2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Unmanned Aerial Vehicles (UAV) Heading Optimal Tracking Control Using Online Kernel-based HDP Algorithm

Fuxiao Tan, Derong Liu, Xinping Guan, and Bin Luo

***Abstract*— UAV can work in places that are dangerous, or not easy to reach for humans. However, due to active control and operating difficulties, it is still a challenge to develop fully autonomous flight in complex environments. This paper applies a novel heuristic dynamic programming for the UAV heading optimal tracking controller design, using kernel-based heuristic dynamic programming (KHDP). Kernel-based HDP is developed by integrating kernel methods and approximately linear dependence (ALD) analysis with the critic learning of HDP algorithm. Compared with conventional HDP where neural networks are widely used and their features were manually designed, the proposed algorithm can obtain better generalization capability and learning efficiency through applying the sparse kernel machine into the critic learning process of HDP algorithm. Simulation and experimental results of UAV heading optimal tracking control problems demonstrate the effectiveness of the proposed kernel-based HDP algorithm.**

I. Introduction

In recent years, with the development of embedded processor, micro-sensor technology, control theory, and Micro-Electro-Mechanical Systems (MEMS) technology widely used in military weapons and civilian products, the application and research of Unmanned Aerial Vehicles (UAV) have attracted great attention, as one kind of hight-speed and high-efficiency agent. UAV can work in the places where are dangerous, or not easy to reach by humans. UAV resembling helicopters become one of the typical unmanned aircraft research platforms, and expand the potential applications of UAV. An UAV containing weather image sensors can implement reconnaissance and surveillance on a target at close range. It can also complete a variety of complex tasks, such as the low altitude reconnaissance, monitoring, which plays an important role in the military. It also provides accurate, real-time target detection information, and implements the control on roll, pitch and rotation by adjusting the rotation speed of the motor, which has strong requirements for controlling frequency and speed.

Many foreign laboratories and universities have UAV research projects, and many research institutions successfully developed with UAV autonomous flight capabilities in a simple constraint environment. However, due to the active control and operating difficulties, it is still a challenge to develop fully autonomous flight in complex environments. Research and development on UAV have been relatively slow in the domestic.

In recent years, many international researchers have been studying UAV control problem using a variety of control algorithms[1], [2], [3], [4]. These methods can control the UAV gesture, but there exists corresponding drawbacks. The DI control theory has no strong robustness, especially under the condition of airflow disturbance and model parameters perturbation. PID and LQ control methods ignore the nonlinear factor in model, which affects the control effect due to poor model accuracy [5]. UAV itself is an unstable system, very vulnerable to the impact of the wind, which will cause the flight direction to change, or even crash.

As is well known, the ADP algorithm improves the control performance by online interaction with the system or environment." Also, ADP is not introduced. Multilayer perceptron neural networks (MLPNNs) [7], [6] were used for policy evaluation in ADP usually [8], [12], [9], [10], [11]. Dierks et al. [13] applied the neural dynamic programming technique into solving the Hamilton-Jacobi-Bellman (HJB) equation for optimal control of unknown affine nonlinear discrete-time systems with proof of convergence. Zhang et al. [14] studied MLPNN-based optimal control algorithm for a class of discrete-time nonlinear affine systems with control constraints. Wang et al. [15] utilized the MLPNNs to design a finite-horizon optimal controller for a class of discrete-time nonlinear systems with ε error bound. Nevertheless, the manual settings of critic networks and empirical design of basis functions were still depended on in the above works. Therefore, we demand to develop automatic feature representation and selection methods for the critic learning module of ADP approaches.

According to the theoretical and empirical results from statistical learning [16], [17], the structures designed by sparse kernel machines have better generalization capability than conventional MLPNNs. Consequently we apply the sparse kernel machines into the critic learning of ADP algorithm. Werbos classified ADP approach into four main schemes [18], [22], [23], [24]: heuristic dynamic programming (HDP), dual-HDP (DHP), action-dependent HDP (ADHDP), and

Fuxiao Tan is with the School of Computer and Information, Fuyang Teachers College, Fuyang, Anhui, China, and Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, Anhui University, Hefei, Anhui, China. (email: fuxiaotan@gmail.com).

Derong Liu is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. (email: derong.liu@ia.ac.cn).

Xinping Guang is with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. (email: xpguan@sjtu.edu.cn).

Bin Luo is with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. (email: luobin@ahu.edu.cn).

This work was supported by Anhui Provincial Natural Science Foundation of China (1208085MF111), the Open Research Project from SKLMCCS (20120102), and National Natural Science Foundation of China (61221003, 61290322).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

action-dependent DHP (ADDHP). Among ADP architectures, HDP is the most popular one. Inspired by [19], this paper integrate HDP algorithm with sparse kernel machines, that is kernel-based HDP algorithm. We apply the kernel-based HDP algorithm to optimal tracking controller design for UAV heading system. As well as we know, there are very few works on kernel-based HDP algorithm in the field and applying it to optimal tracking controller design for UAV heading system. Simulation and experimental results on UAV heading system demonstrate that kernel-based HDP can obtain good performance.

