2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Combining learned controllers to achieve new goals based on linearly solvable MDPs

Eiji Uchibe and Kenji Doya

Abstract— **Learning complicated behaviors usually involves intensive manual tuning and expensive computational optimization because we have to solve a nonlinear Hamilton-Jacobi-Bellman (HJB) equation. Recently, Todorov proposed a class of the so-called Linearly solvable Markov Decision Process (LMDP) which converts a nonlinear HJB equation to a linear differential equation. Linearity of the simplified HJB equation allows us to apply superposition to derive a new composite controller from a set of learned primitive controllers. However, his method was a model-based approach and it was not evaluated in a real domain. This study proposes a model-free method which is similar to the Least Squares Temporal Difference (LSTD) learning. In this method, the exponentially transformed cost function can be regarded as the discount factor in LSTD. Our proposed method is applied to learning walking behaviors with the quadruped robot to evaluate in real robot experiments. The goal of each primitive task is to go to the specific target position in the environment and that of the composite task is to approach arbitrary region represented by the primitives' target positions. Experimental results show that the composite policy can be used as a good initial policy for the new task.**

I. INTRODUCTION

When we want to design an autonomous robot that can act optimally in its environment, the robot should solve nonlinear optimization problems in continuous state and action spaces. If a precise model of the environment is available, then both optimal control [1] and model-based reinforcement learning [2] give a computational framework to find an optimal control policy which minimizes cumulative costs (or maximizes cumulative rewards). If not, model-free reinforcement learning is applicable. In recent years, reinforcement learning algorithms have been applied to a wide range of neuroscience data [3] and model-based approaches have been receiving attention among researchers who are interested in decision making [4], [5].

However, a drawback is the difficulty to find an optimal policy for continuous states and actions. In particular, learning time can become unacceptable long if the model is unknown in advance. There exist several ways to reduce learning time, and we are interested in the reuse of previously learned policies in order to find or design a new optimal policy. If the robot has enough learned policies, no additional learning is required to design the new task by combining learned policies as primitives.

*This work was supported by JSPS KAKENHI Grant Number 24500249 and Grant-in-Aid for Scientific Research on Innovative Areas: Prediction and Decision Making (24120527).

The authors are with the Neural Computation Unit, Okinawa Institute of Science and Technology Graduate University, 1919-1 Tancha, Onna-son, Okinawa 904-0495, Japan {uchibe,doya}@oist.jp

Recently, a new framework of linearly solvable Markov decision process (LMDP) has been proposed, in which a nonlinear Hamilton-Jacobi-Bellman (HJB) equation for continuous systems or Bellman equation for discrete systems is converted into a linear equation under certain assumptions on the action cost and the effect action on the state dynamics [6], [7]. In this approach, an exponentially transformed state value function is defined as a desirability function and it is derived from the linearized Bellman's equation by solving an eigenvalue problem [8] or an eigenfunction problem [9], [10]. One of the benefits is its compositionality. Linearity of the Bellman equation enables deriving an optimal policy for a composite task from previously learned optimal policies for basic tasks by linear weighting by the desirability functions [11], [12]. Although the goal of the composite task should be represented by a linear combination of those of the primitive tasks, there is no need to learn the composite task from scratch. However, their methods assume the environmental dynamics is known and they have so far been tested only in simulation. In this study, we test the applicability of the compositionality theory to real robot control.

Since the environmental dynamics is often unknown in real robot applications, a model free approach is required. We proposed the method which integrates model learning with the LMDP framework [13], but we found that the learned desirability function is biased if the estimated model is not accurate. To overcome this problem, this paper proposes a model-free reinforcement learning of the desirability function and it is combined with the model learning. The proposed learning algorithm is based on the least-squares reinforcement learning algorithm [14], [15] and it can be regarded as the learning method with state-dependent discount factor. In addition, the composite policy derived from the compositionality theory is used as an initial policy to learn the corresponding composite task because the assumptions are sometimes violated.

We test the proposed method in the navigation task using the quadruped robot called the Spring Dog. The goal of the primitive task is to approach the position specified by the landmark while the desired position of the composite task is represented by the set of positions used in the primitive tasks. We compare the following four methods: (1) composite optimal policy derived from the compositionality theory, (2) optimal policy with additional learning from the composite policy, (3) optimal policy learned from scratch, and (4) weighted sum of policies designed in the value function space. Although the composite policy is a weighted sum of primitive policies designed in the desirability function space, our experimental results show that the performance of the proposed method is comparable to that of the optimal policy learned from scratch. Furthermore, it is shown that the additional learning from the composite policy learns faster than learning from scratch while the composite policy designed in the value function space is not a good policy. It suggests that the compositionality is very useful in the real robot experiment and model free learning is promising.

II. Compositionality theory

To begin with, we introduce the basics of the compositionality theory of the LMDP [11], [12] briefly. The basic idea is to apply the principle of superposition of linear differential equations subject to some boundary conditions.

A. Basic idea

Suppose that there exist N learned optimal policies $\boldsymbol{u}_i$ $(i = 1, \ldots, N)$ for a set of specific N **primitive tasks** shown in Fig. 1 (a). Here, the primitive task is characterized by the terminal condition g_i. It means that $\boldsymbol{u}_i$ derived from the value function v_i trained with g_i. When a new **composite task** is created from primitive tasks, our goal is to learn an optimal policy but to compute it represented by

$$\boldsymbol{u}'(\boldsymbol{x}, t) = \sum_{i=1}^{N} m_i(\boldsymbol{x}, t)\boldsymbol{u}_i(\boldsymbol{x}, t), \qquad (1)$$

where $\boldsymbol{x}$ and t denote the state and time, respectively. $m_i(\boldsymbol{x}, t)$ is a mixing weight, and the problem is how we design or optimize it without additional learning. Since the value function is the solution of the nonlinear HJB equation, a linear sum of value functions is not optimal even if the terminal condition of the composite task is given by the sum of that of primitive tasks.

