

Planar Bipedal Locomotion Control Based on State-Discretization

Yuichi Tazaki^a Jun-ichi Imura^a

^a *Tokyo Institute of Technology, Ookayama 2-12-1, Meguro, Tokyo, Japan.*

Abstract

In this paper, a new control method for a planar bipedal robot, which we call Graph-based Model Predictive Control, is proposed. This method makes use of a directed graph constructed on the state space of the robot. The vertices of the directed graph are called *waypoints* and serve as intermediate target states to compose complex motions of the robot. By simply tracing the directed edges of the graph, one can achieve Model Predictive Control over the waypoint set. Such a directed graph is pre-designed and stored into the controller's memory to significantly reduce the computational effort required in real time. In addition, by constructing multiple directed graphs based on different objective functions, one can design multiple motions and switching trajectories among them in a uniform way. The proposed method is applied to variable-speed walking control of a bipedal walker on a 2-dimensional plane, and its effectiveness is verified by numerical simulations.

Key words: Bipedal locomotion, Discretization, Model predictive control, Offline computation

PACS:

1 Introduction

In order for bipedal walking robots to adopt to real environments, it should be able to realize various motions; not only walking in a constant speed, but also stopping, changing the walking direction, and so on. Moreover, the robot must be able to switch smoothly from one motion to another motion. The long term goal of this research is to develop a uniform framework for robot controller design with these capabilities. One of the keys to achieve this goal is the notion of discretization. This is because discretization reduces an originally

Email address: `tazaki@cyb.mei.titech.ac.jp` (Yuichi Tazaki).

continuous (infinite) set into a countable (or sometimes a finite) set, and thus dramatically simplifies planning tasks.

In the field of motion planning, the well known road map method expresses continuous state trajectories by interpolation of discrete sequences of states. Road map methods are often combined with randomized sampling techniques, in which case they are called Probabilistic Road Maps (PRMs) [1][2][3]. On the other hand, in [4][5], grid-based sampling was used in order to solve an obstacle avoidance problem of a 2D vehicle.

To deal with high dimensional systems like bipedal robots, uniform random sampling and grid sampling are ineffective. Instead, some kind of nonuniform discretization reflecting the dynamics and the characteristics of the motion to be realized is needed. For motion planning of humanoid robots, the use of motion primitives is recently being common. For example, Schmidt [6] proposed a motion planning method of 3D bipedal walker that composes walking patterns by concatenating primitive motion fragments, which are pre-designed and stored in a database. However, transition from one (periodic) motion from another is not discussed in depth in existing researches, despite its potential importance.

Some attempts are made in supervised learning of humanoid robots, to construct a symbolized motion space from measured data of various human actions obtained by motion capture [7][8]. Okada et al [9] proposed a method that synthesizes an attractive vector field around a prescribed closed orbit describing a certain motion of the robot. Smooth transitions among multiple kinds of motions are realized by creating multiple attractive fields. These works focus on higher level problems, in the sense that they do not discuss how to actually realize a certain motion under physical constraints (such as equations of motions and contact configurations) involved in humanoid robots. Kovar et al [10] proposed a method of motion generation called Motion Graph, which generates a graph structure from motion captured data by connecting different key frames with sufficiently close posture. However, this work aims at computer animation, rather than robotics.

In this paper, we propose a method of generating multiple motions as well as smooth transitions among them respecting physical constraints. Towards the development of general and systematic framework, this time we focus on a variable-speed walking task of a planar bipedal walker. To this aim, at first we discretize the state space of the bipedal walker by distributing a finite set of representative states, which we call a waypoint set. Next we introduce path cost functions, each of which evaluates the desirability of a path over the waypoint set with respect to the motion it represents. Using path cost functions, periodic motion that realizes each motion (this time steady-state walking in a certain speed) as well as transient trajectories that realize transitions from

