Mixed integer programming-based semi-autonomous step climbing of a snake robot considering sensing strategy

Kazuyuki Kon, *Member, IEEE*, Motoyasu Tanaka, *Member, IEEE* and Kazuo Tanaka, *Fellow, IEEE*

***Abstract*—We propose a control method for semi-autonomous step climbing by a snake robot. Our method is based on mixed integer quadratic programming to generate the reference trajectory of the head of the snake robot online. One of the features of the method is that it determines suitable positions and time duration in which to sense the surroundings before approaching the step. Furthermore, constraints on velocity and acceleration are taken into account, so that the snake robot can securely follow the generated trajectory. Our method was applied to a snake robot equipped with a laser range finder, which is used for step detection. Experiments were performed to verify the efficacy of the method.**

***Index Terms*—Trajectory generation, Step climbing, Mixed integer programming, Sensing strategy, Snake robot**

I. INTRODUCTION

DESPITE their simple limbless bodies, snakes can travel over various terrains, such as grassland, rocky outcrops, sand, water, and trees. Snake robots, i.e., long, thin-bodied robots imitating snake motion, are also expected to be actively used in such terrains. However, because snake robots are constructed with many joints and propel themselves by pushing off surfaces, their operation is necessarily complex and difficult.

Several studies have reported on motion strategies for snake robots in two-dimensional environments [1]−[9], and recently have been reported on locomotion in three-dimensional environments [10]−[19]. Non-smooth environments such as cluttered environment, Liljebäck proposed a control strategy of obstacle-aided-locomotion where the robot generates propelling forces by pushing obstacles in cluttered environment where there are many circular objects [14]. On non-smooth environments such as steps and stairs, the snake robot has to move by changing ground surfaces to land on. In this case, its body is flexed so as to make contact with surfaces in its immediate surroundings, and has to maintain that contact during its forward motion. In climbing steps and stairs, Lipkin proposed a method whereby the body curves are determined experimentally [15], whereas Yamada proposed a method where the body curve is determined by approximately discretizing a continuous curve for an active wheeled snake robot [18]. Liljebäck proposed a framework for the reference trajectory of the three-dimensional motion using transfer points and a continuous shape curve, and demonstrated it on the case descending a stairs by simulations [19]. However, regarding tracking arbitrary trajectories, no mathematical proof has been presented for these methods.

K. Kon, is with NEC Corporation, Tokyo, Japan (e-mail: xeno1983@gmail.com)

M. Tanaka and K. Tanaka are with the Department of Mechanical Engineering and Intelligent Systems, The University of Electro-Communications, Tokyo, Japan (e-mail: mtanaka@uec.ac.jp, ktanaka@mce.uec.ac.jp)

We have also proposed a control method that achieves trajectory tracking of the head of a snake robot in stepped settings, where two horizontal plane surfaces are displaced vertically [20], [21]. In our method, the snake is modeled two-dimensionally using grounding conditions for the wheels of the snake to be switched dynamically and the length of the projections of links onto the plane to be changed. However, we assumed that the height and position of the step were given in advance, and also that the position of the robot can be observed. The assumptions are quite strong (almost unrealistic), and hence the method requires extending so as to achieve step climbing in any surroundings, where in practice height and position are unknown. If the body of the snake robot maintains contact with the two surfaces by using its head adeptly, then the control methods described in [20], [21] can be applied and can accomplish the climb of following links by changing sequentially the connecting part. Thus, traversing an arbitrary step of unknown height and position can be made by planning a trajectory for the head of the robot. Moreover, the difference between climbing and descending a step can be easily observed. The robot can achieve descent by lowering its head until it touches the floor. In contrast, in climbing the robot has to observe the step in advance, and raise its head sufficiently to avoid collision with the step. Thus, climbing is more difficult than descending, and requires a specific planning of head movements.

Nevertheless, research addressing step/stair climbing autonomously/semi-autonomously can be found for search & rescue robots [22], [23], wheeled chairs [24], [25], a jumping robot [26], legged robots [27], [28], and humanoid robots [29], [30]. Unlike these robots, a snake robot moves at a much lower position, from where recognizing surroundings is unsuitable. An elevated head position would make that task much easier. A snake robot can use its body not only for locomotion, but also to change viewing posture. Thus, it is important to consider an observation strategy in which position and time duration for observation is determined appropriately.

Descriptions of the locomotion of snake robots can be found in the literature [31], [32]. Hatton [31] proposed a motion strategy that selects pre-determined gaits for a snake robot. Liljebäck [32] proposed the waypoint guidance control method based on a controller that determines a straight line path for the center of mass to follow [5]. However, the limitations of these methods are that they are restricted to a two-dimensional space and are difficult to apply when precise control of the head is required and only the designed gait or path is given. Moreover, step climbing is difficult to guarantee if loss of traction occur during movements.

In this study, we propose a control method to accomplish semi-autonomous step climbing of a snake robot. The proposed control method consists of three parts: estimation of the step, reference trajectory generation, and trajectory tracking. That is, the snake robot estimates the step height and position based on sensory data, generates a trajectory using the trajectory generation method, and then tracks the reference trajectory. The main contribution of this study is the trajectory generation method based on model predictive control (MPC), which determines on-line the control input by solving a finite horizon open-loop control optimization problem [33], [34]. A feature of MPC-based approaches is that they can take into account various constraint inequalities of optimal control problems. Indeed, the several researches which takes into account the constraints such as velocity and acceleration constraints can be found in the area of the trajectory generation problem of a vehicle robot. Specifically, the obstacle avoidance between a vehicle robot and obstacles can be represented as an inequality constraint including binary variables, the obstacle avoidance problem can be formulated as a mixed integer programming (MIP) [35]−[37]. Moreover, since collision avoidance among vehicles can be also formulated as a MIP, several studies on trajectory generation of multi-vehicle system with collision avoidance have been reported as well (e.g. [38], [39]).

Similarly, for step climbing, there are several conditions (e.g., those associated with collision avoidance) to be satisfied. Since these conditions can be represented as inequality constraints as well, MPC-based approach is a possible systematic tools for step climbing of the snake robot. MPC-based approaches for the snake robot can be found in few literatures such as individual joint control [40] and swimming motion generation [41]. However, to our best knowledge, there are no literatures related to MPC-based approach in three-dimensional environments such as step climbing because of its complexity of the model.

To accomplish feasible step climbing motion based on MPC, the proposed method generates a trajectory only for the head of the snake robot. The trajectory tracking controller is also applied to follow the generated trajectory. More specifically, by representing the necessary conditions for step climbing of the snake robot as inequality constraints and formulating the problem as a MIP, we propose the control method to accomplish the step climbing. With a laser range finder (LRF) attached to the head of the snake robot, our control method uses the robot's characteristics to generate trajectories, taking into account the observed position of the step and time durations. This method takes into account velocity and acceleration constraints, and uncertainties in estimating step height and position. In contrast

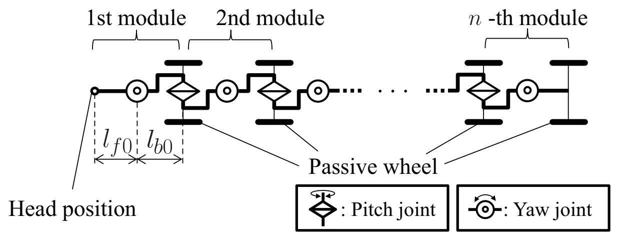


Fig. 1. Diagram of a snake robot with n modules[21].

to previous work [31], [32], because our method updates the reference trajectory on-line, loss of traction poses no difficulty during locomotion. Therefore, step climbing is achieved semi-autonomously, as the operator sets only the forward velocity. The robot moves along at the desired forward velocity and raises its head as the step nears, In applying our method to the snake robot, we performed experiments to evaluate its efficacy.

The organization of the paper is: Section II describes the target robot and surroundings and clarifies the control objective. Section III details the trajectory generation method for step climbing. Then, having applied the method on our snake robot, we present results of tests validating the method in Section IV. Finally, Section V concludes this paper and indicates some future research objectives.

II. Problem formulation

A. The snake robot and step conditions

In our study, we consider a three-dimensional n-module snake robot (Fig. 1). The wheeled module of this robot has a yaw rotational joint and is connected in series via a pitch rotational joint. The module also has a pair of coaxially passive wheels; the velocity constraints imposed assumes no slippage if the wheels are in touch with the ground. The snake robot can perform locomotion similar to a live snake by flexing its joints appropriately within the velocity constraints imposed. Using touch sensors, the robot senses the ground contact of the head and each wheel. We denote l_{f0} as the length from the anterior end of the link to the axis of the yaw joint, and l_{b0} as the length from the posterior end of the link to the axis of the yaw joint. Moreover, let ϕ_i be the yaw joint angle of the i-th module, ψ_i be the pitch joint angle between the i-th and $i+1$-th module, and define vectors $\boldsymbol{\psi} = [\psi_1, \cdots, \psi_{n-1}]^T$, and $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_n]^T$.

Fig. 2 depicts the target conditions of the snake robot in our study. The target conditions consists of two horizontal plane surfaces (set in the xy plane with respect to the global frame Σ_{xyz}) with vertical step height h along the z axis. Because the position and height are *a priori* unknown, the robot must estimate this information from sensory data. In an obvious manner, we call the two plane surfaces the "front plane" and "rear plane".

In this study, we assume the following conditions for the sensors and the surroundings (see Fig. 3):

Assumption 1: The step height h is shorter than the module length $l(:= l_{f0} + l_{b0})$

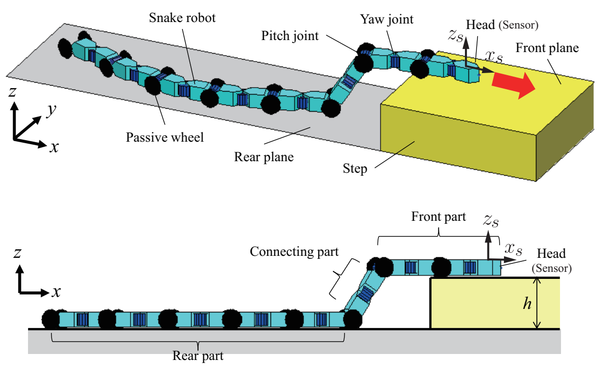


Fig. 2. Snake robot landing on the front plane of a step.