For example, Doya et al. propose a multiple module based reinforcement learning [16] in which the mixing weight is computed from the prediction error of the primitive module. Although their method does not require additional learning for mixing, the combined policy is not necessarily optimal. Thomas and Barto propose a framework based on policy gradient reinforcement learning that learns constant mixing weights as well as optimal policies of primitive tasks [17]. Their method can find a local optimal policy in principle, but the learning mixing weights is always necessary even if the goal of the composite task is slightly modified from one of primitive tasks.

Recently, the compositionality theory is applied to this problem by Todorov [11], in which the optimal mixing weights are designed without additional learning for the composite task in the framework of linearly solvable Markov decision process (LMDP). Under some conditions, the nonlinear HJB equation is transformed into the linear differential equation with respect to a so-called desirability function z, shown in Fig. 1 (b). This linearity allows us to apply superposition of desirability functions. In other words, the weighted sum of the desirability functions is the solution of linearized HJB equation with the weighted sum of terminal conditions.

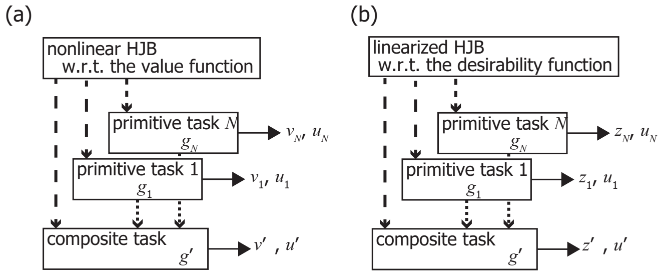

Fig. 1. Basic idea of compositionality. (a) value function. v: value function, $\boldsymbol{u}$: optimal policy, g: boundary condition. (b) desirability function. z: desirability function. See the main text.

B. Markov decision process

We give a brief introduction of Markov Decision Process (MDP) for a continuous domain [1], [18]. Let $\mathcal{X} \subseteq \mathbb{R}^{N_{\boldsymbol{x}}}$ and $\mathcal{U} \subseteq \mathbb{R}^{N_{\boldsymbol{u}}}$ be the continuous state and continuous action spaces, where $N_{\boldsymbol{x}}$ and $N_{\boldsymbol{u}}$ are the dimensionality of the spaces, respectively. At time t, the robot observes the environmental current state $\boldsymbol{x}(t) \in \mathcal{X}$ and executes action $\boldsymbol{u}(t) \in \mathcal{U}$. Consequently, the environment makes a state transition according to the following continuous-time stochastic differential equation,

$$d\boldsymbol{x} = \boldsymbol{a}(\boldsymbol{x})dt + \boldsymbol{B}(\boldsymbol{x})(\boldsymbol{u}dt + \sigma d\boldsymbol{\omega}), \qquad (2)$$

where $\boldsymbol{\omega} \in \mathbb{R}^{N_{\boldsymbol{u}}}$ and σ denote Brownian noise and a scaling parameter for the noise, respectively. $\boldsymbol{a}(\boldsymbol{x})$ describes the passive dynamics of the system while $\boldsymbol{B}(\boldsymbol{x})$ represents the input-gain matrix. Note that Eq. (2) is generally nonlinear with respect to the state $\boldsymbol{x}$ but linear with respect to the action $\boldsymbol{u}$.

It is convenient to represent Eq. (2) in discrete time. By discretizing the time axis with step h, we obtain the following transition probability,

$$p^{\boldsymbol{u}_k}(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_{k+1} \mid \boldsymbol{\mu}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{x}_k, h\boldsymbol{\Sigma}(\boldsymbol{x})), \qquad (3)$$

where $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and

$$\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u}) = h(\boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}), \qquad (4)$$

$$\boldsymbol{\Sigma}(\boldsymbol{x}) = \sigma^2 \boldsymbol{B}(\boldsymbol{x})^\top \boldsymbol{B}(\boldsymbol{x}), \qquad (5)$$

where $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u})$ can be regarded as a deterministic state transition function. Note that $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ represent the state and action at time step k, and they correspond to $\boldsymbol{x}(hk)$ and $\boldsymbol{u}(hk)$ in continuous time, respectively. It should be noted that a state transition probability is defined as an uncontrolled probability when no control is applied ($\boldsymbol{u} = \boldsymbol{0}$), and otherwise, it is called a controlled probability.

A control policy or controller $\boldsymbol{u} = \pi(\boldsymbol{x})$ is defined as a mapping from the state $\boldsymbol{x}$ to the action $\boldsymbol{u}$. The goal of the i-th primitive task is formulated as a finite horizon problem because the boundary conditions of a differential equation play an important role. The expected cumulative cost, which is known as a cost-to-go or value function, is given by

$$v_i^\pi(\boldsymbol{x}, t) = \mathbb{E}\left[\int_t^{T_g} c(\boldsymbol{x}(\tau), \pi(\boldsymbol{x}(\tau)))d\tau + g_i(\boldsymbol{x}(T_g))\right],$$

where $c(\boldsymbol{x}, \boldsymbol{u})$ and $g_i(\boldsymbol{x})$ respectively denote the immediate and terminal cost. T_g is a given fixed evaluation time. It should be noted that $c(\boldsymbol{x}, \boldsymbol{u})$ and T_g are the same among all the primitive tasks, and therefore the primitive task is characterized by g_i. We shall return to this point later. The optimal value function is the minimal expected cumulative cost defined by

$$v_i^*(\boldsymbol{x}, t) = \min_\pi v_i^\pi(\boldsymbol{x}, t).$$

It is known that the optimal value function satisfies the following Hamilton-Jacobi-Bellman's (HJB) equation [6], [18]

$$-\frac{\partial v_i^*(\boldsymbol{x}, t)}{\partial t} = \min_{\boldsymbol{u}} \left(c(\boldsymbol{x}, \boldsymbol{u}) + \mathcal{L}^{(\boldsymbol{u})}[v_i] \right), \tag{6}$$

$$v_i^*(\boldsymbol{x}, T_g) = g_i(\boldsymbol{x}), \tag{7}$$

where $\mathcal{L}^{(\boldsymbol{u})}$ is the second-order linear differential operator defined by

$$\mathcal{L}^{(\boldsymbol{u})}[v_i] = (\boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u})^\top \frac{\partial v_i}{\partial \boldsymbol{x}} + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{B}\boldsymbol{B}^\top \frac{\partial^2 v_i}{\partial \boldsymbol{x}^2} \right).$$