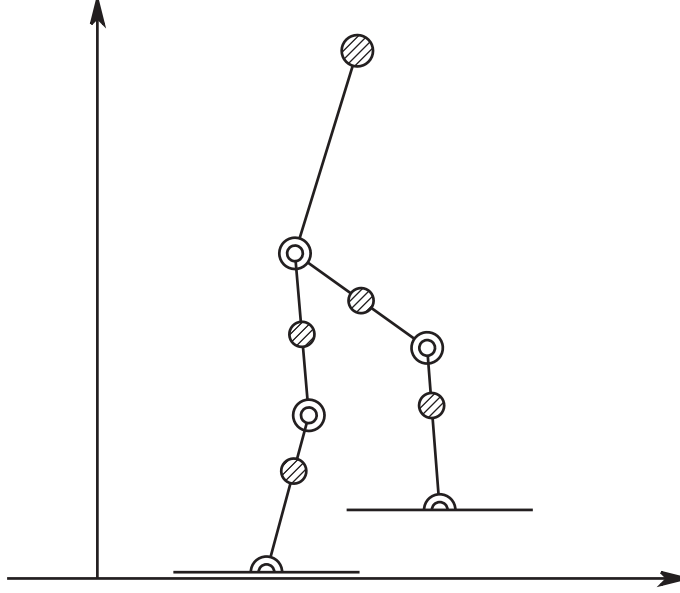


Fig. 1. Planar bipedal robot

among different motions are obtained based on finite-horizon optimality together with the receding horizon policy of model predictive control. For this reason we call the proposed method Graph-based Model Predictive Control.

This paper is organized as follows. In Section 2, we will model a planar bipedal robot as a hybrid system. In Section 3, we will describe the proposed method in detail. Section 4 shows numerical simulations. Concluding remarks are made in Section 5.

Notation: The symbol $[\mathbf{v}_1; \mathbf{v}_2; \dots; \mathbf{v}_N]$ denotes the vertical concatenation of vectors or that of matrices, which is equivalent to $[\mathbf{v}_1^T \mathbf{v}_2^T \dots \mathbf{v}_N^T]^T$. The operation $\|\mathbf{v}\|_M$ for a vector $\mathbf{v}$ and a matrix M is defined as $\sqrt{\mathbf{v}^T M \mathbf{v}}$.

2 Hybrid System Model of Planar Bipedal Robot

Fig.1 shows a planar bipedal walking robot treated in this paper. The dynamics of the robot is modeled as a hybrid system with state jumps. Hybrid systems is a class of dynamical systems consist of both continuous and discrete state variables, and governed by both continuous-time dynamics and discrete events.

To begin with, we make the following assumptions: Each foot is massless; the foot of the swing leg therefore has no effect on the robot's dynamics. The actuator torque of the ankle joint of the swing leg is always 0. The robot always stands on the entire area of one of its feet; it does not stand on tiptoe nor on

a heel. Moreover, no slipping occurs at the contact face between the foot and the ground. The impact between the swing foot and the ground is completely inelastic; the velocity of the swing foot always becomes 0 immediately after the impact.

The continuous state of the robot at time t is defined as

$$\mathbf{x}(t) = [\mathbf{q}(t); \mathbf{v}(t)],$$

where the vector $\mathbf{q}$ denotes the robot's posture given as

$$\mathbf{q} = [\theta_{\text{body}}; \theta_{\text{hip}_l}; \theta_{\text{hip}_r}; \theta_{\text{knee}_l}; \theta_{\text{knee}_r}],$$

and $\mathbf{v}$ denotes the angular velocity; i.e., $\frac{d}{dt}\mathbf{q} = \mathbf{v}$. Every joint is attached an actuator. The input of the robot is defined as a vector of all actuator torques written as

$$\mathbf{u} = [\tau_{\text{hip}_l}; \tau_{\text{hip}_r}; \tau_{\text{knee}_l}; \tau_{\text{knee}_r}; \tau_{\text{ankle}_l}; \tau_{\text{ankle}_r}].$$

The components of $\mathbf{q}$ and $\mathbf{u}$ are given as follows.

