J Bionic Eng 15 (2018) 329–340
DOI: https://doi.org/10.1007/s42235-018-0025-9

Journal of Bionic Engineering
http://www.springer.com/journal/42235

Learning Control of Quadruped Robot Galloping

Qingyu Liu, Xuedong Chen, Bin Han, Zhiwei Luo, Xin Luo*

State Key Laboratory of Digital Manufacturing Equipment and Technology, *Huazhong University of Science and Technology*, *Wuhan* 430074, *China*

Abstract

Achieving galloping gait in quadruped robots is challenging, because the galloping gait exhibits complex dynamical behaviors of a hybrid nonlinear under-actuated dynamic system. This paper presents a learning approach to quadruped robot galloping control. The control function is obtained through directly approximating real gait data by learning algorithm, without consideration of robot's model and environment where the robot is located. Three motion control parameters are chosen to determine the galloping process, and the deduced control function is learned iteratively with modified Locally Weighted Projection Regression (LWPR) algorithm. Experiments conducted upon the bioinspired quadruped robot, AgiDog, indicate that the robot can improve running performance continuously along the learning process, and adapt itself to model and environment uncertainties.

Keywords: quadruped, gallop, dynamic running, LWPR learning, bioinspiration

1 Introduction

Galloping is a fast and energy efficient dynamic locomotion pattern, which relaxes the static equilibrium constraint on how legs can move to support the quadrupeds[1]. Quadrupedal animals always take galloping gait in the highest running speed[2], for example, cheetahs can reach 112 km·h^{-1} in galloping as reported. This advantage arouses great interest of biologists and engineers to imitate the gait in artificial counterparts[3–8].

However, achieving galloping gait in quadruped robots is challenging. A quadruped running in galloping gait acts as an under-actuated and hybrid nonlinear dynamic system from the view point of dynamics[9]. Early investigations were limited to simulation studies of simplified theoretical models with intuitive control strategies[10,11]. More practical control approaches were presented later[8,12–14]. However, all these control approaches have not been applied to real robots. Poulakakis *et al*. implemented the galloping gait on an embodied robot, SCOUT Ⅱ, but the configuration of one actuator per leg limited the gait motions[7]. Park *et al*. presented a control algorithm mimicking leg forces observed in animal running and achieved impressive performance on the MIT Cheetah robot[5], but for simplification, the model used for control ignores the leg inertial and impulse effect of landing. So far, the unique example of robust outdoor galloping gait on real quadruped robot was exhibited on the Boston Dynamics Cheetah robot, WildCat[6], however, little detail of the control algorithm has been revealed.

One disadvantage of above mentioned model-based control approaches is that they cannot handle the high-dimension complete models and model uncertainties. Learning approach provides a promising direction in quadruped control, by approximating the control function directly, avoiding mathematical model of the robot and environment that the robot is situated in, and adapting to model uncertainties[15]. Krasny and Orin, Marhefka and Orin used multiobjective genetic algorithm to search the control parameters embedded in the motion primitives, the controllers worked well in simulation scenario[8,14]. Palmer and Orin used fuzzy control approach and realized turning motion in quadruped galloping gait[13]. Hugh and McMahon characterized the galloping gait with a few parameters and used genetic algorithm to search for the parameter values to control a quadruped[12]. Chae and Park used genetic algorithm to optimize the galloping gait trajectory[16]. However, most of the strategies they used are offline learning approaches and need long time for simulation, limiting their applications on real robots.

***Corresponding author:** Xin Luo
E-mail: mexinluo@hust.edu.cn

Two problems in learning control hinder most of the existing learning methods for online application. One problem is difficulties in acquiring training data and time-consuming when real robot is in control loop, the other is the exponentially growing training data needed to cover the system state space as the system dimension grows[17]. A feasible choice is the Locally Weighted Projection Regression (LWPR) algorithm[18]. The LWPR algorithm is an incremental online learning method presented by Vijayakumar *et al*.[18], which was usually adopted for robot model learning and trajectory optimization. The LWPR algorithm has been used in controlling biped walking[19], however, so far not yet in quadruped galloping control.

In this paper, a learning control approach for quadruped galloping is presented. There are two main contributions in this paper. One is that a control framework is presented to transform the control issue of the quadruped galloping into the determination of three motion control parameters, *i.e*. gait cycle, leg extension length and target speed variation. These parameters can be easily determined online using the modified LWPR. Another contribution is that we modified the LWPR algorithm to map all outputs into one LWPR model while preserving the input projection direction corresponding to each output, which further reduces the computation load. Using the controller, galloping gait of 1 $m \cdot s^{-1}$ was realized on the quadruped robot AgiDog.

The rest of this paper is organized as follows. The learning control framework of the quadruped galloping gait is presented in section **2**. Section **3** presents the formulation of the control modules in the control framework. The experiment results and discussion are presented in section **4**. Conclusions are drawn in section **5**.

2 Learning control framework

2.1 Three-parameter description of galloping locomotion

Transverse gallop is one of the typical galloping gaits. Without loss of generality, this paper will focus on this type of galloping gait. The gait sequence of galloping gait is illustrated in Fig. 1. The trailing hind leg, leading hind leg, trailing fore leg and leading fore leg are abbreviated as TH, LH, TF and LF for short.

Galloping gait is featured by the early retraction of the swing leg[12], that is, the swing leg retracts backward before its landing. This behavior lowers the relative speed of the foot with respect to the ground at the moment of contact, thus reduces the impact with the ground[20]. The events triggering each leg retraction are different, the LH and LF are triggered when the corresponding trailing leg angles with respective to the vertical reduce to zero, the TH is triggered at the time delaying a gait cycle T to its previous retracting, while the TF begins to swing backward after a fix time interval from the swinging backward of the TH which is set to $T/3$ based on Witte's experimental data[21]. The value of T varies with different speeds.