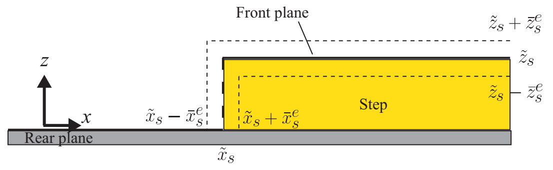

Front plane
$\tilde{z}_s + \bar{z}_s^e$
$\tilde{z}_s$
$\tilde{z}_s - \bar{z}_s^e$
z
x
$\tilde{x}_s - \bar{x}_s^e$
$\tilde{x}_s + \bar{x}_s^e$
Step
Rear plane
$\tilde{x}_s$

Fig. 3. Step estimation and its uncertainty.

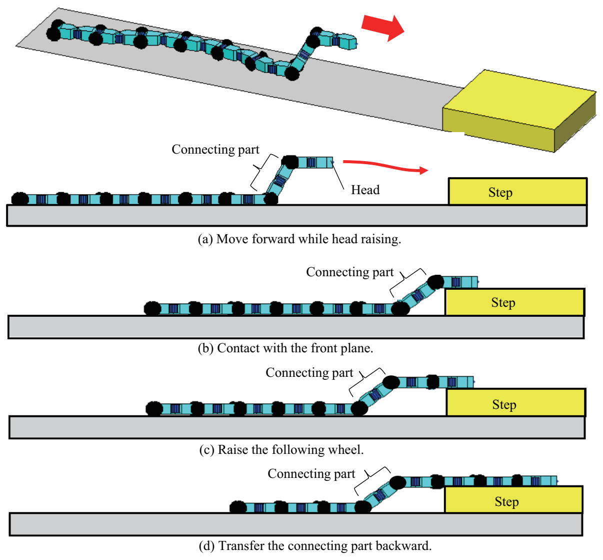

Connecting part
Head
Step
(a) Move forward while head raising.
Connecting part
Step
(b) Contact with the front plane.
Connecting part
Step
(c) Raise the following wheel.
Connecting part
Step
(d) Transfer the connecting part backward.

Fig. 4. Head raising of a snake robot in approaching the step.

Assumption 2: The robot is equipped with a LRF and can obtain the distance information on the sensory frame $\Sigma_{x_s z_s}$.
Assumption 3: The true step height lies within a bounded interval around the final estimated value $\tilde{z}_s$ with an upper error bound of $\bar{z}_s^e$. That is, the true step height lies in interval $[\tilde{z}_s - \bar{z}_s^e, \tilde{z}_s + \bar{z}_s^e]$.
Assumption 4: The distance to the step (i.e., $\tilde{x}_s$) can be estimated from the LRF data, and its upper bound of the error can be given in advance as $\bar{x}_s^e$. That is, the true step distance lies in interval $[\tilde{x}_s - \bar{x}_s^e, \tilde{x}_s + \bar{x}_s^e]$
Assumption 5: The estimation of the step height converges so as to satisfy Assumption 3 within finite number of observations with lower bound denoted by $\underline{m}$.
Assumption 6: Observations with sensors and the trajectory updates are done within sampling time δ.
Assumption 7: The robot detects whether it has settled on the front plane using sensors attached to its body.
Assumption 8: The step position can be established by detecting its lower region, but not its step height.
Assumption 9: The snake robot is controlled so as to maintain a constant relative yaw angle to the step.
Assumption 10: The desired forward velocity v_x^d with respect to the x-direction is given by the operator.

In estimating step position and height, sensory data is needed, but uncertainty in sensory data produces large uncertainties in the estimated step height and position. To achieve better estimates with smaller estimation error, we imposed the condition that the robot records environment data more than $\underline{m}$ times. By Assumption 9, we reduce the problem to a two-dimensional plane. Note that we also assume that the step height is shorter than one link length l (i.e., Assumption 1), because of physical limitations imposed by the actuators. Nevertheless, our method can be applied if $z_s > l$ as well, if we can overcome these hardware limitations. Indeed, the controller in [21] can be also applied to situations where $z_s > l$.

B. Step climbing motion

The snake robot has to raise its head above the level of the step to ensure the transition from the rear plane to the front plane. We therefore apply the control model and the controller described in [20], [21] to achieve this motion. We call the flexed segments of the body of the snake robot ensuring contact with the front plane and the one with the rear plane as "connecting part" as depicted in Fig. 2. The snake robot can move forward between the two planes by shifting the connecting part backward along its body as it advances. The robot's head can track the desired trajectory during climbing a step using control method in [20], [21]. The control method for step climbing begins by changing the connecting part in accordance with the number of its non-contacting wheels. As this method assumes that the step and robot position are given or can be obtained without error, the number of the non-contacting wheels can be calculated. Although the snake robot in this study cannot observe the position directly, it can sense contacts from the sensors it has.

In detail, we accomplish the step climbing by the following procedure. Firstly, the snake robot starts to move while all wheels contacting with the rear plane. The snake robot moves forward while estimating the height and position of the step with the equipped LRF. When the step is found, the snake robot raise its head as depicted in Fig. 4(a). In Fig. 4(a), the connecting part exists just behind the head. The snake robot continues to move forward while raising its head higher than the estimated step height. The head arrives upon the front plane, then the robot descends its head until contacting with the front plane (Fig. 4(b)). After contacting with the front plane, by transferring the connecting part backward along its body as it advances, the following links climb the step (Fig. 4(c), (d)).

The main contributions of this study are the step estimation and trajectory generation in the above procedure. Note that, although we can control the height of the head segment of the snake robot by the method proposed in [4], [8], the method requires switching of the controller after settling on the step. One of the reasons we use the controller in [20], [21] is that it achieves step climbing smoothly without any controller switching.

C. Control objective

As mentioned above, we aim to accomplish step climbing of a snake robot semi-autonomously. The step climbing of the snake robot is to navigate the robot, which starts from the rear plane, to the front plane, i.e. to achieve $x_b \geq \tilde{x}_s + \bar{x}_s^e$ and $z_b \geq h$ where $[x_b, z_b]$ the arbitrary point on the body of the snake robot. Under the assumptions in Section II-A, the control objective in this study is to accomplish semi-autonomous step climbing in a surrounding, where height and distance of the step are unknown with the snake robot by

- moving with the desired forward velocity v_x^d set by the operator as much as possible (i.e. minimize $|v_x - v_x^d|$, where v_x is the forward velocity of the snake robot),
- recognizing surroundings using the sensory data gathered,
- avoiding unintended collisions with environment, and
- raising and lowering the head of the snake robot according to conditions encountered.

To this end, we generate a reference trajectory in which the head of the snake robot is raised and lowered in the xz plane, and apply the trajectory tracking controller of [20], [21] to track the reference trajectory. The reference trajectory is generated based on the MPC method in which the viewing position, the uncertainty of sensory data, and the tracking performance of the snake robot are taken into account. Note that, if the robot's head can travel in a path between the two plane surfaces, our control method can be applied in transitioning its posterior segments. Therefore, we only focus on the trajectory generation problem associated with head transitions.

III. Trajectory generation of the head for step climbing

A. Basic strategy

First, we define five sequential states in step climbing determined in accordance with observations and movements (Fig. 5):

State 1 (Landed state) is the state when the head of the snake robot lands on the front plane or the rear plane.
State 2 (Head-raising state) is the state when the snake robot is raising its head.
State 3 (Observation state) is the state when the snake robot observes its surroundings from a view higher than the step.
State 4 (Approaching state) is the state when the snake robot approaches the step.
State 5 (Landing state) is the state when the head is descending to land on the front plane.

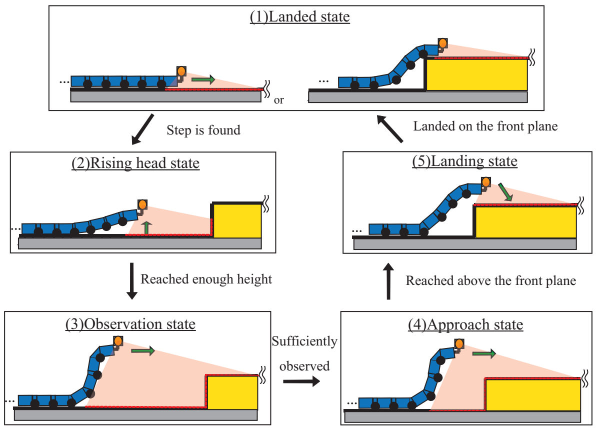


Fig. 5. Defining the inherent states in step climbing.