θ_{body}	:	body inclination
$\theta_{\text{hip}_i} (i \in \{l, r\})$	:	hip joint angle
$\theta_{\text{knee}_i} (i \in \{l, r\})$	:	knee joint angle
$\tau_{\text{hip}_i} (i \in \{l, r\})$	:	hip joint torque
$\tau_{\text{knee}_i} (i \in \{l, r\})$	:	knee joint torque
$\tau_{\text{ankle}_i} (i \in \{l, r\})$	:	ankle joint torque

The subscripts $(\cdot)_l$ and $(\cdot)_r$ denote symbols related to the left and right leg of the robot, respectively.

The *mode* of the robot at time t is denoted by $I(t)$ and takes one of the two symbols: 'L' and 'R'. The symbol L(R) indicates that the robot is standing on its left(right) foot.

The continuous behavior of the system is described by the following ordinary differential equations.

$$\begin{aligned} \dot{\mathbf{q}}(t) &= \mathbf{v}(t) \\ \dot{\mathbf{v}}(t) &= F_{I(t)}(\mathbf{q}(t))\mathbf{u}_{I(t)}(t) + G_{I(t)}(\mathbf{q}(t), \mathbf{v}(t)) \end{aligned} \tag{1}$$

Here, the matrix-valued function $F_{I(t)}(\mathbf{q}(t))$ and the vector-valued function $G_{I(t)}(\mathbf{q}(t), \mathbf{v}(t))$ will be described in Appendix. The symbol $\mathbf{u}_{I(t)}$ is obtained by removing from $\mathbf{u}$ the ankle joint torque of the swing leg, which is always 0 by the assumption.

Discrete events involved in the robot dynamics is explained below. When the mode is L and the swing foot (the right foot) hits the ground, an impulsive

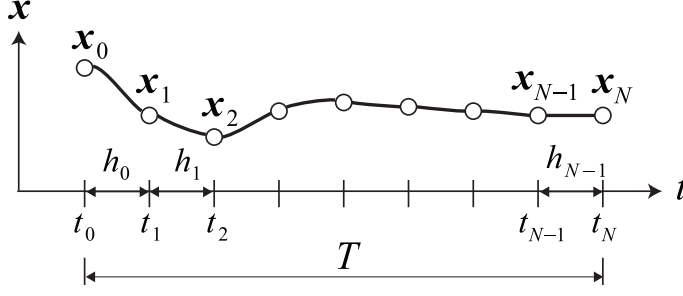


Fig. 2. Continuous state trajectory passing waypoints

force acts on the right foot and the velocity of the robot changes instantaneously. As a result, the stance leg switches from the left leg to the right leg, which indicates the change of the mode to R. We denote this event by the symbol LR. Similarly, the event RL denotes a discrete event resulting in the transition of the mode from R to L. We denote by I^-I^+ the event causing the change of the mode from I^- to I^+ . Only the event LR and RL are of consideration. The set of continuous states that trigger the event I^-I^+ is denoted by $\mathcal{S}_{I^-I^+}$. Consequently, the instantaneous jump of the velocity and the change of the mode at the discrete event I^-I^+ is expressed as follows.

$$\begin{aligned} \mathbf{v}(t^+) &= H_{I^-I^+}(\mathbf{q}(t^-))\mathbf{v}(t^-) \\ I(t^+) &= I^+ \end{aligned} \quad \text{if } (\mathbf{q}(t^-), \mathbf{v}(t^-)) \in \mathcal{S}_{I^-I^+} \quad (2)$$

The symbol t^- and t^+ denotes the time immediately before and after the discrete event, respectively. The matrix-valued function $H_{I^-I^+}(\mathbf{q}(t^-))$ will be given in Appendix.

3 Graph-based Model Predictive Control

In this section, we will explain the proposed method. After providing a brief overview, we will give a further description of each building block composing the method.

