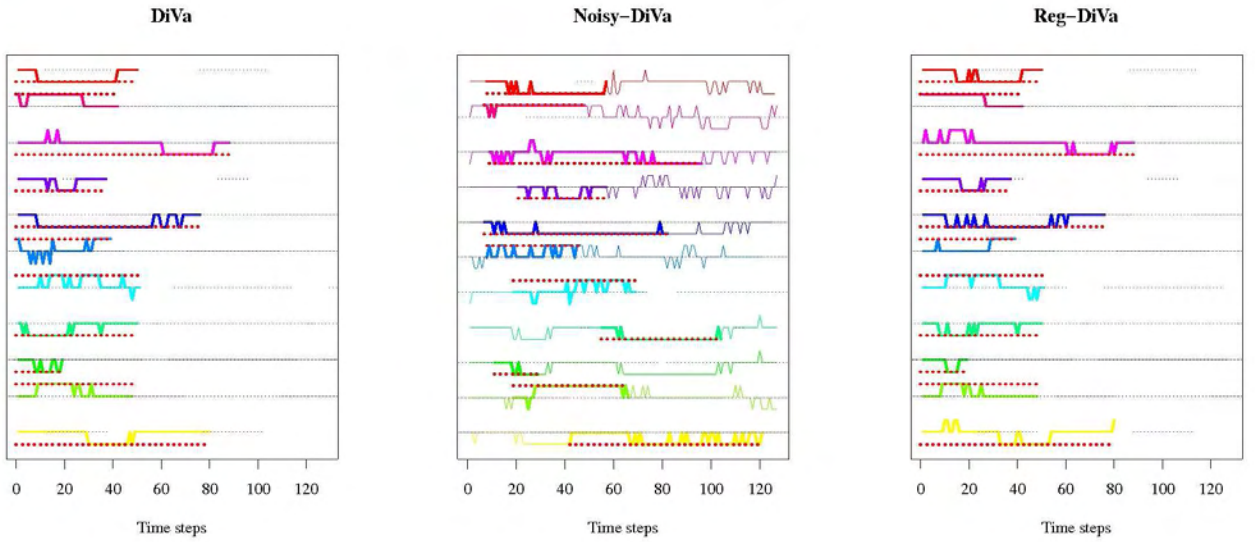

Figure 5: Comparative behavior of controller $\pi_{\widehat{V^*}_i}$ on the test trajectory along time, for DIVA (left), NOISY-DIVA (medium) and REG-DIVA (right).

The corresponding policy $\widehat{\pi}^{*(i)}$is plot in Fig. 5, where $\widehat{\pi}^{*(i)}(s_t^{(i)})$ is indicated by a cross respectively on, above or below over the central line depending on whether the selected action is *Ahead*, *Right* or *Left*. The ground truth obtained by visual inspection of the logs is reported below. It is seen that the policy generally selects the relevant action, and becomes chaotic in the end of the trajectory as the follower does no longer see the leader.

Finally, the working assumption that peripheral pixels hardly matter for the value function (section 4.3), implying that the noise of the delayed transition model estimate does not harm the controller, is confirmed by inspecting the average weight of the pixels in Fig. 6. As could have been expected, the most important region is the central upper one, where the leader is seen at the beginning of each training trajectory; the low region overall is considered uninformative. In the upper regions, the weight decreases from the center to the left and right boundaries.

5 Discussion and perspectives

This paper has presented a proof of principle of the DIVA approach, showing that elementary experiments can be used to "prime the pump" of reinforcement learning and train a mildly competent value function. Indeed, a controller with comparable performance might have been manually written (hacked) easily. Many a student experience suggests however that the manual approach is subject to the law of diminishing returns, as more and more efforts are required to improve the controller as its performance increases.

The next step thus is to study the scalability of DIVA, e.g. by gradually adding new logs to the training set. On-going experiments consider the effects of adding the test trajectories (with Braitenberg-

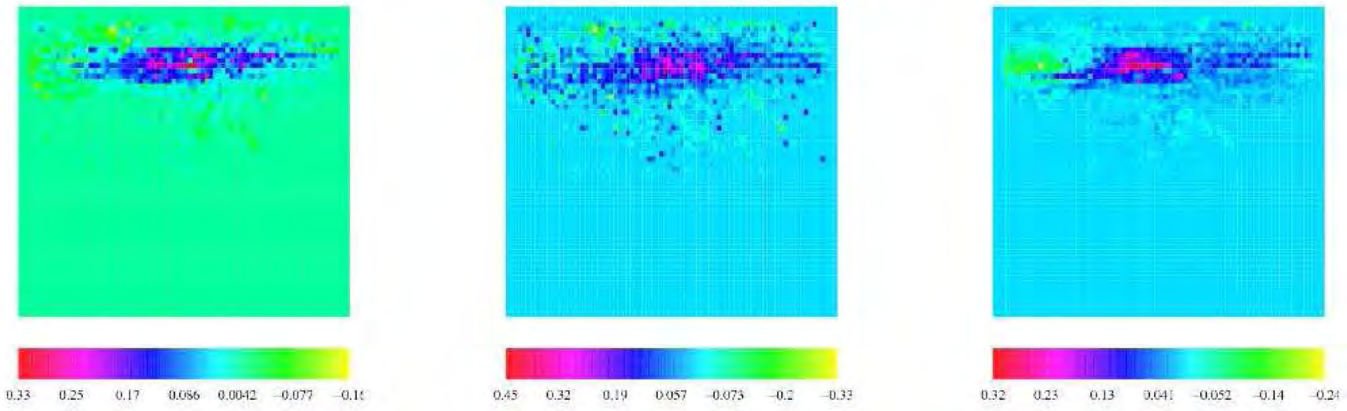

Figure 6: Average relevance of the visual regions to the value function for DIVA (left), NOISY-DIVA (medium) and REG-DIVA (right).

operated leader) to the training trajectories to retrain the value function.