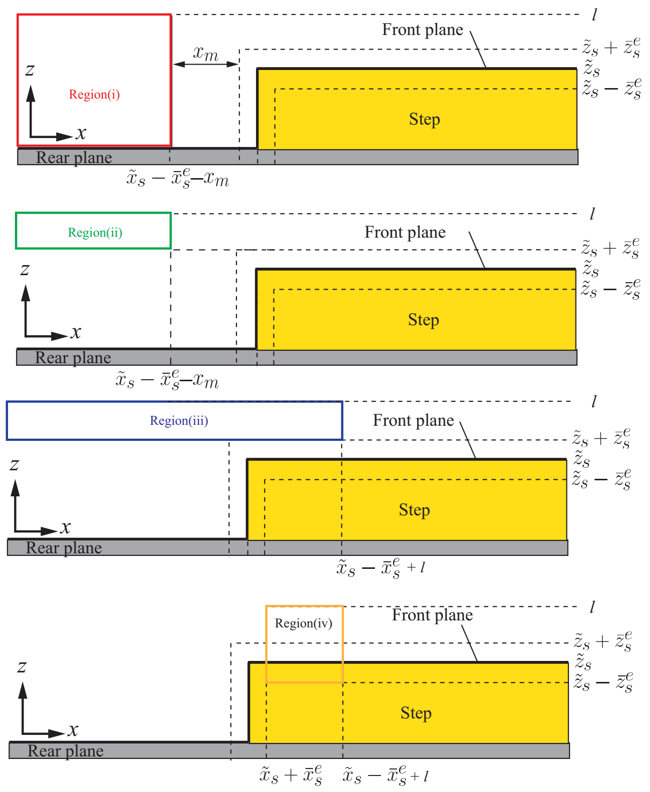
Our method generates a trajectory that sequentially transits States 1, 2, 3, 4, 5, and 1 (i.e., landed on the front plane). These states correspond to the following regions defined by the robot's relative position to the step (Fig. 6)

Region (i) is the area where the robot raises its head toward Region (ii) so as to observe the front plane associated with State 2. x_m denotes the margin from the step.
Region (ii) is the area in front of the step and above the front plane associated with State 3.
Region (iii) is the area above the front plane associated with State 4.
Region (iv) is the area directly above the front plane from where the head can reach from the corner point (i.e. from $\tilde{x}_s + \bar{x}_s^e$ to $\tilde{x}_s - \bar{x}_s^e + l$). It is associated with State 5.

To summarize, by transiting the four regions sequentially, we achieve the control objective, i.e., observation of the step, climbing of the step, and landing on the front plane.

The overall outline of the step climbing process is shown in Fig. 7. Our method consists of step detection, trajectory generation and trajectory tracking. The details of the procedures are also shown in Algorithm 1. More precisely, our method obtains the sensory data from the LRF, and estimates the height and position of the step according to the method summarized in Appendix A. If a step is found, the method solves the optimization problem given in Section III-E to generate a reference trajectory after updating the necessary number of observations. If no step is found, the reference trajectory is generated according to equation (26). The snake robot follows the reference trajectory under guidance from the tracking controller, as described in Appendix B. The above procedures are repeated every sampling period until the front plane is reached.

The main contribution of this paper is the optimization problem for step climbing (i.e., line 7 in Algorithm 1). The proposed optimization problem consists of the constraint inequalities that describe the above-mentioned conditions. In the following subsections, we introduce a prediction model, a cost function, and constraints used in the optimization problem.



l
Front plane
$\tilde{z}_s + \bar{z}_s^e$
$\tilde{z}_s$
$\tilde{z}_s - \bar{z}_s^e$
x_m
z
Region(i)
x
Step
Rear plane
$\tilde{x}_s - \bar{x}_s^e - x_m$
Region(ii)
Region(iii)
$\tilde{x}_s - \bar{x}_s^e + l$
Region(iv)
$\tilde{x}_s + \bar{x}_s^e$

Fig. 6. Definition of the four regions in step climbing.

B. Prediction model

To generate a reference trajectory to be tracked by the snake robot, we use the following discretized time-invariant double-integral model,

$$\begin{aligned} X(k+1) &= A_d X(k) + B_d u(k), \quad (1)\\ A_d &:= \exp(A_c\delta),\\ B_d &:= \int_0^{\delta} \exp(A_c\tau)d\tau B_c,\\ A_c &:= \begin{bmatrix} 0 & 0 & 1 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & -1 \end{bmatrix}, \ B_c := \begin{bmatrix} 0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1 \end{bmatrix}, \end{aligned}$$

where $X(k) := [x(k)\ z(k)\ v_x(k)\ v_z(k)]^T$ is the state vector of the position and velocity at $t = k\delta$, $u(k) := [u_x(k)\ u_z(k)]^T$ the control input at $t = k\delta$, and δ the sampling period. The prediction model (1) is the common control model which is used for path planning [42], [43] and swarm control [44]. One of the necessary conditions included in the trajectory tracking controller and used in [6] is that the function giving the reference trajectory is piecewise continuous. In the above model, we stipulate that this function is differentiable to achieve a smoother trajectory. In addition, we introduce the upper bound for the velocity and control input described as

$$\begin{aligned} |v_x(k)| &\le \bar{v}_x,\ |v_z(k)| \le \bar{v}_z, \quad (2)\\ |u_x(k)| &\le \bar{a}_x,\ |u_z(k)| \le \bar{a}_z, \quad (3) \end{aligned}$$

where the upper bounds $\bar{v}_x$, $\bar{v}_z$, $\bar{a}_x$, and $\bar{a}_z$ in the constraints (2) and (3) are determined according to the tracking performance of the snake robot.

C. Cost function

Next, we introduce a cost function associated with trajectory generation. For clarity, we design separate cost functions with respect to the x- and z-directions, and then integrate them into one cost function.

With respect to the x-direction, we consider a trajectory along the given reference velocity v_x^d that is as optimal as possible given the control objective. The cost function with respect to x-direction, which determines the error between the reference position x^d derived by integrating v_x^d and the position on the trajectory from $t = k\delta$ to $t = (k+N)\delta$, can be written as follows:

$$\sum_{\tau=k}^{k+N} \|x(\tau) - x^d(\tau)\|_{Q_1} + \|v_x(\tau) - v_x^d(\tau)\|_{Q_2}, \quad (4)$$

where N is the prediction horizon, $\|\mathbf{x}\|_{\mathbf{A}} := \mathbf{x}^{\,T}\mathbf{A}\mathbf{x}$ with a vector $\mathbf{x}$ and a matrix $\mathbf{A}$, and Q_1, Q_2 are the weight coefficients. Note that the reference position x^d along the x-direction is derived by integrating the reference velocity v_x^d as follows:

$$x^d(k+i) := x(k) + \int_{k\delta}^{(k+i)\delta} v_x^d(\tau)d\tau,\ \ i \in [0, N]. \quad (5)$$

Similarly, with respect to the z-direction, the reference is changed depending on the state (Fig. 5). For instance, maintaining a higher viewpoint during the head-raising and observation states is important. Thus, a height close to l, which is the upper bound, is the optimal height. Nevertheless, the purpose during approach/landing is to finally land on the front plane. The estimated step height $\tilde{z}_s$ should be a reference during the approaching/landing state. With a switching period $t_c(\in [k\delta, (k+N)\delta])$ which is the time interval in which the robot changes from its observation state to the approaching state, the reference trajectory with respect to the z-direction can be written as follows,

$$z^d(\tau) = \begin{cases} l & \tau \in (k, k_c] \\ \tilde{z}_s - \bar{z}_s^e & \tau \in (k_c, k+N] \end{cases}, \quad (6)$$

where the reference velocity v_z^d is assumed to be 0. Therefore, from equation (6), the cost function with respect to the z-direction can be derived in the same manner as for the x-direction,

$$\sum_{\tau=k}^{k+N} \|(z(\tau) - z^d(\tau))\|_{Q_3} + \|(v_z(\tau) - v_z^d(\tau))\|_{Q_4}, \quad (7)$$

where Q_3, Q_4 are the weight coefficients.

To summarize, by equations (4) and (7), we have a cost function for trajectory generation

$$\sum_{\tau=k}^{k+N} \|X(\tau) - X^d(\tau)\|_Q, \quad (8)$$

where $X^d(k) := [x^d(k)\ z^d(t)\ v_x^d(k)\ v_z^d(k)]^T$ and $Q :=$ diag$(Q_1,\ Q_3,\ Q_2,\ Q_4)$. The cost function (8) is to be

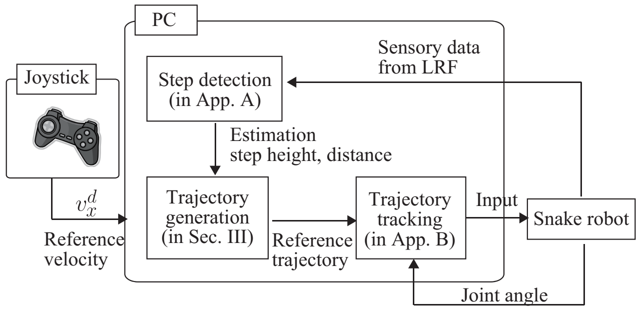

PC
Joystick
Step detection (in App. A)
Sensory data from LRF
Estimation step height, distance
v_x^d
Reference velocity
Trajectory generation (in Sec. III)
Reference trajectory
Trajectory tracking (in App. B)
Input
Snake robot
Joint angle

Fig. 7. Outline of the propose system including the trajectory generation method.

Algorithm 1 Proposed semi-autonomous step climbing method

```
1:  Initialize the necessary number of observations as $m = \bar{m}$
2:  while Not arrived on the front plane do
3:      Get LRF sensory data
4:      Detect step by Algorithm 2 (Appendix A)
5:      if the step is found then
6:          Update the necessary observation times as $m \leftarrow m-1$
7:          Solve the optimization problem (Sec. III-E).
8:      else
9:          Generate reference trajectory [Eq. (26)]
10:     end if
11:     Apply the tracking controller (Appendix B) with the generated trajectory
12: end while
```

minimized so as to reduce the errors from reference trajectory. Note that the switching period k_c in equation (6) is not given in advance, but is the variable determined by the optimization described below.

D. Inequality constraints

To generate the trajectory in accordance with the above-mentioned strategy, we introduce two inequality constraints named “Region constraint” and “Observation constraint”.