The rest of this paper is organized as follows. In Section 1, the problem statement is given. The Online kernel-based HDP optimal tracking controller design is presented in Section 2. The simulation results of UAV heading optimal tracking controller using kernel-based HDP algorithm are presented to show the satisfactory performance of the proposed scheme in Section 3. Finally, the conclusions are drawn in Section 4.

II. Problem Statement

This paper introduces a detailed mathematical UAV model, as is shown in (1) [20].

$$x_(k+1) = f(x_k) + g(x_k)u_{pk} = A(x_k) + Bu_{pk} \quad (1)$$

Here, the system state is determined by the heading angle θ and heading angular velocity $\dot{\theta}$, i.e. $x = [\theta; \dot{\theta}] \in \Re^2$ is system state vector. The tail total moment $u_p \in \Re$ is the only control input of the system. Parameters $A = \begin{bmatrix} 0.9993 & 0.0197 \\ -0.0657 & 0.9721 \end{bmatrix}$ and $B = \begin{bmatrix} 0.0143 \\ 1.4251 \end{bmatrix}$ are system parameter constants. Assume that the system (1) is controllable.

Optimal tracking control problem of UAV heading aims to determine optimal control law u_p^*, so as to make the UAV heading system (1) to track a desired trajectory r_k satisfying

$$r_{k+1} = \phi(r_k) \quad (2)$$

where $r_k \in \Re^2$ and $\phi(r_k) \in \Re^2$. Then, the tracking error is defined as

$$e_k = x_k - r_k \quad (3)$$

From [25], [26], the steady control corresponding to the reference trajectory r_k is defined as

$$u_{dk} = g^{-1}(r_k)(\phi(r_k) - f(r_k)) \quad (4)$$

where $g^{-1}(r_k)g(r_k) = I_m$ and I_m is an $m \times m$ identity matrix.

By denoting

$$u_k = u_{pk} - u_{dk} \quad (5)$$

and using equations (1)-(4), a new system can be obtained as follows

$$\begin{cases} e_{k+1} = f(e_k + r_k) + g(e_k + r_k)g^{-1}(r_k)(\phi(r_k - f(r_k)) \\ \qquad - \phi(r_k)) + g(e_k + r_k)u_k \\ r_{k+1} = \phi(r_k) \end{cases} \quad (6)$$

Note that in system (6), we regard $[e_k; r_k]$ and u_k as the system variables and system input, respectively.

Definition 1: The system (6) is said to be stabilizable on a compact set $\Omega \in \Re^2$, if for all random initial conditions $e_0 \in \Omega$, there exists a control sequence $u_0, u_1, \ldots, u_\infty \in \Re$, such that the final state $e_k \to 0$ as $k \to \infty$.

For infinite-horizon optimal tracking control problem, the purpose is to find the control sequence which can minimize the value function as follows:

$$Q(e_k, \underline{u}_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(e_i, u_i) \quad (7)$$

where $U(e_i, u_i) \geq 0$ is the utility function, which has the quadratic form as follows:

$$U(e_i, u_i) = e_i^T Q e_i + u_i^T R u_i$$

This value function can force both the UAV heading system state to track the desired trajectory and the tail total moment $u_p \in \Re$ to approximate to the steady value.

Definition 2: A control sequence $u_k, u_{k+1}, \ldots, u_\infty \in \Re$ is said to be admissible for a state $e_k \in \Re^n$ with respect to (7) on Ω if $u_k, u_{k+1}, \ldots, u_\infty \in \Re$ is continuous on a compact set $\Omega_u \in \Re$, $u_0 = 0$, $e_\infty = 0$ and $Q(e_k, u_k)$ is finite.

According to Bellman's optimality principle, the optimal cost function $Q^*(e_k)$ satisfies the discrete-time HJB equation

$$Q^*(e_k) = \min_{u_k} \left\{ e_k^T Q e_k + u_k^T R u_k + Q^*(e_{k+1}) \right\} \quad (8)$$

The optimal control u^* satisfies the first-order necessary condition, which is given by the gradient of the right-hand side of (8) with respect to u_k. Then,

$$u^*(e_k) = -\frac{1}{2} R^{-1} g^T(e_k + r_k) \frac{\partial Q^*(e_{k+1})}{\partial e_{k+1}} \quad (9)$$

Then, the optimal tracking control input for UAV heading system (1) can be computed by

$$u_{pk}^* = u^*(e_k) + u_{dk} = u^*(e_k) + g^{-1}(r_k)(\phi(r_k) - f(r_k)) \quad (10)$$

