Interactive Learning in Continuous Multimodal Space: A Bayesian Approach to Action-Based Soft Partitioning and Learning

Hadi Firouzi, Majid Nili Ahmadabadi, Babak Nadjar Araabi, *Member, IEEE*, Saeed Amizadeh, Maryam S. Mirian, and Roland Siegwart, *Fellow, IEEE*

***Abstract*—A probabilistic framework for interactive learning in continuous and multimodal perceptual spaces is proposed. In this framework, the agent learns the task along with adaptive partitioning of its multimodal perceptual space. The learning process is formulated in a Bayesian reinforcement learning setting to facilitate the adaptive partitioning. The partitioning is gradually and softly done using Gaussian distributions. The parameters of distributions are adapted based on the agent's estimate of its actions' expected values. The probabilistic nature of the method results in experience generalization in addition to robustness against uncertainty and noise. To benefit from experience generalization diversity in different perceptual subspaces, the learning is performed in multiple perceptual subspaces—including the original space-in parallel. In every learning step, the policies learned in the subspaces are fused to select the final action. This concurrent learning in multiple spaces and the decision fusion result in faster learning, possibility of adding and/or removing sensors—i.e., gradual expansion or contraction of the perceptual space-, and appropriate robustness against probable failure of or ambiguity in the data of sensors. Results of two sets of simulations in addition to some experiments are reported to demonstrate the key properties of the framework.**

***Index Terms*—Adaptive partitioning, decision fusion, multimodal perception, reinforcement-based Bayesian learning, subspace learning.**

I. Introduction

REINFORCEMENT LEARNING (RL) methods are widely used in different disciplines because of being interactive and unsupervised. Nevertheless, complications in dealing with continuous, multimodal, and complex perceptual spaces in addition to having a relatively slow learning speed have hindered RL methods from being employed in situated systems. Different discretization and function approximation methods, in addition to mixtures of learning, modeling, and planning frameworks—see Dyna-Q [1] as an example—have been proposed so far to partially solve the mentioned problems. However, more effective solutions are yet needed to sufficiently increase the appropriateness of the RL methods for real world applications.

Uniform discretization of the perceptual space with sufficiently small quantization levels hypothetically results in good learning performance in low-dimensional and compact perceptual spaces. Nevertheless, fine quantization causes the curse of dimensionality in large and multimodal spaces.

Function approximators on the other hand are used to reduce the need for fine discretization. This lessens the curse of dimensionality and results in generalization and faster learning. Widely employed function approximators are artificial neural networks, fuzzy systems, linear, and radial basis functions (RBF), etc.; see [2]–[8]. However, the quality and speed of learning highly depend on the suitability of the selected function approximator, its initial settings, and the appropriateness of the state representation method for the problem at hand.

To deal with the mentioned problems, a learning approach capable of adaptive partitioning of continuous perceptual spaces is required. The partitioning is task-based and should be inline with the expected reward maximization. Moreover, the learned policy should be close enough to the optimal policy all over the perceptual space. The ability of generalization [9] and robustness to noise and uncertainty are other important requirements.

From the cognitive psychological perspective, a metaknowledge can be used to classify or partition things into categories [10]. Accordingly, task-based partitioning of the perceptual space is a categorization process based on some metaknowledge about the task. On the other hand, some biological researches show that mirror neurons [11]–[13] classify the perceptual space based on the affordable actions [14], [15]. It seems that this categorization is the result of learning and is done gradually. We adopt this finding and design a perceptual space partitioning method accordingly.

In this paper, we propose a framework in which partitioning continuous and multimodal perceptual spaces is performed automatically while the task learning is done interactively. The learning process is formulated in a Bayesian RL setting. This

Manuscript received December 14, 2010; revised May 20, 2011 and July 30, 2011; accepted September 16, 2011. Date of publication October 03, 2011; date of current version June 08, 2012. This work was supported in part by University of Tehran, as well as Iran Telecommunication Research Center (ITRC). This research has also been realized in close collaboration with the BACS project supported by EC-contract number FP6-IST-02'140, Action line: Cognitive Systems.

H. Firouzi, S. Amizadeh, and M. S. Mirian are with the Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran (e-mail: h.firouzi@ece.ut.ac.ir; amizadeh@ut.ac.ir; mmirian@ut.ac.ir).

M. N. Ahmadabadi and B. N. Araabi are with the Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran. They are also with the School of Cognitive Sciences, IPM, Tehran, Iran (e-mail: mnili@ut.ac.ir; araabi@ut.ac.ir).

R. Siegwart is with the Autonomous Systems Lab. ETHZ, Zurich, Switzerland (e-mail: rsiegwart@ethz.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAMD.2011.2170213

1943-0604/$26.00 © 2011 IEEE

facilitates soft and adaptive partitioning of the perceptual space in addition to experience generalization. The partitioning technique is action-based and is done using a mixture of Gaussian distribution functions. By action-based we mean that, a novel partition is generated only if a considerable change is observed in the policy. The Gaussian functions are matched with the Bayesian formulation and benefit the framework in terms of generalization, gradual adaptability, and further mathematical analysis.

Another major aspect of the framework is supporting concurrent learning in multiple perceptual subspaces and the fusion of policies learned in those subspaces. This property results in faster learning; as the learning speed and generalization capability of our Bayesian RL varies from one subspace to another. By the concurrent learning and the policy fusion we exploit this diversity.

The organization of the paper is as follows. Related researches are reviewed in the next section. In Section III, the problem under consideration is described in more details. In addition, the basic idea of the proposed approach is discussed and formulated. The learning algorithm is introduced in Section IV. The simulation and the experimental results are reported in Section V. Finally, some discussions and conclusions are given in Section VI.

II. Related Works

The first set of related works mainly includes state abstraction and perceptual space partitioning methods. In these methods the space is partitioned and abstracted manually or by using supervised methods, e.g., [2]–[4], [7], and [16]–[23]. These methods either result in the curse of dimensionality in large spaces, require prior knowledge or need training data. To partially resolve these problems, adaptive discretization methods, such as U-Tree and its extensions [24], [25], variable resolution methods [26], and the VQQL algorithm [27], [28], are proposed. Such mapping methods are not directly inline with the expected reward maximization, they do not consider the natural uncertainty and noise existing in observations of the real world, or their effectiveness highly depends on the designer's knowledge. In addition, revising the partitions adaptively is a very difficult task in most of these methods.

Incorporating self-adaptability, Smith [29] has proposed to abstract and quantize the continuous perceptual and action spaces using two self organizing maps (SOMs) and then relates them via a Q-table. Mobahi *et al.* [19], inspired by the mirror neurons, have reported a single step imitative learning approach for the phoneme acquisition problem. They used deterministic and crisp mapping functions to partition the perceptual space.

The topics of the second set of related researches are mainly on learning in multiple perceptual spaces and fusing the learned policies. The key idea here is to scale ordinary RL methods for learning tasks with complex and high-dimensional state spaces.

In [30] a framework named Cooperative-Competitive-Concurrent Learning with Importance Sampling (CLIS) has been proposed. In this framework, the agent has multiple learning modules with different numbers of parameters and learning algorithms. Learning in the modules is done concurrently. In the CLIS framework, some heterogeneous reinforcement learning modules share the same sensory-motor space to obtain a good performance. However, in our method, the agent policy is concurrently learned in multiple heterogeneous sensory spaces. As a result, the learning performance and speed is improved. Also in our framework additional sensory spaces can be added or detached on-the-fly.

In this paper, we model the existing uncertainty in the agent's perceptual space by using a Bayesian framework. It helps the agent to convert unmanageable incompleteness into a manageable uncertainty; see [31]–[36] for the concept and some examples. Additionally, unlike [19], we consider multi-step tasks Furthermore, the proposed framework provides the capacity for concurrent learning in multiple and multimodal perceptual spaces; see [37] for the benefits.

III. Interactive Learning Via Action-Based Soft Partitioning in Multiple Spaces

Consider a robot equipped with S sensors and assume that the world can be modeled by a Markov Decision Process (MDP) through the union of those sensors. The perceptual space of the robot is the Cartesian product of the sensory spaces, i.e., $X^1 \times X^2 \times \ldots \times X^S$ where X^i is the i^{th} sensory space. The sensory spaces are continuous and include uncertainty. It is assumed that the robot has A actions. The robot should learn the best action in response to each of its perceptual points. Learning is based on the feedback the robot receives from the environment in terms of the reinforcement signal r in response to its actions. Therefore, an RL setting has been employed. The first step in learning is partitioning the perceptual space in a way that uncertainty in the sensory information is considered, generalization of experiences is possible, and the curse of dimensionality is reduced.

From the multiagent point of view, the agent's (i.e., robot's) mind can be composed of tiny agents, where each of them observes the environment through a specific perceptual space and learns to suggest a suboptimal action based on its partial observation. In each learning step the suggested actions are fused using an expertness criterion. Thus, the main agent's mind structure can be considered as a cooperative multiagent system. A schematic overview of the proposed framework is illustrated in Fig. 1. In the following subsections both the probabilistic modeling of the tiny agents and the expertise measure will be explained in detail.

A. Modeling

In the probabilistic formalism, the suitability of performing the ith action $\mathrm{a_i}$ by the sth agent $\mathrm{T_s}$ in perceptual space X^s can be encoded as the probability distribution $P(a_i|X^s)$. Therefore, our action selection problem by tiny agent T^s is reduced to computing the posterior probabilities $P(a_i|X^s)$. On the other hand, by applying the Bayes rule, we have

$$P(a_i|X^s) = \eta.P(X^s|a_i)P(a_i) \tag{1}$$

where $P(X^s|a_i)$ and $P(a_i)$ are the likelihood of action a_i and the prior probability of action a_i, respectively. Also η is a normalization factor. Based on (1), for estimating distribution $P(a_i|X^s)$, distributions $P(X^s|a_i)$ and $P(a_i)$ should be

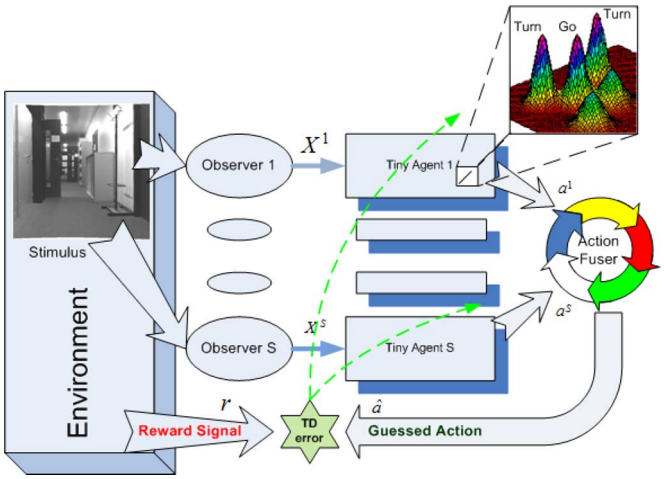

Fig. 1. Approximate scheme of the proposed framework for learning via action-based soft partitioning the multiple perceptual spaces.

estimated as well, which is the main focus of the next section. However, before estimating these distributions, we should first specify a parametric form for the likelihood $P(X^s|a_i)$.