The region constraint narrows the approach and landing onto the step so as to avoid collisions with the step. It confines the head of the snake robot to one of the four regions (Fig. 6) described as follows:

$$\begin{aligned} [C_0^T, C_1^T]^T X(\tau) &\leq [D_0^T, D_1^T]^T && (9)\\ \text{or } [C_0^T, C_2^T]^T X(\tau) &\leq [D_0^T, D_2^T]^T && (10)\\ \text{or } [C_0^T, C_3^T]^T X(\tau) &\leq [D_0^T, D_3^T]^T && (11)\\ \text{or } [C_0^T, C_4^T]^T X(\tau) &\leq [D_0^T, D_4^T]^T, && (12) \end{aligned}$$

where

$$C_0=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, D_0=\begin{bmatrix} l \\ 0 \end{bmatrix}$$

$$C_1=\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, D_1=\begin{bmatrix} \tilde{x}_s-\bar{x}_s^e-x_m \end{bmatrix}$$

$$C_2=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, D_2=\begin{bmatrix} \tilde{x}_s-\bar{x}_s^e-x_m \\ -\min(l, \tilde{z}_s+\bar{z}_s^e) \end{bmatrix}$$

$$C_3=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, D_3=\begin{bmatrix} \tilde{x}_s-\bar{x}_s^e+l \\ -\tilde{z}_s-\bar{z}_s^e \end{bmatrix}$$

$$C_4=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, D_4=\begin{bmatrix} -\tilde{x}_s-\bar{x}_s^e \\ \tilde{x}_s-\bar{x}_s^e+l \\ -\tilde{z}_s+\bar{z}_s^e \end{bmatrix},$$

The inequality constraints (9)–(12) correspond to Regions (i), (ii), (iii), and (iv), respectively (Fig. 6). Essentially, collisions with the step are avoided using these region constraints, and at least one of these inequalities is satisfied for that to occur.

From the literature, constraints (9)–(12) can be written as linear inequalities by defining binary variables $\kappa_i(\tau), (i = 1,2,3,4)$ [42].

$$\begin{aligned} CX(\tau) &\leq D + M_1[\kappa_1(\tau)\ \kappa_2(\tau)\ \kappa_3(\tau)\ \kappa_4(\tau)]^T && (13)\\ \sum_{i=1}^{4} \kappa_i(\tau) &\leq 3, && (14) \end{aligned}$$

where

$$M_1 = \begin{bmatrix} 0 & 0 & M & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & M & M & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & M & M & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & M & M & M \end{bmatrix}^T,$$

$C := [C_0^T,\ C_1^T,\ C_2^T,\ C_3^T,\ C_4^T]^T$, $D := [D_0^T,\ D_1^T,\ D_2^T,\ D_3^T,\ D_4^T]^T$, and M is a sufficiently large positive number.

Next, we introduce the observation constraint. To take into account the uncertainty of the sensory data, the robot must observe its environment a sufficient number of times when in Region (ii) before entering Regions (iii) and (iv). The binary variables in the region constraint (13) and (14) correspond to Regions (i)–(iv). For instance, if κ_1 is 0 then the position on the trajectory is within Region (i). Also, if both κ_1 and κ_2 are 0, then the position on the trajectory is within the intersection of Regions (i) and (ii). Therefore, it is possible to know whether the position on the trajectory is within the observable region (i.e., Region (ii)) or not using the value of κ_2. In other words, it is possible to generate a trajectory that stays in Region (ii) over a necessary period by constraining the transition of κ_2 between steps. More precisely, focusing on κ_2, we introduce the following observation constraints,

$$\begin{aligned} \sum_{\hat{\tau}=k}^{\hat{\tau}} (\kappa_2(\hat{\tau})) &\leq (\tau - k) - m(\tau) + \kappa_5(\tau) && (15)\\ \kappa_1(\tau) + \kappa_2(\tau) &\leq 1 + M - M\kappa_5(\tau), && (16) \end{aligned}$$

where $\kappa_5(\tau)$ is an additional binary variable. The inequality constraint (15) constrains the position to within Region (i) at least for the m predetermined times. The constraint (16) forces the head position to be within Region (i) or (ii), if constraint (15) cannot be satisfied. These two constraints become active or inactive by the value of κ_5.

Imposing the above constraints in the optimization problem, a collision-avoiding trajectory that ends on the front plane can be generated after having made a sufficient number of observations.

E. Optimization problem

As stated in Algorithm 1, a trajectory from step detection to landing on the front plane is generated by solving the optimization problem. This is formulated as the mixed integer quadratic programming (MIQP) that consists of the above-mentioned prediction model (1), velocity and acceleration constraints (2) and (3), and the cost function (8).

The optimization problem at $t = k\delta$ is described as follows:

$$\min_{\hat{U},\hat{\kappa},\hat{W}} \sum_{\tau=k}^{k+N} \|\hat{X}(\tau|k) - \hat{X}^d(\tau|k)\|_Q + \|[\hat{U}^T(\tau|k)\ \hat{\kappa}^T(\tau|k)\ \hat{W}^T(\tau|k)]^T\|_R$$

subject to

$$\hat{X}(\tau+1|k) = A_d\hat{X}(\tau|k) + B_d\hat{U}(\tau|k) \tag{17}$$

$$\hat{X}(k|k) = [x(k),\ z(k),\ v_x(k),\ v_z(k)]^T \tag{18}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \hat{X}(\tau+1|k) \leq \begin{bmatrix} \bar{v}_x \\ \bar{v}_x \\ \bar{v}^z \\ \bar{v}^z \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \hat{U}(\tau+1|k) \leq \begin{bmatrix} \bar{a}_x \\ \bar{a}_x \\ \bar{a}_z \\ \bar{a}_z \end{bmatrix} \tag{20}$$

$$\hat{W}(\tau+1|k) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \hat{X}(\tau+1|k) - 2l + \begin{bmatrix} l & l & 0 & 0 & 0 \end{bmatrix} \hat{\kappa}(\tau+1|k) \tag{21}$$

$$C\hat{X}(\tau+1|k) \leq D + [{M_1}^T \mathbf{0}]^T \hat{\kappa}(\tau+1|k) \tag{22}$$

$$\sum_{i=1}^{4} \hat{\kappa}_i(\tau+1|k) \leq 3 \tag{23}$$

$$\sum_{\hat{\tau}=k}^{\tau} \hat{\kappa}_2(\hat{\tau}) \leq (\tau - k) - m(k) + M\hat{\kappa}_5(\tau) \tag{24}$$

$$\hat{\kappa}_1(\hat{\tau}) + \hat{\kappa}_2(\hat{\tau}) \leq 1 + M - M\hat{\kappa}_5(\tau) \tag{25}$$

$$\tau = k\delta, \cdots, k\delta + N,$$

where $\hat{X}(\tau|k)$ denotes the predicted values of $X(\tau)$ at $t = k\delta$, R is the weight matrix, and $\hat{\kappa}(\tau) := [\hat{\kappa}_1(\tau),\ \hat{\kappa}_2(\tau),\ \hat{\kappa}_3(\tau),\ \hat{\kappa}_4(\tau),\ \hat{\kappa}_5(\tau)]^T$. The equality constraint (17) correspond to the prediction model stated in (1) and (19), (20) is the velocity and acceleration constraints in (2) and (3) respectively. The equality constraint (21) is introduced to change the reference height depending on the state. The constraints (22) and (23) are the region constraints in (13); the switching of these is done by binary variables $\hat{\kappa}$. The inequality constraints (24) and (25) are the observation constraints in (15).

In our system, the step climbing and landing motion is achieved by following the trajectory obtained by solving the optimization problem. The constraints on the step height are set as conservative conditions (Appendix A) when the number of observations is insufficient, but with repeated observations, the constraint is also relaxed. Note that, before finding the step and after landing on the front plane, the following simple controller is used:

$$u(t) = \begin{bmatrix} k_1(v_x^d(t) - v_x(t)) + k_2(x^d(t) - x(t)) \\ 0 \end{bmatrix}, \tag{26}$$

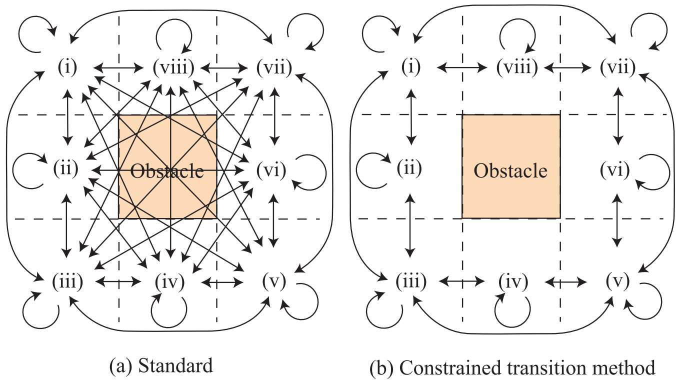


Fig. 8. Transition condition for the obstacle avoidance constraint.

where $k_x(> 0)$ is the feedback gain.

Remark 1: By replacing the sampling period δ in the optimization problem with the longer prediction period $\delta_c(> \delta)$, it is possible to generate a trajectory which predicts much longer terms. However, this might cause collisions with the surroundings between sampling periods if the sampling period δ_c becomes bigger. In such cases, imposing transition constraints that constrain the transition of the binary variables in the region constraints is worthwhile. The transition constraint [36] is introduced to eliminate the unnecessary transition which includes possible collisions as depicted in Fig. 8. In general, the condition that $\text{sgn}(v_x(t)) = \text{const.}, t \in (k\delta_c, (k+1)\delta_c]$ during a sampling period is necessary to guarantee collision avoidance. Despite this necessary condition cannot in theory guarantee within the proposed system that some trajectories, not satisfying $\text{sgn}(v_x(t)) = \text{const.}, t \in (k\delta_c, (k+1)\delta_c]$ near the step, will not be generated because of the prescribed cost function. In practice, the transition constraint works well to prevent the above-mentioned problem.

Remark 2: Although our trajectory generation method was applied to a snake robot, the method can also be applied to hypermobile robots whose heads can be arbitrarily controlled (e.g. [45]).

For reader convenience, we show numerical examples illustrating how our method generates a trajectory for the head of the snake robot.