The second characteristic of the galloping gait is the hip thrusting and shoulder braking effect[12]. The Center of the Gravity (COG) is not coincidence with the hip/shoulder joints, thus the robot has tendencies to pitch downward in hind leg stance phase and upward in fore

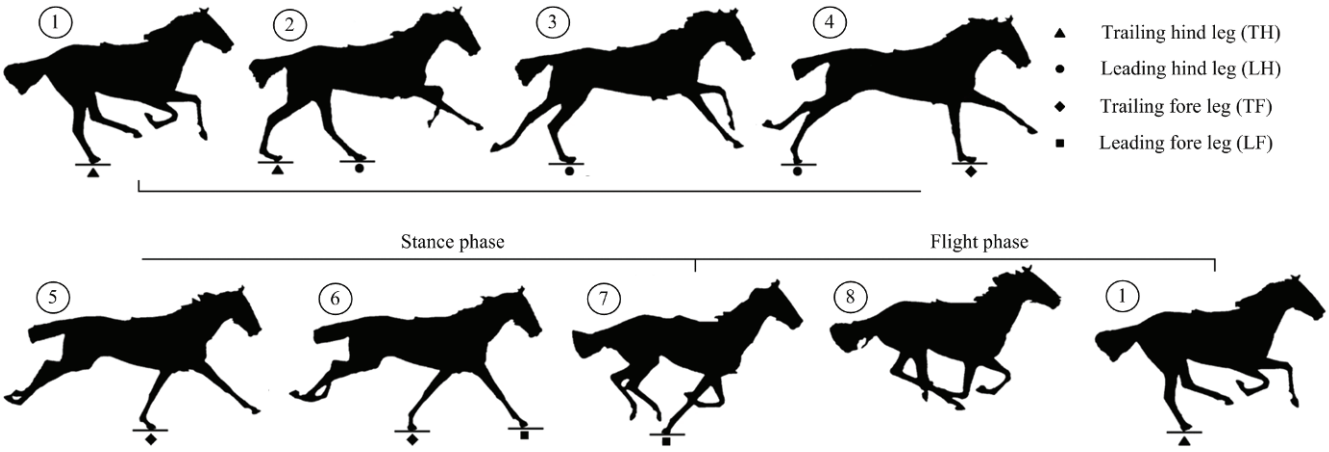

Fig. 1 Four-beat transverse galloping gait.

leg stance phase. To remain stable, the quadruped would exert extra torques in hip and shoulder joints, accelerating the robot in hind leg stance phase and decelerating the robot in fore leg stance phase. We use V_{var} to represent the difference between the target speed in stance phase and the target average speed.

In stance phase, dynamical behaviour of the leg is equivalent to that of a spring-damp system[1]. Besides, there is energy loss due to friction, impact, *etc*. during galloping. Thus, in galloping gait control, we reset the target leg length L_{var} longer than the landing length at maximum compression, thereby increasing the nominal length of the equivalent spring. In the process of elongation, the leg will inject extra energy to the robot system to compensate for the energy dissipation.

Based on the gait characteristics above described, the leg motion in one gait cycle can be characterized as follows:

(1) The TH begins early retraction at a certain moment $t_p = 0$. When the TH touches down, it continues retracting to maintain target speed $v_h = v_d + V_{var}$, where v_d is the desired speed.

(2) The LH begins early retraction with the same target speed at the time when the angle of TH with respect to the vertical reduces to zero.

(3) The TF begins early retraction at the time $t_p = T/3$. When the TF contacts the ground, it retracts to maintain target speed $v_f = v_d - V_{var}$.

(4) The LF begins early retraction with target speed v_f when the angle of TF reduces to zero.

(5) The stance leg length motion is defined by a predefined trajectory, in which the leg extends L_{var} longer than the landing length. Then the robot enters flight phase, and the TH begins early retraction at the time $t_p=T$, and a new gait cycle begins.

Thus, we can find that the galloping gait can be characterized by three parameters, *i.e*. gait cycle T for coordination of the leg movement sequences, target velocity variation V_{var} for stance leg motion control, and the leg extension length L_{var} for system energy compensation. It is naturally to use these parameters for galloping gait control. Though Krasny and Orin also parameterized the galloping gait to control a quadruped model, the parameters are too many to calculate online[8]. Herr and McMahon also controlled the quadruped gallop by determined three parameters[12], but his control strategy cannot actively regulate the gait cycle and robot system energy as in ours by gait cycle T and leg extension length L_{var}.

2.2 Control framework

The object of learning is to determine the three parameters of the galloping gait for leg motion control, thus, the outputs of the learning module are T, V_{var} and L_{var}, while the inputs are robot state variables in apex, including height of the COG h, horizontal velocity of the COG v, pitch angle p, angular velocity ω and desired speed v_d. The vertical speed and horizontal position are mutually excluded as vertical speed is always zero in apex and the horizontal position has no effect on the dynamics of the robot. Thus, the control function to be learned is defined as:

$$\left(T,\, V_{var},\, L_{var}\right)^{T} = f\left(h, v, p, \omega, v_d\right). \tag{1}$$

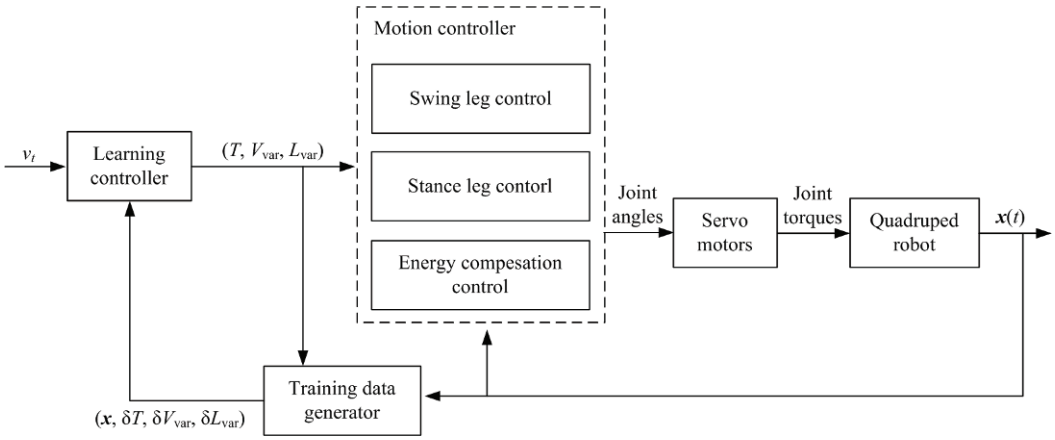

Fig. 2 Control framework of galloping gait.

When the control parameters are determined, the leg movements will be generated. The control function is updated every gait cycle according to the control performance. Thus, the proposed control framework consists of motion controller, training data generator, and learning controller, as shown in Fig. 2. The motion controller receives the control parameters from the learning controller to coordinate motion of legs. The training data generator outputs the system state $\boldsymbol{x}$ and the corresponding parameters offset δT, δV_{var}, δL_{var} relative to the values used in previous gait cycle as the training data. The learning controller is activated once in each gait cycle in the apex of the flight phase and outputs new control parameters for the next gait cycle.