An agent's actions may be associated with different components (soft partitions) in the perceptual space. Thus, for each action a_i, there may be more than one region in the state space where the likelihood $P(X^s|a_i)$ is high (As an example in Fig. 1 the "Turn" action has two apparent representatives in the first perceptual space). We call these regions the modes or the modals of the probability distribution $P(X^s|a_i)$. As a result, $P(X^s|a_i)$ must be modeled as a multimodal distribution to capture the scattered nature of the optimal action in the perceptual space. To do so, the mixture densities model [38] is used. As a result, the likelihood of the action a_i is decomposed as

$$P(X^s|a_i) = \sum_{j=1}^{q^s} P\left(X^s|M_j^s\right) P\left(M_j^s|a_i\right) \tag{2}$$

where the M_j^s are the components of the mixture model of the tiny agent T^s. Distribution $P(X^s|M_j^s)$ conveys the probability of observing X^s when the observation is generated by the component M_j^s and $P(M_j^s|a_i)$ is the contribution weight of the component M_j^s for the action a_j. For the sake of simplicity, all the components of different actions are unified in the set M^s whose cardinality is q^s (We simply set $P(M_j^s|a_i)$ to zero if the component M_j^s does not belong to the action a_j.) By substituting (2) into (1), we get

$$P(a_i|X^s) = \frac{P(a_i)\sum_{j=1}^{q^s} P\left(X^s|M_j^s\right) P\left(M_j^s|a_i\right)}{\sum_{k=1}^{A}\left\{P(a_k)\sum_{j=1}^{q^s} P\left(X^s|M_j^s\right) P\left(M_j^s|a_k\right)\right\}}. \tag{3}$$

According to (3), we can conclude that to compute $P(a_i|X^s)$, we should first estimate the probability distributions $P(a_i)$, $P(M_j^s|a_i)$ and $P(X^s(t)|M_j^s)$. In the next subsection, the method used to model these distributions is described.

B. The Parametric Forms

In this section, the parametric forms used for each of the three mentioned probability distributions are explained in details. Here the Bayesian approach is adopted to estimate the parametric forms.

1) Modeling of $P(M_j^s|a_i)$: This distribution can be parameterized as

$$P\left(M = M_j^s|a = a_i, f_{ij}^s\right) = P\left(M_j^s|a_i, f_{ij}^s\right) = f_{ij}^s \tag{4}$$

where f_{ij}^s $(i \in [1, A], j \in [1, q^s])$ encodes the normalized belief of component M_j^s belonging to action a_i. To estimate this probability distribution, we define matrix $B^s = [b_{ij}^s \in \Re]_{A\times q^s}$ where b_{ij}^s is the nonnormalized version of f_{ij}^s. Subsequently, we need a normalization function to map b_{ij}^s to f_{ij}^s. This normalization function is defined as follows:

$$f_{ij}^s = \Phi\left(b_{ij}^s\right); \quad \sum_j f_{ij}^s = 1; \Phi : \Re \to [0, 1] \tag{5}$$

where $\Phi(\cdot)$ is the normalization function. There are many functions—like the Boltzmann function—to map real values into probabilities; however, choosing a good one which normalizes truthfully is somewhat tricky. In the proposed framework, a heuristic normalization factor is employed to normalize belief values. This normalization factor is defined as follows:

$$\hat{b}^s = \frac{\left|\min\left(\left\{b_{ij}^s\right\}\right)\right|}{1 + \left|\min\left(\left\{b_{ij}^s\right\}\right)\right|^{-\min\left(\left\{b_{ij}^s\right\}\right)}} \quad i \in [1, A], j \in [1, q^s]. \tag{6}$$

As a result, the normalized belief of the component M_j^s belonging to the action a_i is calculated based on the following equation

$$P\left(M_j^s|a_i\right) = \left(b_{ij}^s + \hat{b}^s\right) \Big/ \sum_{k=1}^{q^s}\left(b_{ik}^s + \hat{b}^s\right). \tag{7}$$

2) Modeling of $P(a_i)$: From (7), b_{ij}^s can be interpreted as the ratio of observed stimuli which simultaneously belongs to the action a_i and is absorbed by the component M_j^s. Therefore, the probability $P(a_i)$ can be calculated directly from matrix B

$$P(a_i) = \sum_{j=1}^{q^s}\left(b_{ij}^s + \hat{b}^s\right) \Big/ \sum_{k=1}^{A}\sum_{j=1}^{q^s}\left(b_{kj}^s + \hat{b}^s\right). \tag{8}$$

3) Modeling of $P(X^s|M_j^s)$: Due to the fact that distribution $P(X^s|M_j^s)$ measures the proximity of stimulus X to the center of the component M_j^s in a nonlinear fashion, a symmetric uni-modal distribution is suitable to model it. Among different symmetric uni-modal distributions, the normal distribution is selected. Such a distribution can be used to describe, approximately, any variable that tends to cluster around the mean. This characteristic of Gaussian distributions facilitates generalization of observed states into un-observed ones.

To employ a normal distribution, its mean vector and its covariance matrix need to be known in advance, but this is not possible in our problem domain. Thus, we should define parametric

distributions for these unknown parameters to encode our belief about them. As a result, the covariance matrix and the mean vector are set to take the *Wishart* distribution and the normal distribution conditional on the covariance matrix, respectively [38]. Using these distributions, it can be proved that $P(X^s|M_j^s)$ takes a *multivariate* t distribution with parameters α_j^s, β_j^s, μ_j^s and v_j^s (interested readers can refer to [38] for the proof)

$$P\left(X^s|M_j^s\right)=t\left(X^s;\alpha_j^s-n^s+1,\mu_j^s,\frac{v_j^s\left(\alpha_j^s-n^s+1\right)}{v_j^s+1}\beta_j^{s-1}\right). \tag{9}$$

To explain each of the four parameters of distribution $P(X^s|M_j^s)$, we first define S_j^s to be the set of the perceptual vectors based on which the distribution $P(X^s|M_j^s)$ is estimated. Then the parameters are defined as follows: v_j^s is the cardinality of S_j^s, μ_j^s and β_j^s are the empirical mean vector and the nonnormalized empirical covariance matrix of the S_j^s's members, respectively. Normally, α_j^s is set to v_j^s-1.

Now we derive parametric forms for $P(a_i)$, $P(M_j^s|a_i)$ and $P(X^s|M_j^s)$. Parameters are the matrixes $B^s=[b_{ij}^s]_{A\times q^s}$ and $D^s=[\alpha_j^s\mu_j^s\beta_j^s\mathrm{v_j^s}]_{q^s\times 4}$, which should be learned to compute the distribution $P(a_i|X^s(t))$. Moreover, the number of the components (q^s) used to cluster the s^{th} perceptual space should also be determined gradually from observations. In Section IV, a reinforcement-based algorithm is proposed for learning the presented model interactively.

IV. The Learning Algorithm

The general scenario in each learning step is that, the tiny agent T^s perceives perceptual vector X^s from the environment and based on its current estimation of $P(a_i|X^s)$, it stochastically finds the most promising action a_g^s (g stands for guessed) to which the vector X^s belongs, See Fig. 2. These guessed actions are fused by a fusion method. Then, by performing the fused action $a_g(t)$, the tiny agent T^s receives the reinforcement signal r. The learning algorithm walks through different steps in accordance with the reinforcement signal, which is discussed in the following subsections. Fig. 2 shows a schematic block diagram of the learning algorithm.

A. Initialization

In this step, the initial values for parameters q^s, $B^s=[b_{ij}^s]_{A\times q^s}$ and $D^s=[\alpha_j^s\mu_j^s\beta_j^s\mathrm{v_j^s}]_{q^s\times 4}$ are set

$$\begin{aligned}&A\leftarrow(\text{The number of actions})\\&q^s\leftarrow 0\\&B^s\leftarrow Rand_{r\times q}\\&\forall j\in[1\dots q^s]:\\&\quad\alpha_j^s\leftarrow -1,\quad \mu_j^s\leftarrow[0\,0\,0\,\dots\,0]_{1\times \mathrm{n^s}}^T\\&\quad v_j^s\leftarrow 0,\quad \beta_j^s\leftarrow \text{InitialVariance}\times I_{n^s}\end{aligned}$$

where *InitialVariance* is a scalar representing the initial variance of the components, n^s is the s^{th} perceptual space dimension, *Rand* is a random matrix, and I is the identity matrix. Note that the number of actions is equal to the size of the agent's action set, which is known in advance.

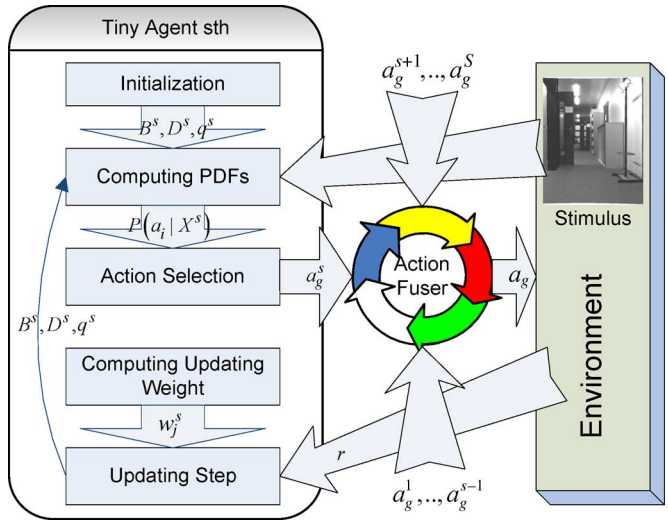

Fig. 2. Block diagram of the proposed framework—different parts and their relationships.

B. Estimating Pdfs

In addition to $P(a_i)$, $P(M_j^s|a_i)$ and $P(X^s|M_j^s)$, some other probability distributions, i.e., $P(M_j^s|X^s)$, $P(M_j^s)$, $P(M_j^s|a_i,X^s)$, and $P(M_j^s|\bar{a}_i,X^s)$ need to be estimated in advance. In the following subsections these distributions are considered in detail.

1) Estimating Distribution $P(M_j^s|X^s)$: This distribution encodes the probability of component M_j^s belonging to perceptual vector X^s. By applying Bayes rule we have

$$\begin{aligned}P\left(M_j^s|X^s\right)&=\frac{P\left(X^s|M_j^s\right)P\left(M_j^s\right)}{P(X^s)}\\&=\frac{P\left(X^s|M_j^s\right)P\left(M_j^s\right)}{\sum_{k=1}^{q^s}P\left(X^s|M_k^s\right)P\left(M_k^s\right)}\\&=\sigma P\left(X^s|M_j^s\right)P\left(M_j^s\right)\end{aligned} \tag{10}$$

where σ is a normalization factor.