Example 1: Suppose the step position is at $x = 1$[m] and its height is $h = 0.15$[m]. We assume a range of uncertainty such that position is within $0.97 \leq x \leq 1.03$ and height within $0.12 \leq z \leq 0.18$. With only this information, a trajectory is generated using our method. The head of snake is located at $(x,\ z) = (0.7, 0.0)$[m], and the system has already detected the step. Values for other parameters used in this simulation are as follows: $l = 0.25$[m], $\underline{m} = 15$, $\delta = 0.2$[s], $N = 20$, $Q = I$, $R = 0.1I$, $v_x^d = 0.2$[m/s], $x_m = 0.2$[m], $\bar{x}_s^e = 0.03$[m], $\bar{z}_s^e = 0.03$[m], $\bar{v}_x = \bar{v}_z = 0.2$[m/s] and $\bar{a}_x = \bar{a}_z = 0.2$[m/s^2]. Fig. 9 shows the simulation results without the transition constraints mentioned in Remark 1. This trajectory is one planned at the initial position. In an actual procedure, the reference trajectory is updated every sampling period as the robot moves. Moreover, from this trajectory, the head moves to the region above $\tilde{z}_s + \bar{z}_s^e$, and remains there while observing its surroundings at least 15 times. After taking

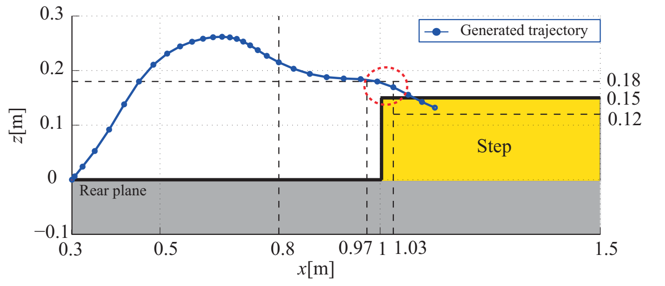


Fig. 9. Generated trajectory using our method without the transition constraint.

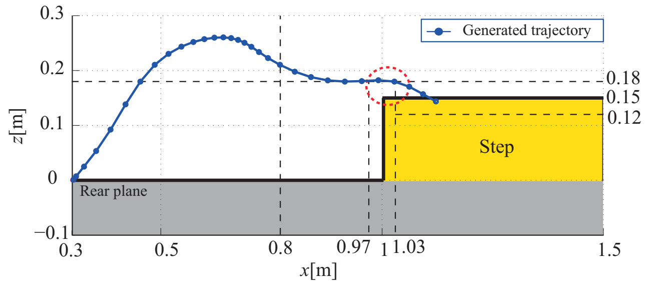


Fig. 10. Generated trajectory using our method with transition constraints.

sufficient observations, the head approaches the step and lands on the front plane. As the exact step height is not known in this simulation, the robot would land in practice on the front plane in mid-trajectory. Note that, although we omit the time response of the control input owing to space limitation, we confirmed that the trajectory satisfies the velocity and acceleration constraints as well. From this result, our method appears to work well. However, the trajectory does enter the region of uncertainty in regards to step height and position, i.e., inside the circle shown in Fig. 9. This may cause collisions between head and facing board of the step. To prevent such possible collisions, the transition constraints are imposed in a subsequent simulation (Fig. 10). In this case, we confirmed that the trajectory avoids the region of uncertainty to avoid possible collision and the head settles on the front plane. Hence, from this numerical example, the transition constraints have worked well in practice.

IV. Experiments

In this section, we apply our method to a real robot, and evaluate its effectiveness in experiments.

A. Outline of the experiments

Fig. 11 is a photo of the snake robot in its experimental surroundings. The snake robot consists of 8 modules with 15 joints, each of size $l_{f0} = l_{b0} = 0.088$[m]. The servo motors, Dynamixel MX-64R and MX-106R (ROBOTIS Inc.), are used as actuators in each joint. Because of their ease of implementation, we used infrared distance sensors with photoreflectors LBR-127HLD (Letex Technology 2 Corp) to obtain contact information. Wheel contact with the environment can be obtained based on the preliminary experimental results. Micro-computers SEED-MS1A (THK CO., LTD.) are

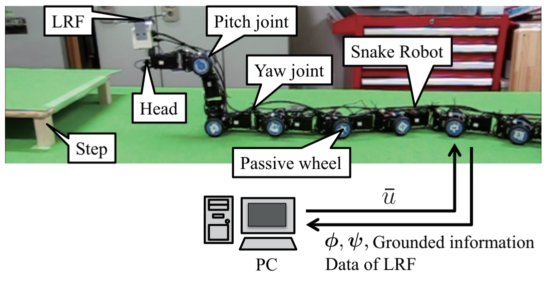


Fig. 11. Photo of the experimental system.

installed in each module, and communicate with external personal computers via CAN communications to transmit sensory data and control signals to the actuators, The snake robot estimates the distance to the step, the height of the step, and the height from the rear plane using a LRF (URG-LX04, Hokuyo Automatic) attached to the head. The details regarding the detection method are summarized in Appendix A.

The experiments were performed in surroundings with one step of height 0.1[m]. The initial head position is located 0.8[m] in front of the step, and all links including the head are in contact with the rear plane. Experiments were judged completed when the head lands on the front plane. The snake robot follows the reference trajectory generated according to the internal model, as mentioned in Appendix B. The parameter $\hat{m}$ was determined experimentally and set at $\hat{m} = 10$. Values for the other parameters are $\delta = 0.3$[s], $\delta_c = 1.2$[s], $N = 10$, $Q = I$, $R = 0.1I$, $v_x^d = 0.04$[m/s], $x_m = 0.05$[m], $\bar{x}_s^e = 0.03$[m], $\bar{z}_s^e = 0.03$[m], $\bar{v}_x = \bar{v}_z = 0.05$[m/s] and $\bar{a}_x = \bar{a}_z = 0.25$[m/s^2].

Note that we did not use the transition constraint mentioned in Remark 1 because the velocity of the robot is slow and the distance moved between sampling periods is not large. We used CPLEX [46] to solve the optimization problem, and implemented the algorithm on a personal computer (Windows 7 64bit, CPU: Intel Core i7, RAM 8GB) with MATLAB 2013a. To evaluate the trajectory of the snake robot, we observed the posture of the head using the motion capture system OptiTrack (NaturalPoint, Inc.), which was not used for guidance control.

B. Experimental results

Fig. 12 shows snapshots of an overview of a typical experiment. Figs. 13–17 presents the experimental results. As can be seen in Fig. 12, the snake robot started moving with its head down (t = 0.0[s]), and raised its head after finding the step using its LRF (t=4.0[s]). It then moved forward to approach the step with a raised head (t=10.0–50.0[s]), and finally settled its head on the front plane after reaching it (t = 55.0[s]). The trajectory generated by our method is shown in Fig. 13; the velocity and acceleration constraints are clearly satisfied. Figs. 14 and 15 show the time response of the yaw ($\phi_i, i = 1, \ldots, 8$) and pitch ($\psi_i, i = 1, \ldots, 7$) joint angles of the robot, respectively. From Fig. 14, ϕ_2, which is the yaw angle that is included among the controlled variables is set to reference value 0, whereas the rear joint angles

$\phi_3, \ldots, \phi_8$, which make contact with the rear plane, generate a periodic motion to avoid singular configurations. Moreover, from Fig. 15, the first and second pitch joint angles correspond to raising and lowering motion of the head. Fig. 16 shows the estimated step height from the LRF and the relative distance to the step. The robot is seen to find the step after around 4[s], and by raising its head detects the front plane around 12[s]. Also the estimation error becomes bigger as the robot approaches the step. This is because the LRF used in these experiments tends to generate bigger measurement error near targets. However, the errors were within ranges that did not affect the control performance. Fig. 17 shows the posture of the head of the snake robot obtained from the external motion capture system. Here we were able to confirm that the robot had settled its head on the front plane after hovering above the step (i.e., z =0.1[m]).

In Fig. 18 the differences between the x coordinate estimated by the internal model and the reference trajectory are presented. We see that the robot moves about 3[m] before landing on the front plane, whereas the actual distance is 0.8[m]. Thus, the robot has falsely recognized that it traveled almost four times the distance because of wheel slippage. However, we found that the effect of the slippage was reduced by solving the optimization problem and updating the reference trajectory online. Of course, we can reduce slippage by selecting an appropriate velocity, but we used a velocity that tended to cause loss of traction to evaluate the effectiveness of trajectory updating.

To summarize, we found our method works well. Note that, although the experiment shown in Figs. 12–18 were deemed completed when the head landed on the front plane, step climbing of the following links as well can be accomplished by applying the control strategy proposed in our previous work [20], [21] for successive motion. Indeed, we could confirmed that the proposed method accomplished the whole body step climbing. Fig. 19 shows the snapshots of an overview a whole body step climbing experiment. In Fig. 19, the inside of the red circles show the connecting part. After arriving on the front plane, the snake started to swing its head to avoid singular configuration, and shifted the connecting part using information of the ground contact of each wheel sensed by range sensors mounted on each module. As can be seen from Fig. 19, by shifting the connecting part as the snake robot moves forward, the whole step climbing was accomplished.

V. Conclusion

We have proposed the trajectory generation method for step climbing of the snake robot. Our method realizes the semi-autonomous step climbing motion based on sensory data and the given desired velocity from the operator. The step climbing problem of the snake robot is described as a MIQP. The step climbing motion is generated by solving the optimization problem on-line. One of the advantages of our method is that it takes into account velocity and acceleration constraints that depend on the tracking performance of the snake robot, and can generate a trajectory in which the robot observes its surroundings from appropriate viewing positions a sufficient number of times to minimize uncertainties in the sensory data. Furthermore, we applied our method to a real snake robot, and showed the effectiveness of our method in experiments. From results, we confirmed that our method works well even when slipping occurs.

In future work, we will extend the method to three-dimensional problems. In general, as a snake robot can use its head as a manipulator, one expects use in object recognition in three-dimensional environments. Based on advanced recognition, we want to realize more advanced motions such as stair climbing and trajectory generation with obstacle avoidance in three-dimensional spaces, so as to extend the snake robot's range of capabilities. Moreover, in order to realize more precise and effective motion planning, we would like to take into account dynamics of the snake robot in future works as well, whereas the proposed method in study is based on the kinematics of the snake robot.