Since Eq. (6) is nonlinear with respect to the value function, it is difficult to solve the optimal value function in general even though the model parameters $\boldsymbol{a}(\boldsymbol{x})$ and $\boldsymbol{B}(\boldsymbol{x})$ are known in advance.

C. Linearly solvable MDP

Next, we show how the nonlinear HJB equation (6) can be made linear. Suppose that the cost function is given by

$$c(\boldsymbol{x}, \boldsymbol{u}) = q(\boldsymbol{x}) + \frac{1}{2\sigma^2}\boldsymbol{u}^\top \boldsymbol{u},$$

where $q(\boldsymbol{x})$ denotes a non-negative state dependent cost function. It should be noted that $c(\boldsymbol{x}, \boldsymbol{u})$ is nonlinear with respect to $\boldsymbol{x}$ but quadratic with respect to $\boldsymbol{u}$. In this case, it is possible to minimize the right hand side of Eq. (6) with respect to $\boldsymbol{u}$ analytically, and Eq. (6) becomes linear as follows:

$$-\frac{\partial z_i(\boldsymbol{x}, t)}{\partial t} = \mathcal{L}^{(\boldsymbol{0})}[z_i] - q(\boldsymbol{x}) z_i(\boldsymbol{x}, t), \tag{8}$$

$$z_i(\boldsymbol{x}, T_g) = \exp(-g_i(\boldsymbol{x})), \tag{9}$$

where $z_i(\boldsymbol{x})$ is the desirability function transformed by

$$z_i(\boldsymbol{x}, t) = \exp(-v_i^*(\boldsymbol{x}, t)). \tag{10}$$

Eq. (8) plays an important role for compositionality because superposition of the desirability functions is considered using this equation. Therefore, the immediate cost function must be shared not to change the equation among primitive tasks. Hereafter Eq. (8) is called the linearized HJB equation and its counterpart in discrete time is the linearized Bellman equation given by

$$z_i(\boldsymbol{x}, t) = \exp(-hq(\boldsymbol{x}))\mathcal{G}[z_i](\boldsymbol{x}, t) \tag{11}$$

$$z_i(\boldsymbol{x}, T_g) = \exp(-g_i(\boldsymbol{x}_g)). \tag{12}$$

The operator $\mathcal{G}$ shown on the right hand side of the linearized Bellman's equation (11) is the integral operator given by

$$\mathcal{G}[\phi](\boldsymbol{x}) = \int p^{\boldsymbol{0}}(\boldsymbol{x}' \mid \boldsymbol{x})\phi(\boldsymbol{x}')d\boldsymbol{x}',$$

where $p^{\boldsymbol{0}}$ is the uncontrolled probability when no control is applied ($\boldsymbol{u} = \boldsymbol{0}$) in Eq. (3). See [6] for more details. It should be noted that Eqs (8) and (11) are always satisfied by the trivial solution ($z_i(\boldsymbol{x}) \equiv 0$ for all $\boldsymbol{x}$) if no boundary conditions (9) and (12) are taken into account.

D. Control policy

In the LMDP framework, the optimal control policy of the i-th primitive task is given by

$$p^{\boldsymbol{u}^*}(\boldsymbol{x}' \mid \boldsymbol{x}) = \frac{p^0(\boldsymbol{x} \mid \boldsymbol{x})z(\boldsymbol{x}, t)}{\mathcal{G}[z](\boldsymbol{x}, t)}.$$

Specifically, if the dynamics are represented in the form of the stochastic differential equation (2), then the optimal control policy is represented by

$$\boldsymbol{u}_i^*(\boldsymbol{x}, t) = \sigma^2 \boldsymbol{B}(\boldsymbol{x})^\top \frac{\partial \ln z_i(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}. \tag{13}$$

We assume that the terminal cost of the composite task $g'(\boldsymbol{x})$ is given by the weighted sum

$$\exp\left(-g'(\boldsymbol{x})\right) = \sum_{i=1}^N w_i \exp\left(-g_i(\boldsymbol{x})\right) \tag{14}$$

in the nonlinear transformed space, where w_i is a mixing weight of terminal conditions. If the set of terminal costs $\{g_i(\boldsymbol{x})\}_{i=1}^N$ is sufficiently rich, we can represent an arbitrary cost. If a desired $g'(\boldsymbol{x})$ is given, $\{w_i\}_{i=1}^N$ can be optimized by the least-squares method. Because of linearity, the desirability function of the composite task is also given by

$$z'(\boldsymbol{x}, t) = \sum_{i=1}^N w_i z_i(\boldsymbol{x}, t). \tag{15}$$

As a result, the optimal policy of the composite task is derived as follows:

$$\boldsymbol{u}'^*(\boldsymbol{x}, t) = \sigma^2 \boldsymbol{B}(\boldsymbol{x})^\top \frac{\partial \ln z'(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} = \sum_{i=1}^N m_i(\boldsymbol{x}, t)\boldsymbol{u}_i^*(\boldsymbol{x}, t), \tag{16}$$

where the mixing weight of the policy is defined by

$$m_i(\boldsymbol{x}, t) = \frac{w_i z_i(\boldsymbol{x}, t)}{\sum_j w_j z_j(\boldsymbol{x}, t)}. \tag{17}$$

We call Eq. (16) the composite optimal policy designed in the desirability function space.

