On-board Robot Interactive Training

Riad Akrour, Marc Schoenauer, and Michèle Sebag
TAO, CNRS − INRIA − LRI
Université Paris-Sud, F-91405 Orsay Cedex
FirstName.Name@lri.fr

***Abstract*—This paper presents a frugal approach hybridizing preference learning, active learning and reinforcement learning to build an accurate robotic controller by direct interaction with the human expert, emitting preferences among the behaviors demonstrated by the robot. A preference learning approach is used, allowing the robot to both gradually refine its policy utility estimate, and select a new policy after an Expected Utility of Selection criterion.**

The strengths and limitations of the approach are experimentally investigated in simulation and in-situ on the e-puck robotic platform.

I. Introduction

Reinforcement learning (RL) [16, 18], a well founded approach to optimal sequential decision making, has been extensively investigated in robotics (see e.g. [15]). From the applicative standpoint, RL relies on the smart design of the state and action spaces, and of the reward function, actually defining the optimization objective. Not only should the reward function reflect the target robot behavior; it should also induce a robust and tractable optimization problem. Since the 90s, some approaches have been proposed to learn an appropriate reward function, based on the demonstration of the target behavior by the expert (e.g. inverse reinforcement learning [14], learning by imitation [5], or learning by demonstration [12]). Since the late 2000s, alternative approaches have been proposed, based on the expert's preferences about the robot demonstrations and aimed at learning a reward function [6], a posterior over the parametric policy space [21] or a policy return function [2, 3].

This paper specifically investigates the strengths and limitations of preference-based reinforcement learning. Taking inspiration from the active preference-based reinforcement learning setting (April) [3], it proposes a Bayesian formulation aimed at fast satisfactory policy building, focussing on the policy robustness w.r.t. noisy expert's preferences. The practical merits of the new approach, called May, are investigated *in-situ*, establishing a proof of principle of the approach while shedding some light on the type of noise involved in the expert's judgments.

This paper is organized as follows. Section II briefly discusses work related to preference-based reinforcement learning and active preference learning. Section III gives an overview of May. Section IV is devoted to the empirical validation of the approach and the paper concludes with some perspectives for further research.

II. State of the art

This section briefly introduces the notations used throughout the paper, assuming the reader's familiarity with reinforcement learning and referring to [16] for a comprehensive presentation. Active preference learning-based reinforcement learning is discussed with respect to inverse reinforcement learning [1, 11], preference-based value learning [6] and Bayesian direct policy learning [21]. Lastly, the section introduces related work in active ranking, specifically in interactive optimization and online recommendation.

A. Formal background

Reinforcement learning classically considers a Markov decision process framework $(\mathcal{S}, \mathcal{A}, p, r, \gamma, q)$, where $\mathcal{S}$ and $\mathcal{A}$ respectively denote the state and the action spaces, p is the transition model ($p(s, a, s')$ being the probability of being in state s' after selecting action a in state s), $r : \mathcal{S} \mapsto \mathbb{R}$ is a bounded reward function, $0 < \gamma < 1$ is a discount factor, and $q : \mathcal{S} \mapsto [0, 1]$ is the initial state probability distribution. To each policy π ($\pi(s, a)$ being the probability of selecting action a in state s), is associated a policy return $J(\pi)$ being the expected discounted reward collected by π over time:

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

RL aims at finding the optimal policy $\pi^* = \arg\max J(\pi)$.

B. Policy Learning with no rewards

IRL takes as input an MDP$\backslash r$ (reward is unknown), a set of state sequences $c_k = (s_0^{(k)}\ s_1^{(k)}\ s_2^{(k)} \dots s_h^{(k)} \dots)$ generated by an expert, and a feature function $\phi(.)$ that maps each state onto a vector in $\mathbb{R}^D$; in the following the vectorial description $\mathbf{u}(c_k)$ of trajectory c_k is defined as $\mathbf{u}(c_k) = \sum_{h=0}^{\infty} \gamma^h \phi(s_h^{(k)})$. The exert is assumed to generate trajectories according to a policy maximizing a hidden reward r^*, though the goal of IRL is not to find r^* *per se* but only a policy that performs (near) optimally under it. In [1] the authors iteratively construct a set of policies performing as well as the expert policy. In [11], IRL is cast as a structured classification problem with the actions having the role of labels and expert trajectory being the learning set.