III. Implementation for UAV Heading Optimal Tracking Controller Using Kernel-based HDP Algorithm

A. Framework of Kernel-based HDP

A general framework of the HDP algorithm with sparse kernel machines is shown in Fig. 1. Its main components consist of a model of the system (6), a reward function, an actor, a critic module and a kernel-based feature learning module. The kernel-based feature learning module aims to implement data-driven feature representation and learning so that we can obtain better generalization performance and learning efficiency for HDP. The objective of the critic is to approximate the performance index function. In kernel-based HDP algorithm, the kernel function and its induced feature space play very important roles in the critic learning process. Because of kernel-based features which are in linear forms,

the critic can employ the RLS-TD learning algorithms. The actor receives the current tracking error state e_k and outputs the control u_k. The output of the critic is used in the actor training process to compute the policy gradients, which will be shown in the following section. Given the control u_k, the plant model estimates the next trajectory tracking error e_{k+1}. The state tracking error data are input to the critic and to the utility function. The solid lines represent signal flow, while the dashed lines are the paths for network weight updating in the critic and actor networks.

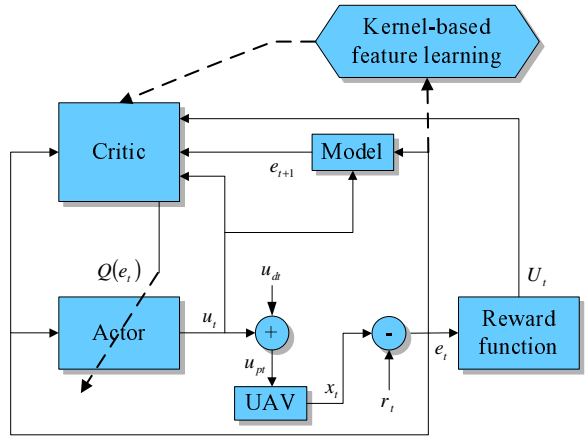


Fig. 1. Learning tracking control structure based on KHDP

There are two main procedures included in the proposed kernel-based HDP algorithm, that is, a kernel-based feature construction process and an online learning control process. The sample collection process for kernel feature construction is realized by observing the MDP running with an initially random control policy of the actor in this paper. The data samples are in the form of state transitions $\{(e_1,u_1),(e_2,u_2),\ldots,(e_n,u_n)\}$. Before the online control learning process of HDP, we need to perform the ALD-based kernel sparsification procedure offline on the data samples.

After the sample collection process, we construct the kernel-based features in a data-driven way in the ALD analysis[27]. Let $S_n = \{s_1, s_2, \cdots, s_n\}$ denote a set of data samples and ϕ be a feature mapping on S_n, which can be determined by the Mercer kernel function defined in equation (11). We can obtain a feature vector set as $\Phi_n = \{\phi(s_1), \phi(s_2), \cdots, \phi(s_n)\}$, $\phi(s_i) \in R^{m\times 1}, i = 1,2,\ldots,n$. According to the Mercer theorem [16], there exists a Hilbert space H and a mapping ϕ from S to H such that

$$k(s_i, s_j) = \langle \phi(s_i), \phi(s_j) \rangle \tag{11}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product in H. The dimension of H may be infinite and the nonlinear mapping ϕ is usually unknown, but all the computation in the feature space can still be performed if it is in the form of inner products.

To perform the ALD analysis on the feature vector set, we need to define a data dictionary as a subset of the feature vector set, denoted by D. The data dictionary D is initialized as empty and the ALD analysis is implemented by testing