2) Estimating Distribution $P(M_j^s)$: This function is the probability distribution of component M_j^s and it can be directly estimated by matrix B.

$$P\left(M_j^s\right)=\left(\sum_{i=1}^{A}\left(b_{ij}^s+\hat{b}^s\right)\right)\Big/\left(\sum_{i=1}^{A}\sum_{k=1}^{q^s}\left(b_{ik}^s+\hat{b}^s\right)\right) \tag{11}$$

3) Estimating Distribution $P(M_j^s|a_i,X^s)$: This distribution is the probability of X^s being absorbed by component M_j^s when action a_i is performed. $P(M_j^s|a_i,X^s)$ is estimated as

$$\begin{aligned}P\left(M_j^s|a_i,X^s\right)&=\frac{P\left(X^s|M_j^s\right)P\left(M_j^s|a_i\right)}{\sum_{k=1}^{q^s}P\left(X^s|M_k^s\right)P\left(M_k^s|a_i\right)}\\&=\sigma P\left(X^s|M_j^s\right)P\left(M_j^s|a_i\right).\end{aligned} \tag{12}$$

4) Estimating Distribution $P(M_j^s|\bar{a}_i, X^s)$: This distribution is the probability of X^s being absorbed by component M_j^s when action a_i is not performed. $P(M_j^s|\bar{a}_i, X^s)$ is computed as

$$\begin{aligned} P\left(M_j^s|\overline{a}_i, X^s\right) &= \frac{P\left(X^s|M_j^s\right)P\left(M_j^s|\overline{a}_i\right)}{\sum_{k=1}^{r} P\left(X^s|M_k^s\right)P\left(M_k^s|\overline{a}_k\right)} \\ &= \frac{P\left(X^s|M_j^s\right)\left[1-P\left(M_j^s|a_i\right)\right]}{\sum_{k=1}^{A} P\left(X^s|M_k^s\right)\left[1-P\left(M_k^s|a_k\right)\right]} \\ &= \sigma P\left(X^s|M_j^s\right)\left[1-P\left(M_j^s|a_i\right)\right]. \end{aligned} \quad (13)$$

C. Action Selection

In the proposed framework, the suitability of an action is defined in terms of the expected reward and is encoded in the distribution $P(a_i|X^s)$. However, the guessed action is made stochastically, based on another distribution (i.e., $P_e(a_i|X^s)$) which is calculated based on distribution $P(a_i|X^s)$. An extra variable T (temperature) is employed in distribution $P_e(a_i|X^s)$ in order to move smoothly from random action selection to greedy action selection by decreasing temperature T during the learning stages. Distribution $P_e(a_i|X^s)$ is defined as

$$P_e(a_i|X^s) = \exp\left(\frac{P(a_i|X^s)}{T}\right) \Big/ \sum_{i=1}^{A} \exp\left(\frac{P(a_i|X^s)}{T}\right). \quad (14)$$

Then the guessed action a_g^s in the s^{th} perceptual space is selected stochastically, based on the following equation

$$a_g^s = \arg soft\max_i \left\{P_e(a_i|X^s)\right\}. \quad (15)$$

D. Computing the Updating Weight

Similar to the Temporal Difference algorithm [16], in the proposed framework a TD-like error is computed in order to calculate the updating weight. This weight is used for updating a specific component. Before computing the TD error let's define the *most absorbent component* for the vector $X^s(t)$ as

$$M_{\text{mac}}^s(t) = \arg\operatorname{Max}_j P\left(M_j^s|a_g, X^s\right). \quad (16)$$

Here, we merely update the most absorbent component M_{mac}^s. More specifically, after fusing the guessed actions and activating the fused action a_g, the next perceptual vector X'^s is observed and both the next greedy guessed action $a_g'^s$ and the next most absorbent component $M_{\text{mac}}'^s$ are found according to the following equations

$$a_g'^s = \arg\operatorname{Max}_i P(a_i|X'^s) \quad (17)$$

$$M_{\text{mac}}'^s = \arg\operatorname{Max}_j P\left(M_j^s|a_g'^s, X'^s\right). \quad (18)$$

Having a_g^s, M_{mac}^s, $C_g'^s$ and $M_{mac}'^s$ the TD-like error is computed as

$$\text{TD}_{\text{err}}^s = \alpha\left(r + \gamma b_{a_g'^s, M_{\text{mac}}'^s}^s - b_{a_g, M_{\text{mac}}^s}^s\right) \quad (19)$$

where α, γ, and r are the learning rate, the discount factor, and the received reward in response to the active action a_g, respectively.

Based on the TD error value, the updating weights are computed by different probability distributions

$$\begin{aligned} &\forall j \in [1 \ldots q^s] \\ &if\left(TD_{\text{err}}^s > thr_{\text{positive\_sample}}\right) \\ &\quad w_j^s = P\left(M_j^s|a_g, X^s\right) \\ &\text{else if}\left(TD_{\text{err}}^s < thr_{\text{negative\_sample}}\right) \\ &\quad w_j^s = P\left(M_j^s|\overline{a}_g, X^s\right) \\ &\text{else} \\ &\quad w_j^s = P\left(M_j^s|X^s\right) \end{aligned} \quad (20)$$

where w_j^s is the updating weight of component M_j^s and $thr_{\text{positive\_sample}}$, $thr_{\text{negative\_sample}}$ are thresholds that indicate the correctness of the guessed action. In fact, as a more suitable action results in a higher reward, when the TD error value is above $thr_{\text{positive\_sample}}$, most probably the guessed action is selected suitably. Accordingly, the performed action should be considered in computing the updating weight and the distribution $P(M_j^s|a_g, X^s)$ is employed to estimate the updating weights. On the other hand, a TD error value which is below $thr_{\text{negative\_sample}}$ implies that the guessed action has been chosen improperly and it should not contribute in estimating the updating weight. In this case the distribution $P(M_j^s|\bar{a}_g, X^s)$ is used to estimate the updating weights. The action $\bar{a}_g$ indicates all other actions except a_g. In other cases, as there is no evidence of properness or improperness of the guessed action, the components' updating weights are computed from the distribution $P(M_j^s|X^s)$.

E. Updating Step

In this step, matrixes B and D are updated. First the updating mechanism of B is considered

$$\begin{aligned} b_{a_g,J}^s &\leftarrow b_{a_g,J}^s + TD_{err}^s \times w_J^s \\ \text{where } J &= \arg\max_j \left\{w_j^s\right\}. \end{aligned} \quad (21)$$

In this step, only $b_{g,J}$ which is related to the maximum weight is updated. It means that instead of updating all $b_{a_g,j}^s; j \in [1, q^s]$ just $b_{a_g,J}^s$ is updated. It is called *hard updating*

At the second substep of updating, matrix $D^s = [\alpha_j^s \mu_j^s \beta_j^s \mathrm{v}_\mathrm{j}^s]_{q^s \times 4}$ is considered. Basically the probability distribution function $P(X^s|M_j^s)$ is estimated from the value of this matrix. Actually, there are two subtasks when a new state X^s is observed; it can be used either for updating the existing components or creating a new one. To distinguish these subtasks, some criteria should be satisfied based on the following equation

$$\begin{aligned} &if\left(\min\left\|X^s - M_j^s\right\|_{j=1}^{q^s} < thr_{\text{new\_component}}\right. \\ &\qquad \left. OR\, TD_{err} > thr_{\text{negative\_sample}}\right) \\ &\qquad \text{Update components} \\ &\text{else} \\ &\qquad \text{Add a new component.} \end{aligned} \quad (22)$$

In (22), two conditions are evaluated to determine the updating or adding of a component. The first criterion is the minimum Euclidean distance between the vector X^s and component M_j^s.

In fact component M_j^s is a prototype vector in the perceptual space

$$\min \left\| X^s - M_j^s \right\|_{j=1}^{q^s} < thr_{\text{new\_component}} \tag{23}$$

where the value of $thr_{\text{new\_component}}$ determines the density of components distributed in the perceptual space. In other words, if this threshold is set to a very large value, most of the observations will be used to update the existing components. Then, the perceptual space will be partitioned by a few large components. On the other hand, there will be a lot of small components, if the threshold is chosen too small. Therefore, this threshold controls the density of components in the perceptual space. The second condition is

$$TD_{\text{err}} > thr_{\text{negative\_sample}} \tag{24}$$

where $thr_{\text{negative\_sample}}$ is the threshold to determine wrongly selected actions.

1) Updating Components: Similar to the updating of matrix B, in this step just one component is updated in order to build a more local model of the perceptual space. The component which maximizes $P(X^s|M_j^s)$ is the one that is updated

$$\hat{j} = \arg\max_j \left\{ P\left(X^s|M_j^s\right) \right\}. \tag{25}$$

Then by choosing the most appropriate component $\hat{j}$ for updating, the following steps are taken

$$\beta_{\hat{j}}^s \leftarrow \beta_{\hat{j}}^s + \frac{v_{\hat{j}}^s w_{\hat{j}}^s}{v_{\hat{j}}^s + w_{\hat{j}}^s} \left(X^s - \mu_{\hat{j}}^s\right)\left(X^s - \mu_{\hat{j}}^s\right)^T \tag{26}$$

$$\mu_{\hat{j}}^s \leftarrow \mu_{\hat{j}}^s + \frac{w_{\hat{j}}^s}{v_{\hat{j}}^s + w_{\hat{j}}^s} \left(X^s - \mu_{\hat{j}}^s\right) \tag{27}$$

$$\alpha_{\hat{j}}^s \leftarrow \alpha_{\hat{j}}^s + w_{\hat{j}}^s$$
$$v_{\hat{j}}^s \leftarrow v_{\hat{j}}^s + w_{\hat{j}}^s. \tag{28}$$

2) Adding a New Component: As mentioned before, a new component is created if some criterion [see (23), (24)] is satisfied (There are also other criteria like ones mentioned in Adaptive Mixtures [39]).

Once a new component is created, its parameters are initialized, as in the initialization step of learning, except for b_{a_g,q^s}^s, which is initialized based on the TD_{err}^s

$$q^s \leftarrow q^s + 1$$
$$\alpha_{q^s}^s \leftarrow n^s, \qquad v_{q^s}^s \leftarrow n^s + 1$$
$$\mu_{q^s}^s \leftarrow X^s, \qquad \beta_{q^s}^s \leftarrow initialVariance \times I_{n^s} \tag{29}$$
$$B_{:q^s}^s \leftarrow Rand_{1\times q^s} \tag{30}$$
$$if\,(TD_{err}^s > thr_{\text{positive\_sample}}) \quad b_{a_g,q^s}^s \leftarrow 1$$
$$elseif\,(TD_{err}^s < thr_{\text{negative\_sample}}) \quad b_{a_g,q^s}^s \leftarrow -1. \tag{31}$$

In (31) if TD_{err}^s is greater than $thr_{\text{positive\_sample}}$ most probably the performed action a_g has been selected correctly and accordingly by initializing b_{a_g,q^s}^s to "1," the probability of selecting a_g correctly in the next learning steps will be increased. On the other hand, by setting b_{a_g,q^s}^s to "-1" selection of a_g in the next learning steps becomes less probable.