In [6] the authors use preference learning in replacement of classification algorithms in *Rollout Classification Policy Iteration* [9]. The authors advocate that action ranking is more

flexible and robust than a supervised learning based approach. In [3], the authors use preference learning to define a policy return estimate; the main difference with [6] is that the former defines an order relation on the action space depending on the current state and the current policy, whereas the later defines an order relation on the trajectories. In [21], the authors advocate the use of comparisons between short trajectories (as opposed to the full trajectories in APRIL ([3]) provided they can sample initial states from a distribution $P(s_0)$, increasing the probability of visiting interesting parts of the state space.

Formally, [21] maintains a distribution on the parametric policy search space, whereas [3] maintains a posterior on trajectory utilities. On the one hand, the former approach enables the direct sampling of policies from the posterior; in contrast, the latter one needs an additional step to get a policy maximizing a sampled utility from the posterior. On the other hand, in [21], it is more expensive to update the posterior distribution after a constraint is added, as a good amount of rollouts is necessary to compute the similarity between the trajectories generated from a certain policy parameter and the two trajectories of the added constraint.

C. Interactive optimization

The preference-based reinforcement learning approach requires the agent to find a new policy, expectedly relevant w.r.t. the current objective function J_t, with the goal of finding as fast as possible a (quasi) optimal solution policy. Interestingly this same goal, cast as an interactive optimization problem, has been tackled by [20, 4] in a Bayesian setting.

In [20], the system iteratively provides the user with a choice query, that is a (finite) set S of items described as vectors in $\mathcal{X} = \mathbb{R}^D$, of which the user selects the item she prefers. The ranking constraints are used to learn a linear utility function J on $\mathbb{R}^D$, with $J(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$ and $\mathbf{w}$ a vector in $\mathbb{R}^D$. Within the Bayesian setting, the uncertainty about the utility function is expressed through a belief θ defining a distribution over the space of utility functions.

Formally, the problem of (iterated) optimal choice queries is to simultaneously learn the user's utility function, and present the user with a set of good items, such that she can select one with maximal expected utility. Viewed as a single-step (greedy) optimization problem, the goal thus boils down to finding a items $\mathbf{x}$ with maximal expected utility $EU^*(\theta)$. In a global optimization perspective however [20], the goal is to find a choice query $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ with maximum *expected posterior utility*, defined as the expected gain in utility of the next decision $EPU(S;\theta) = \sum_{i=1}^{i=k} P_R(S \rightsquigarrow x_i;\theta) EU^*(\theta | S \rightsquigarrow x_i)$, where $S \rightsquigarrow x_i$ designs the event of the expert selecting the i$^{\text{th}}$ element of S, and P_R its probability under the expert response model. The (more tractable) expected utility of selection $EUS(S;\theta) = \sum_{i=1}^{i=k} P_R(S \rightsquigarrow x_i;\theta) EU(x_i;\theta | S \rightsquigarrow x_i)$ is shown to be quasi optimal and close to EPU in terms of optimal query set.

In [19], the issue of the maximum *expected value of information* (EVOI) is tackled, aimed at selecting $\mathbf{x}$ such that (where $\mathbf{x}^*$ is the current best solution):

$$EUS(\mathbf{x};\theta) = \mathbb{E}_{\theta, x > x^*}[\langle \mathbf{w}, \mathbf{x} \rangle] + \mathbb{E}_{\theta, x < x^*}[\langle \mathbf{w}, \mathbf{x}^* \rangle] \quad (1)$$

Eq. (1) thus measures the expected utility of $\mathbf{x}$, distinguishing the case where $\mathbf{x}$ actually improves on $\mathbf{x}^*$ and the case where $\mathbf{x}^*$ remains the best solution. This criterion can be understood by reference to active learning and the so-called splitting index criterion [8]. In active ranking, any instance $\mathbf{x}$ likewise splits the version space into two subspaces: the challenger subspace of hypotheses ranking $\mathbf{x}$ higher than the current best instance $\mathbf{x}^*$, and its complementary subspace. In the Bayesian setting, considering an interactive optimization goal, the action selection criterion to be maximized is set to the expected utility of $\mathbf{x}$ on the challenger subspace, plus the expected utility of $\mathbf{x}^*$ on the complementary subspace.

III. MAY OVERVIEW

The MAY algorithm elaborates on the active preference reinforcement learning (APRIL) setting presented in [3]. After briefly describing the latter for the sake of self-containedness, this section focuses on the handling of the expert preference noise. A noise model is proposed, supporting the EUS-optimal selection of the next policy to be demonstrated to the expert.