3 Formulation of the control approach

3.1 Motion control of the legs

Leg motions are coordinated based on three control parameters, that is, T, V_{var}, L_{var}. T is used to coordinate the leg retraction motion. For the TF and TH, the control strategy is described as:

$$\begin{cases} \alpha_{\text{T}} = \alpha_{\text{flight}}, \ \text{if} \ (flag = 0) \\ \alpha_{\text{T}} = \alpha_{\text{flight}} - \dfrac{v}{l_{\text{T}}} t_{\text{p}}, \ \text{if} \ (flag = 1) \end{cases}, \tag{2}$$

where α_{T} is the trailing leg angle with respect to the vertical, α_{flight} is the target value, l_{T} is the leg length of the trailing leg, *flag* is a binary-valued indicator. For the TH, *flag* is assigned to 1 at time t_{p}=0, while for the TF, *flag*=1 from t_{p}=T/3, and is set to 0 when they reach back limit positions. The retractions of the LH and LF are triggered when the corresponding trailing legs are perpendicular to the ground. Thus the leading leg control strategy is deduced as:

$$\begin{cases} \alpha_{\text{L}} = \alpha_{\text{flight}}, \ \text{if} \ (\alpha_{\text{t}} \geq 0) \\ \alpha_{\text{L}} = \alpha_{\text{flight}} - \dfrac{v}{l_{\text{L}}} t_{\text{p}}, \ \text{if} \ (\alpha_{\text{t}} \leq 0) \end{cases}, \tag{3}$$

where α_{L} is the leading leg angle with respect to the vertical, l_{L} is the leg length of the leading leg. Based on the gait characteristics discussed in section **2.1**, we assign different target running speeds in hind leg and fore leg stance phases. Thus, the deduced leg control strategy in stance phase is:

$$\begin{cases} v_{\text{h}} = v_{\text{d}} + V_{\text{var}} \\ v_{\text{f}} = v_{\text{d}} - V_{\text{var}} \end{cases}. \tag{4}$$

As described in section **2.1**, the stance leg dynamics is equivalent to a spring-damp system, the leg length function of time is similar to trigonometric function. So we approximate the function by a 2-order interpolation function as shown in Fig. 3. The formulation of the leg length control strategy is:

$$\begin{aligned} l(t, l_2) = & \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} l_0 \\ & + \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} l_1 + \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} l_2, \end{aligned} \tag{5}$$

where l is the leg length, (t_0, l_0), (t_1, l_1), (t_2, l_2) are the characteristic points in the function curve, where l_0 is the nominal leg length, l_1 is the minimum leg length estimated using the model presented by Heglund and Taylor[22], and l_2= l_0+ L_{var}, L_{var} is determined by learning.

3.2 Modification of the LWPR for multi-output systems

3.2.1 Parameter learning algorithm

The LWPR is a fast, incremental and reasonably accurate function approximate method for high dimension nonlinear functions. Here we explain some basic concepts in the LWPR, for details see Ref.[18]. The modified LWPR algorithm will be presented in section 3.2.2.

In the LWPR algorithm, one LWPR model is built for each output. the input space in each LWPR model is divided into different regions called receptive fields (RFs), each RF is determined by a positive definite distance matrix $\boldsymbol{D}$. The weight w of input $\boldsymbol{x}$ is calculated

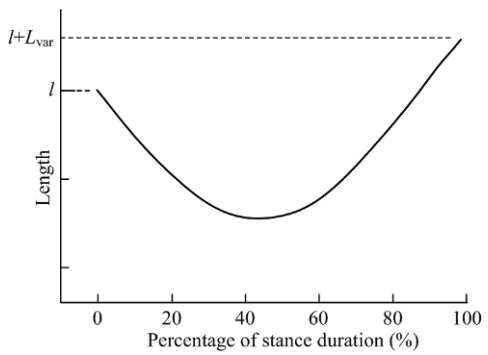

Fig. 3 Leg length trajectory in stance phase.

using the Gaussian function

$$w(\boldsymbol{x})=\exp[0.5\cdot(\boldsymbol{x}-\boldsymbol{c})^{\mathrm{T}}\cdot\boldsymbol{D}\cdot(\boldsymbol{x}-\boldsymbol{c})],$$

where $\boldsymbol{c}$ is the center of the RF.

In each RF, a linear function $y=\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}$ is used to predict the output, where $\boldsymbol{\beta}$ is coefficient vector of the linear model calculated by regression method, y is the estimated output. To reduce the input dimension, the input vector is projected to several directions which are most relevant to the output using partial least square regression, thus, a lower dimensional input vector can be obtained and then regression analysis of the lower dimensional input vector and output can be done.

The LWPR algorithm updates the distance matrix $\boldsymbol{D}$ every learning step to improve the predictive performance, the update rule is:

$$\boldsymbol{m}^{n+1}=\boldsymbol{m}^{n}+\alpha\frac{\partial L}{\partial \boldsymbol{m}}, \tag{6}$$

where $\boldsymbol{m}$ is the vectorization of a matrix $\boldsymbol{M}$, $\boldsymbol{m}=\mathrm{Vec}(\boldsymbol{M})$, $\boldsymbol{D}=\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M}$, $\boldsymbol{M}$ is an upper triangular matrix to ensure $\boldsymbol{D}$ is symmetric and positive definite, L is the cost function, α is the learning rate. As the training data comes, new RFs will be added and some RFs may be deleted as needed to fit the target model, and the final output is the weighted sum of the local model outputs of the RFs.

3.2.2 Modification

There are two alternatives to handle multi-output function learning in original LWPR algorithm, one is building a LWPR model for each output, and the other is learning one single LWPR model for all outputs using the same projection directions in each RF[18]. However, the modified algorithm determines different outputs in one model while preserving the projection directions for each output, *i.e*., the algorithm updates the local linear models for different outputs in one RF using different projection directions. The sum of the residual errors of all outputs is used to adjust the shape and size of RF. The modified algorithm is more efficient.

When a training data set ($\boldsymbol{x}$, $\boldsymbol{y}$) is obtained, weight of the $\boldsymbol{x}$ in each RF is calculated as described in section 3.2.1, then the RFs in the LWPR model will be updated one by one using the data set ($\boldsymbol{x}$, $\boldsymbol{y}$). The weighted means are firstly calculated as:

$$\boldsymbol{x}_0^{n+1}=\frac{1}{W^{n+1}}(\lambda W^{n}\boldsymbol{x}_0^{n}+w\boldsymbol{x}), \tag{7}$$

$$y_{0,i}^{n+1}=\frac{1}{W^{n+1}}(\lambda W^{n}y_{0,i}^{n}+wy_i),\ \ i=1,\cdots,K, \tag{8}$$

where n is corresponding to the nth training data, i indicates the ith element of $\boldsymbol{y}$. The forgetting factor λ makes it possible for the algorithm to adapt to the change of the function to be learned over time. The coefficients of the local linear model will be calculated using the iterative form of the partial least square regression analysis. The coefficients are calculated as:

$$\beta_{m,i}^{n+1}=\frac{R_{m,i}^{n+1}}{S_{m,i}^{n+1}}, \tag{9}$$

where

$S_{m,i}^{n+1}=\lambda S_{m,i}^{n}+ws^2$, $R_{m,i}^{n+1}=\lambda R_{m,i}^{n}+wsz_{\mathrm{res}}$, the s is calculated as:

$$\boldsymbol{u}_{m,i}^{n+1}=\lambda\boldsymbol{u}_{m,i}^{n}+wzz_{\mathrm{res}}, \tag{10}$$

$$s=\boldsymbol{z}^{\mathrm{T}}\boldsymbol{u}_{m,i}^{n+1}, \tag{11}$$

where $\boldsymbol{u}$ is the projection direction, s is the Coordinate on projected direction, the subscript i indicates the ith output, the m is corresponding to the mth coefficient of the linear model, and the superscript n is corresponding to the nth update of the LWPR model. $\boldsymbol{z}$ and z_{res} are the offset vectors of the $\boldsymbol{x}$ and y_i to the weighted mean points x_0^{n+1} and $y_{0,i}^{n+1}$ .The regression analysis is performed on the offset vectors other than the original input $\boldsymbol{x}$ and y_i, which is to guarantee that the linear models in RFs are estimates of the nonlinear function around the weighted means. For one output in a RF, the calculation processes of Eq. (7) to Eq. (11) are repeated till the coefficients are solved, and the $\boldsymbol{z}$ and z_{res} will be updated in each iterative step, the update rule is:

$$z_{\mathrm{res}}=z_{\mathrm{res}}-s\beta_{m,i}^{n+1}, \tag{12}$$

$$\boldsymbol{z}=\boldsymbol{z}-s\boldsymbol{p}_{m,i}^{n+1}, \tag{13}$$

where $\boldsymbol{p}_{m,i}^{n+1}=\boldsymbol{E}_{m,i}^{n+1}/s_{m,i}^{n+1}$, $\boldsymbol{E}_{m,i}^{n+1}=\lambda\boldsymbol{E}_{m,i}^{n}+wzs$, and the subscripts i, m and the superscript n are the same as in Eq. (7) to Eq. (11). The update process of the linear

model for all outputs in one RF is summarized as:

//----------

INPUT: training data set $(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{x}=(h,v,p,\omega,v_{\mathrm{d}})^{\mathrm{T}}, \boldsymbol{y}=(T,V_{\mathrm{var}},L_{\mathrm{var}})^{\mathrm{T}}$

PROCESS:

Compute the weight w of $\boldsymbol{x}$ in RF.

Update the sum of the weight W^{n+1}.

Update the weight means of data $\boldsymbol{x}_0^{n+1}$.

for i = 1 to K

Update the weight means $\beta_{0,i}^{n+1}$ of the ith element of $\boldsymbol{y}$.

end for

for i = 1 to K

Initialize the residuals of the $\boldsymbol{x}$ and y_i.

$\boldsymbol{z} \leftarrow \boldsymbol{x} - \boldsymbol{x}_0^{n+1}, z_{\mathrm{res}} \leftarrow y_i - y_{0,i}^{n+1}$.

for m = 1 to R

Update the projection direction $\boldsymbol{u}_{m,i}^{n+1}$.

Project the residual $\boldsymbol{z}$ to $\boldsymbol{u}_{m,i}^{n+1}$.

Take regression analysis and obtain the rth coefficient of the linear model corresponding to the ith output.

Update the residual $\boldsymbol{z}$ and z_{res}.

end for

end for

OUTPUT: update the coefficients $\beta_{m,i}^{n}$ of the linear models in RF.

//----------

The calculation is similar to the original LWPR algorithm, except that, w, W^{n+1}, x_0^{n+1} are shared by all outputs while $y_{0,i}^{n+1}$ is corresponding to the ith output, and the regression calculation process is repeated K times for K outputs in one single LWPR model. What's more, in RF update procedure, the prediction error in cost function L is the sum of the errors of all outputs, that is:

$$L = \frac{1}{\Sigma_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^T (\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)}{(1 - w_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{P} \boldsymbol{x}_i)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2, \tag{14}$$

where the first term is the mean leave-one-out cross-validation error of all predictive outputs in one LWPR model, and the second term is the penalty term to prevent the RF shrinking indefinitely, otherwise the number of the LWPR models will grow infinitely. After the training process, the derived LWPR model which minimizes the cost function L can make a balance between predictive precision and computation load.

3.3 Generation of training data set

The training data generator module will monitor the control performance under current parameters to generate a training data set. The training data set includes an input vector $\boldsymbol{x}$ and the corresponding output offset vector $(\delta T \ \delta V_{\mathrm{var}} \ \delta L_{\mathrm{var}})^{\mathrm{T}}$, which is used to update the LWPR model in learning control module.

The learning algorithm is invoked when the robot is at the apex after the fore leg stance phase. During one galloping gait, the max height $h_{\max}$, max pitch angle $p_{\max}$ and minimal pitch angle $p_{\min}$ are recorded as indicators of the control performance, and the actual gait duration is used to renew the gait cycle. To make the robot galloping in a more natural way, we set $(15\ 30\ -30)^{\mathrm{T}}$ as the target value of $(h_{\max}\ p_{\max}\ p_{\min})^{\mathrm{T}}$ in galloping gait, the data is based on the zoology experiment on mammals about the same size[23]. The control function outputs will be updated according to the error vector of the performance indicators to their target values. The update rule target function output is derived as:

$$\begin{pmatrix} \delta T \\ \delta L_{\mathrm{var}} \\ \delta V_{\mathrm{var}} \end{pmatrix} = \boldsymbol{J}^{-1} \begin{pmatrix} \Delta h_{\max} \\ \Delta p_{\max} \\ \Delta p_{\min} \end{pmatrix}, \tag{15}$$

where the $(\Delta h_{\max}\ \Delta p_{\max}\ \Delta p_{\min})^{\mathrm{T}}$ is calculated as $(15\ 30\ -30)^{\mathrm{T}} - (h_{\max}\ p_{\max}\ p_{\min})^{\mathrm{T}}$. The transformation matrix $\boldsymbol{J}$ is the Jocobian matrix of the function expressed as:

$$(h_{\max}\ \ p_{\max}\ \ p_{\min})^{\mathrm{T}} = \boldsymbol{g}(T, L_{\mathrm{var}}, V_{\mathrm{var}}), \tag{16}$$

where the function $\boldsymbol{g}$ is a function relative to the galloping gait. We need not to know the exact expression of $\boldsymbol{g}$ and we just estimate its Jocobian matrix. The Jocobian matrix is estimated using the latest three data points as:

$$\boldsymbol{J} = \begin{bmatrix} \Delta h_{\max 1} & \Delta h_{\max 2} & \Delta h_{\max 3} \\ \Delta p_{\max 1} & \Delta p_{\max 2} & \Delta p_{\max 2} \\ \Delta p_{\min 1} & \Delta p_{\min 2} & \Delta p_{\min 3} \end{bmatrix} \begin{bmatrix} \Delta T_1 & \Delta T_2 & \Delta T_3 \\ \Delta L_{\mathrm{var}1} & \Delta L_{\mathrm{var}2} & \Delta L_{\mathrm{var}3} \\ \Delta V_{\mathrm{var}1} & \Delta V_{\mathrm{var}2} & \Delta V_{\mathrm{var}3} \end{bmatrix}^{-1}. \tag{17}$$

4 Experimental test and discussion

4.1 Experimental setup

To verify the effectiveness of the proposed approach, a small-sized, bioinspired quadruped robot, AgiDog, was built with lightweight legs and powerful actuators, thus obtaining running frequency up to 5 Hz[24]. The leg configuration was inspired by the terrestrial mammalian animals[25]. The trunk of the robot is stiff, and the structural material of the legs is fibreglass. The robot is shown in Fig. 4, and its design specification is listed in Table 1.