F. Action Fuser

According to Fig. 2, in each action selection step all suggested actions $(a_g^1, a_g^2 \ldots a_g^S)$ are input to the action fusion unit and subsequently, the fused action is performed. In fact, the decision fuser selects the action that belongs to the perceptual space in which the tiny agent T^s is the most expert. Therefore, a proper expertness criterion is essential in choosing the best suggested action [40], [41].

Basically, the expertness of a reinforcement learning agent in a specific state can be characterized by the following two criteria: 1) the agent should have visited the state a reasonable number of times, 2) the agent should have received higher rewards from that adequate number of visits [40].

In the proposed framework, the value of the distribution $P(X^s|M_j^s)$ encodes the amount of visiting of state X^s, when absorbed by the component M_j^s. Based on (26), (27), and (28), when the component M_j^s absorbs a state vector such as X^s, not only the cardinality of the distribution $P(X^s|M_j^s)$ is increased, but also the covariance of this distribution is decreased. Accordingly, the value of $P(X^s|M_j^s)$ for states which are absorbed by component M_j^s will be increased. Thereby, this distribution can satisfy the first mentioned expertness condition properly. On the other hand, according to (7), (8), and (21) the value of distributions $P(M_j^s|a_i)$ and $P(a_i)$ are indirectly updated by the TD_{err}^s. Thus, a higher value of these distributions indirectly means that these states have been rewarding enough so far. Therefore, $P(M_j^s|a_i)$ and $P(a_i)$ can satisfy the second condition of the suitable expertness measure.

Regarding (1) and (2), and also the above discussion, distribution $P(a_i|X^s)$ can be a proper expertness measure.

In sum, the action fusion process can be reduced to selecting that action (a_g^s) whose $P(a_g^s|X^s)$ value is higher than other ones. That means

$$a_g = \arg\max_{a_g^s} \left[P\left(a_g^s|X^s\right) \right]_{s=1}^S. \tag{32}$$

Note that the max operator is just one possible method to fuse actions; other suitable methods such as min-entropy and voting are also applicable. However, regarding the above mentioned expertness criteria, the max operator is found to be a more appropriate candidate in the proposed framework.

V. Simulation and Experimental Results

A set of simulations and experiments on different mazes and single car driving tasks have been performed to test the performance and the general applicability of the proposed methods, as well as revealing their advantages and limitations.

The physical robot employed in the experiments is an E-puck robot [42] (see Fig. 12). This mobile robot is equipped with two stepper motors by which it can move around. Moreover, the robot has a USB Bluetooth communication system which permits us to run the learning algorithm on our PC instead of on the limited hardware of the robot.

A. Maze Task

In this problem a mobile robot is placed in an area of specific width and length in which there are some obstacles and a goal,

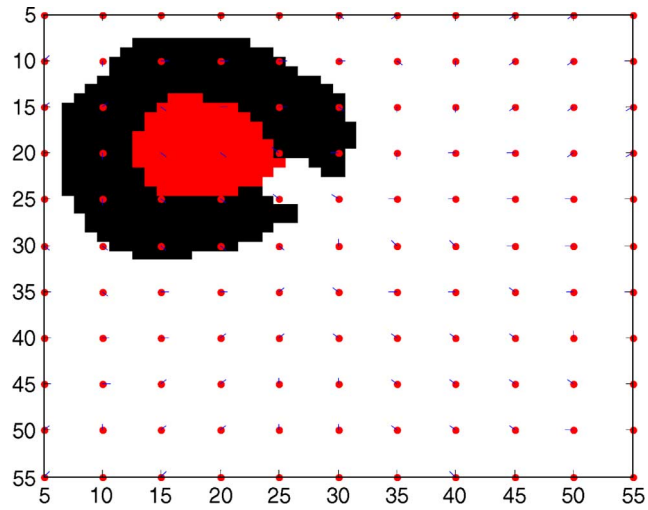

Fig. 3. Environment 1—Small arrows at the selected points show the learned optimal policy by the proposed algorithm. The red, the black, and the white areas are the goal, the obstacle, and the free spaces respectively.

see Fig. 3. The robot is supposed to learn the shortest path to the goal while avoiding the obstacles. In each learning step, we let the robot move five units in one of the 8 ($r = 8$) predetermined directions (i.e., 0, $\pi/4$, $\pi/2$, $3\pi/4$, π, $5\pi/4$, $3\pi/2$, and $7\pi/4$). It means that the robot can cluster the perceptual space into at most 8 distinct categories. Indeed, this correspondence comes from our action-based partitioning view.

1) Simulation Result I: In this simulation the following conditions are set: the robot's perceptual space is its 2D (x,y) position; the value of $thr_{\text{new\_component}}$ is gradually reduced from 0.25 to 0.2 during learning. In addition, $thr_{\text{positive\_sample}}$ and $thr_{\text{negative\_sample}}$ are set to 6 and -3 respectively and the initial variance is set to 0.1. After every movement of the robot, three different internal reinforcement signals are generated; if it comes into an empty space (neither goal nor obstacle) it gets -0.1, if it faces an obstacle, the punishment is -5 and finally if it reaches the goal, it receives 20 as the reward. Besides, the learning rate (α) is changed from 0.9 to 0.1 and the forgetting factor is set to 0.8.

The proposed method is benchmarked against Q-learning with discrete states. In order to have a fair comparison, four different quantization levels (i.e., 2, 2.5, 3.3, and 5) are designed for the Q-learner and since the environment's size is 50×50, the Q-table size of the four Q-learners are $25 \times 25 = 625$, $20 \times 20 = 400$, $15 \times 15 = 225$, and $10 \times 10 = 100$, respectively.

In Fig. 4, average reward (window size $= 20$) and accumulated reward of the proposed approach against the episode number are shown and compared with that of Q-Learners having the mentioned quantization factors. As the figure shows, our method demonstrates faster learning and higher rewards during the initial stages of learning; which is quite important in practical applications of RL.

As shown in Fig. 5, the number of produced components is 126. That number represents the Q-Table size and it is close to that of the coarsest applied quantization level, which does not lead to an acceptable learning. On the other hand, the best

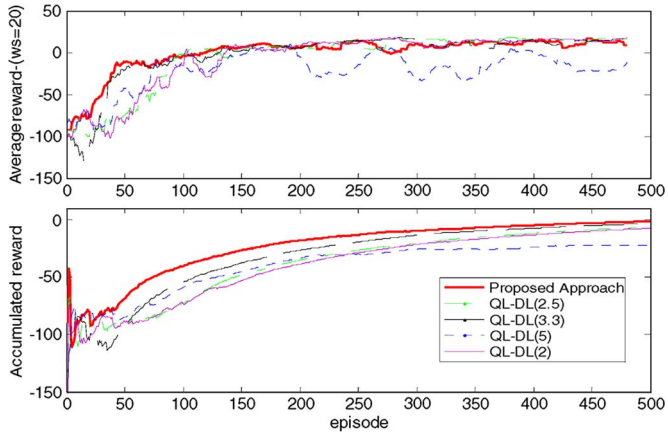

Fig. 4. Average (window size $= 20$) and accomulated rewards gained by the proposed algorithm in comparison to those obtained by the Q-Learners with different state quantization factors (i.e., 5, 3.3, 2.5, and 2).

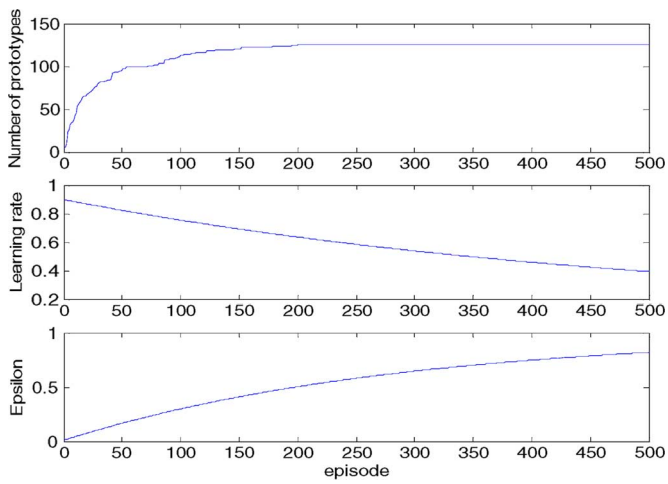

Fig. 5. Number of components, learning rate and value of ε during the learning.

Q-learner quantization factor (3.3) is determined by trial and error which is not possible in practice. In fact our approach gains this performance without any prior knowledge about the best distribution of prototypes. Also the distribution of the learned components—as illustrated in Fig. 6—shows that the proposed algorithm controls the density of the components properly. For example in Fig. 6, in an area like A, where the optimal actions are the same, fewer components are generated in comparison with the number of components in an area like B. This adaptive generation of components is the main reason for faster convergence and for gaining more rewards during early stages of learning. In other words, the agent tries to assume components that are as general as possible. This leads to consuming fewer episodes for learning components in the areas where optimal actions are the same; e.g., area A in Fig. 6. Let us call this property *generalization ability*.

Now the question is "to what extent does this generalization ability hold?" To answer this question, some other environments (see Figs. 7, 8, and 9) are employed. The results are compared with those of a Q-learner with 3.3 quantization factor. The simulations show that by scattering more obstacles in the environment, the generalization ability decreases,

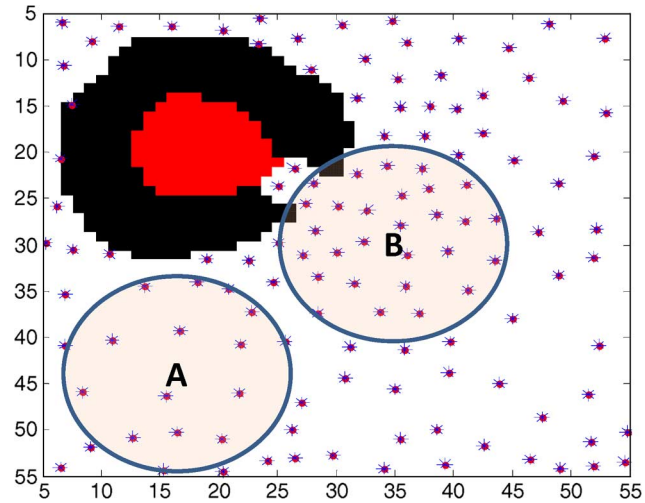

Fig. 6. Environment 1; Final distibution of produced components by the proposed framework.