A. APRIL

Let us briefly remind that APRIL considers two search spaces. The first one noted $\mathcal{X}$, referred to as input space or parametric space, is meant to describe and generate policies. In the following $\mathcal{X} = \mathbb{R}^d$; policy $\pi_{\mathbf{x}}$ is represented as a vector $\mathbf{x}$ (e.g. the weight vector of a neural net or the parameters of a control pattern generator (CPG) [13], mapping the current sensor values onto the actuator values). The subscript $\mathbf{x}$ will be omitted for readability when clear from the context.

Another space noted $\Phi(\mathcal{X})$ and referred to as feature space or behavioral space, is meant to describe a *policy behavior* or trajectory. The rationale for using both an input and a feature space is that the expert's preferences depend on the policy behavior only; they are independent from the policy parametric description, conditionally to the policy behavior. Indeed, the policy behavior depends on both the policy parametric description and on the robot environment; but the policy behavior actually depends in an arbitrarily non-smooth way on the parametric policy representation.

Formally, a robot trajectory generated from policy $\pi_{\mathbf{x}}$ and given as a state-action sequence $(s_0, \pi_{\mathbf{x}}(s_0), s_1 \ldots \pi_{\mathbf{x}}(s_{H-1}), s_H)$ is represented as a vector $\mathbf{u} \in \mathbb{R}^D$, with

$$\mathbf{u} = \sum_{h=0}^{h=H} \gamma^h \phi(s_h), \quad 0 < \gamma \leq 1$$

and ϕ a feature function mapping the state space onto $\mathbb{R}^D$.

The utility space $\mathbf{W}$ is set to the unit sphere of $\mathbb{R}^D$. Given a utility function $\mathbf{w} \in \mathbf{W}$, the utility of a behavior $\mathbf{u}$ is measured from the scalar product $\langle \mathbf{w}, \mathbf{u} \rangle$.

APRIL thus defines an iterative 3-step process: at time t i) the selected policy π is demonstrated to the expert,

generating a behavior $\mathbf{u}$; ii) the expert assesses this new behavior comparatively to (her memory of) the previous best behavior $\mathbf{u}_t$; iii) depending on the comparison, the associated ranking constraint (e.g. $\mathbf{u} \prec \mathbf{u}_t$) is added to the APRIL archive $\mathcal{U}_t$, a new policy return model is estimated and a new policy is selected (see section III-C).

As discussed in [3], the use of both the parametric and the behavioral or feature space aims at addressing the expressiveness/tractability dilemma. On the one hand, a high dimensional continuous search space is required to express competent policies. But such a high-dimensional search space makes it difficult to learn a preference-based policy return from a moderate number of preferences. On the other hand, the behavioral space does enable to learn a preference-based policy return from the little evidence provided by the expert, despite the fact the behavioral description is insufficient to describe a flexible policy.

B. Preference and noise model

In the APRIL framework, the expert provides a binary feedback and the feedback noise is not considered. As could have been expected however, the experimentation of APRIL in a robotic setting shows that the preference noise cannot be discarded, and must be carefully taken into consideration.

Letting $\mathbf{w}^*$ denote the true (hidden) utility of the expert, her feedback is modelled as a noisy perturbation of the difference $\langle \mathbf{w}^*, (\mathbf{u} - \mathbf{u}_t) \rangle$ between the utility of the previous best and current trajectories. This perturbation is usually modelled in the literature as a Gaussian one ([7, 21]) or following the Luce-Shepard rule [20]. In both cases however, the noise model involves an extra-parameter (respectively the standard deviation of the Gaussian perturbation or the temperature of the Luce-Shepard rule) controlling the magnitude of the noise.

A simpler model is proposed in MAY, involving a scale parameter δ. The merit of this noise model is to allow for a full Bayesian treatment of the expert's preferences, supporting the integration of the preference noise over the scale parameter δ and thereby avoiding the burden of adjusting it. Formally, the probability of the expert preferring $\mathbf{u}$ over $\mathbf{u}_t$ given the threshold $\delta \in \mathbb{R}, \delta > 0$ and the true preference $\mathbf{z} = \langle \mathbf{w}^*, (\mathbf{u} - \mathbf{u}_t) \rangle$ is defined as:

$$P(\mathbf{u} \succ \mathbf{u}_t | \mathbf{w}^*, \delta) = \begin{cases} 1 & \text{if } \mathbf{z} > \delta \\ 0 & \text{if } \mathbf{z} < -\delta \\ \frac{1}{2\delta}\mathbf{z} + \frac{1}{2} & \text{else} \end{cases}$$