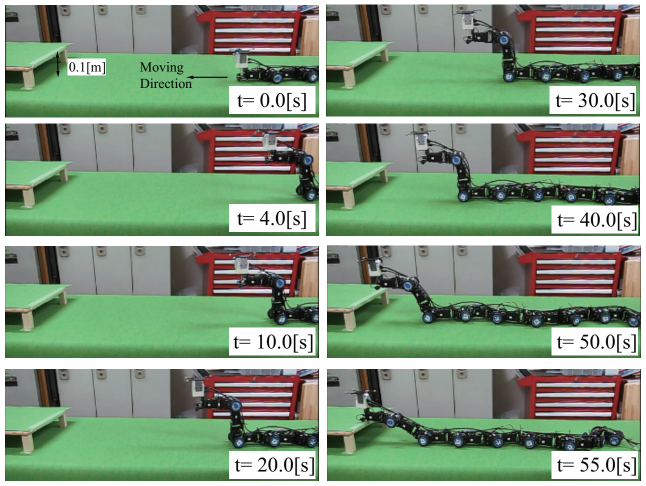


Fig. 12. Snapshots of a typical step climbing experiment.

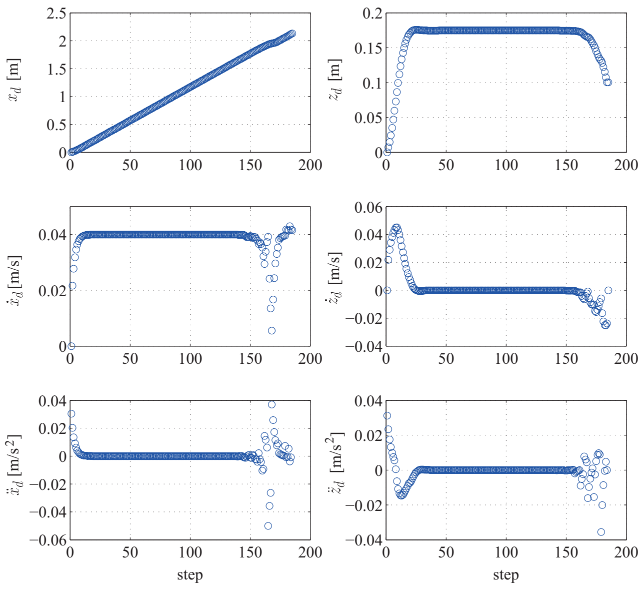


Fig. 13. Reference trajectory generated by our method.

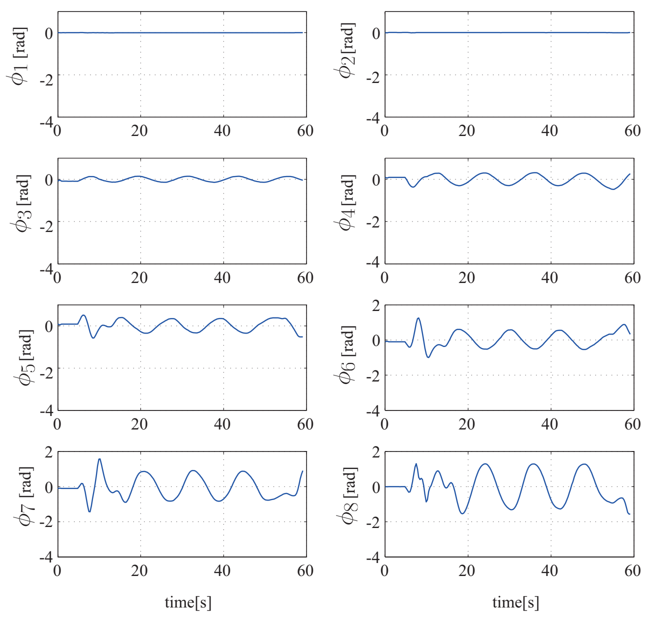


Fig. 14. Time response of the state variables (ϕ) in the experiment.

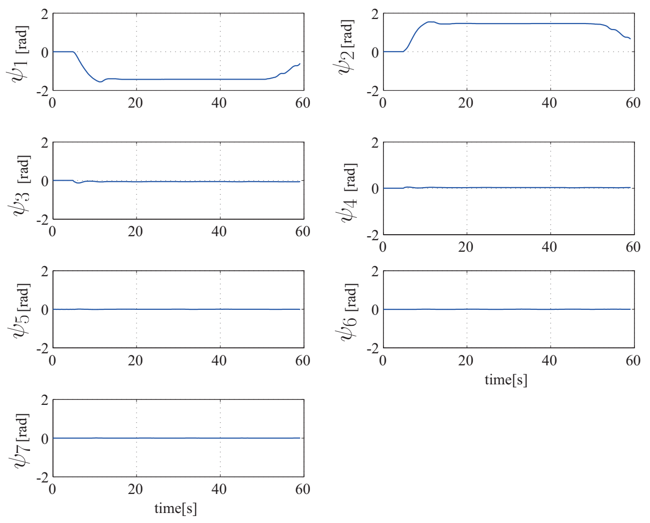


Fig. 15. Time response of the state variables (ψ) in the experiment.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 26870198, and ImPACT Program of Council for Science, Technology and Innovation (Cabinet Office, Government of Japan).

APPENDIX A
STEP DETECTION FROM LRF DATA

The snake robot is equipped with an LRF on its head (Fig. 11). It is attached so as coordinates x and x_s, and z and z_s are matched without rolling and pitching, respectively. Algorithm 2 outlines the estimation of the step height $\tilde{z}_s$, distance to step $\tilde{x}_s$, and height from the rear plane $\tilde{z}_r$ using the sensory data from the LRF.

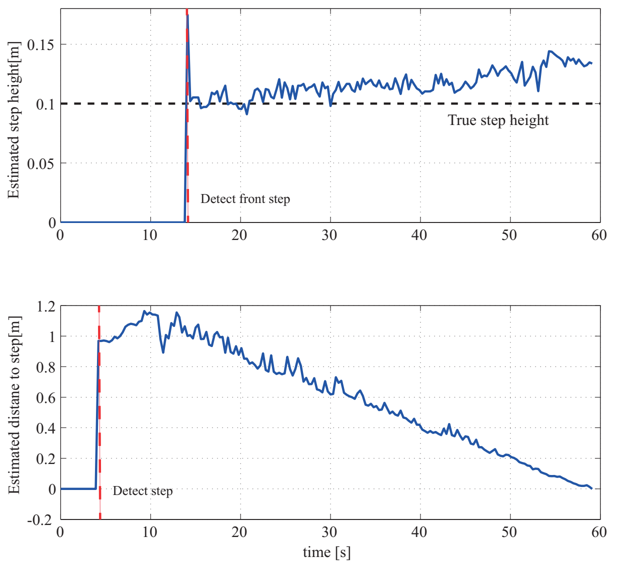


Fig. 16. Estimation of the step height.

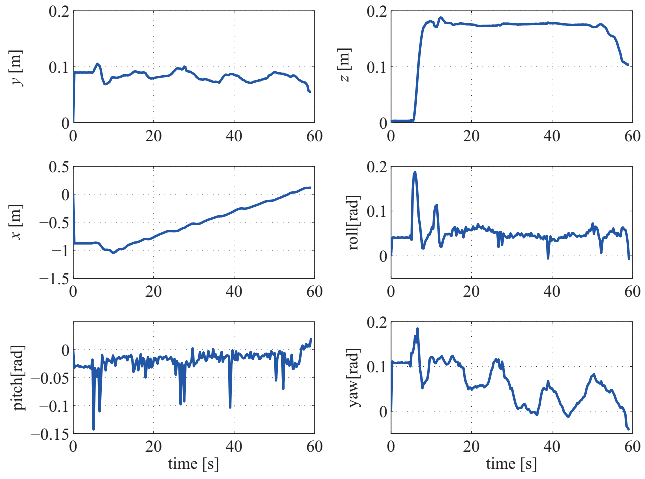


Fig. 17. Time response of the head position of the snake robot.

In Algorithm 2, the line segments corresponding to the front plane and the rear plane are determined, and then the step height is derived from the distance between the two line segments (Fig. 20). Because this distance is defined with respect to the sensory frame $\Sigma_{x_s z_s}$., we translate $\tilde{h}_s$ into a height $\tilde{z}_s$ along the z axis using the roll angle θ of the head which is calculated by integrating the joints angles.

$$\tilde{z}_s = \frac{\tilde{h}_s}{\cos(\theta)} \tag{27}$$

Note that, because the head of the snake robot is a cantilever structure, the posture of the head will include uncertainty. Also sensor noise will affect the estimation of the step height. Therefore, the step height in particular will have a bigger uncertainty. To take into account such observation errors, we estimate the step height using the extended Kalman filter.

This filter uses the roll angle of the head and the distance between the two line segments as observations. The state and

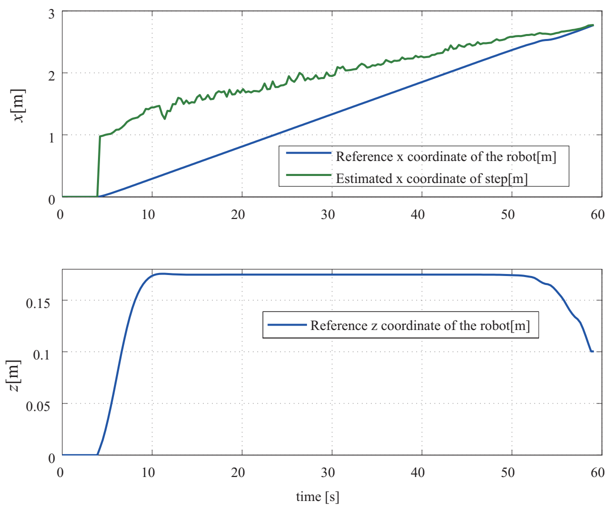


Fig. 18. Tracking error based on the internal model.