Each leg is tri-segmented, *i.e.* the proximal segment d_1, the middle segment d_2 and the distal segment d_3. The middle segment is a parallelogram mechanism, which connects the proximal and the distal using two parallel pairs, so the proximal is always parallel to the distal throughout the leg movement. Calculation of the leg length and leg angle can be seen in our previous paper on bounding gait control[24]. Each leg is actuated by two DC servo motors mounted on the trunk. While the hip motor drives the hip joint co-axially to protract or retract during running, the knee motor drives the knee joint through a cable mechanism. As shown in Fig. 5, there is a boom connected to the robot so the robot can run around another end of the boom. The control algorithm was implemented using the NI PXI 8119 control board, the data acquisition and command output were implemented using the NI PXI 6123/6133 cards, for details about the robot design and experimental setup, refer to Ref. [24].

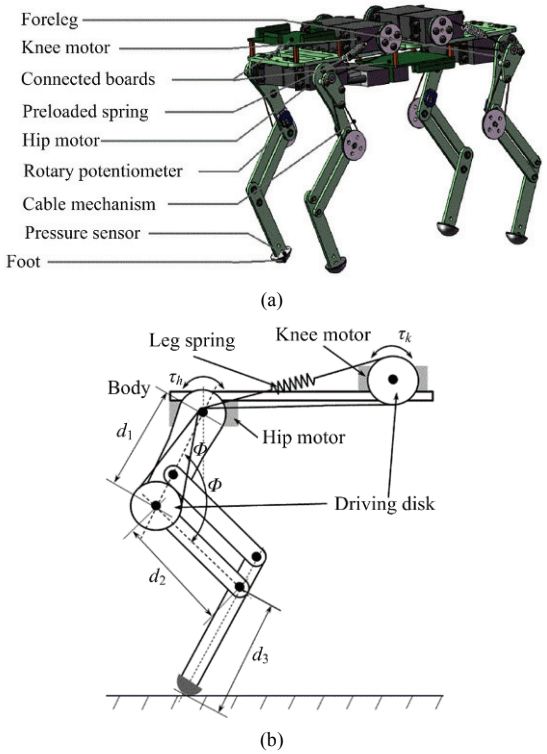

Fig. 4 AgiDog robot[24]. (a) Solid model; (b) leg design.

Table 1 Design specification of the AgiDog

Item	Type/Value
Total mass	1.1 kg
Leg mass	0.03 kg
Nominal leg length	0.145 m
Body length (hip to shoulder)	0.2 m
Body width (hip to hip)	0.1 m
Leg spring stiffness	15000 $N \cdot m^{-1}$
Cable material	Nylon
Cable diameter	0.8 mm
Servo motor	servoKing DS-695HV
Power supply	8.4 V
Motor mass	68 g
Stall torque	2.13 Nm at 8.4 V
Max speed	0.05 s/60° at 8.4 V
Control board	NI PXI 8119
Input/output card	NI PXI 6123/6133
Rotary potentiometer	SV01A103
Pressure transducer	FSR402

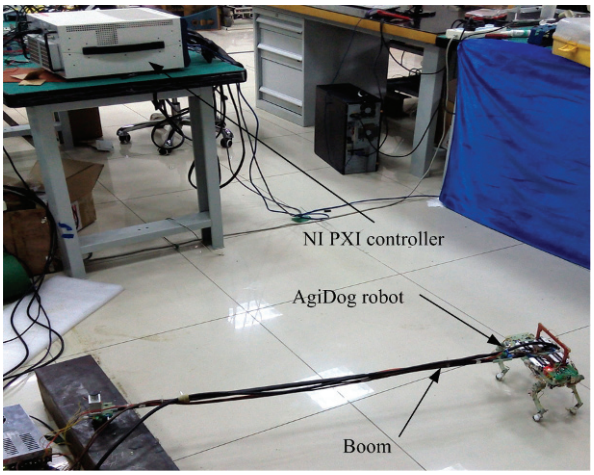

Fig. 5 The experimental setup.

4.2 Experimental results

4.2.1 Galloping before learning

To test the control framework presented in section **2**, the control strategy without learning control was applied to the AgiDog robot, the results are showed in Fig. 6. The control parameter vector (T, L_{var}, V_{var}) was set to value (350, 15, 0.3) which can be chosen by trial and

error. The motions of the robot did not seem very smooth. The pitch angle motions varied dramatically from one cycle to another and were not symmetrical relative to the zero. The average horizontal velocity deviated far from the target speed, and the height variation amplitude of the COG reached up to 50 mm. However, the AigDog did not fall down which indicates that the control framework is effective and good enough to initialize the galloping gait for learning.

4.2.2 Performances of the modified LWPR algorithm

Fig. 7 shows the comparison of the computational efficiency between the LWPR and modified LWPR algorithms. The two algorithms learnt with the same sample data, the modified algorithm adopted here took 641 ms to learn 1000 sample data, while the original LWPR algorithm took 1669 ms. This is mainly because the LWPR algorithm builds one LWPR model for each output, but the modified algorithm approximates all outputs in one LWPR model while preserving the projection directions. So the computational efficiency of the modified LWPR algorithm is almost 3 times higher than the original one.

RFs are the segmented regions of input space for better estimating the target function. Both the number of the RFs and metric of the RFs adapt for the prediction error. As shown in Fig. 8, the number of the RF grew fast before 1000 training data, because more training data was out of the existing RFs at the beginning of the learning progress and the learning algorithm would assign new RFs continuously. When the number of RFs reached 35 at 1000 training data, the growing rate slowed down. The average Frobenius norm of the RFs showed a decline at 1000 training data as the existing RFs began to stop shrinking while the initial Frobenius norm of the newly added RF was small.