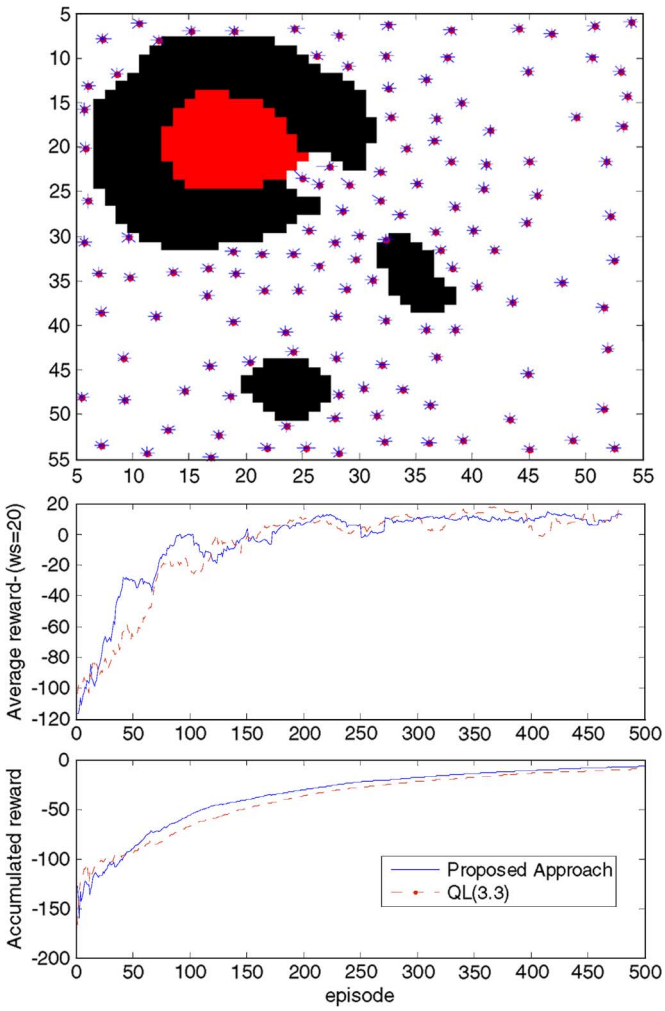

Fig. 7. Environment 2; Distibution of learned components, average reward (window size = 20), and accumulated reward by the proposed framework in comparison with the Q-learner (quantization factor = 3.3).

and the performance of the proposed approach comes closer to—but still better than—that of the Q-learner. In fact scattered

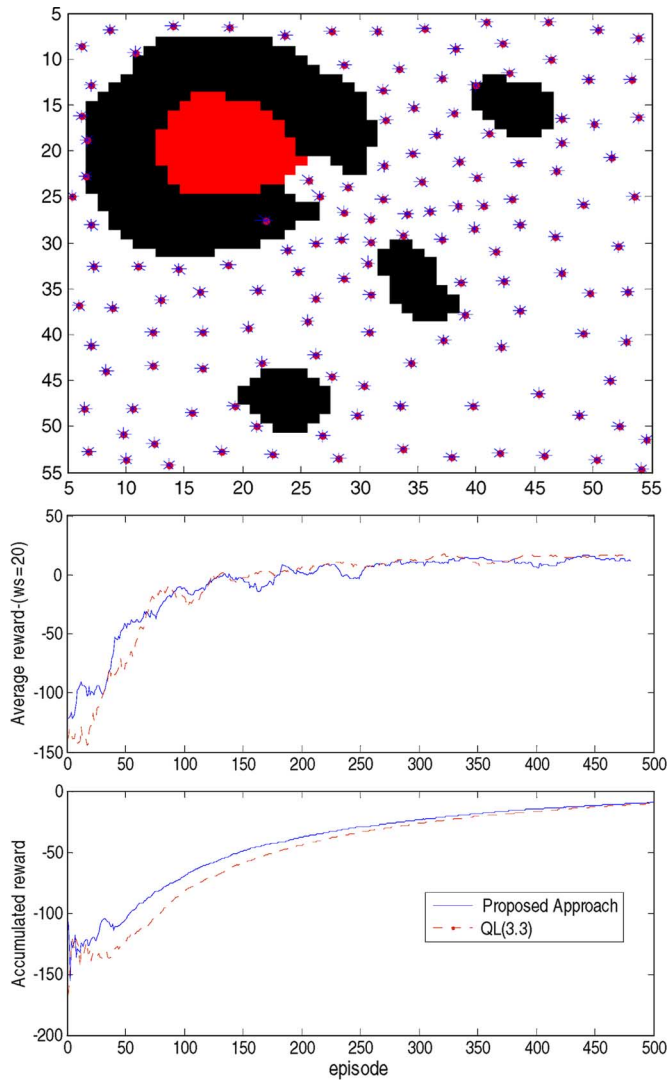

Fig. 8. Environment 3; Distribution of learned components, average reward (window size = 20), and accumulated reward by the proposed framework in comparison with the Q-learner (quantization factor = 3.3).

obstacles worsen the generalization ability by decreasing the number of free spaces. It means that the selected space representation—XY coordinate—does not code a strong enough generalization property in terms of the discussed action-based partitioning and learning when the obstacles are distributed in the environment. This is due to the fact that optimal actions in neighboring points of an XY coordinate are different; which means that more components are required (129, 139, and 138 components for environments represented in Figs. 7, 8, and 9, respectively), and consequently there is less generalization capability and slower learning speed.

2) Simulation Result II: This simulation is designed to demonstrate the advantage of learning in multiple perceptual spaces. All conditions are similar to the *Simulation I*, except that we have two extra perceptual spaces here. In other words, there are three tiny agents with different state spaces; one with the XY robot position, one with its X coordinate, and the third one with robot's position in the Y direction.

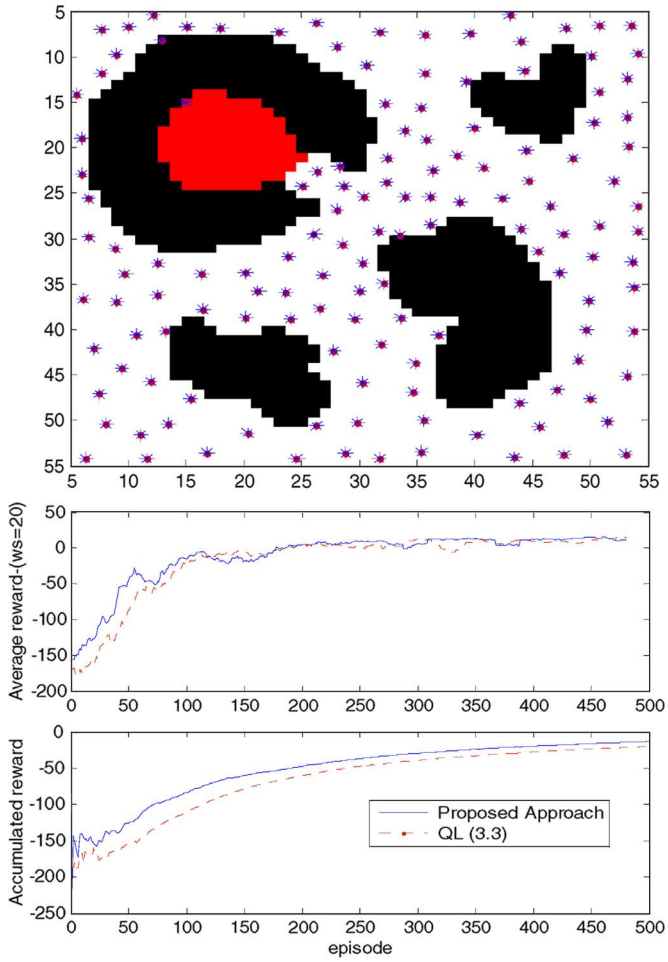

Fig. 9. Environment 4; Distibution of learned components, average reward (window size = 20), and accumulated reward by the proposed framework in comparison with the Q-learner (quantization factor = 3.3).

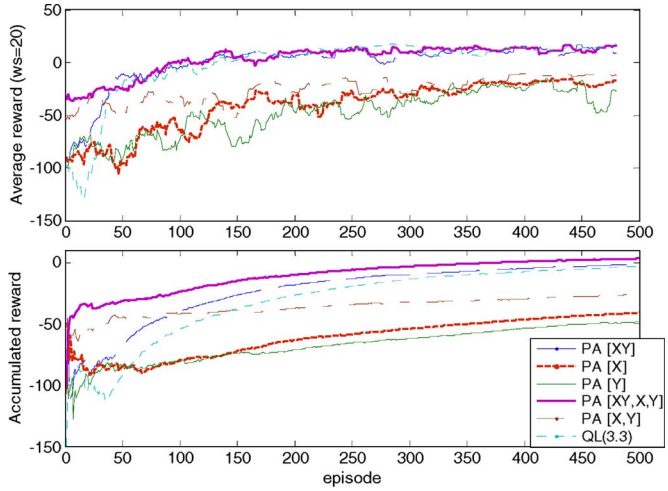

Fig. 10. Average reward (window size = 20) and accumulated reward gained by the proposed approach and the Q-learner in the Environment 1. PA[XY]= learning in XY sapce, PA[X] = learning in X sapce, PA[Y] = learning in Y space, PA[XY,X,Y] = learning in multiple perceptual spaces XY, X, and Y, PA[X, Y]= learning in multiple perceptual spaces X and Y, and QL(3.3)= a typical Q-learner with quantization level 3.3.

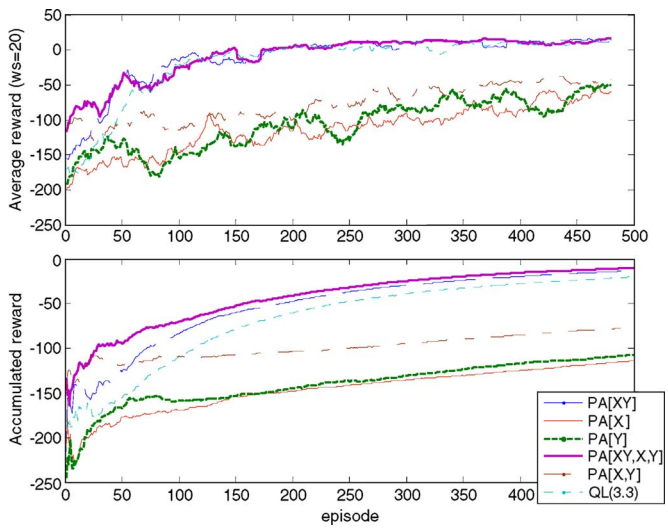

Fig. 11. Average reward (window size = 20) and accomulated reward gained by the proposed approach and the Q-learning in the Environment 4. PA[XY] = learning in XY sapce, PA[X] = learning in X sapce, PA[Y] = learning in Y space, PA[XY,X,Y] = learning in multiple perceptual spaces XY, X, and Y, PA[X, Y] = learning in multiple perceptual spaces X and Y, and QL(3.3) = a typical Q-learner with quantization level 3.3.