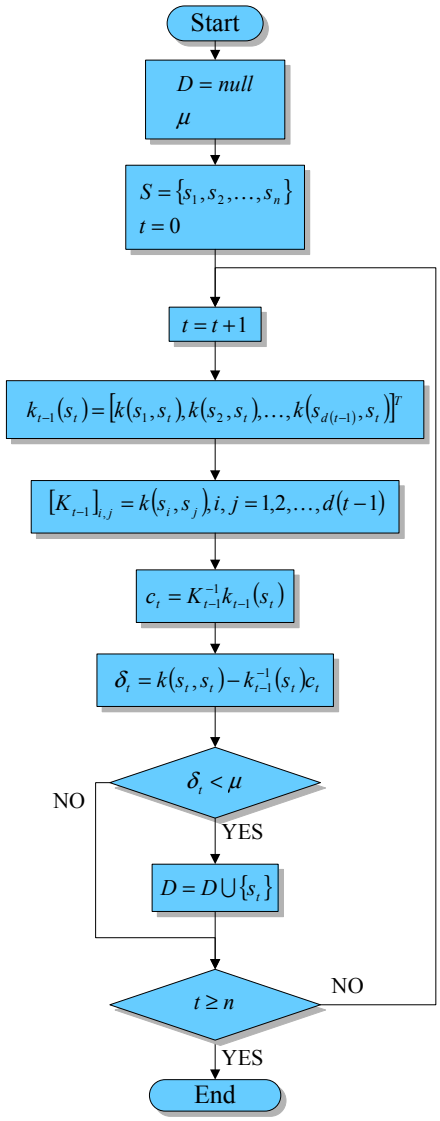


Fig. 2. The ALD analysis procedure

every feature vector in Φ_n, one at a time. The detailed steps are shown in Fig. 2, where $d(t-1)$ is the length of the data dictionary, μ is a predefined threshold. Consequently, after the ALD analysis process, all the feature vectors of the data samples in S_n can be approximately represented by linear combinations of the feature vectors in the dictionary within μ.

After the sparsification procedure, we can obtain a data dictionary D_n with less number of data sample vectors, and the approximated value function is represented as follows:

$$\tilde{Q}(s) = \sum_{j=1}^{d(n)} \alpha_j k(s, s_j) \tag{12}$$

where $d(n)$, usually much smaller than the original sample size n, is the length of the dictionary D_n. $s_j = s(e_j, u_j)$, and e_j $(j = 1,2,\ldots,d(n))$ are the elements of the data dictionary.

The critic aims at approximating the performance index functions. Hence, in the proposed kernel-based HDP, we will apply a recursive algorithm of KLSTD [28], [29]. The kernel-based HDP algorithm is implemented as follows.

B. Implementation of Kernel-based HDP Algorithm

Now we implement the kernel-based HDP algorithm using NNs and sparse kernel function. In the kernel-based HDP algorithm, there are three modules, which are model module, critic module and action module. The model module and action module are chosen as three-layer feedforward NNs. The flowchart of the proposed algorithm is shown in Fig. 3.

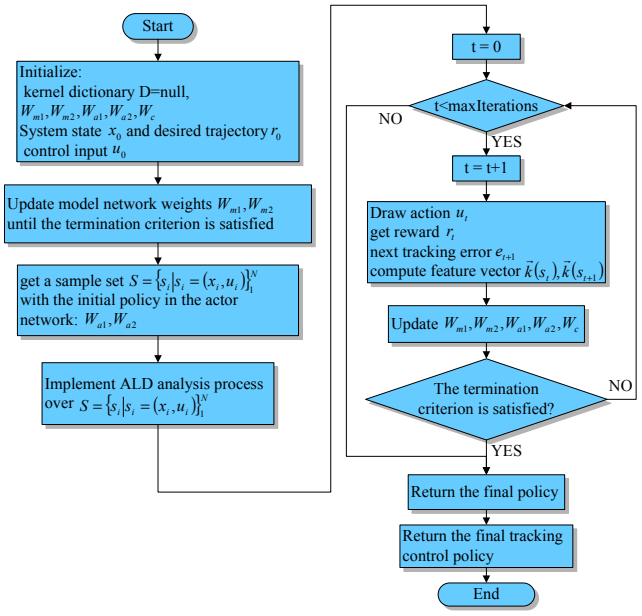


Fig. 3. The flowchart of the kernel-based HDP algorithm

1) The model network: The model network is designed to approximate the error dynamics. We should train the model network before carrying out the kernel-based HDP algorithm. For given e_k and u_k, we can obtain the output of the model network as

$$\hat{e}_{k+1} = W_{m2}\sigma\left(W_{m1}z_k\right) \tag{13}$$

where $z_k = \left[\begin{array}{cc} e_k^T & u_k^T \end{array}\right]^T$. The error function of the model network is defined as follows

$$e_{mk} = \hat{e}_{k+1}\text{-}e_{k+1}$$

The weights of the model network are updated to minimize the following performance measure:

$$E_{mk} = \frac{1}{2}e_{mk}^T e_{mk} \tag{14}$$

Using the gradient-based adaptation rule, the weights can be updated as

$$W_{m1}\left(t+1\right) = W_{m1}\left(t\right) - l_m\left[\frac{\partial E_{mk}}{\partial W_{m1}\left(t\right)}\right] \tag{15}$$

$$W_{m2}\left(t+1\right) = W_{m2}\left(t\right) - l_m\left[\frac{\partial E_{mk}}{\partial W_{m2}\left(t\right)}\right] \tag{16}$$

where $l_m > 0$ is the learning rate of the model network. After the model network is trained, we should keep its weights unchanged.