We can also create a composite policy designed in the value function space if the value function is approximated by

$$v'(\boldsymbol{x}, t) = \sum_{i=1}^N w_i v_i(\boldsymbol{x}, t). \tag{18}$$

It is possible to derive the policy from v', but it should be noted that linearity does not hold in the value function space.

III. LEARNING PARAMETERS

This section describes how the optimal policy (13) of the primitive task is learned from samples. Specifically, we introduce a model free learning of the desirability function $z(\boldsymbol{x}, t)$ and a simple learning of the input-gain matrix $\boldsymbol{B}$.

A. *Model free learning of the desirability*

In our previous work [13], we proposed a model-based framework in which the desirability function was computed from the estimated dynamics. In this paper, we propose a novel model-free framework of learning the desirability function based on the least squares reinforcement learning algorithms [14], [15]. We assume that the desirability function is approximated by the following linear approximator

$$z_i(\boldsymbol{x}, t; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{j=1}^{N_z} w_{ij}\phi(\boldsymbol{x}, t; \boldsymbol{\theta}_j) = \boldsymbol{w}_i^\top \boldsymbol{\phi}(\boldsymbol{x}, t; \boldsymbol{\theta}), \tag{19}$$

where w_{ij} and $\boldsymbol{w}_i$ are a learning weight and its vector representation of the i-th primitive task, $\phi(\boldsymbol{x}, t; \boldsymbol{\theta}_j)$ is a basis function parameterized by $\boldsymbol{\theta}_j$, and $\boldsymbol{\phi}(\boldsymbol{x}, t; \boldsymbol{\theta})$ is the vector consisting of basis functions $[\phi(\boldsymbol{x}, t; \theta_1), \ldots, \phi(\boldsymbol{x}, t; \theta_{N_z})]^\top$.

Let us denote the robot's experience in discrete time by $d = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N_{\text{step}}})$ where $hN_{\text{step}} = T_g$. From the linearized Bellman's equation (11) and the boundary condition (12), the temporal difference error of state transition to non-terminal and terminal states are given by

$$\begin{aligned}\delta_{i,k+1} &= (\boldsymbol{\phi}_k - \exp(-q(\boldsymbol{x}_k))\boldsymbol{\phi}_{k+1})^\top \boldsymbol{w}_i,\\ \delta_{i,N_{\text{step}}} &= \exp(-g_i(\boldsymbol{x}_{N_{\text{steps}}})) - \boldsymbol{\phi}_{N_{\text{steps}}}^\top \boldsymbol{w}_i,\end{aligned}$$

where $\boldsymbol{\phi}_k = \boldsymbol{\phi}(\boldsymbol{x}_k, hk)$. According to the LSTD [14], the update of $\boldsymbol{w}_i$ at the end of episode is calculated by

$$\Delta\boldsymbol{w}_i = \sum_{k=0}^{N_{\text{steps}}-1} \boldsymbol{\phi}_k \delta_{i,k+1} = \boldsymbol{A}\boldsymbol{w}_i - \boldsymbol{b}_i,$$

where $\boldsymbol{\phi}_{N_{\text{steps}}+1} = \boldsymbol{0}$ and $\boldsymbol{A}$ and $\boldsymbol{b}_i$ are computed recursively by

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + \sum_{k=0}^{N_{\text{step}}} \boldsymbol{\phi}_k \left(\boldsymbol{\phi}_k - \exp(-q(\boldsymbol{x}_k))\boldsymbol{\phi}_{k+1}\right)^\top, \tag{20}$$

$$\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + \boldsymbol{\phi}_{N_{\text{step}}} \exp(-g_i(\boldsymbol{x}_{N_{\text{step}}})). \tag{21}$$

Instead of using a stochastic gradient method with $\delta\boldsymbol{w}_i$, the least-squares method solves $\boldsymbol{w}_i = \boldsymbol{A}^{-1}\boldsymbol{b}_i$ directly. As compared with the standard LSTD, $\exp(-q(\boldsymbol{x}_k))$ in Eq. (20) corresponds to the discount factor and it can be regarded as a state-dependent discount factor. In addition $\boldsymbol{A}$ is independent of the primitive tasks while $\boldsymbol{b}_i$ is dependent on them and it is updated at the end of episode.

B. *Learning model parameters*

In the LMDP framework, the system dynamics (2) is assumed to be known in advance. When it is unknown, estimating the dynamics is required from samples collected by the passive dynamics. Although we propose the model-free learning of a desirability function in Section III-A, the Jacobian matrix $\boldsymbol{B}(\boldsymbol{x})$ is required to compute the optimal policy (13). This section describes how to learn $\boldsymbol{B}(\boldsymbol{x})$ from samples.

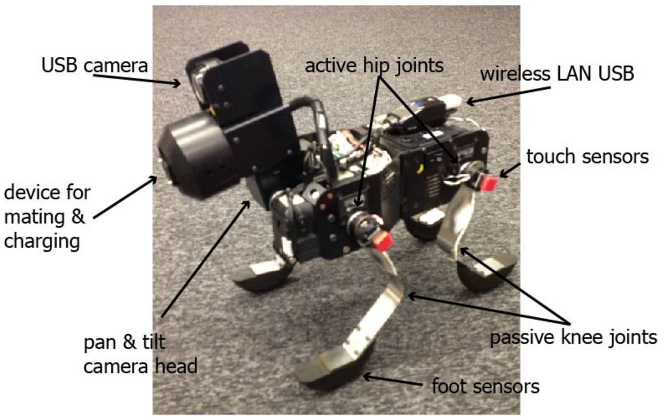

Fig. 2. Quadruped robot, Spring Dog

Many methods exist which can estimate the system dynamics [19], [20], and we adopt a simple least squares method to estimate $\boldsymbol{\mu}(\boldsymbol{x})$ with basis functions. Specifically, we estimate a deterministic state transition (4). It should be noted that the scale parameter of noise σ is generally unknown, but it is determined by the experimenters here since it can be regarded as the parameter that controls exploration of the environment.