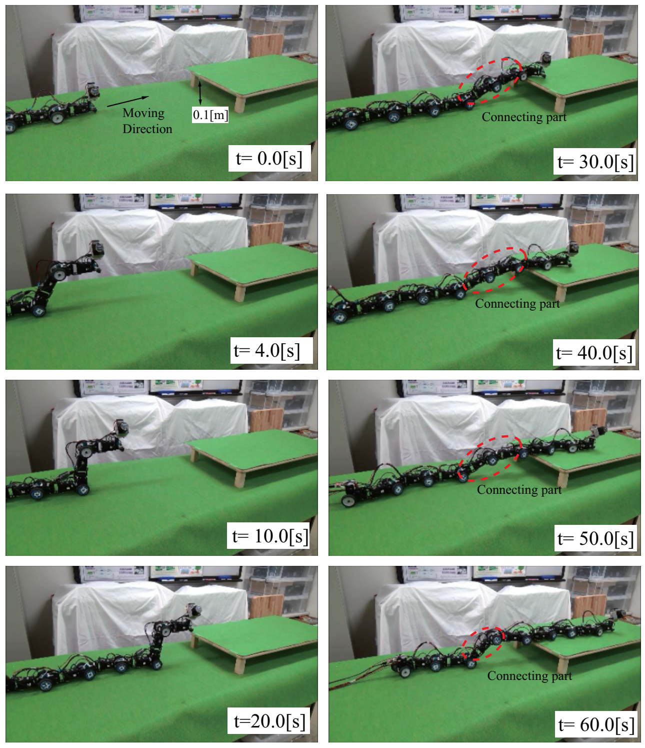


Fig. 19. Snapshots of a whole body step climbing experiment.

observation equations in the extended Kalman filter are as follows:

$$x_{k+1}^F = A^F x_k^F + B^F u_k^F + w_k \tag{28}$$
$$y_k^F = h(x_k) + v_k, \tag{29}$$

where $x_k := [\theta, z_s]^T$, $A^F = \mathbf{I}_{2\times2}$, $B^F = \mathbf{1}$, $h(x_k) := [\theta, \tilde{h}_s/\cos(\theta)]$ and $w_k \sim N(0, Q_k)$, $v_k \sim N(0, R_k)$. The initial step estimation x_0^F is set as the admissible maximum height l so as to start the estimation with conservative initial conditions with respect to trajectory generation. The above procedure is done as new sensory data becomes available, and is used in trajectory generation.

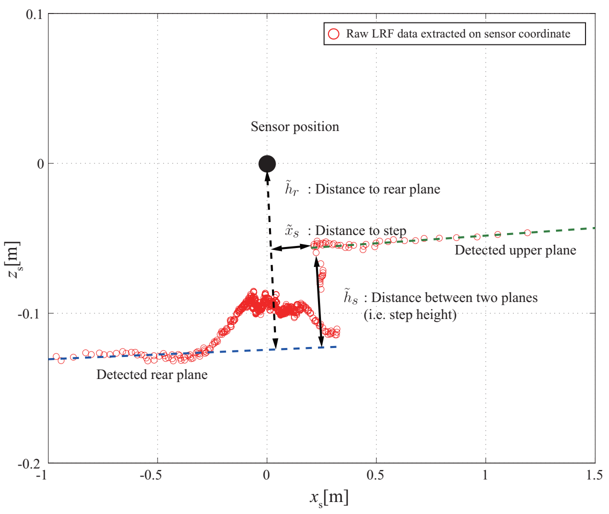


Fig. 20. Example of step detection. Distances to front and rear planes are estimated.

Algorithm 2 Find the step from LRF data

```
Require: Sensory data from LRF
1: Detect front and rear planes from LRF data based on line fitting algorithm (e.g. [47], [48])
2: if Both front plane and rear plane are detected then
3:     Calculate the distance $\tilde{h}_s$ between two planes as in Fig. 20
4:     Get $\tilde{z}_s$ from $\tilde{h}_s$ and eq. (27)
5:     Calculate the height $\tilde{h}_r$ from the rear plane in sensory frame
6:     Get $\tilde{z}_r$ from $\tilde{h}_r$ and eq. (27)
7:     Estimate step height by updating EKF defined in eq. (28) and (29)
8: end if
9: if Only front plane is detected then
10:    Calculate the distance $\tilde{x}_s$ to the front plane
11: end if
12: if Only rear plane is detected then
13:    Calculate the height $\tilde{h}_r$ from the rear plane
14:    Get $\tilde{z}_r$ from $\tilde{h}_r$ and eq. (27)
15: end if
16: return $\tilde{x}_s$, $\tilde{z}_s$, $\tilde{z}_r$.
```

Appendix B
Trajectory Tracking Controller for Snake Robot[20], [21]

In [20], [21], we proposed the tracking controller with a two-dimensional model, in which the transition of the connecting part is planned using a desired pitch angle, so that the controlled variables such as the head position and pitch joint angles converge to the reference values. In this paper, the height of the snake robot is controlled by changing the desired pitch angle of the connecting part. We omit the details of the reference pitch angle with respect to the reference height because one can derive it easily using simple kinematic relations. We refer to the unique integers derived from both the allocation state of wheeled links and that of the connecting part as “modes”, and denote the discrete mode number as σ. With the robot mode σ, the kinematic model of the robot can be described as

$$\tilde{\boldsymbol{A}}_\sigma(\boldsymbol{\theta},\boldsymbol{\psi})\dot{\bar{\boldsymbol{w}}}_\sigma = \tilde{\boldsymbol{B}}_\sigma(\boldsymbol{\theta},\boldsymbol{\psi})\bar{\boldsymbol{u}}, \tag{30}$$

where $\bar{\boldsymbol{u}}$ denotes the vector of pitch and yaw angular velocities, $\boldsymbol{\theta} = [\theta_h, \boldsymbol{\phi}^T]^T$, $\bar{\boldsymbol{w}}_\sigma$ denote the vectors of the controlled variables including the posture of the head $\boldsymbol{w} = [x_h, y_h, \theta_h]^T$ and pitch joint angles.

Let the control input $\bar{\boldsymbol{u}}$ have the following form:

$$\bar{\boldsymbol{u}} = \tilde{\boldsymbol{B}}_\sigma^{-1}\tilde{\boldsymbol{A}}_\sigma\left\{\dot{\bar{\boldsymbol{w}}}_{\sigma d} - \boldsymbol{K}_\sigma\left(\bar{\boldsymbol{w}}_\sigma - \bar{\boldsymbol{w}}_{\sigma d}\right)\right\}, \tag{31}$$

where $\boldsymbol{K}_\sigma > \boldsymbol{0}$ is the feedback gain and $\bar{\boldsymbol{w}}_{\sigma d}$ denotes the reference value of the controlled variable $\bar{\boldsymbol{w}}_\sigma$. Note that, as $\boldsymbol{w}$ cannot be observed directly, we numerically derive $\boldsymbol{w}$ by integrating the closed loop system according to equations (30) and (31). The snake robot cannot move given singular configurations such as line or circular shapes. The reference trajectory of θ_h is set as a sinusoidal wave to prevent such singularities.

However, in this study, it is not suitable to control θ_h as sinusoidal wave before the head lands on the front plane, because the LRF mounted on the head observes the step. Thus, we set the reference trajectory of ϕ_3, which is introduced in the variable $\bar{\boldsymbol{w}}_\sigma$ as an additional shape controllable point (these are the directly controllable angles corresponding to the wheelless links in the robot’s body), instead of θ_h as a sinusoidal wave before the head lands on the front plane. After the head lands on the front plane, we set the reference trajectory of θ_h as a sinusoidal wave.

REFERENCES

[1] S. Hirose, *Biologically Inspired Robots (Snake-like Locomotor and Manipulator)*, Oxford University Press, 1993.

[2] P. Prautsch, T. Mita and T. Iwasaki, “Analysis and Control of a Gait of Snake Robot”, *Trans. of IEEJ*, Vol.120-D, pp. 372–381, 2000.

[3] H. Date, Y. Hoshi and M. Sampei, “Locomotion Control of a Snake-Like Robot based on Dynamic Manipulability”, *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 2236–2241, 2001.

[4] M. Yamakita, M. Hashimoto and T. Yamada, “Control of Locomotion and Head Configuration for 3D Snake Robot”, *Proc. of IEEE Int. Conf. Robotics and Automation*, Vol. 2, pp. 2055–2060, 2003.

[5] P. Liljebäck, I. U. Haugstuen and K. Y. Pettersen, “Path Following Control of Planar Snake Robots Using a Cascaded Approach,” *IEEE Trans. on Control Systems Technology*, 20(1), pp. 111–126, 2012.

[6] F. Matsuno and K. Mogi, “Redundancy Controllable System and Control of Snake Robot with Redundancy based on Kinematic Model”, *Proc. of IEEE Conf. on Decision and Control*, pp. 4791–4796, 2000.

[7] F. Matsuno and H. Sato, “Trajectory tracking control of snake robot based on dynamic model,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 3040–3046, 2005.

[8] M. Tanaka and F. Matsuno, “Modeling and Control of Head Raising Snake Robots by Using Kinematic Redundancy,” *Journal of Intelligent and Robotic Systems*, 75(1), pp.53-69, 2014.

[9] M. Tanaka and F. Matsuno, “Control of Snake Robots with Switching Constraints: trajectory tracking with moving obstacle,” *Advanced Robotics*, 28(6), pp. 415–429, 2014.

[10] H. Date and Y. Takita, “Control of 3D snake-like locomotive mechanism based on continuum modeling,” *Proc. of ASME Int. Design Engineering Technical Conf.*, No. DETC2005-85130, 2005.

[11] H. Tsukano, M. Tanaka and F. Matsuno, “Control of a Snake Robot on a Cylindrical Surface Based on a Kinematic Model,” *Proc. of the 9th IFAC Symposium on Robot Control*, 2009.

[12] H. Yamada and S. Hirose, “Study of Active Cord Mechanism – Generalized Basic Equations of the Locomotive Dynamics of the ACM and Analysis of Sinus-lifting–,” *Journal of the Robotics Society of Japan*, 26(7), pp. 801–811, 2008. (In Japanese)

[13] H. Yamada and S. Hirose, “Study on the 3D Shape of Active Cord Mechanism,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2890–2895, 2006.