2) The critic network: In the critic of kernel-based HDP algorithm, the performance index function $Q\left(e,u\right)$ is approximated in a linear weighted form, where a Mercer kernel function $k\left(x,y\right) = \left\langle \phi\left(x\right), \phi\left(y\right)\right\rangle$ is employed to realize the feature mapping in a reproducing kernel Hilbert space (RKHS). Let $s_t = \left(e_t, u_t\right)$ denote the trajectory tracking error-action pair at time step t. Then, the performance index function $Q\left(e_t, u_t\right)$ can also be expressed as $Q\left(s_t\right)$. As studied in [29], the regression equation for the linear LSTD(0) $\left(\lambda = 0\right)$ algorithm is

$$\phi\left(s_t\right)\left(\tilde{Q}\left(s_t\right) - \gamma\tilde{Q}\left(s_{t+1}\right)\right) = \phi\left(s_t\right)r_t \tag{17}$$

and

$$\tilde{Q}\left(s\right) = \phi^T\left(s\right)W, \qquad \phi, W \in \Re^{q\times 1} \tag{18}$$

Equation (17) can be rewritten as

$$\left[\phi\left(s_t\right)\left(\phi^T\left(s_t\right) - \gamma\phi^T\left(s_{t+1}\right)\right)\right]W = \phi\left(s_t\right)r\left(s_t\right) \tag{19}$$

The observation equation of (19) is as follows:

$$\phi\left(s_t\right)\left(\phi^T\left(s_t\right) - \gamma\phi^T\left(s_{t+1}\right)\right)W = \phi\left(s_t\right)r\left(s_t\right) + \varepsilon_t \tag{20}$$

where ε_t is the one-step observation noise.

According to the property of RKHS, the weight vector W in equation (20) can be represented by the weighted sum of the state feature vectors

$$W = \sum_{i=1}^{d(n)}\phi\left(s_i\right)\alpha_i \tag{21}$$

where $s_i\left(i = 1, 2, \ldots, d\left(n\right)\right)$ are the selected tracking error-action pairs after the ALD analysis, $d\left(n\right)$ is the length of the dictionary D_n, and α_i are the coefficients.

Let

$$\Phi_T = \left(\phi^T\left(s_1\right), \phi^T\left(s_2\right), \ldots, \phi^T\left(s_T\right)\right)^T \tag{22}$$

$$\vec{k}\left(s_t\right) = \left(k\left(s_1, s_t\right), k\left(s_2, s_t\right), \ldots, k\left(s_T, s_t\right)\right)^T \tag{23}$$

By multiplying Φ_T to both sides of the observation equation (20), we get

$$\vec{k}\left(s_t\right)\left[\vec{k}\left(s_t\right)\vec{\alpha} - \gamma\vec{k}^T\left(s_{t+1}\right)\vec{\alpha}\right] = \vec{k}\left(s_t\right)r_t \tag{24}$$

and

$$\vec{\alpha} = \left[\alpha_1, \alpha_2, \ldots, \alpha_T\right]^T \tag{25}$$

Let

$$A_T = \sum_{t=1}^{N}\vec{k}\left(s_t\right)\left[\vec{k}^T\left(s_t\right) - \gamma\vec{k}^T\left(s_{t+1}\right)\right] \tag{26}$$

$$b_T = \sum_{t=1}^{N}\vec{k}\left(s_t\right)r_t \tag{27}$$

where N is the total number of samples.

Then, the kernel-based least-squares fixed point solution is as follows:

$$\vec{\alpha} = A_T^{-1} b_T \tag{28}$$

To realize online learning in the critic, the following update rules on the basic of kernel RLS-TD(0) algorithm are applied in the critic of kernel-based HDP, as are shown in equations (29)-(31).

$$\beta_{t+1} = P_t \vec{k}(s_t) \Big/ \left(\mu + \left(\vec{k}^T(s_t) - \gamma \vec{k}^T(s_{t+1})\right) P_t \vec{k}(s_t)\right) \tag{29}$$

$$\vec{\alpha}_{t+1} = \vec{\alpha}_t + \beta_{t+1}\left(r_t - \left(\vec{k}^T(s_t) - \gamma \vec{k}^T(s_{t+1})\right)\vec{\alpha}_t\right) \tag{30}$$

$$P_{t+1} = \frac{1}{\mu}\left[P_t - \frac{P_t \vec{k}(s_t)\left(\vec{k}^T(s_t) - \gamma \vec{k}^T(s_{t+1})\right) P_t}{\left[\mu + \vec{k}\left(\vec{k}^T(s_t) - \gamma \vec{k}^T(s_{t+1})\right) P_t \vec{k}(s_t)\right]}\right] \tag{31}$$

where β_t is the step size in the critic, $\mu\,(0 < \mu \leq 1)$ is the forgetting factor, $P_0 = \delta I$, δ is a positive number, and I is the identity matrix.