Let us suppose that the deterministic state transition $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u})$ is approximated by the linear function with N_φ basis functions $\varphi_i(\boldsymbol{x}, \boldsymbol{u})$,

$$\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{W}) = \boldsymbol{W}^\top \boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{u}). \tag{22}$$

where $\boldsymbol{W}$ is a weight matrix and $\boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{u})$ is a vector consisting of basis functions. Suppose that the training samples $\{\boldsymbol{x}_1, \boldsymbol{u}_1, \ldots, \boldsymbol{x}_{N_s}, \boldsymbol{u}_{N_s}, \boldsymbol{x}_{N_s+1}\}$ are obtained by the passive dynamics. The objective function of model learning is given by the following sum-of-squares error function,

$$E = \frac{1}{2}\sum_{k=1} \left\|\Delta\boldsymbol{x}_k - \boldsymbol{W}^\top \boldsymbol{\varphi}(\boldsymbol{x}_k, \boldsymbol{u}_k)\right\|^2,$$

where $\Delta\boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$. Setting $\partial E/\partial\boldsymbol{W} = \boldsymbol{0}$ yields

$$\boldsymbol{W} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \Delta\boldsymbol{X},$$

where $\Delta\boldsymbol{X}$ is the matrix whose a row vector consisted of state transition in each sample $\Delta\boldsymbol{x}_k$ and $\boldsymbol{\Phi}$ is also the matrix whose a column vector consisted of the basis functions in each sample $\boldsymbol{\varphi}(\boldsymbol{x}_k, \boldsymbol{u}_k)$. The detail is as follow,

$$\begin{aligned}\Delta\boldsymbol{X} &= \begin{bmatrix}\Delta\boldsymbol{x}_1 & \cdots & \Delta\boldsymbol{x}_{N_s}\end{bmatrix}^\top,\\ \boldsymbol{\Phi} &= \begin{bmatrix}\boldsymbol{\varphi}(\boldsymbol{x}_1, \boldsymbol{u}_1) & \cdots & \boldsymbol{\varphi}(\boldsymbol{x}_{N_s}, \boldsymbol{u}_{N_s})\end{bmatrix}.\end{aligned}$$

See our previous study of model learning [13] for more details.

IV. Experiments

A. Locomotion Task

We conduct a control task of a quadruped robot called the "Spring Dog" to evaluate the proposed model free method and the compositionality of the LMDP in the real environment. Fig. 2 shows a hardware of the Spring Dog. The Spring Dog is endowed with a variety of sensory inputs, including a USB camera, touch sensors, foot sole sensors, gyros, and accelerometer. The Spring Dog is a quadruped robot and each leg has one active hip joint and one passive knee joint while its head has two degrees of freedom (DOFs): pan and tilt joints. As a whole the Spring Dog has six DoFs. The programs are coded in C++ on top of a standard linux operating system.

Fig. 3 shows the experimental field, in which three landmarks with LED are located in front of the Spring Dog. The task of the robot is to walk from the fixed starting position to the goal position determined by the landmarks. Specifically, the Spring Dog learns to approach one of landmarks in the primitive task. Since there exist three landmarks, three primitive tasks should be optimized by the LMDP framework. Blue, green, and red curves shown in Fig. 4 represent the exponentially transformed terminal cost function $\exp(-g_i(\boldsymbol{x}))$ of primitive tasks, projected to the x position in the global coordinate system and it is maximum when the Spring Dog arrives at the specified landmark while $q(\boldsymbol{x}) = 0.95$ for all $\boldsymbol{x}$. The composite task is go to the position between the blue and green landmarks and its corresponding terminal cost function is shown in black in Fig. 4. This terminal cost is created by setting $w_1 = w_2 = 0.86, w_3 = 0$ in Eq. (14). It should be noted that the shape of the terminal cost of the composite task (black line) is different from those of the component tasks because it is unable to represent a complicated function by a weighted sum of three functions as shown in Eq. (14).

Fig. 5 shows the control architecture. The state vector $\boldsymbol{x}$ consists of six components,

$$\boldsymbol{x} = \left[\boldsymbol{\theta}_{\text{hip}}^\top, \boldsymbol{p}^\top\right]^\top,$$

where $\boldsymbol{\theta}_{\text{hip}} \in \mathbb{R}^4$ is the current angles of the hip joints, and $\boldsymbol{p} = [x, y]^\top$ represents the position of the Spring Dog in the global coordinate system shown in Fig. 3. The camera head module computes $\boldsymbol{p}$ from captured images of the environment and generates the desired angle of the pan and tilt joints. On the other hand, the desired angle of the hip joint $\boldsymbol{\theta}_{\text{hip},d} \in \mathbb{R}^4$ is determined by

$$\boldsymbol{\theta}_{\text{hip},d} = \boldsymbol{u} + \boldsymbol{\theta}_{\text{CPG}}, \tag{23}$$

where $\boldsymbol{u}$ and $\boldsymbol{\theta}_{\text{CPG}} \in \mathbb{R}^4$ denote the action from the reinforcement learning module and the output from the CPG, respectively. The role of the CPG is to generate a stable movement as a base controller and it helps the robot to collect good experience for learning. The equations of motion with the CPG control is interpreted as the uncontrolled probability. The modified Hopf oscillator [21] is applied in this implementation.