The above model can be viewed as a ridged version of the Gaussian noise model[1]. As mentioned, the ridged version allows one to handle the uncertainty on the threshold δ through integration over δ. A first option is to consider that the expert consistently answers according to a hidden but fixed δ^*. The second option assumes that δ can vary over time. Indeed the first option is less robust as an otherwise consistent ranking can be misidentified due to a single large mistake. Inversely, the second option being less restrictive might slow down the identification of the expert's utility function; however it is clearly more appropriate in the cases where the task (or the expert's understanding thereof, or her preferences) might change over time. In the remainder of the paper, a distinct noise scale parameter δ_i is considered to model the expert preference noise upon seeing the i-th demonstration with $0 \leq \delta_i \leq M$ and M a hyper-parameter of the algorithm.

Additionally, the experimentations suggest that in some cases the current demonstration $\mathbf{u}$ is neither better nor worse than the previous best demonstration $\mathbf{u}_t$; rather, both demonstrations are equally inappropriate. A ternary preference model is thus considered, where the expert is allowed to express that $\mathbf{u}$ is better (resp. worse) than $\mathbf{u}_t$, or that both are comparable. In the latter case, the two ranking constraints $\mathbf{u} \succ \mathbf{u}_t$ and $\mathbf{u}_t \succ \mathbf{u}$ are added to $\mathcal{U}_t$, thereby focussing the search on utilities which are orthogonal to $\mathbf{u} - \mathbf{u}_t$.

C. Active policy selection

Let $\mathcal{U}_t = \{\mathbf{u}_0, \ldots \mathbf{u}_t; (\mathbf{u}_{i_1} \succ \mathbf{u}_{i_2}), i = 1 \ldots k\}$ denote the archive of all demonstrations seen by the expert up the t-th iteration, and the ranking constraints defined from the expert preferences ($k \leq 2t$). With no loss of generality, the best demonstration in $\mathcal{U}_t$ is noted $\mathbf{u}_t$.

Let us set a uniform prior of the utility function $\mathbf{w}$ over the unit sphere in $\mathbb{R}^D$, and let us likewise assume that the prior over the noise scale parameter δ_i is uniform on interval $[0, M]$. Given $\mathcal{U}_t$, the posterior distribution of the utility function reads:

$$p(\mathbf{w}; \mathcal{U}_t) \propto \prod_{i=1}^{k} \left(\frac{1}{2} + \frac{\mathbf{z}_i(\mathbf{w})}{2M} \left(1 + \log \frac{M}{|\mathbf{z}_i(\mathbf{w})|} \right) \right)$$

where $\mathbf{z}_i$ is set to $\langle \mathbf{w}, (\mathbf{u}_{i_1} - \mathbf{u}_{i_2}) \rangle$ if it is not greater (resp. lower) than M (resp. $-M$) in which case it takes the value M (resp. $-M$).

The above utility posterior distribution is used to select a promising policy along the line of Expected Utility of Selection (Section II-C):

$$\begin{aligned} \pi_x &= \arg\max_{\pi_x} \quad EUS(\{\mathbf{u}_x, \mathbf{u}_t\}; \mathcal{U}_t) \\ &= \arg\max_{\pi_x} \quad P_R(\mathbf{u}_x \succ \mathbf{u}_t | \mathcal{U}_t) EU(\mathbf{u}_x; \mathcal{U}_t \cup \{\mathbf{u}_x \succ \mathbf{u}_t\}) + \\ &\qquad\qquad P_R(\mathbf{u}_t \succ \mathbf{u}_x | \mathcal{U}_t) EU(\mathbf{u}_t; \mathcal{U}_t \cup \{\mathbf{u}_t \succ \mathbf{u}_x\}) \end{aligned} \tag{2}$$

As shown in [20], the EUS criterion not only supports the selection of the most promising items w.r.t. the current belief; it also increases the expected utility of the posterior distribution.

In the robotic context, the EUS criterion however raises specific difficulties.

The first difficulty is that the policy selection operates in the parametric space $\mathcal{X}$, while the utility function is defined on the behavioral space $\Phi(\mathcal{X})$. This issue is dealt with in [21] by assuming that a pool of trajectories, sampled from the current posterior over the parametric space, is available; one out of the pool trajectory is selected according to the

[1] Where the Gaussian noise model reads:

$$P(\mathbf{u} \succ \mathbf{u}_t | \mathbf{w}^*, \delta) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{\langle \mathbf{w}^*, \mathbf{u} - \mathbf{u}_t \rangle}{\sqrt{2}\delta}\right)$$

considered active selection criterion defined in the behavioral space. In an *in-situ* robotic setting, where a robot trajectory is hardly reproducible (even when freezing the starting point of the robot given the environment noise), this approach however requires every trajectory in the pool to be recorded beforehand to be able to display one of them to the expert.