3) The actor network: In the actor network in kernel-based HDP, e_t is used as input to obtain the optimal control. The output can be formulated as

$$\hat{u}_k = W_{a2}\sigma(W_{a1} e_k) \tag{32}$$

The target control input is shown as

$$u_k = \frac{1}{2} R^{-1} g^T(e_k + r_k) \frac{\partial Q(s_{k+1})}{\partial e_{k+1}} \tag{33}$$

When Gaussion kernels are applied, the approximated value function is

$$\tilde{Q}(e,u) = \sum_{i=1}^{d(n)} \alpha_i k(s, s_i) = \sum_{i=1}^{d(n)} \alpha_i e^{-\|\vec{s} - \vec{s}_i\|/\sigma^2} \tag{34}$$

where $\vec{s} = \left(e_{(1)}, e_{(2)}, \ldots, e_{(m)}, u\right)$ is the combined vector of the tracking error-action pair. The dimension of the tracking error state is defined as m. $\|\cdot\|$ is defined as

$$\|\vec{s} - \vec{s}_i\| = \sqrt{\sum_{j=1}^{m}\left(e_{(j)} - e_{i(j)}\right)^2 + (u - u_i)^2} \tag{35}$$

On the basic of the definition in equation (34), we have

$$\frac{\partial \tilde{Q}(e,u)}{\partial e} = \sum_{i=1}^{T} 2\alpha_i \frac{(e - e_i)}{\sigma^2} e^{-\|\vec{s} - \vec{s}_i\|/\sigma^2} \tag{36}$$

We can define the error function of the action network as

$$e_{ak} = \hat{u}_k - u_k \tag{37}$$

The actor network is trained to minimize the following performance error measure:

$$E_{ak} = \frac{1}{2} e_{ak}^T e_{ak} \tag{38}$$

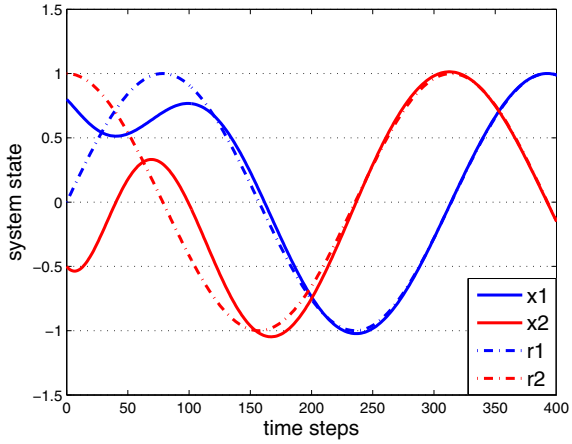
Similarly, the actor learning rule in KHDP is as follows.

$$W_{a1}(t) = W_{a1}(t) - l_a \left[\frac{\partial E_{ak}}{\partial W_{a1}(j)}\right] \tag{39}$$

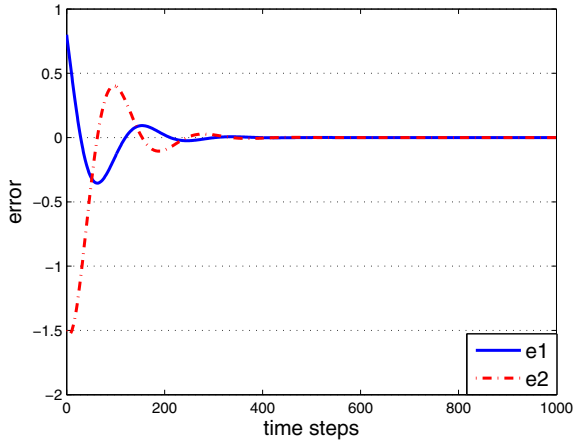
$$W_{a2}(t) = W_{a2}(t) - l_a \left[\frac{\partial E_{ak}}{\partial W_{a2}(j)}\right] \tag{40}$$

IV. Simulation Study

We set the original system state as [0.8;-0.5]. The reference trajectory for the above system is selected as $r = [\sin(t); \cos(t)]$. In the UAV heading system tracking simulation experiment, the optimal trajectory tracking control problem is transformed into designing an optimal regulator for the tracking error dynamics firstly. Note that the weights in the action and the model networks are trained using their internal cycles. The weights in the model and the actor networks are initialized randomly in the range of (-1,1). The simulation results are shown in the following figures. We only give part-time steps in order to better observe the results in Fig. 4(a).



(a) x and r



(b) The tracking error e_k

Fig. 4. The system state trajectories and tracking error

In the example, we used Gaussian kernel $k(s_i, s_j) = \exp\left(-(s_i - s_j)^T (s_i - s_j)\right) \Big/ \left(2\sigma^2\right)$. Thus, in the tracking control procedure is used to find the value of the kernel width parameter σ for which each algorithm performed best. In this experiment, we set $\sigma = 6$.