Fig. 9 shows the *nMSE* of the prediction error versus the number of training data. The *nMSE* is the average Mean Square Error (MSE) as an indication of

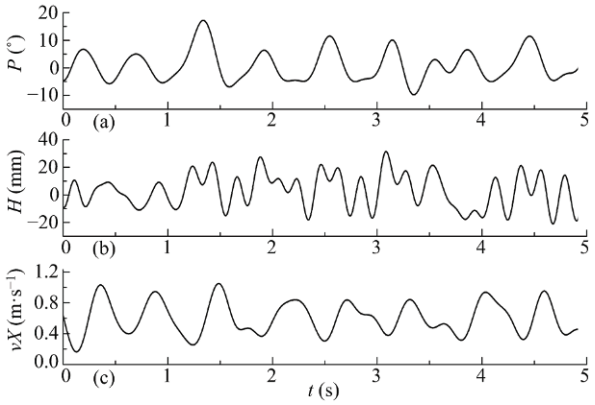

Fig. 6 Motion trajectories of the trunk of AgiDog in 1 m·s^{-1} galloping before learning. (a) Pitch of the trunk; (b) height of the COG; (c) horizontal speed.

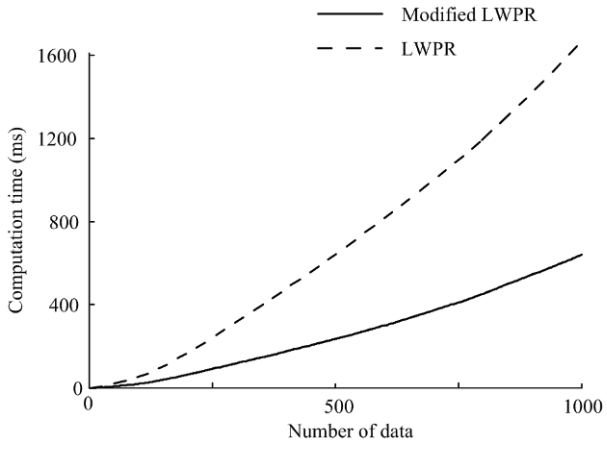

Fig. 7 Comparison of the computational efficiency between the LWPR and modified LWPR algorithms.

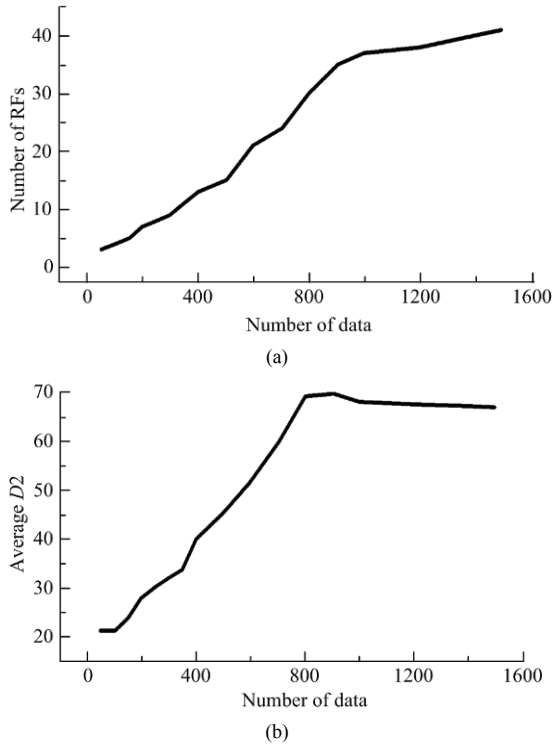

Fig. 8 Number of RFs and average Frobenius norm of the RF respect to number of training data. (a) Number of RFs; (b) frobenius norm of the RF.

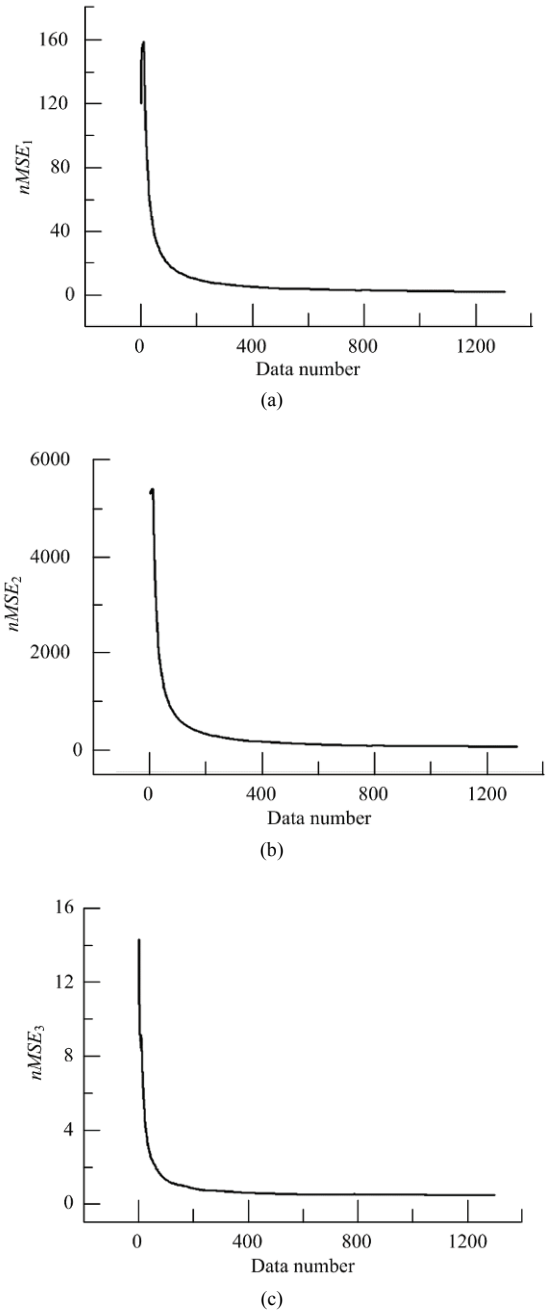

Fig. 9 Average Mean Square Error (MSE) of the prediction error. (a) Leg extension length offset; (b) target gait cycle offset; (c) velocity variation offset.

the prediction error. The *nMSE* is calculated as:

$$nMSE_{n+1} = \frac{nMSE_n + (y_o - \hat{y}_o)^2}{\mathrm{var}(Y) \cdot n}, \qquad (18)$$

where y_o is the target control function output in training data, $\hat{y}_o$ is the estimated control function output given by the LWPR model, $\boldsymbol{Y}$ is the vector consisting of all existing y_o, var(Y) is the variance of the vector $\boldsymbol{Y}$, n is the number of training data. The *nMSE* decreased dramatically at the beginning of learning process, and remained steady after 800 training data, indicating that the LWPR model had converged.