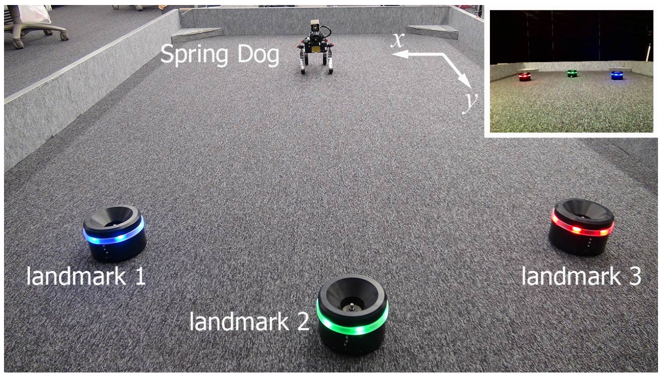

Fig. 3. The Spring Dog and three landmarks in the experimental field. The top right image represents the view from the Spring Dog. The global coordinate system is also shown in this figure.

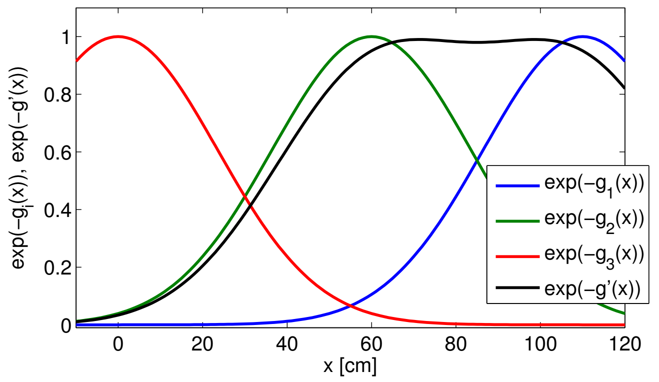

Fig. 4. Three terminal cost functions of primitive tasks (blue, green, and red) and that of the composite task (black).

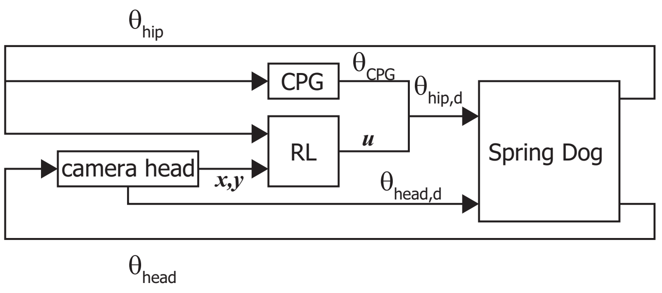

Fig. 5. Control architecture.

In order to represent the desirability function, we adopt an unnormalized Gaussian function given by

$$\phi(\boldsymbol{x}, t; \boldsymbol{\theta}_i) = \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{c}_i(t)\right)^\top \boldsymbol{S}_i \left(\boldsymbol{x} - \boldsymbol{c}_i(t)\right)\right),$$

where $\boldsymbol{\theta}_i = \{\boldsymbol{c}_i, \boldsymbol{S}_i\}$ and $\boldsymbol{c}_i$ and $\boldsymbol{S}_i$ denote a center position and a precision matrix of the i-th basis function, respectively. On the other hand, the linear basis function used in our previous study [13] to approximate $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u})$. In this case, $\boldsymbol{B}(\boldsymbol{x})$ can be represented a constant matrix.

B. Experimental results

It took about three hours for the Spring Dog to learn appropriate behavior in each primitive task. To see the performance

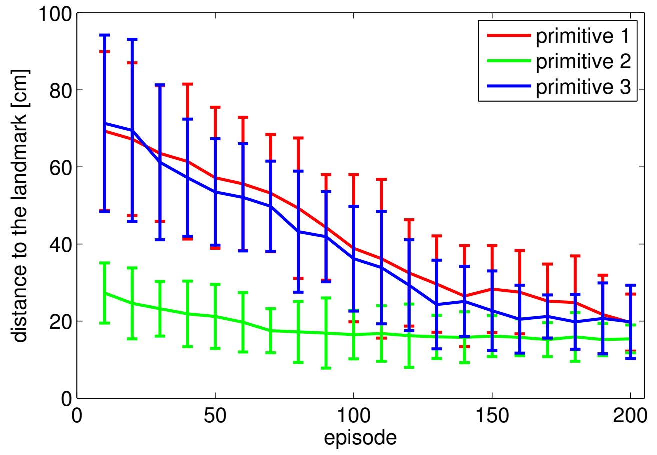

Fig. 6. Learning curves of primitive tasks. The error bars represent the standard deviation of 10 experimental runs.

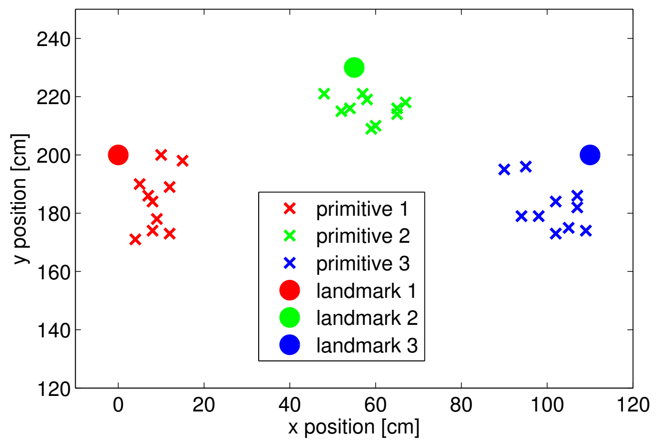

Fig. 7. Positions at the end of episodes generated by one learned policy of the primitive tasks in a typical experiment.

of the behaviors during learning we measured the distance between the landmark and the position of the Spring Dog at the end of episodes. Fig. 6 shows the experimental results of learned primitive tasks, in which error bars represent the standard deviation of 10 runs. Since the landmark 2 (blue) was located in front of the Spring Dog shown in Fig. 3, the basic controller $\boldsymbol{\theta}_{\mathrm{CPG}}$ in Eq. (23) was close to the optimal controller, and the performance at the early stage of learning was better than those in other primitive tasks. Fig. 7 shows the final positions controlled by the learned primitive behaviors. Each circle represents the position of the landmark. To evaluate the learned primitive behaviors, we conducted 10 evaluation runs for each primitive task. The colored crosses show the final position of the Spring Dog at the end of episode in a typical experiment. It is confirmed that the proposed model free learning explained in Section III-A could learn good behaviors even though the dynamics of the Spring Dog was not well approximated by the form of Eq.(2).