3.1 Basic Concept of the Method

The fundamental concept underlying this method is the discretization of time and space. Fig. 2 illustrates this concept. Consider a continuous-time continuous state trajectory $\mathbf{x}(t)$ over the time interval $[0, T]$. Moreover, consider a discrete sequence of sample points $\{t_i\}$ over the time axis and a sampled sequence $\mathbf{x}(t_i)$ of the state trajectory. We may say that the discrete sequence $\mathbf{x}(t_i)$ is, in some sense, abstracting the continuous trajectory $\mathbf{x}(t)$. Thinking in the opposite way, if we design a discrete sequence of sampling times $\{t_i\}$ and

that of intermediate states $\{\mathbf{x}_i\}$ under a certain criteria, we may expect that a continuous trajectory obtained by interpolation is sufficiently close to the *true* solution, which could be obtained by optimization over the continuous time and space. In this way, we can decompose the entire control problem into the design of discrete sequences ($\{t_i\}$ and $\{\mathbf{x}_i\}$) and continuous-time control for transferring the state from one intermediate state to the next one in a specified time.

In general, one needs to perform a search in continuous space to obtain the optimal sequence of $\{t_i\}$ and $\{\mathbf{x}_i\}$ for a given initial state, which often requires an excessive computation cost. To overcome this problem, we *discretize the time axis and the state space*. By saying discretize the time axis, we mean that transition time length from one intermediate state to the another is chosen from a prescribed finite set. We denote such a set by $\mathcal{T}$. On the other hand, discretization of the state space means that we also choose intermediate states from a prescribed finite set. We call such a set a *waypoint set*, and denote it by $\mathcal{X}$. The finite discretization of time and state naturally leads to a graph structure; that is, the graph $\mathcal{G} = \langle \mathcal{X}, \mathcal{E} \rangle$ constituted by the nodes $\mathcal{X}$ and the edge set

$$\mathcal{E} = \{(\mathbf{x}, \mathbf{x}', h) \in \mathcal{X} \times \mathcal{X} \times \mathcal{T} \mid \mathbf{x} \text{ is reachable to } \mathbf{x}' \text{ in the time period } h.\}. \quad (3)$$

By introducing such a graph structure with finitely many nodes, one no longer needs to perform an expensive search in continuous space, but can simply calculate path planning problems over a graph, which requires definitely less computation effort. In this framework, the following key questions arise.

- How do we design a graph structure without knowing the initial state?
- How do we relate a node of the graph from an initial state that is arbitrarily given in the continuous state space?
- What type of path planning problem shall we solve on the graph, in order to realize a certain motion on the robot?
- How do we generate an actual continuous trajectory from a path on the graph?

This paper does not provide complete and general answers to these questions. Instead, we will focus on the variable-speed walking problem of a planar bipedal robot as a particular example and propose one heuristic approach based on the above concept. At first, in Subsection 3.2, we will discuss how to generate a continuous-time control input trajectory that drives the state from an initial waypoint to a target waypoint in a certain transition time period. The method presented here will produce the intermediate trajectory and the transition cost as explicit functions of waypoints and transition time periods. Using this as a building block, in Subsection 3.3, we discuss how to

design the directed graph: the set of waypoints $\mathcal{X}$, the set of transition periods $\mathcal{T}$, and the set of directed edges $\mathcal{E}$. This graph design procedure will be completely done off line, and the resultant directed graph will be stored in the controller's memory. Finally, in Subsection 3.4, we will show the entire controller architecture.

3.2 Inter-waypoint Control

This subsection describes what we call *inter-waypoint control*. Inter-waypoint control is the problem of transferring the state from an initial state $\mathbf{x}_k$ to a target state $\mathbf{x}_{k+1}$ in a time period h_k , when $\mathbf{x}_k$, $\mathbf{x}_{k+1}$ and h_k are all given. There are mainly two difficulties involved in this control problem; one is the existence of discrete events, and the other is the nonlinearity of the continuous dynamics. The former problem is, in some sense, avoided by simply regarding that discrete events will not occur while the state travels through waypoints. Instead, this is treated as constraints in the design of waypoint set, which will be explained in the next subsection. For the latter problem, we employ the inverse dynamics technique to derive an approximate solution.