4.2.3 Performances of the learning controller

The control parameters were determined by the learning algorithm to improve the galloping gait. As shown in Fig. 10a, the average speed before learning was smaller than 0.6 $m \cdot s^{-1}$, while the average speed reached 0.9 $m \cdot s^{-1}$ after learning, though it still had a steady-state error to the target speed 1 $m \cdot s^{-1}$, the performance had a significant improvement. The apex height in galloping gait rose to 10 mm height, which meant that the robot began to have the flight phase. The average pitch angle did not stay near to the zero horizontal line, but the absolute value had a decreasing trend.

To test the robust of the controller, the AgiDog was commanded to run over a stair obstacle and the snapshots are shown in Fig. 11. The stair has two steps while each step is 3 mm height. The robot experienced a little unstable when it firstly ran up the stairs, but regained

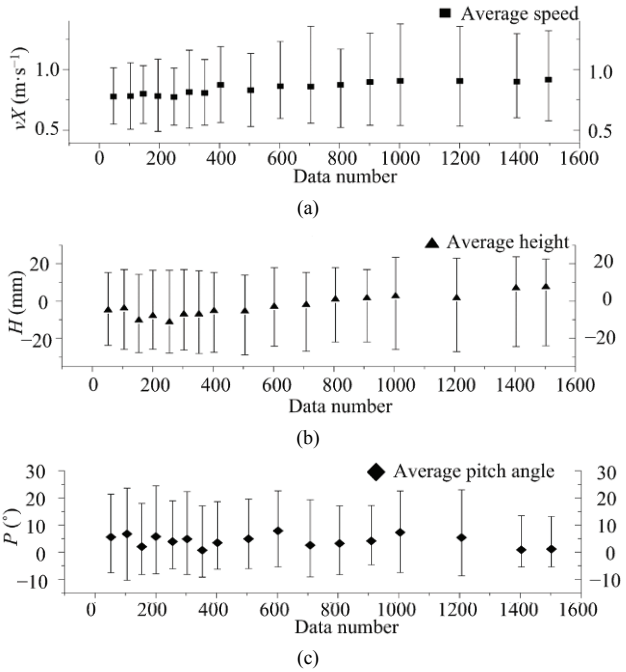

Fig. 10 Evolution of the robot motions during learning process, the scatter lines indicate the average value in the gait, the upper bound is the maximum value, and the lower bound is the minimum value. (a) Horizontal speed; (b) height of the COG of robot trunk; (c) pitch angle of the robot.

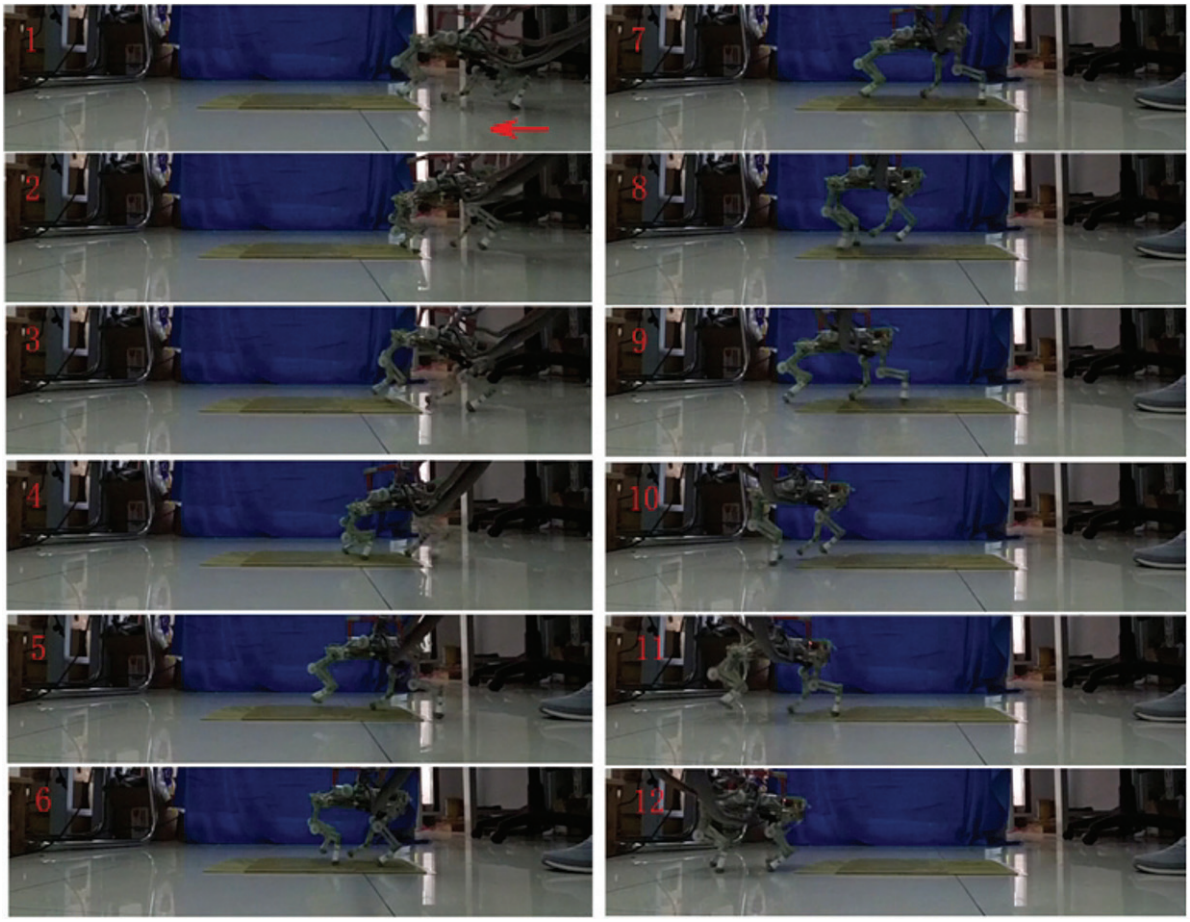

Fig. 11 AgiDog runs over obstacle under the learned controller at the speed of 0.8 $m \cdot s^{-1}$.

dynamic stability after it went to the top of the stair, which shows that the controller is robust to the external disturbances.

4.3 Discussion

Though machine learning is a powerful tool to solve many control problems, it is less successful in quadruped control, especially in quadruped galloping control. One reason is that acquiring sample data is difficult when real quadruped is in the control loop. The quadruped may fall down before the control strategy is learned, which will cause damage of the robot or at least a restart of the system. So most of the researchers study learning galloping control in simulation scenario[8,12–14]. Thus, in learning control approach, the ability to keep the robot from falling before learning is crucial for the control strategy. As show in Fig. 6, the control strategy presented in this paper successfully stabilized the galloping quadruped, which makes it possible for the learning algorithm to improve the controller performance. As far as we know, this is the first learning control framework applied on real quadruped galloping control.