Next, the composite optimal policy was designed from three primitive policies using Eq. (16). To evaluate the compositionality theory in the real environment, we compared the following policies:

1) **composite policy** $\boldsymbol{u}'_z$: the composite policy computed by Eq. (16).
2) **composite policy with additional learning** $\boldsymbol{u}'^*_z$: the policy learned with the composite policy $\boldsymbol{u}'_z$. That is, the base controller $\boldsymbol{\theta}_{\mathrm{CPG}}$ in Eq. (23) was replaced by the composite controller $\boldsymbol{u}'_z$.
3) **optimal policy learned from scratch** $\boldsymbol{u}^*_z$: the policy learned with the base controller $\boldsymbol{\theta}_{\mathrm{CPG}}$.
4) **value-weighted policy** $\boldsymbol{u}'_v$: the composite policy designed in the value function space. The value function is created by a weighted sum of the value functions of primitive tasks by Eq. (18) and the weights were optimized by an exhaustive search.

Note that the same terminal condition g' was used to train $\boldsymbol{u}'^*_z$ and $\boldsymbol{u}^*_z$ in the composite task. Theoretically speaking, the composite policy $\boldsymbol{u}'_z$ should be optimal with respect to the terminal cost function shown in Fig. 4 and it is identical to the optimal policy $\boldsymbol{u}^*_z$. However, it was not necessarily true because there exist approximation errors in the desirability function in practice and the dynamics of the Spring Dog was not completely expressed by Eq.(2).

Fig. 8 compares the learning curves of the composite policy with additional learning $\boldsymbol{u}'_z$ and the optimal policy learned from scratch $\boldsymbol{u}^*$. As we expected, the composite policy $\boldsymbol{u}'_z$ was not optimal in the composite task, and the performance of $\boldsymbol{u}'^*_z$ was improved by additional learning. The learning curve of $\boldsymbol{u}^*_z$ was similar to those of the primitive tasks shown in Fig.6. The two policies eventually obtained the same performance, but the composite policy with additional learning learned much faster than the optimal policy learned from scratch. It suggests that the composite policy $\boldsymbol{u}'_z$ was a good approximation of the optimal policy as compared with the pre-defined base controller $\boldsymbol{\theta}_{\mathrm{CPG}}$.

Fig. 9 compares the final positions in the composite task. Cyan crosses represent the positions generated by $\boldsymbol{u}^*_z$ and they can be regarded as the results of the optimal policy in the composite task. The final positions generated by $\boldsymbol{u}'_z$ are shown as black circles and its distribution was slightly different from that of the positions generated by $\boldsymbol{u}^*_z$. The final positions generated by $\boldsymbol{u}'^*_z$ are shown as black crosses in Fig. 9 and they were distributed in the similar areas of $\boldsymbol{u}^*_z$. On the other hand, the final positions obtained by the value-weighted policy $\boldsymbol{u}'_v$ were quite different from those of $\boldsymbol{u}^*_z$.

V. Conclusion

This paper evaluates how the learned optimal policies are combined to generate a new optimal policy based on the compositionality theory of the LMDP framework. One of contributions of this paper is to propose the model free learning algorithm based on the least squares reinforcement learning method for approximating the desirability function. Although the composite policy is slightly different from the optimal one due to the violation of the assumptions, our experimental results suggest that the composite policy is a

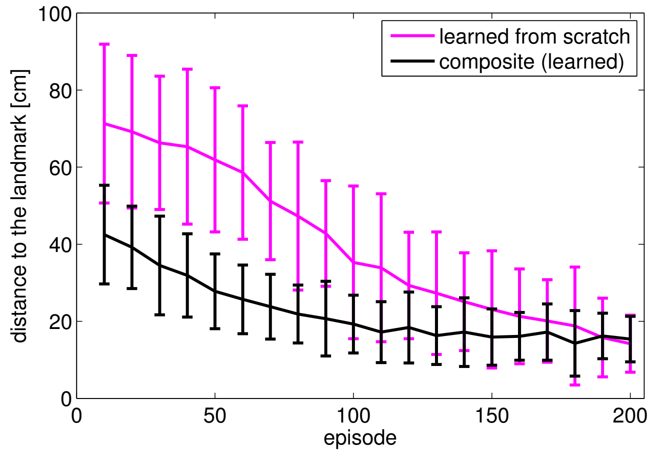

Fig. 8. Learning curves of composite tasks. The black line shows the result of $\boldsymbol{u}'^*$ when the composite controller was used as the base controller while the magenta shows the result of $\boldsymbol{u}^*$ when the base controller used in the primitive tasks were used in Eq.(23). The error bars represent the standard deviation of 10 experimental runs.