It should be mentioned that the model network should be trained first. 1000 trials of samples are collected by a random

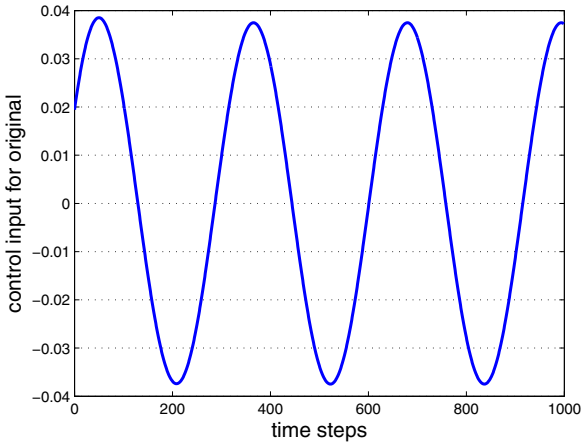
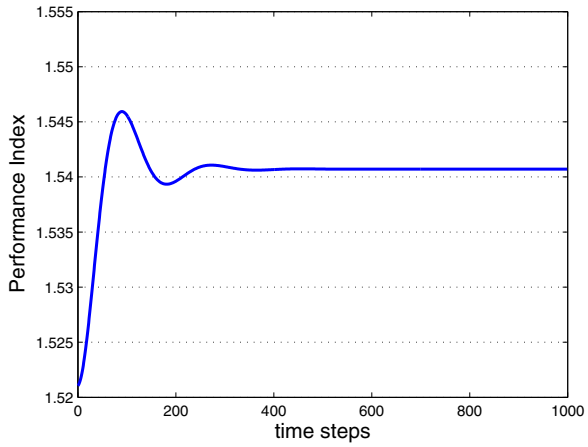
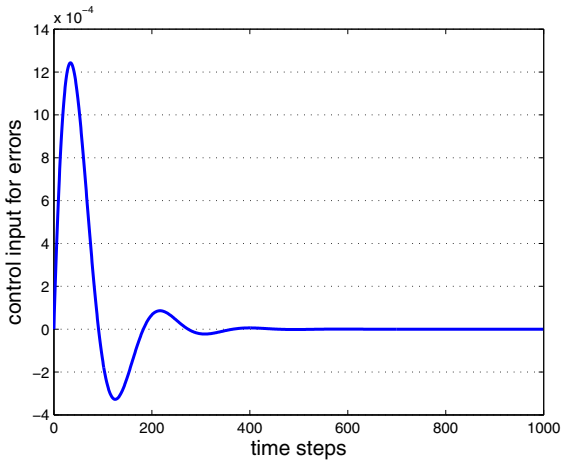


Fig. 5. The tracking control law for the UAV heading system



(a) The trajectory of action value function



(b) The tracking control law for the error system

Fig. 6. The convergence process

policy to construct the dictionary of kernel features. The threshold parameter for the ALD analysis is set as $\mu = 0.001$.

The convergence processes of the value function of the kernel-based HDP algorithm and optimal tracking control for the error dynamics are shown in Fig. 6(a) and Fig. 6(b). We can see that the iterative value function sequence does converge to the optimal cost function quite rapidly, which indicates the effectiveness of the kernel-based HDP algorithm. Next, we compute the near-optimal tracking control law for original system (1) using equation (10) and apply it to the controlled system. The obtained UAV system state curves are shown in Fig. 4(a). In order to evaluate the tracking performance, the corresponding reference trajectories are also plotted simultaneously in Fig. 4(a). The tracking control curves and the tracking errors are shown in Fig. 5 and 4(b). These simulation results verify the excellent performance of the tracking controller developed by the KHDP algorithm.

V. Conclusion

In this paper, the kernel-based HDP algorithm is introduced to design the optimal tracking controller for UAV heading system, which obtains a infinite-horizon near-optimal tracking controller that makes both the performance index function close to its optimal value and the UAV heading system dynamics tracking the desired trajectories simultaneously. Due to MLPNNs with manually designed features, conventional HDP has difficulties in improving the generalization capability and learning efficiency. This paper presents a novel framework of HDP by integrating kernel methods and ALD analysis into the critic learning for the optimal tracking controller design. The kernel methods can obtain better generalizaiton capability than MLPNNs. Moreover, the simulation results confirmed the validity of the proposed tracking control approach. This research in this paper shows that it is a very promising work to integrate sparse kernel machines into online learning ADP control algorithms.

References

[1] Erjie Cui. Intelligent micro air vehicle-inspiration from bionics. Scientific Chinese, no. 4, pp. 7–8, 2004

[2] Bowen Nie, etc. Research status and key technology of the micro-rotor aircraft. Electronics Optics & Control. vol. 14, no. 6, pp. 113–117, 2007.

[3] Sentao Hong, Zhihui Jin and Zhiqiang Li. Modeling and posture stability analysis of the quad-rotor aircraft. Electronic and Electro-optical Systems, no. 2, pp. 34–37, 2008.