As shown in Fig. 10, learning in three spaces at the same time (PA[XY, X, Y]), accelerates the learning speed by employing the generalization ability of the proposed method in low dimensional perceptual spaces (i.e., X and Y). In fact, as the figure shows, PA[X] and PA[Y] agents cannot learn the task completely as the world is not fully observable by them. However, they speed up the learning process of PA[XY,X,Y] agent at its early stages. Note that this achievement does not necessitate any more learning episodes for the robot, because learning in all spaces is done in parallel.

Fig. 11 shows the same results in the Environment 4 (Fig. 9). As the obtained results indicate, the generalization ability of learning in subspaces (X, Y) is decreased; which is due to a reduction in the size of the free areas. Therefore, in Fig. 11, the learning speed in multiple perceptual spaces (PA [XY, X, Y]) is closer to that in PA [XY], compared with the result illustrated in Fig. 10.

3) Experimental Result: In this section, an E-puck robot [42] is employed to examine the learned policy in a noisy and real environment. This experiment has been executed to show the ability of the proposed framework to manage uncertainties and to generalize the learned knowledge.

As in the simulation, the robot's goal is to reach a specific area while it is avoiding the obstacles. This experiment includes two steps. In the first one, a model of the real environment—except the model of input sensory noise—is employed for learning (see Fig. 13). In other words, the robot first learns the desired behavior in the simulation environment. Then, in the second step, the robot employs the learned behaviors in the real environment, see Fig. 12. In fact, in this step the robot selects greedy actions. A camera is installed above the maze to send feedback to the robot of its position.

Fig. 14 illustrates the received rewards in the learning step during the 600 episodes and Fig. 15 demonstrates the received rewards in the real environment during 50 episodes.

Fig. 12. E-puck robot and the *experimental* environment.

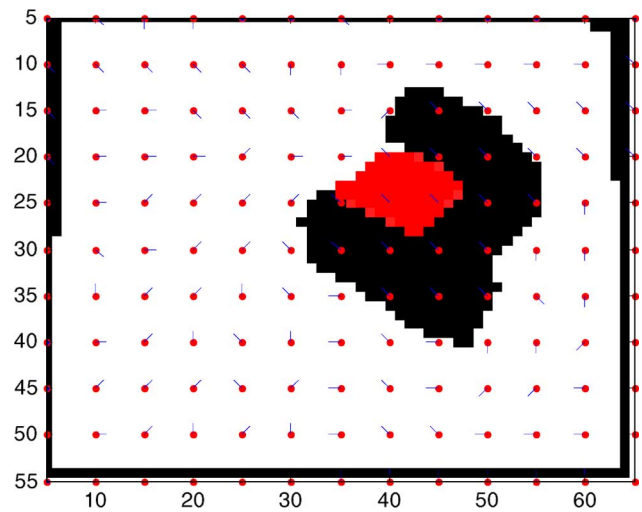

Fig. 13. Small arrows at the selected points show the learned optimal policy by the proposed algorithm. The red, the black and the white areas are the goal, the obstacle, and the free spaces respectively (model of the real environment which is used in the first step).

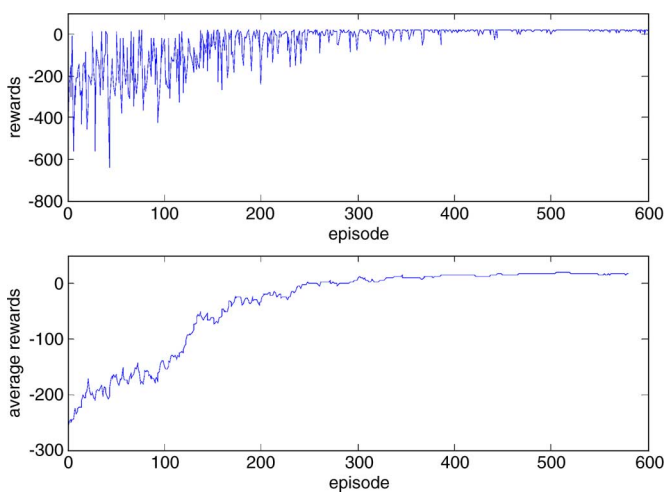

Fig. 14. Received and average reward during the learning in the first step (simulation).

It is noticeable that the simulations report the learning process (see Fig. 14) while the experimental results show the robot performance using the already learned policy in the simulation. Therefore, the final parts of graphs in Fig. 14 should be compared with the data shown in Fig. 15. Considering the graph

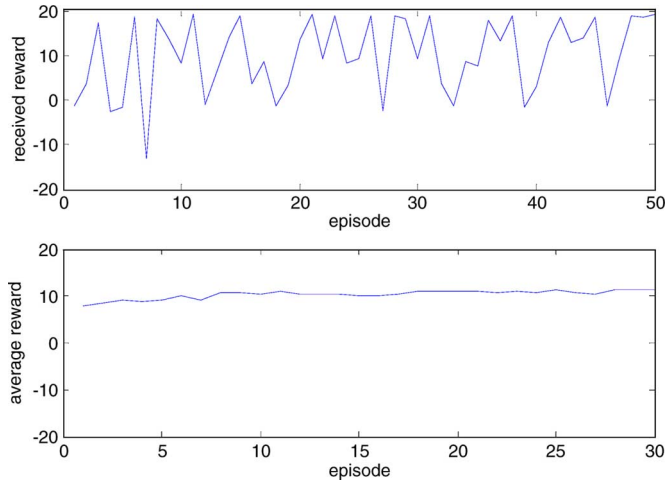

Fig. 15. Received and average reward in the real environment (greedy action selection).

scales, this comparison shows that there is no significant performance drop from the simulations to the experiment; which is a good achievement. In addition please note that due to the averaging window size of 20, the lower diagram is sketched for 30 episodes.

It is a common practice to learn the task in the simulation and then employ the learned knowledge in the real world, when learning from scratch is experimentally not feasible. Since the real world and simulators differ to a considerable extent, it is highly probable that the optimal policy found in the simulations does not result in proper behavior in the experiments. However, we showed that our method is a positive exception. This means that, in our approach, learning in simulations results in good performance in experiments as well. It is due to the fact that by soft and probabilistic partitioning of the state space, we make our agent robust against uncertainties and noise that exist in the real world. One of the major sources of uncertainty is the difference between the real world and its model used in the simulations in addition to error in the agent's sensory systems.

B. Single Car Driving Task

A simplified driving simulator is written in MATLAB. It contains a road and a small car in it. The robot (small car) can visually observe three available spatial areas; front, left, and right. A simple view of its visual field and the roads is shown in Fig. 16.

The robot's motor actions are setting the forward speed to (1.5, 3 cm/s) and the steering angle to (−pi/8, 0, pi/8).

The reward function is defined as

$$\begin{aligned} r(t) = {} & 200 \times (\text{body\_in\_road} - 1) \\ & + 120 \times (\text{front\_body\_in\_road} - 1) \\ & + 5 \times (\text{cur\_speed} - \max\{\text{robot\_speed}\}) \\ & - 0.1 \times |\text{cur\_str\_angle}| \\ & - 300 \times (\text{is\_out\_of\_boundary}). \end{aligned} \tag{33}$$

The parameters in the reward function are defined as:

- *Body_in_road*: percentage of the robot's body in road. The more off-road body, the more punishment will be assigned;

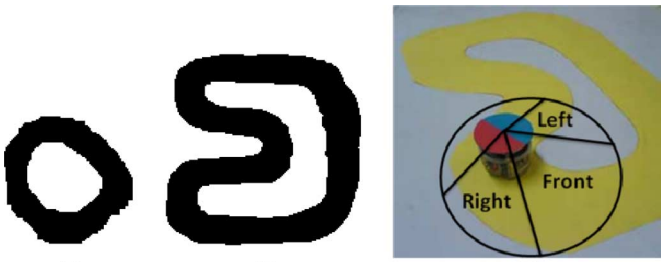

Fig. 16. a) the road map used for training called learning road, b) the road map used for simulation test called test road, c) the E-puck robot driving on the road map used for experiments. The spatial visual areas are specified: Front, Left and Right.

- *Front_body_in_road*: percentage of the robot's front perceptual space in road. It is important that both the robot and its perceptual areas to be in the road boundary to keep the road observable;
- *Cur_speed* is the current velocity of the robot and *max{robot_speed}* is its maximum velocity. It means the robot should move faster to get less punishment;
- *Cur_str_angle* is the current steering angle of the robot which is less rewarding if it is far from zero;
- *is_out_of_boundary* is set to one if the robot goes out of the environment by performing its suggested action. In other cases it is set to zero.

Therefore, the robot's driving task is summarized into *learning how to drive within the road boundary at maximum speed while trying not to go out of the environment's boundary*.

1) Simulation Result: In this simulation the following conditions are set:

The perceptual space is a three dimensional space which is defined as:

$$X = [\text{front,left,right}] \in \Re^3$$

- front: the road percentage in robot's front visual area.
- left: the road percentage in robot's left visual area.
- right: the road percentage in robot's right visual area.

$$thr_{\text{new\_component}} = 0.6 \rightarrow 0.25$$
$$thr_{\text{positive\_sample}} = -10$$
$$thr_{\text{negative\_sample}} = -50$$

Fig. 17 illustrates the average reward $(\text{window size} = 100)$ and accumulated reward during the learning in the learning environment [see Fig. 16(a)]. After learning the optimal policy in the learning environment, the learned behavior is tested on a different road [the test road in Fig. 16(b)] to demonstrate the generalization ability of the proposed framework (see Fig. 18 for the result). The robot greedily selects actions during the test. The suitable average and accumulated reward on the test road [see Fig. 16(b)] show the applicability and the robustness of the proposed framework against new roads.

2) Experimental Result: The learned policy is employed on the E-puck robot driving on a real miniature road [see Fig. 16(c)] in order to demonstrate robustness of the learned behavior against noise and uncertainty in the real world. Because of the very limited visual field of the E-puck camera, we used an external camera to feedback the robot's front, left and right visual areas. The results are shown in Fig. 19.

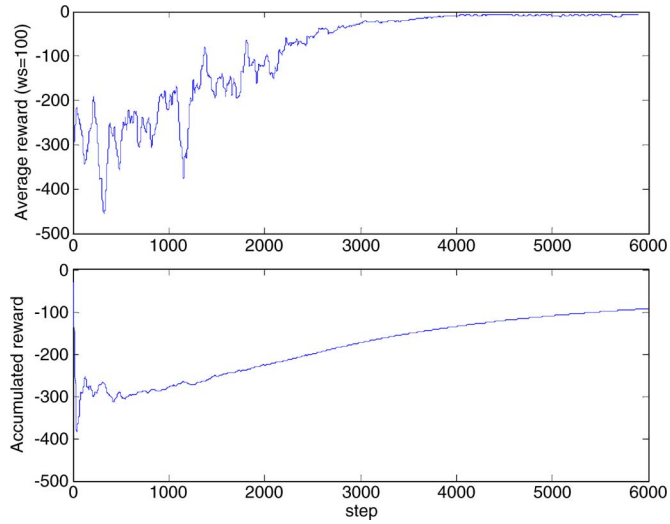

Fig. 17. Average reward $(\text{window size} = 100)$ and accumulated reward during learning on the learning road.