The second difficulty is related to the parametric-to-behavioral mapping. Ideally, a policy π_x would be associated a unique trajectory $\mathbf{u}_x$; this however requires the trajectory to be long enough comparatively to the mixing time of the MDP. In the case of a convex selection criterion a possibility is to select a representative trajectory (close to the average trajectory associated to a policy); thereby the policy optimization process would optimize a lower bound on the selection criterion, after the Jensen inequality. When none of those conditions are fulfilled, a careful study of the trajectories generated for a given policy and the trajectory distribution is required before selecting the policy to be demonstrated.

IV. Experimental results

This section presents two experimental validations of the MAY approach. The first setting considers the in-situ interaction between a robot and a human expert, where the focus is on the noise on the expert preferences. The second setting considers a grid world, where the focus is on the noise in the transition model.

A. Reaching a target robot

The first problem, inspired from the swarm robotics framework [17], aims at having a swarm robot aligned in a very precise way with another robot to dock to each other and form a multi-robot organism. The simplified setting considered here involves a single e-puck robot equipped with a (52x39, 4 img/s) camera; the target behavior is to reach an immobile robot with its leds turned on. The starting state, from the perspective of the expert and through the e-puck camera, is displayed in Fig. 1. Sixteen states and five macro-actions are manually defined. The macro-actions involve: stay motionless for one time step, moving forward (resp. backward) for 3 time steps, and rotate to the left (resp. to the right) for one time step. The mapping from the robot camera image onto the state space $\{1, \dots 16\}$ is defined as follows. Given the set of pixels $S_{lum} = \{(x_i, y_i)\}$ associated to the target (the stationary robot) leds, the distance to the target is defined by discretizing $max(y_i)$ in 5 values; the orientation w.r.t. the target is defined by discretizing $\frac{1}{2}(min(x_i) + max(x_i))$ in 3 values. The case where S_{lum} is the empty set (no target in sight) corresponds to the 16th state.

The robot demonstrates the selected policy for 80 time steps; it then gets the expert's feedback and the expert sets the robot back to its initial position. The feedback is interpreted using a built-in procedure: the expert activates the front (resp.) the back e-puck sensors to indicate that the current behavior is better (resp. worse) than the previous best one, and the side sensors are activated to indicate that the current behavior is neither better nor worse than the previous best one.

The parametric policy space is finite, associating one action to each state. The parametric-to-behavioral space is defined as follows. Letting the current trajectory generated from policy $\pi_{\mathbf{x}}$ be noted $s_0, \dots s_{H-1}$, where $s_i \in \{1, \dots, 16\}$ is the robot state at time i, the associated behavioral representation is set to $\Phi(\mathbf{x}) \in \mathbb{R}^{16}$ with $\Phi(\mathbf{x})[j] = \sum_{h=0}^{H-1} \gamma^h \delta_{j,s_h}$ where δ_{j,s_h} is 1 iff the robot is in state j at time h, and 0 otherwise. Parameter γ is set to .95 in the experiment.

The utility of a trajectory is integrated according to the posterior probability distribution $p(w|\mathcal{U}_t)$, estimated using importance sampling with a set of $50,000$ particles drawn uniformly from the sphere in $\mathbb{R}^{16}$. The policy maximizing the EUS cannot be directly found as the EUS is not linear in $\mathbf{u}$. For this reason, the *Query Iteration* defined by [20] was used, combined with an RL step. Firstly, 50 candidate utilities $\mathbf{w}_i$ are drawn according to $p(w|\mathcal{U}_t)$; for each such $\mathbf{w}_i$, an optimal policy π_i is computed using policy iteration and the predefined transition model p. The average trajectory $\mathbf{u}_{xi}$ associated to π_i is generated and $\text{EUS}(\mathbf{u}_{xi}; \mathcal{U}_t)$ is computed. A new utility function

$$\mathbf{w} = \int_W p(\mathbf{w}|\mathcal{U}_t) \mathbb{1}_{\{w \cdot \mathbf{u}_{x\,i} > w \cdot \mathbf{u}_t\}} \mathrm{d}\mathbf{w}$$

is defined and the process is iterated while the EUS increases.