Calculation efficiency is critical for a learning algorithm to learning online in real quadruped control. The gait frequency may reach 5 Hz in our galloping control experiment. The algorithm receives the gait data for learning at the apex of the flight phase, and renews the learning model to output the control parameters for the next gait cycle before landing. In the original LWPR algorithm[18], the algorithm will build one LWPR model for each output in multi-output case or, for simplification, determine all outputs in one LWPR model by using the same projection directions. In this paper, we combine the two advantages of the existing method to learn the control function in one LWPR model while preserving the projection directions for each output. As shown in Fig. 7, the modified algorithm is almost three times faster than the original one. And the Figs. 8 and 9 show that the

modified algorithm also convergences very fast and does not lose much calculation precision.

An advantage of the learning approach is that it can improve the control performance by finding appropriate control parameters for different states during running. As shown in Fig. 10, the average speed, height of the apex in flight phase and the pitch angle are all improved after learning. Lately, the MIT cheetah realized a high speed trot-galloping gait transition, but many parameters in the control strategy rely much on manual tuning[26]. Thus, learning approach is a very promising method to quadruped galloping control.

Fig. 11 shows the quadruped galloping over a stair like obstacle. The purpose of the obstacle negotiating experiment is to demonstrate the robustness of the presented control strategy. Though galloping over a 3 mm height obstacle is not very impressive for a 15 cm height quadruped, it does show that the quadruped can gallop under the ground disturbances. To negotiate more challenging obstacles, the quadruped needs visual perception to identify the obstacles, which is not discussed in this paper.

5 Conclusion

This paper presents a learning control approach to quadruped robot galloping. The control framework is designed to mimic gait characteristics of the quadrupedal animals, in which the control issue of the quadruped galloping is converted to the determination of three motion control parameters. To determine the parameters, the control function mapping from five state variables to the three control parameters is defined. We modify the LWPR learning algorithm so that it can approximate multi-output control function in one single LWPR model while ensuring each output has its own projection directions, which improves the calculation efficiency. The learning control approach controlled the bioinspired quadruped robot AgiDog to gallop successfully. The running performance is improved continuously during learning. The robot could run over a stair-like obstacle which shows that the controller is robust. The learning approach presented in this paper provides a practical control framework for galloping gait, which may even extend to other running gait. In the future, an on board controller and power supply will be implemented, so the quadruped could run without tether and maintain a 3D galloping gait.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant numbers 61175097 and 51475177, and the Research Fund for the Doctoral Programme of Higher Education of China (RFDP) under grant number 20130142110081.

References

[1] Raibert M H. *Legged Robots That Balance*, The MIT Press, Massachusetts, USA, 1986.

[2] Muybridge E. *Horses and Other Animals in Motion*: 45 *Classic Photographic Sequences*, Dover Publications, New York, USA, 1985.

[3] Chen D, Li N, Wang H, Chen L. Effect of flexible spine motion on energy efficiency in quadruped running. *Journal of Bionic Engineering*, 2017, **14**, 716–725.

[4] Nie H, Sun R, Hu L, Su Z, Hu W. Control of a cheetah robot in passive bounding gait. *Journal of Bionic Engineering*, 2016, **13**, 283–291.

[5] Park H W, Wensing P M, Kim S. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *International Journal of Robotics Research*, 2017, **36**, 167–192.

[6] WildCat, *The World's Fastest Quadruped Robot*, [2017-12-06], https://www.bostondynamics.com/wildcat

[7] Poulakakis I, Smith J A, Buehler M. On the dynamics of bounding and extensions: Towards the half-bound and gallop gaits. *Adaptive Motion of Animals and Machines*, 2006, 79–88.

[8] Krasny D P, Orin D E. Evolution of a 3D gallop in a quadrupedal model with biological characteristics. *Journal of Intelligent & Robotic Systems*, 2010, **60**, 59–82.

[9] Leonov G, Nijmeijer H, Pogromsky A, Fradkov A. *Dynamics and Control of Hybrid Mechanical Systems*, World Scientific, Singapore, 2010.

[10] Nanua P. *Dynamics of a Galloping Quadruped*, Ohio State University, Ohio, USA, 1992.

[11] Ringrose R. Self-stabilizing running. *Proceedings of the IEEE International Conference on Robotics and Automation* (*ICRA*), New Mexico, USA, 1997, 487–493.

[12] Herr H M, McMahon T A. A galloping horse model. *International Journal of Robotics Research*, 2001, **20**, 26–37.

[13] Palmer L R, Orin D E. Intelligent control of high-speed

turning in a quadruped. *Journal of Intelligent & Robotic Systems*, 2010, **58**, 47–68.

[14] Marhefka D W, Orin D E. Fuzzy control of quadrupedal running. *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000, 3063–3069.

[15] Wright J, Jordanov I. Intelligent approaches in locomotion – A review. *Journal of Intelligent & Robotic Systems*, 2015, **80**, 255–277.

[16] Chae G, Park J H. Galloping trajectory optimization and control for quadruped robot using genetic algorithm. *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Sanya, China, 2007, 1166–1171.

[17] Kober J, Bagnell J A, Peters J. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 2013, **32**, 1238–1274.

[18] Vijayakumar S, D'Souza A, Schaal S. Incremental online learning in high dimensions. *Neural Computation*, 2005, **17**, 2602–2634.

[19] Missura M, Behnke S. Online learning of bipedal walking stabilization. *KI-Künstliche Intelligenz*, 2015, **29**, 401–405.

[20] Reiser R F, Peterson M L, Kawcak C E, Mcllwraith C W. Forelimb hoof landing velocities in treadmill trotting and galloping horses. *Society for Experimental Mechanics*, Portland, USA, 2005.

[21] Witte T H, Hirst C V, Wilson A M. Effect of speed on stride parameters in racehorses at gallop in field conditions. *Journal of Experimental Biology*, 2006, **209**, 4389–4397.

[22] Heglund N C, Taylor C R. Speed, stride frequency and energy cost per stride: How do they change with body size and gait? *Journal of Experimental Biology*, 1988, **138**, 301–318.

[23] Smith J L, Chung S H, Zernicke R F. Gait-related motor patterns and hindlimb kinetics for cat trot and gallop. *Experimental Brain Research*, 1993, **94**, 308–322.

[24] Liu Q, Chen X, Han B, Luo Z, Luo X. Virtual constraint based control of bounding gait of quadruped robots. *Journal of Bionic Engineering*, 2017, **14**, 218–231.

[25] Fischer M S, Blickhan R. The tri-segmented limbs of therian mammals: Kinematics, dynamics, and self-stabilization – A review. *Journal of Experimental Zoology Part A*: *Comparative Experimental Biology*, 2006, **305**, 935–952.

[26] Hyun D J, Lee J, Park S I, Kim S. Implementation of trot-to-gallop transition and subsequent gallop on the MIT Cheetah I. *International Journal of Robotics Research*, 2016, **35**, 1627–1650.