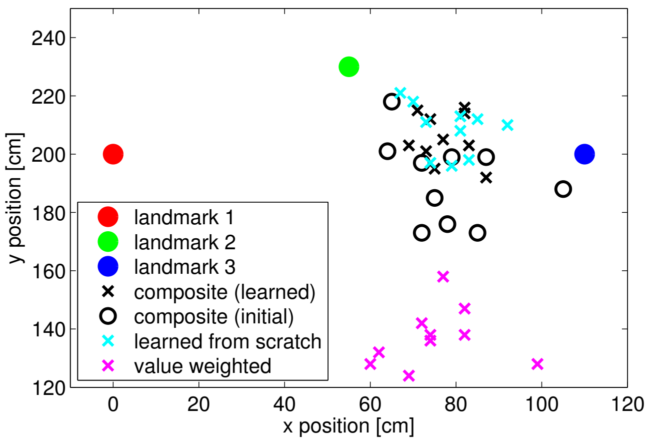

Fig. 9. Comparison among the composite policy, the optimal policy learned from the composite cost function, and the composite policy designed in the value function space.

good approximation of the optimal policy and the optimal policy can be learned much faster than the case without using the composite policy. To our best knowledge, this is the first time that the compositionality in LMDP is demonstrated to be successful in real robot experiments.

Historically, the basis idea of linearization of the HJB equation using logarithmic transformation has been shown in the book written by Fleming and Soner and its connection to risk sensitive control has been discussed in the field of control theory [18]. Their study has been receiving attention recently in the field of robotics and machine learning fields [22] because there exist a number of interesting properties in the linearized Bellman equation [6]. This paper focused on the desirability function approach, but the compositionality in the path integral approach [23], [24] is also promising. In the path integral approach, the linearized Bellman is computed along paths starting from given initial states using sampling methods. The path integral approach has been successfully applied to learning of stochastic policies for robots with large degrees of freedom [25], [26], [27], and it is best suited for optimization around stereotyped motion trajectories. However, an additional learning is needed when a new initial state or a new goal state is given. Two approaches are closely related and new theoretical findings are reported [22], but there are some differences in practice.

Further empirical and theoretical analysis is required to show the efficiency of the proposed model-free learning of the desirability function. In particular, we are interested in the relation to the state-dependent discount factor in MDP [28], [29]. Next, we plan to extend the proposed method to construct the set of primitive tasks incrementally. Since the terminal cost is approximated by Eq.(14), the approximation error can be used as a criterion to add a new primitive task.

References

[1] E. Todorov. Optimal control theory. In K. Doya et al., editors, *Bayesian Brain: Probabilistic Approaches to Neural Coding*, chapter 12, pages 269–298. MIT Press, 2006.

[2] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press/Bradford Books, 1998.

[3] Y. Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, June 2009.

[4] N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, March 2011.

[5] B. B. Doll, D. A. Simon, and N. D. Daw. The ubiquity of model-based reinforcement learning. *Current Opinion in Neurobiology*, 22(6):1075–1081, September 2012.

[6] E. Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences of the United States of America*, 106(28):11478–83, July 2009.

[7] K. Doya. How can we learn efficiently to act optimally and flexibly? *Proceedings of the National Academy of Sciences of the United States of America*, 106(28):11429–30, July 2009.

[8] E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems 19*, pages 1369–1376. 2007.

[9] E. Todorov. Eigenfunction approximation methods for linearly-solvable optimal control problems. In *Proc. of the 2nd IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 161–168, 2009.

[10] M. Zhong and E. Todorov. Aggregation Methods for Lineary-solvable Markov Decision Process. In *Proc. of the World Congress of the International Federation of Automatic Control*, 2011.

[11] E. Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems 22*, pages 1856–1864. 2009.

[12] M. da Silva, F. Durand, and J. Popović. Linear Bellman combination for control of character animation. *ACM Transactions on Graphics*, 28(3), July 2009.

[13] K. Kinjo, E. Uchibe, and K. Doya. Evaluation of linearly solvable Markov decision process with dynamic model learning in a mobile robot navigation task. *Frontiers in Neurorobotics*, 7(7), 2013.

[14] J. A. Boyan. Technical Update: Least-Squares Temporal Difference Learning. *Machine Learning*, 49(2/3):233–246, 2002.

[15] M. G. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[16] K. Doya, K. Samejima, K. Katagiri, and M. Kawato. Multiple model-based reinforcement learning. *Neural Computation*, 14(6):1347–1369, June 2002.

[17] P. S. Thomas and A. G. Barto. Motor primitive discovery. In *Proc. of IEEE International Conference on Development and Learning and Epigenetic Robotics*. IEEE, November 2012.

[18] W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions*. Springer, 2 edition, 2006.

[19] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–40, November 2011.

[20] O. Sigaud, C. Salaün, and V. Padois. On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, December 2011.

[21] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 819–824. IEEE, May 2008.

[22] E. A. Theodorou and E. Todorov. Relative entropy and free energy dualities: Connections to Path Integral and KL control. In *Proc. of the 51st IEEE Conference on Decision and Control*, pages 1466–1473. IEEE, December 2012.

[23] H. Kappen. Linear Theory for Control of Nonlinear Stochastic Systems. *Physical Review Letters*, 95(20), November 2005.

[24] H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11011–P11011, November 2005.

[25] E. Theodorou, J. Buchli, and S. Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

[26] F. Stulp and O. Sigaud. Path Integral Policy Improvement with Covariance Matrix Adaptation. In *Proc. of the 10th European Workshop on Reinforcement Learning (EWRL 2012)*, 2012.

[27] N. Sugimoto and J. Morimoto. Phase-dependent trajectory optimization for periodic movement using path integral reinforcement learning. In *Proc. of the 21st Annual Conference of the Japanese Neural Network Society*, 2011.

[28] N. Yoshida, E. Uchibe, and K. Doya. Reinforcement learning with state-dependent discount factor. In *Proc. of IEEE Joint International Conference on Development and Learning and Epigenetic Robotics*, pages 1–6. IEEE, August 2013.

[29] Q. Wei and X. Guo. Markov decision processes with state-dependent discount factors and unbounded rewards/costs. *Operations Research Letters*, 39(5):369–374, July 2011.