Note that an alternative would be to use *Direct Policy Search* (DPS, e.g. [10]) to optimize the EUS. Further work is concerned with comparatively assessing *Query Iteration* and DPS.

Fig. 1 displays the performance of MAY as follows. Figure 1.c shows the visiting frequency to the goal state (averaged out of 5 runs) *vs* the number of expert's feedback, a.k.a. number of interactions. Interestingly, all 5 policies were found to reach the goal state after 5 interactions; the observed performance decrease in step 6 is explained by inspecting the logs, showing that the expert made an error in one of the runs, favoring a trajectory that spent significantly less time in the goal state. Figs. 1.d, 1.e, 1.f show the utility weight of each coordinate *vs* the number of interactions for three independent runs. The red curve depicts the weight associated to the goal state. Interestingly, the weight associated to some states increases after the first interactions, and thereafter decreases, due to the fact that these states are intermediate between the starting and the goal states. Fig. 1.d displays the run where the expert made a mistake (at the third feedback), causing the weight associated to the "nothing in sight" state to increase and out-pass the other weights; this in turn leads astray the optimal policy; however MAY recovers one iteration later.

Overall, this task requires a relatively low number of interactions with the expert to be successfully achieved (circa 6 expert's feedbacks; in APRIL [3] circa 10-15 expert's feedbacks were needed to learn e.g. the mountain car policy). The docking task is indeed easy as safe trajectories, not visiting the "nothing in sight" state, can be found from the initial to the target states. Further experiments will consider more difficult target states.

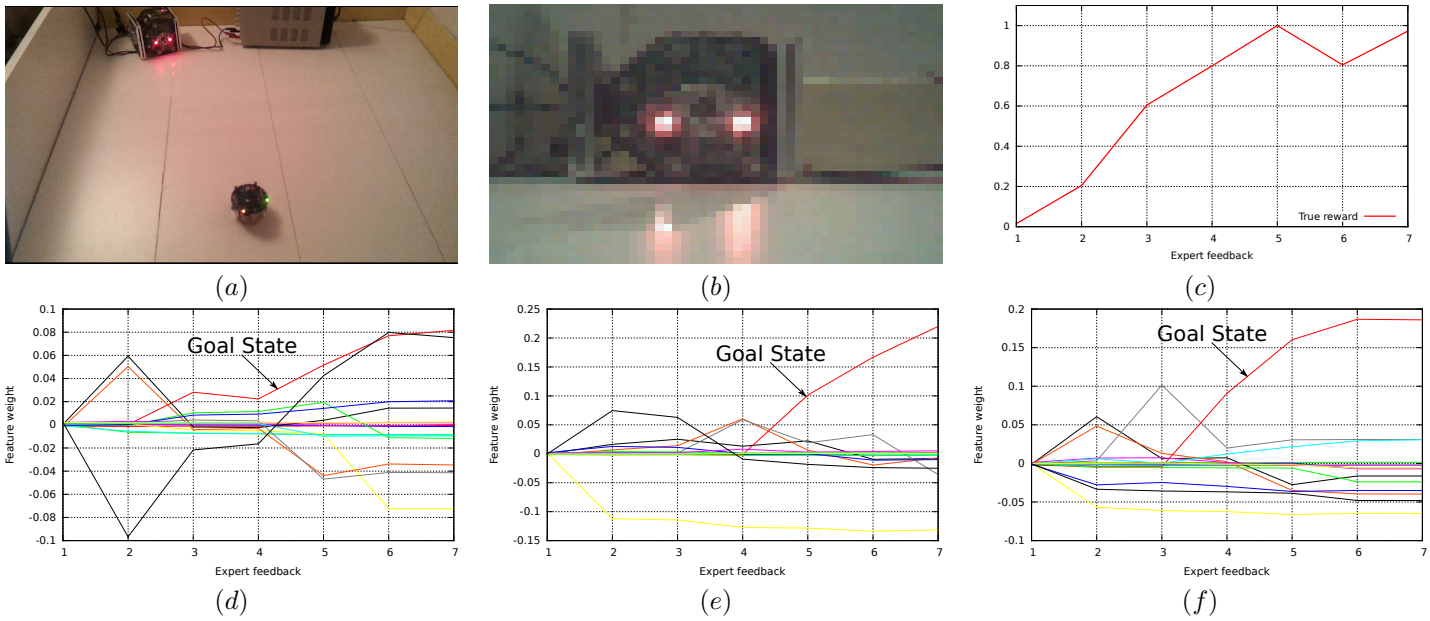

Fig. 1. a) Initial state from an observer's perspective. b) Initial state from the e-puck camera perspective. c) Performance of the policy maximizing the average utility *vs* number of interactions with the expert, averaged out of 5 runs. d, e, f) Utility weight vector in $\mathbb{R}^{16}$ *vs* number of interactions for three different runs. For each run, 16 curves, representing the weight of the 16 states, are displayed