[14] K. Liljebäck, Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, “Snake Robots - Modelling, Mechatronics, and Control,” Springer, 2013.

[15] K. Lipkin, I. Brown, A. Peck, H. Choset, J. Rembisz, P. Gianfortoni and A. Naaktgeboren, “Differentiable and piecewise differentiable gaits for snake robots,” *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1864–1869, 2007.

[16] R. L. Hatton and H. Choset, “Generating gaits for snake robots: annealed chain fitting and keyframe wave extraction,” *Autonomous Robots*, Vol. 28-3, pp. 271–281, 2010.

[17] T. Kamegawa, T. Baba and A. Gofuku, “V-shift control for snake robot moving the inside of a pipe with helical rolling motion,” *Proc. of IEEE Int. Symp. on Safety, Security and Rescue Robotics*, pp. 1–6, 2011.

[18] H. Yamada, S. Takaoka and S. Hirose, “A snake-like robot for real-world inspection applications (the design and control of a practical active cord mechanism),” *Advanced Robotics*, 27(1), pp. 47–60, 2013.

[19] K. Liljebäck, Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, “A 3D Motion Planning Framework for Snake Robots,” *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 1100-1107, 2014.

[20] M. Tanaka and K. Tanaka, “Climbing and Descending Control of a Snake Robot on Step Environments based on Kinematics,” *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3285–3290, 2013.

[21] M. Tanaka and K. Tanaka, “Control of a Snake Robot for Ascending and Descending Steps,” *IEEE Trans. on Robotics*, accepted.

[22] D. M. Helmick, S. I. Roumehotis, M. C. McHenry and L. Matthies, “Multi-Sensor, High Speed Autonomous Stair Climbing,” *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 733–742, 2002.

[23] A. Kalantari, E. Mihankhah and S. A. A. Moosavian, “Safe Autonomous Stair Climbing for Tracked Mobile Robot Using a Kinematics based Controller,” *Proc. of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, pp. 1891–1896, 2009.

[24] M. J. Lawn and T. Ishimatsu, “Modeling of a Stair-Climbing Wheelchair Mechanism With High Single-Step Capability,” *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 11(3), pp. 323–332, 2003.

[25] H. Ikeda, Z. Wang, T. Takahashi and E. Nakano, “Stable Step Climbing and Descending for Tandem Wheelchairs Connected by a Passive Link,” *Proc. of IEEE/ICME Int. Conf. on Complex Medical Engineering*, pp. 1345–1350, 2007.

[26] S. A. Stoeter and N. Papanikolopoulos, “Autonomous Stair-Climbing With Miniature Jumping Robots,” *IEEE Trans. on Systems, Man, and Cybernetics–Part B: Cybernetics*, 35(2), pp. 313–325, 2005.

[27] S. D. Herbert, A. Drenner and N. Papanikolopoulos, “Loper: A Quadruped-Hybrid Stair Climbing Robot,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 799–804, 2009.

[28] M. Eich, F. Grimminger and F. Kirchner, “A Versatile Stair-Climbing Robot for Search and Rescue Applications,” *Proc. of IEEE Int. Workshop on Safety, Security and Rescue Robotics*, pp. 35–40, 2008.

[29] C. Fu and K. Chen, “Gait Synthesis and Sensory Control of Stair Climbing for a Humanoid Robot,” *IEEE Trans. on Industrial Electronics*, 55(5), pp. 2111–2120, 2008.

[30] S. Oßwald, A. Görög, A. Hornung and M. Bennewitz, “Autonomous Climbing of Spiral Staircases with Humanoids,” *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 4844–4849, 2011.

[31] R. L. Hatton, R. A. Knepper, H. Choset, D. Rollinson, C. Gong, and E. Galceran, “Snakes on a Plan: Toward Combining Planning and Control,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 5174–5181, 2013.

[32] P. Liljebäck and K. Y. Pettersen, “Waypoint guidance control of snake robots,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 937-944, 2011.

[33] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. Scokaert, “Constrained model predictive control: Stability and optimality”, *Automatica*, 36(6), pp. 789–814, 2000.

[34] J. M. Maciejowski, “Predictive Control with Constraints”, Prentice Hall, 2002.

[35] M. G. Earl and R. D’Andrea : Iterative MILP Methods for Vehicle-Control Problems, *IEEE Trans. on Robotics*, 21(6), pp. 1158–1167, 2005.

[36] K. Kon, H. Fukushima and F. Matsuno, “Trajectory generation based on Model Predictive Control with Obstacle Avoidance between Prediction Time Steps”, *Proc. of the 9th IFAC Symposium on Robot Control*, F3B3, 2009.

[37] L. Blackmore, M. Ono, A. Bektassov and B. C. Williams, A probabilistic particle-control approximation of chance-constrained stochastic predictive control, *IEEE Trans. on Robotics*, 26(3), pp. 502–517, 2010.

[38] Y. Kuwata, A. Richards, T. Schouwenaars and J. P. How, Distributed robust receding horizon control for multivehicle guidance, *IEEE Int. Journal of Control Systems Technology*, 15(4), pp. 627–641, 2007.

[39] H. Fukushima, K. Kon, and F. Matsuno, “Model Predictive Formation Control Using Branch-and-Bound Compatible With Collision Avoidance Problems”, IEEE Transactions on Robotics, 29(5), pp. 1308–1317, 2013.

[40] F. Cortes, D. Linares, D. Patino and K. Melo, “A Disrtibuted Model Predictive Control (D-MPC) for Modular Robots in Chain Configuration”, *IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications*, pp. 1–6, 2011.

[41] Y. Tassa, T. Erez and B. Smart, “Receding Horizon Differential Dynamic Programming”, *Advances in neural information processing systems*, pp. 1465-1472, 2008.

[42] T. Schouwenaars, B. D. Moor, E. Feron and J. How, “Mixed integer programming for multi-vehicle path planning”, *Proc. of the European Control Conference*, pp. 2603–2608, 2001.

[43] M. Athans and P. L. Falb. “Optimal Control, An Introduction to the Theory and Its Applications”, *McGraw-Hill*, 1966.

[44] R. Olfati-Saber, “Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory”, *IEEE Transactions on Automatic Control*, 51(3), pp. 401–420, 2006.

[45] G. Granosik, “Hypermobile Robots –the Survey,” *Journal of Intelligent and Robotic Systems*, 75(1), pp. 147–169, 2014.

[46] IBM CPLEX, http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

[47] V. Nguyen, A. Martinelli, N. Tomatis and R. Siegwart, “A comparison of line extraction algorithms using 2D laser range finder for indoor mobile robotics”, *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 1929–1934, 2005.

[48] J. Poppinga, N. Vaskevicius, A. Birk and K. Pathak, “Fast Plane Detection and Polygonalization in noisy 3D Range Images”, *Proc. of Int. Conf. on Intelligent Robots and Systems*,pp. 3378–3383 2010.

Kazuyuki Kon (S’07 - M’10) received his B.S. and M.S. in Engineering from University of Electro-Communications, Japan, in 2005 and 2007, and Ph.D. degree in Engineering from Kyoto University, Japan, in 2010, respectively. From 2009 to 2011, he was as a Research Fellow of Japan Society for the Promotion of Science. From 2011 to 2014 he was an Assistant Professor of Kyoto University, Japan. Currently he is currently with NEC Corporation, Tokyo. His research interests include autonomous mobile robots and formation control. He is a member of the IEEE, SICE and RSJ.

Motoyasu Tanaka (S’05 - M’12) received his B.S., M.S., and Ph.D. in Engineering from the Department of Mechanical Engineering and Intelligent Systems at the University of Electro- Communications, Japan in 2005, 2007, and 2009, respectively. From 2009 to 2012, he worked at Canon, Inc., Tokyo, Japan. He is currently an Assistant Professor in the Department of Mechanical Engineering and Intelligent Systems at the University of Electro-Communications. His research interests include biologically inspired robotics and dynamic-based nonlinear control. He received the IEEE Robotics and Automation Society Japan Chapter Young Award from the IEEE Robotics and Automation Society Japan Chapter in 2006. He is a member of the IEEE, SICE, and RSJ.

Kazuo Tanaka (S’87 - M’91 - SM’09 - F’14) received the B.S. and M.S. degrees in Electrical Engineering from Hosei University, Tokyo, Japan, in 1985 and 1987, and Ph.D. degree, in Systems Science from Tokyo Institute of Technology, in 1990, respectively. He is currently a Professor in Department of Mechanical Engineering and Intelligent Systems at The University of Electro-Communications. He was a Visiting Scientist in Computer Science at the University of North Carolina at Chapel Hill in 1992 and 1993. He received the Best Young Researchers Award from the Japan Society for Fuzzy Theory and Systems in 1990, the Outstanding Papers Award at the 1990 Annual NAFIPS Meeting in Toronto, Canada, in 1990, the Outstanding Papers Award at the Joint Hungarian-Japanese Symposium on Fuzzy Systems and Applications in Budapest, Hungary, in 1991, the Best Young Researchers Award from the Japan Society for Mechanical Engineers in 1994, the Outstanding Book Awards from the Japan Society for Fuzzy Theory and Systems in 1995, 1999 IFAC World Congress Best Poster Paper Prize in 1999, 2000 IEEE Transactions on Fuzzy Systems Outstanding Paper Award in 2000, the Best Paper Selection at 2005 American Control Conference in Portland, USA, in 2005, the Best Paper Award at 2013 IEEE International Conference on Control System, Computing and Engineering in Penang, Malaysia, in 2013, the Best Paper Finalist at 2013 International Conference on Fuzzy Theory and Its Applications, Taipei, Taiwan in 2013. His research interests include intelligent systems and control, nonlinear systems control, robotics, brain-machine interface and their applications. He co-authored (with Hua O. Wang) the book Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach (Wiley-Interscience, 2001). He has served as an Associate Editor for Automatica and for the IEEE Transactions on Fuzzy Systems, and is on the IEEE Control Systems Society Conference Editorial Board. He is a fellow of IEEE and IFSA.