Based on the above considerations, inter-waypoint control is formulated as follows.

Problem 3.1 *Inter-waypoint Control*

Suppose the mode $I \in \{\text{L}, \text{R}\}$, the initial state $\mathbf{x}_s \in \mathcal{S}_I$, the target state $\mathbf{x}_f \in \mathcal{S}_I$, the transition time $h > 0$, and the positive definite matrix $R > 0$ are given. Then, find the control input trajectory $\mathbf{u}_I$ that minimizes the cost function

$$J(\mathbf{u}_I; h) = \int_0^h \mathbf{u}_I(\tau)^T R \mathbf{u}_I(\tau) d\tau \quad (4)$$

while satisfying

$$\mathbf{x}(0) = \mathbf{x}_s, \mathbf{x}(h) = \mathbf{x}_f, \quad (5)$$

and

$$\begin{aligned} \dot{\mathbf{q}}(t) &= \mathbf{v}(t), \\ \dot{\mathbf{v}}(t) &= F_I(\mathbf{q}(t))\mathbf{u}_I(t) + G_I(\mathbf{q}(t), \mathbf{v}(t)). \end{aligned} \quad (6)$$

Here, the initial time and the arrival time are set as 0 and h , respectively, without loss of generality. The cost function (4) represents an objective to minimize the input energy. Let us remark that one can select other criteria, such as minimum acceleration and minimum jerk. The procedure explained in the following also apply to such cases with minor modifications.

In the following, we derive an approximate, explicit solution to Problem 3.1. First, let us introduce a virtual input variable denoted by $\hat{\mathbf{u}}$, related to the actual input $\mathbf{u}_I$ by the following nonlinear input transformation.

$$\mathbf{u}_I(t) = F_I(\mathbf{q}(t))^{-1}(\hat{\mathbf{u}}(t) - G_I(\mathbf{q}(t), \mathbf{v}(t))) \quad (7)$$

This means that we directly control the acceleration of the robot by means of inverse dynamics technique. Thus we obtain the double integrator system

$$\dot{\mathbf{q}}(t) = \mathbf{v}(t), \quad \dot{\mathbf{v}}(t) = \hat{\mathbf{u}}(t). \quad (8)$$

On the other hand, by substituting (7) into (4) we obtain

$$\begin{aligned} \hat{J}(\hat{\mathbf{u}}; h) &= \int_0^h \|\hat{\mathbf{u}}(\tau) - G_I(\mathbf{q}(\tau), \mathbf{v}(\tau))\|_{\tilde{R}_I(\mathbf{q})}^2 d\tau \\ (\hat{R}_I(\mathbf{q}) &= F_I(\mathbf{q})^{-T} R F_I(\mathbf{q})^{-1}), \end{aligned} \quad (9)$$

which is a cost function of $\hat{\mathbf{u}}$. The cost function (9) has nonlinear terms on $\mathbf{q}(\tau)$ and $\mathbf{v}(\tau)$. To remove this nonlinearity, we approximate the cost function by replacing $\mathbf{q}(\tau)$ and $\mathbf{v}(\tau)$ with $(\mathbf{q}_s + \mathbf{q}_f)/2$ and $(\mathbf{v}_s + \mathbf{v}_f)/2$, respectively, where $\mathbf{x}_s = [\mathbf{q}_s; \mathbf{v}_s]$ and $\mathbf{x}_f = [\mathbf{q}_f; \mathbf{v}_f]$. Thus we obtain