B. Reaching a target state in a grid world

The second problem (Fig. 2) is concerned with reaching a goal state in a grid world, in simulation, where the state space involves 25 states and the action space involves 5 actions (up, down, left, right or stay motionless). In this latter experiment, much less time-consuming than the former one, a longer time horizon ($H = 300$ instead of $H = 80$) is considered. Additionally, the impact of a stochastic transition model is examined: the robot stays motionless with probability 1/2 if it can execute the selected action up, down, left or right (respectively with probability 1 if the selected action would send it through the wall). This setting aims at investigating in depth the exploration of the search space achieved by MAY.

The expert's feedback is emulated, preferring the trajectory reaching at the earliest the target state, and otherwise, the trajectory reaching the state nearest to the target state along the 300 time steps.

As shown by Fig. 2.b, displaying the visiting frequency of the goal state, averaged out of 41 independent runs, vs the number of emulated expert's feedbacks, this problem is more difficult than the former one. Formally, after 60 feedbacks 75% of the runs only reach the target behavior. This shortcoming is blamed on the limited size of the particle set and its steep weight decay. Indeed, a better approximation of the EUS criterion requires the particles to be re-sampled (e.g. using MCMC), all the more so as the size of the state space increases. Figs 2.d, (respectively e. and f) display the frequency of visits to the states during the 1-15 (resp. 15-30 and 31-60) iterations of MAY. As could have been expected, Fig. 2.d shows that initial policies mostly visit the regions close to the starting state (iterations 1-15). Fig. 2.e shows that policies explore farther regions at a later stage (iterations 16-30) and correctly visit more the most interesting, upper rightmost region. Later on however, it seems that the MAY strategy is overly biased toward the exploration of the grid world (Fig. 2.f). A tentative interpretation blames the over-exploration on the inappropriate cumulative preference; trajectories *staying longer* in the goal state should be preferred.

V. DISCUSSION AND PERSPECTIVES

The main contribution of the present paper is to show the feasibility of training a robot on-board, online, using an active preference learning framework. The former APRIL framework assumes that the expert's preferences are noiseless; as could have been expected, this assumption hardly holds in *in-situ* experiments.

An extended Bayesian setting incorporating an integrable noise model, is therefore proposed and investigated, defining the MAY system. Several lessons are learned from the preliminary experiments done with MAY in-situ and in simulation.

A first lesson is that experts (the authors) do make mistakes, visibly leading astray the found policies; but MAY is shown to recover from such errors in an easy, in-situ setting. A second lesson, equally unsurprising, is that fine-grained information should be leveraged by the expert, when comparatively assessing policies: it was known that the expert must be able to make differences between bad and very bad behaviors, to prevent the system from facing a "Needle in the Haystack" problem (everything is bad, except the exact target behavior). Symmetrically, it is suggested that the expert must be able to make differences between good and very good behaviors, between visiting the target state and staying in it, to limit the over-exploration phenomenon.