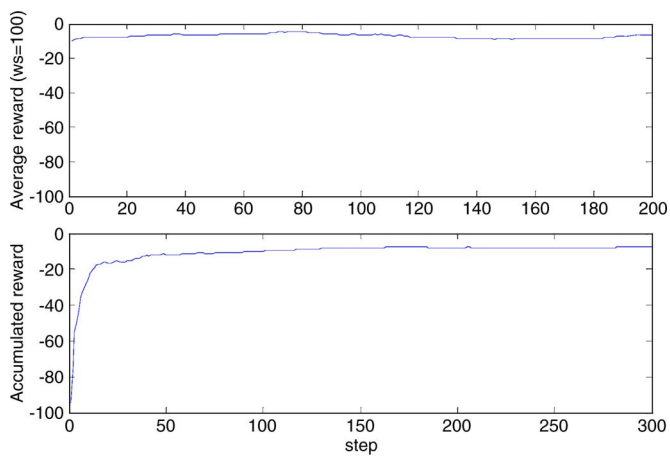

Fig. 18. Average reward $(\text{window size} = 100)$ and accumulated reward during test (greedy run) on the test road.

In spite of the E-puck's noisy observations and facing a new environment, comparing the experimental result (see Fig. 19) with the simulation result (see Fig. 18) shows the proper robustness of the proposed framework in real problems. In fact this reduction in average rewards is happened because of a smaller road width and imperfect image processing.

3) Simulation Result II: To show the advantage of learning in multiple perceptual spaces, the behavior in the front visual area is learned in parallel with learning in the entire robot perceptual space $(X \in \Re^3)$. It means that the robot's mind contains two tiny agents; agent_1 observes state $X = [front, left, right] \in \Re^3$ and the perceptual space of agent_2 is $X' = [front] \subset X$.

Based on Fig. 20, the generalization ability of the second perceptual space (X') speeds up the learning and increases the accumulated reward especially during the early stages of learning. The road shown in Fig. 16(a) is used in this simulation.

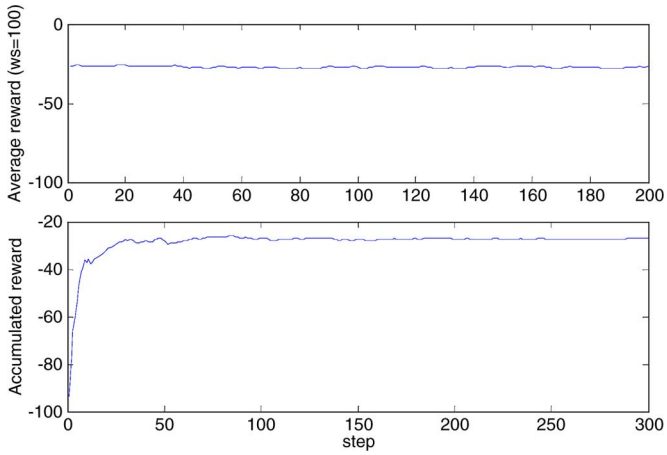

Fig. 19. Average reward (window size = 100) and accumulated reward on the real environment (greedy run).

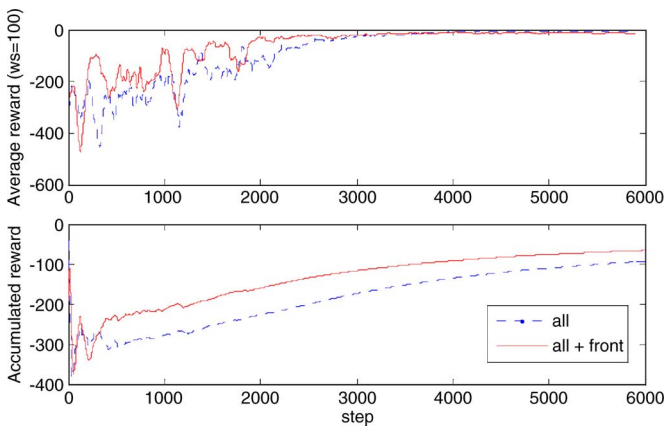

Fig. 20. Comparing average reward (window size = 100) and accumulated reward for learning using single perception (All) with multiple perceptions in driving simulation (All + Front).

VI. Further Discussion and Conclusion

A new method for learning in continuous and multimodal spaces was presented. The learning algorithm is constructed from the probabilistic formalism which enables the agent to face uncertainty in its perception and noisy environments. Utilizing the mixture density model with adaptive number of components, the developed model is capable of soft partitioning the perceptual space while learning.

On the other hand, the learning algorithm is designed to learn through received rewards and punishments and handles the multistep problems as well as discounted rewards.

The framework we propose for learning in multiple perceptual spaces is a Mixture of Experts strategy in essence. This brings many advantages such as alleviating the high dimensionality problem, providing a higher degree of fault tolerance and robustness, and speeding up the learning. In addition, the proposed method does not impose any extra learning trials to the learning agent; however, it increases the computational cost of learning because of learning in multiple perceptual spaces. Simulation and experimental results demonstrated not only the effectiveness of our approach in a noisy environment, but also the superiority of learning in multiple perceptual spaces. In sum, the theoretical discussions, the results of simulations, and the outcomes of experiments, reveal the following major points:

- Table I shows the learning speed in terms of the number of episodes to attain the defined percentage of accumulated reward by different learning methods in four maze environments. The reference for calculation of those percentages is the reward accumulated by multiple perceptual spaces (MPA) method—which is the best method—at episode 500.
  Based on Table I, learning in MPA is the best method in terms of the learning speed in all four maze environments. In addition, the proposed approach (PA) is faster than Q-learning. The difference between learning speeds for Q-learner, PA, and MPA decreases with increase in the complexity of the environment. In fact, environments 2 and 3 are more complex—thus less generalizable—as small obstacles distort the optimal policies around them while in environments 1 and 4, the free and obstacle areas are larger. Therefore, PA and MPA show their advantages far distinguishing in environments 1 and 4.
- Table II compares the learning speeds of PA and MPA methods in the single car driving task. The reference for calculation of those percentages is the reward accumulated by the MPA method at learning step 6000. As the table shows, MPA is faster than PA and the difference in their learning speeds decreases as the learning proceeds. Faster learning in the early stages of learning is essential for the application of RL methods in the real world, which is attained by our MPA method.
- The proposed approach automatically and—we would claim—properly partitions the perceptual space, while in existing researches it is mostly done by hand. Of course there are some methods that partition the state-space adaptively; however, their partitioning methods are deterministic and crisp. Indeed an efficient partitioning mechanism if performed in accordance with the agent's needs (action-based) in a soft manner (Gaussian distribution) can gain the proper level of achievable reward. On the other hand, to have faster learning and to save memory, a proper partitioning method should find coarse segments in some areas in order to exploit the generalization property that is encoded in the environment. Therefore, contrary to hand-designed partitioning mechanisms which require preknowledge about the task and the environment; our approach partitions the state-space interactively by some soft modules according to the expected reward.
- The proposed approach needs a far smaller number of partitions in comparison to a Q-learner with the same level of performance. It is due to the fact that our method partitions the environment by soft entities according to the expected reward. It means that the state partitioning is in-line with reward maximization which is the ultimate goal of any RL agent.
- One of the advantages of the proposed approach is faster learning when generalization of experiences is possible. The possibility of generalization of past experiences depends on the environment and the state representation

TABLE I
ANALYZING THE LEARNING SPEED OF DIFFERENT METHODS IN TERMS OF ACCUMULATED REWARD IN THE MAZE TASK, PA = PROPOSED APPROACH IN XY SPACE, QL = THE Q-LEARNER WITH QUANTIZATION FACTOR 3.3, MPA = PROPOSED APPROACH IN {XY, X, Y} SPACES

Percentage of accumulated reward	60 %			70 %			80 %			90 %		
Method	PA	QL	MPA	PA	QL	MPA	PA	QL	MPA	PA	QL	MPA
Environment 1 (number of episodes)	103	145	11	142	188	78	201	260	133	345	400	229
Environment 2 (number of episodes)	121	160	119	168	203	152	230	269	229	329	376	317
Environment 3 (number of episodes)	151	182	118	196	224	166	258	284	228	346	372	314
Environment 4 (number of episodes)	120	165	97	163	212	138	228	289	200	334	425	294

TABLE II
ANALYZING THE LEARNING SPEED OF DIFFERENT METHODS IN TERMS OF ACCUMULATED REWARD IN THE SINGLE CAR DRIVING, PA = PROPOSED APPROACH IN {FRONT_LEFT_RIGHT} SPACE, MPA = PROPOSED APPROACH IN { FRONT_LEFT_RIGHT , FRONT} SPACES,

Percentage of accumulated reward	60 %		70 %		80 %		90 %	
Method	PA	MPA	PA	MPA	PA	MPA	PA	MPA
Single Car Driving (number of steps)	2701	1469	3353	2034	4269	2680	5767	3762

method. Therefore, state representation highly affects the learning speed of the proposed method.

- The possibility of concurrent learning in multiple subspaces is helpful specifically for dealing with the real world's uncertain challenges. In fact, our method can start learning with simpler perceptual subspaces and then gradually add more complex perceptual ones. This gradual learning results in a higher—but limited—level of rewards in the early stages of learning—which is very important for situated and interactive agents—as learning in simpler spaces is expected to converge faster. Moreover, adding a new perceptual space does not disturb the agent's performance, since the Max operator in the decision fusion stage automatically selects the decisions made in the simpler spaces which have previously resulted in a dominant policy. By dominant policy we mean a policy in which the probability of selecting an action is higher than the probability of selecting any other action in the previously learned spaces.
- No divergence was observed in the simulations and the proposed methods converged to the solutions obtained by the Q-learner agents. Providing convergence proofs for the proposed methods is among our future research goals.
- The max operator in the action fuser works well; however, it is not an optimal action fuser in general. Leaning to fuse the actions—especially when the environment is not fully observable—is the next step of this research.
- Setting the thresholds is a hard task in all learning methods and our method is not an exception. Therefore, finding a proper and computationally nonintensive method for adaptive and automatic selection of the thresholds is highly desired.
- Actually, the real robot's experiment has been executed to show the ability of the proposed framework in managing uncertainties and generalizing the learned knowledge. It shows the robustness of the proposed framework against noise and uncertainty in this real problem. This robustness is the result of our employing the Bayesian approach for soft partitioning and learning. In addition, we have showed by a series of simulations that learning in subspaces and fusing the corresponding decisions results in faster learning; which is crucial in real world tasks. The next stage of this research is to demonstrate this characteristic in the experiments as well.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments, which have helped us to greatly improve this article.