[4] S. Bouabdallah, A. Noth and R. Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor.In Intelligent Robots and Systems. In Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, Sendal, Japan, pp. 2451–2456.

[5] Haiyan Shan. Combined DI/QFT flight control for a quad-rotor unmanned helicopter. Electronics Optics & Control, vol. 15, no. 12, pp. 68–71, 2008.

[6] S. Zhong, X. Zeng, S. Wu, and L. Han. Sensitivity-based adaptive learning rules for binary feedforward neural networks. IEEE Transactions On Neural Networks and Learning Systems, vol. 23, no. 3, pp. 480-491, 2012.

[7] Ding Wang, Derong Liu, Qinglai Wei, Dongbin Zhao, and Ning Jin. Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. Automatica, vol. 48, pp. 1825-1832, 2012.

[8] V. Yadav, R. Padhi and S. N. Balakrishnan. Robust/Optimal temperature profile control of a high-speed aerospace vehicle using neural networks," IEEE Transactions on Neural Networks, vol. 18, no. 4, pp. 1115-1128, 2007.

[9] F. L. Lewis, D. Liu and G. G. Lendaris. Guest editorial - special issue on adaptive dynamic programming and reinforcement learning in feedback control. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 38, no. 4, pp. 896 - 897, 2008.

[10] Derong Liu. Approximate dynamic programming for self-learning control. Acta Automatic Sinica, vol. 31, no. 1, pp. 13-18, 2005.

[11] Yanhong Luo, Huaguang Zhang, Ning Cao, and Bing Chen. Near-optimal stabilization for a class of nonlinear systems with control constraint based on single network greedy iterative DHP algorithm. Acta Automatic Sinica, vol. 35, no. 11, pp. 1436-1445, 2009.

[12] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits and Systems Magazine, vol. 9, no. 3, pp. 32-50, 2009.

[13] Travis Dierks, Balaje T. Thumati and S. Jagannathan. Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. Neural Networks, vol. 22, no. 5-6, pp. 851–860, 2009.

[14] Huaguang Zhang, Yanhong Luo and Derong Liu. Neural network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. IEEE Transactions on Neural Networks, vol. 20, no. 9, pp. 1490 - 1503, 2009.

[15] Fei-Yue Wang, Ning Jin, Derong Liu, and Qinglai Wei. Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε - error bound. IEEE Transactions On Neural Networks, vol. 22, no. 1, pp. 24-36, 2011.

[16] B. Schlkopf and A. J. Smola. *Learning With Kernels*. Cambridge, Massachusetts: The MIT Press, 2002.

[17] V. Vapnik. *Statistical Learning Theory*. New York: Wiley, 1998.

[18] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. In: Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches, D. A. White and D. A. Sofge, D. A. White and D. A. Sofge. New York: Van Nostrand Reinhold, 1992, pp. 493–525.

[19] Xin Xu, Z. Hou, C. Lian, and H. He. Online learning control using adaptive critic designs with sparse kernel machines. IEEE Transactions on neural networks and learning systems, vol. 24, no. 5, pp. 762-775, 2013.

[20] Jianda Han, Yuqing He and Xingang Zhao. Mobile robot systems : modeling , estimation and control. Beijing:Science Press, 2011.

[21] Huaguang Zhang, Qinglai Wei and Derong Liu. On-line learning control for discrete nonlinear systems via an improved ADDHP method, In Proc. International Symposium on Neural Networks, Nanjing, China, June 2007, vol. 1, pp.387-396.

[22] Huaguang Zhang, Qinglai Wei and Yanhong Luo. A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 38, no. 4, pp. 937–942, 2008.

[23] T. Dierks and S. Jagannathan. Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics. In Proceedings of Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, Shanghai, China, 2009, pp. 6750–6755.

[24] D. Vrabiea, O. Pastravanub, M. Abu-Khalafc, and F. L. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. Automatica, vol. 45, no. 2, pp. 477-484, 2009.

[25] Huaguang Zhang, Lili Cui, Xin Zhang, Yanhong Luo. Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. IEEE Transactions on Neural Networks, vol. 22, no. 12, pp. 2226-2236, 2011.

[26] Murad Abu-Khalaf and Frank L. Lewis. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. Automatica. vol. 41, no. 5, pp. 779–791, 2005.

[27] Y. Engel, S. Mannor and R. Meir. The kernel recursive least-squares algorithm. IEEE Transactions On Signal Processing, vol. 52, no. 8, pp. 2275-2285, 2004.

[28] X. Xu, T. Xie, D. Hu, and X. Lu. Kernel least-squares temporal difference learning. International Journal of Information Technology, vol. 11, no. 9, pp. 54-63, 2005.

[29] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. Journal of Machine Learning Research, vol. 4, pp. 1107-1149, 2003.