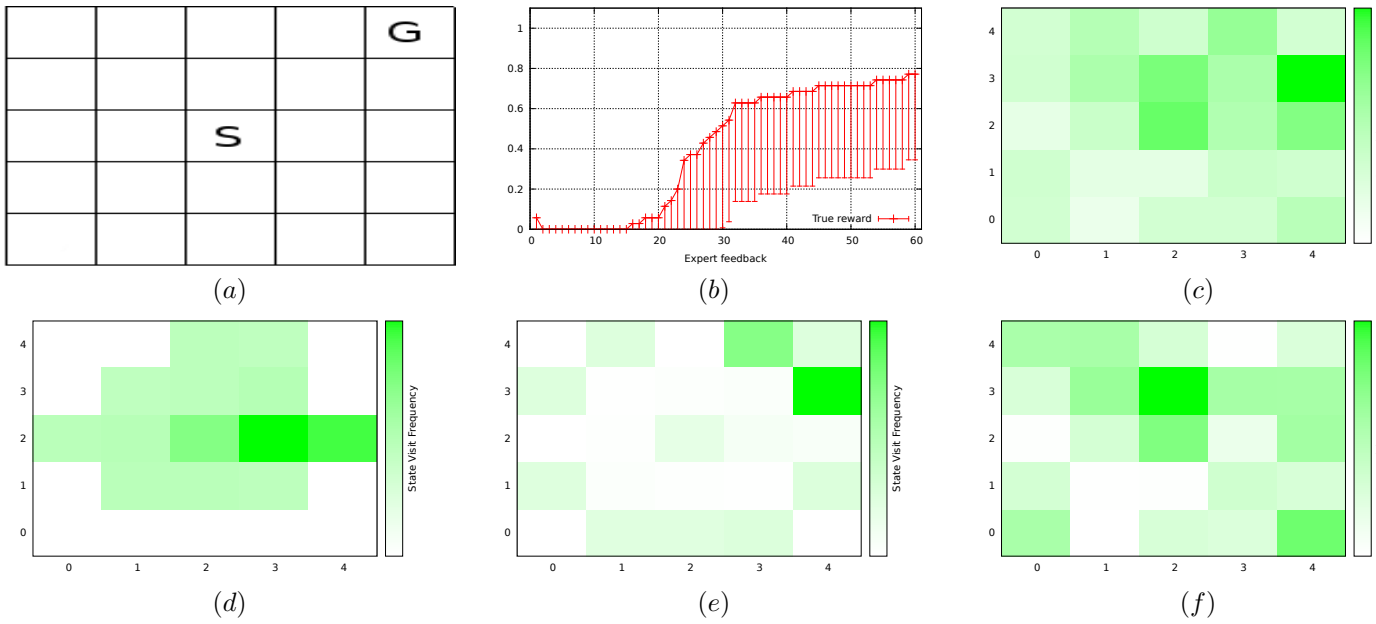

Fig. 2. a) The grid world: the agent initially in the start state S must visit the goal state G. b) Performance of the policy maximizing the average utility against the number of expert's feedback. c, d, e, f) Frequency of the state visits, averaged over all trajectories (c), during iterations 1-15 (d), during iterations 16-20 (e) and during iterations 31-60 (f).

Further work, inspired from [21], will investigate how to extend MAY in order to learn from short and well focused sub-behaviors. Indeed, the better framed the sub-behaviors, the more informative the expert's preferences will be. A challenging issue is however related to the choice of the starting state, conditioning the behavior framing. We shall tackle the preference learning over (unknown) sub-behaviors as a Multiple Instance ranking problem, considering that a long behavior actually involves a set of sub-behaviors, and that some sub-behaviors among the set, are responsible for the expert's ranking preferences. Incidentally this approach will address the dependency of MAY onto the starting state.

Acknowledgments. The first author is funded by FP7 European Project *Symbrion*, FET IP 216342, http://symbrion.eu/.

REFERENCES

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

[2] Riad Akrour, Marc Schoenauer, and Michèle Sebag. Preference-based policy learning. In *ECML/PKDD (1)*, pages 12–27. Springer Verlag, 2011.

[3] Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *ECML/PKDD (2)*, pages 116–131. Springer Verlag, 2012.

[4] E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *NIPS 20*, pages 409–416, 2008.

[5] 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.

[6] Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *ECML/PKDD (1)*, pages 312–327. Springer Verlag, 2011.

[7] Wei Chu and Zoubin Ghahramani. Preference learning with Gaussian processes. In *ICML*, pages 137–144, 2005.

[8] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS*, 2005.

[9] Christos Dimitrakakis and Michail G. Lagoudakis. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, 2008.

[10] Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML*, page 51, 2009.

[11] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. In *NIPS*, pages 1016–1024, 2012.

[12] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *NIPS*, pages 1162–1170, 2010.

[13] Chengju Liu, Qijun Chen, and Danwei Wang. Locomotion control of quadruped robots based on cpg-inspired workspace trajectory generation. In *Proc. ICRA*, pages 1250–1255. IEEE, 2011.

[14] A.Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In P. Langley, editor, *ICML*, pages 663–670. Morgan Kaufmann, 2000.

[15] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

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

[17] Symbrion. FP7 European Project FET IP 216342, June 2009. URL http://symbrion.eu/.

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

[19] Paolo Viappiani. Monte-Carlo methods for preference learning. In *Proc. Learning and Intelligent OptimizatioN, LION 6*. LNCS, Springer Verlag, 2012.

[20] Paolo Viappiani and Craig Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*, pages 2352–2360, 2010.

[21] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A Bayesian approach for policy learning from trajectory preference queries. In *NIPS*, pages 1142–1150, 2012.