REFERENCES

[1] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. Int. Conf. Mach. Learn.*, May 1990, vol. 11, p. 92.
[2] S. Singh and T. Jaakkola, "Reinforcement learning with soft state aggregation," *Adv. Neural Inform.*, 1995.
[3] S. Pareigis, "Adaptive choice of grid and time in reinforcement learning," *Adv. Neural Inform. Process. Syst.*, pp. 1036–1042, 1998.
[4] A. Persson, "Using temporal difference methods in combination with artificial neural networks to solve strategic control problems," KTH Numerical Analysis and Computer Science Royal Institute of Technology. Stockholm, Sweden.
[5] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Netw.*, vol. 3, pp. 724–740, 1992.
[6] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, 2000.
[7] H. R. Berenji, "Fuzzy Q-learning for generalization of reinforcement learning," in *Proc. Fifth IEEE Int. Conf. Fuzzy Syst.*, 1996, pp. 2208–2214.
[8] D. Vengerov, N. Bambos, and H. R. Berenji, "A fuzzy reinforcement learning approach to power control in wireless transmitters," *IEEE Trans. Syst., Man, Cybernet. Part B: Cybernet.,* , vol. 35, pp. 768–778, 2005.
[9] S. Hart and R. Grupen, "Learning Generalizable Control Programs," *Trans. Auton. Mental Develop.*, pp. 1–16, 2010.
[10] T. Zentall and M. Galizio, "Categorization, concept learning, and behavior analysis: An introduction.," *Anal. Behav.*, 2002.
[11] G. Buccino, S. Vogt, A. Ritzl, G. Fink, and K. Zilles, "Neural circuits underlying imitation learning of hand actions: An event-related fMRI study," *Neuron*, 2004.

[12] M. A. Arbib, "The mirror system, imitation, and the evolution of language," *Imitation in Animals Artifacts*, p. 229, 2002.
[13] G. Rizzolatti and M. A. Arbib, "Language within our grasp," *Trend. Neurosci.*, vol. 21, pp. 188–194, 1998.
[14] A. Billard and M. Mataric, "Automatic learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture," *Robot. Autonom. Syst.*, vol. 941, pp. 1–16, 2001.
[15] C. Keysers, E. Kohler, M. Umiltà, and L. Nanetti, "Audiovisual mirror neurons and action recognition," *Exp. Brain*, 2003.
[16] R. S. Sutton and A. G. Barto*, Reinforcement Learning: An Introduction*. , MA: MIT Press, 1998.
[17] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Systems, Man, Cybernet., Part C: Appl. Rev.,* , vol. 28, pp. 338–355, 1998.
[18] S. Whiteson, M. E. Taylor, and P. Stone*, Adaptive Tile Coding for Value Function Approximation*. : Citeseer, 2007.
[19] H. Mobahi, M. N. Ahmadabadi, and B. N. Araabi, "A biologically inspired method for conceptual imitation using reinforcement learning," *Appl. Artif. Intell.*, vol. 21, pp. 155–183, 2007.
[20] S. Mahadevan, "Automatic programming of behavior-based robots using reinforcement learning," *Artif. Intell.*, 1992.
[21] J. K. Kruschke and M. K. Johansen, "A model of probabilistic category learning," *J. Exp. Psychol.: Learn., Memory, Cogn.*, vol. 25, p. 1083, 1999.
[22] M. Asada, S. Noda, and K. Hosoda, "Action-based sensor space categorization for robot learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. '(IROS 96)*, 1996, pp. 1502–1509.
[23] J. Millá;n and D. Posenato, "Continuous-action Q-learning," *Mach. Learn.*, 2002.
[24] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proc. 1993 Connectionist Models Summer School*, 1993, Citeseer.
[25] M. Asadpour, M. N. Ahmadabadi, and R. Siegwart, "Heterogeneous and hierarchical cooperative learning via combining decision trees," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2006, pp. 2684–2690.
[26] A. Moore, L. Birnbaum, and G. Collins, "Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces," in *Proc. 8th Int. Workshop,*, San Mateo, CA, 1991, pp. 333–337.
[27] F. Ferná;ndez and D. Borrajo, "VQQL. Applying vector quantization to reinforcement learning," in *Proc. Robocup-99: Robot Soccer World Cup iii*, 2000, pp. 49–57.
[28] F. Ferná;ndez and L. E. Parker, "Learning in large cooperative multi-robot domains," *Int. J. Robot. Autom.*, pp. 217–226, 2001.
[29] A. Smith, "Applications of the self-organising map to reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 15, pp. 1107–1124, Oct. 2002.
[30] E. Uchibe and K. Doya, "Competitive-cooperative-concurrent reinforcement learning with importance sampling," in *Proc. Int. Conf. Simulation Adapt. Behav.*, 2004, p. 287.
[31] O. Lebeltel, P. Bessière, and J. Diard, "Bayesian robot programming," *Autonom. Robot.*, 2004.
[32] R. E. Neapolitan*, Learning Bayesian Networks*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.
[33] P. Bessière, Survey: Probabilistic Methodology and Techniques for Artefact Conception and Development 2003.
[34] D. M. Chickering, "A transformational characterization of equivalent Bayesian network structures," in *Proc. UAI'95*, 1995, pp. 87–98.
[35] C. Koike, C. Pradalier, and P. Bessière, "Proscriptive bayesian programming application for collision avoidance," *Intell. Robot.*, 2003.
[36] J. Tenenbaum, "Bayesian modeling of human concept learning," *Adv. Neural Inform. Process. Syst.*, vol. 11, 1999.
[37] D. Roy, "Learning From Sights and Sounds: A Computational Model," Ph.D. dissertation, MIT, , 1999.
[38] R. Duda and P. Hart*, Pattern Classification*, 2nd ed. New York: , 2001.
[39] C. E. Priebe, "Adaptive mixtures," *J. Amer. Statist. Assoc.*, vol. 89, pp. 796–806, 1994.
[40] M. N. Ahmadabadi, A. Imanipour, B. N. Araabi, M. Asadpour, and R. Siegwart*, Knowledge-Based Extraction of Area of Expertise for Cooperation in Learning*. Piscataway, NJ: IEEE, 2006.
[41] M. N. Ahmadabadi and M. Asadpour, "Expertness based cooperative Q-learning," *IEEE Trans. Syst., Man, Cybernet., Part B: Cybernet.*, 2002.
[42] F. Mondada, M. Bonani, and X. Raemy, "The e-puck, a robot designed for education in engineering," *Autonom. Robot.*, 2009.

Hadi Firouzi received the M.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2008. He is currently working towards the Ph.D. degree at Okanagan School of Engineering, The University of British Columbia, Canada, where he is focusing on the autonomy of robots in dynamic environments using visual object tracking.

Since then, he has been working on different areas of research, which include mobile robot localization and mapping, interactive machine learning and attention control, vision-based robot control, visual object detection, and tracking. http://acis.ok.ubc.ca/hadifirouzi.htm.

Majid Nili Ahmadabadi was born in 1967. He received the B.Sc. degree from Sharif University of Technology of Iran, Tehran, Iran, in 1990. He received the M.Sc. and Ph.D. degrees in information sciences from the Graduate School of Information Science, Tohoku University, Tohoku, Japan, in 1994 and 1997, respectively.

In 1997, he joined the Advanced Robotics Laboratory at Tohoku University. Later he moved to School of Electrical and Computer Engineering, College of Engineering, University of Tehran where he is a Professor and the Head of the Robotics and AI Laboratory. He is the Founder and Director of Cognitive Robotics Laboratory as well. He is also a Senior Researcher at School of Cognitive Sciences, Institute for Research in Fundamental Sciences (IPM), Iran. In the summers of 2005 and 2008, he worked with Autonomous System Laboratory at EPFL and ETHZ as an invited visiting professor. He was one of the Distinguished Lecturers selected by IEEE Robotics and Automation Society for the years 2007–2009. His main research interests are cognitive robotics and modeling cognitive systems, learning systems, distributed robotics, object manipulation, and mobile robots.

Dr. Nili served as a member of the Engineering board of Iranian National Science Foundation for the period 2005–2011. He is one of the founders of Robotics Society of Iran and a member of board of directors of Mechatronics Society of Iran.

Babak Nadjar Araabi (S'98–M'01) was born in 1969. He received the B.Sc. degree from Sharif University of Technology, Tehran, Iran, in 1992, the M.Sc. degree from the University of Tehran, Tehran, Iran, in 1996, and the Ph.D. degree from Texas A&M University, College Station, in 2001, all in electrical engineering.

In January 2002, he joined the School of Electrical and Computer Engineering at University of Tehran, where he is currently an Associate Professor and the Head of the Control Systems Division. He is also a Research Scientist with the School of Cognitive Sciences, Institute for Studies in Theoretical Physics and Mathematics, Tehran. He is the author of more than 100 international journals and conference papers in his research areas, which include machine learning, pattern recognition, neurofuzzy control, predictive control, and system modeling and identification.

Saeed Amizadeh is currently working towards the Ph.D. degree in intelligent systems program at University of Pittsburgh, Pittsburgh, PA. Saeed has completed his second masters in Intelligent Systems at University of Pittsburgh and his first masters in Artificial Intelligence and Robotics at University of Tehran, Tehran, Iran. His B.S degree is in Computer Science from University of Tehran. He has also worked at Intel Labs and Microsoft Research as 'Research Intern'. Saeed's areas of interest are Artificial Intelligence, Machine Learning and Data Mining. More recently, he has been working on learning large-scale graphical models, dimensionality reduction and spectral analysis of large datasets.

Maryam S. Mirian has received her Ph.D. degree from University of Tehran in Artificial Intelligence and robotics in September 2010. Her master degree is in the same field while she studied hardware engineering during the undergraduate level. She had worked and researched in the department of IT in Iran Telecom Research Center since 2003 on different projects focusing on various areas such as text mining, learning organizations, ontological research content generation and knowledge networking. She is joining the AI group of ECE Department of University of Tehran as a faculty member.

Roland Siegwart (M'90–SM'00–F'08) is a full professor for Autonomous Systems and Vice President Research and Corporate Relations at ETH Zurich since 2006 and 2010 respectively. He has a Master in Mechanical Engineering (1983) and a PhD in Mechatronics (1989) from ETH Zurich. In 1989/90 he spent one year as postdoctoral fellow at Stanford University. After that he worked part time as R&D director of MECOS Traxler AG and lecturer at the Institute of Robotics, ETH Zürich. From 1996 to 2006 he was associate and later full professor for Autonomous Microsystems and Robots at the Ecole Polytechnique Fédérale de Lausanne (EPFL). Roland Siegwart is member of the Swiss Academy of Engineering Sciences, IEEE Fellow and officer of the International Federation of Robotics Research (IFRR). He served as Vice President for Technical Activities (2004/05) and was awarded Distinguished Lecturer (2006/07) and is currently an AdCom Member (2007–2011) of the IEEE Robotics and Automation Society. He leads a research group of around 30 people working in the fields of robotics, mechatronics and product design. Roland Siegwart was a general chair of several conferences in robotics including IROS 2002, AIM 2007, FSR 2007, ISRR 2009 and is a cofounder of multiple successful spin-off companies in robotics. http://www.asl.ethz.ch/