$$\begin{aligned} \tilde{J}(\hat{\mathbf{u}}; \mathbf{x}_s, \mathbf{x}_f, h) &= \int_0^h \|\hat{\mathbf{u}}(\tau) - \tilde{G}_I\|_{\tilde{R}_I}^2 d\tau \\ &\left(\tilde{G}_I = G_I\left(\frac{\mathbf{q}_s + \mathbf{q}_f}{2}, \frac{\mathbf{v}_s + \mathbf{v}_f}{2}\right), \quad \tilde{R}_I = F_I\left(\frac{\mathbf{q}_s + \mathbf{q}_f}{2}\right)^{-T} R F_I\left(\frac{\mathbf{q}_s + \mathbf{q}_f}{2}\right)^{-1} \right), \end{aligned} \quad (10)$$

which is a quadratic cost function of $\hat{\mathbf{u}}$. As a result, Problem 3.1 is approximately transformed into a fixed-terminal optimal control problem of a double integrator system (8) with the cost function (10) and the terminal constraint (5). This problem has an analytical solution, meaning that the optimal virtual control input (acceleration) trajectory $\hat{\mathbf{u}}^*$, the optimal state trajectory $\mathbf{x}^* = [\mathbf{q}^*; \mathbf{v}^*]$, and the minimum value of the cost function $\tilde{J}^*$ are expressed as explicit functions of time, waypoints and transition times. First, the optimal posture trajectory is given by

$$\mathbf{q}^*(t; \mathbf{x}_s, \mathbf{x}_f, h) = \begin{bmatrix} \mathbf{q}_s & \mathbf{v}_s & \mathbf{q}_f & \mathbf{v}_f \end{bmatrix} \begin{bmatrix} 1 & 0 & -\frac{3}{h^2} & \frac{2}{h^3} \\ 0 & 1 & -\frac{2}{h} & \frac{1}{h^2} \\ 0 & 0 & \frac{3}{h^2} & -\frac{2}{h^3} \\ 0 & 0 & -\frac{1}{h} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}. \quad (11)$$

Accordingly, $\mathbf{v}^*$ and $\hat{\mathbf{u}}^*$ are given as the first and the second time derivative

of $\mathbf{q}^*$; that is,

$$\mathbf{v}^*(t; \mathbf{x}_s, \mathbf{x}_f, h) = \begin{bmatrix} \mathbf{q}_s & \mathbf{v}_s & \mathbf{q}_f & \mathbf{v}_f \end{bmatrix} \begin{bmatrix} 0 & -\frac{6}{h^2} & \frac{6}{h^3} \\ 1 & -\frac{4}{h} & \frac{3}{h^2} \\ 0 & \frac{6}{h^2} & -\frac{6}{h^3} \\ 0 & -\frac{2}{h} & \frac{3}{h^2} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}, \quad (12)$$

$$\hat{\mathbf{u}}^*(t; \mathbf{x}_s, \mathbf{x}_f, h) = \begin{bmatrix} \mathbf{q}_s & \mathbf{v}_s & \mathbf{q}_f & \mathbf{v}_f \end{bmatrix} \begin{bmatrix} -\frac{6}{h^2} & \frac{12}{h^3} \\ -\frac{4}{h} & \frac{6}{h^2} \\ \frac{6}{h^2} & -\frac{12}{h^3} \\ -\frac{2}{h} & \frac{6}{h^2} \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix}. \quad (13)$$