The main current limitation of the approach concerns the small size of the action space. Still, it must be noted that the delayed transition model estimate naturally extends to fine-grained actions, by mapping the group of movement actions on the group of image translation vectors (section 4.2), thus enabling to determine the optimal action in a large action space. For computational efficiency, the underlying requirement is that the target behavior be sufficiently smooth. Currently, this requirement is hardly compatible with the low number of frames per second, due to the on-board computation of the SURF descriptors. On-going study is considering more affordable image pre-processings.

Another applicative perspective is to learn a docking controller, enabling robots in a swarm to dock to each other. This goal is amenable to a DIVA approach, as training trajectories can be easily obtained by initially setting the robots in a docked position, and having every robot to undock and start wandering along time. Like for the follower problem, the best state is the initial one and the value of the robot state decreases along time, in the spirit of Murphy's law.

Acknowledgments

The authors gratefully acknowledge the support of the FP7 European Project *Symbrion*, FET IP 216342, `http://symbrion.eu/`, and the ANR Franco-Japanese project Sydinmalas ANR-08-BLAN-0178-01. A part of this research was supported by JST and the grants-in-aid for scientific research on fundamental research (B) 18300047 and on challenging exploratory research 24650070.

REFERENCES

[1] P. Abbeel and A.Y. Ng, 'Apprenticeship learning via inverse reinforcement learning', in *ICML*, ed., Carla E. Brodley, volume 69 of *ACM International Conference Proceeding Series*. ACM, (2004).

[2] Riad Akrour, Marc Schoenauer, and Michèle Sebag, 'Preference-based policy learning', In Gunopulos et al. [12], pp. 12–27.

[3] Riad Akrour, Marc Schoenauer, and Michèle Sebag, 'April: Active preference-based reinforcement learning', in *ECML/PKDD*, p. to appear, (2012).

[4] G. Bakir, T. Hofmann, B. Scholkopf, A.J. Smola, B. Taskar, and S.V.N. Vishwanathan, *Machine Learning with Structured Outputs*, MIT Press, 2006.

[5] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool, 'Surf: Speeded up robust features', in *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pp. 404–417, (2006).

[6] Steffen Bickel, Christoph Sawade, and Tobias Scheffer, 'Transfer learning by distribution matching for targeted advertising', in *Advances in Neural Information Processing Systems, NIPS 21*, pp. 145–152. MIT Press, (2008).

[7] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Santa Fe Institute Studies in the Sciences of Complexity, Oxford University Press, 1999.

[8] Lipson H. Bongard J., Zykov V., 'Resilient machines through continuous self-modeling', *Science*, **314**(5802), 1118 – 1121, (2006).

[9] E. Brochu, N. de Freitas, and A. Ghosh, 'Active preference learning with discrete choice data', in *Advances in Neural Information Processing Systems 20*, pp. 409–416, (2008).

[10] S. Calinon, F. Guenter, and A. Billard, 'On Learning, Representing and Generalizing a Task in a Humanoid Robot', *IEEE transactions on systems, man and cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, **37**(2), 286–298, (2007).

[11] Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park, 'Preference-based policy iteration: Leveraging preference learning for reinforcement learning', In Gunopulos et al. [12], pp. 312–327.

[12] Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, eds. *Proc. European Conf. on Machine Learning and Knowledge Discovery in Databases, ECML PKDD, Part I*. LNCS 6911, Springer Verlag, 2011.

[13] Thorsten Joachims, 'A support vector method for multivariate performance measures', in *ICML*, eds., Luc De Raedt and Stefan Wrobel, pp. 377–384, (2005).

[14] Thorsten Joachims, 'Training linear svms in linear time', in *KDD*, eds., Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, pp. 217–226. ACM, (2006).

[15] J. Zico Kolter, Pieter Abbeel, and Andrew Y. Ng, 'Hierarchical apprenticeship learning with application to quadruped locomotion', in *NIPS*. MIT Press, (2007).

[16] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen, 'Constructing skill trees for reinforcement learning agents from demonstration trajectories', in *Advances in Neural Information Processing Systems 23*, pp. 1162–1170, (2010).

[17] Michail Lagoudakis and Ronald Parr, 'Least-squares policy iteration', *Journal of Machine Learning Research (JMLR)*, **4**, 1107–1149, (2003).

[18] A.Y. Ng and S. Russell, 'Algorithms for inverse reinforcement learning', in *Proc. of the Seventeenth International Conference on Machine Learning (ICML-00)*, ed., P. Langley, pp. 663–670. Morgan Kaufmann, (2000).

[19] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[20] Csaba Szepesvári, *Algorithms for Reinforcement Learning*, Morgan & Claypool, 2010.

[21] Gerald Tesauro, 'Programming backgammon using self-teaching neural nets', *Artif. Intell.*, **134**(1-2), 181–199, (2002).

[22] Paolo Viappiani and Craig Boutilier, 'Optimal Bayesian recommendation sets and myopically optimal choice query sets', in *NIPS*, pp. 2352–2360, (2010).