Finally, $\tilde{J}^*$ is given by

$$\tilde{J}^*(\mathbf{x}_s, \mathbf{x}_f, h) = \begin{bmatrix} \mathbf{q}_s \\ \mathbf{v}_s \\ \mathbf{q}_f \\ \mathbf{v}_f \\ \tilde{G}_I \end{bmatrix}^T \begin{bmatrix} \tilde{R}_I \frac{12}{h^3} & \tilde{R}_I \frac{6}{h^2} & -\tilde{R}_I \frac{12}{h^3} & \tilde{R}_I \frac{6}{h} & 0 \\ \tilde{R}_I \frac{6}{h^2} & \tilde{R}_I \frac{4}{h} & -\tilde{R}_I \frac{6}{h^2} & \tilde{R}_I \frac{2}{h} & \tilde{R}_I \\ -\tilde{R}_I \frac{12}{h^3} & -\tilde{R}_I \frac{6}{h^2} & \tilde{R}_I \frac{12}{h^3} & -\tilde{R}_I \frac{6}{h^2} & 0 \\ \tilde{R}_I \frac{6}{h^2} & \tilde{R}_I \frac{2}{h} & -\tilde{R}_I \frac{6}{h^2} & \tilde{R}_I \frac{4}{h} & -\tilde{R}_I \\ 0 & \tilde{R}_I & 0 & -\tilde{R}_I & \tilde{R}_I h \end{bmatrix} \begin{bmatrix} \mathbf{q}_s \\ \mathbf{v}_s \\ \mathbf{q}_f \\ \mathbf{v}_f \\ \tilde{G}_I \end{bmatrix} \quad (14)$$

The above explicit solutions combined with the input transformation (7) provides an approximate solution to Problem 3.1. The method presented here is equipped with the following useful aspects: i) it guarantees the terminal condition $\mathbf{x}(h) = \mathbf{x}_f$, and ii) the solution is given in an explicit form. These aspects play a fundamental role in the optimization of the waypoints and the transition times discussed in 3.3. It will also be utilized to improve the robustness of the real-time controller, as discussed in 3.4. However, there are some shortcomings: First, the obtained solution is suboptimal due to the cost function approximation (10). Second, the model of the robot should be full-actuated and sufficiently accurate. These technical issues should be discussed further in future works.

3.3 Design of Directed Graphs

In this subsection, we address the design of the transition time set $\mathcal{T}$ and the waypoint set $\mathcal{X}$. For this purpose, we start with classifying the whole motion space of the robot into two categories: periodic motions and transient motions. In this research, periodic motions correspond to steady-state walking

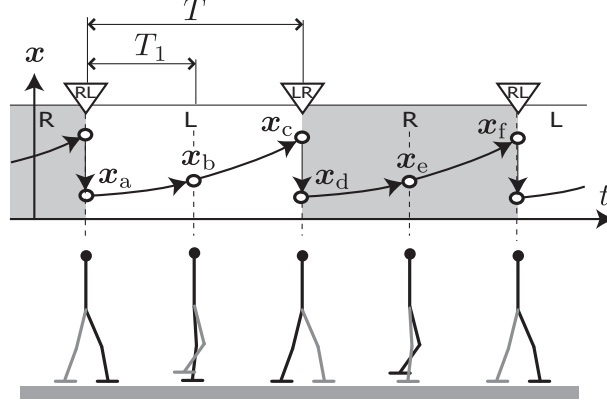


Fig. 3. Arrangement of waypoints and transition times

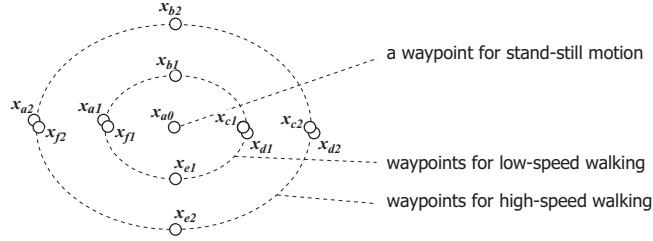


Fig. 4. Waypoints for a stand-still motion and steady-state walking motions

in a constant speed, while transient motions describe switching between one walking speed to another. The method discussed here is intended to optimize periodic motions. This is because in most applications, one can regard that the robot performs one of the periodic motions most of the time, while transient motions take place in relatively a short time period. The following assumptions are made for simplicity of discussion.

Assumption 3.2 (a) *The mode sequence in steady-state locomotion is $L \rightarrow R \rightarrow L \rightarrow \dots$.*

(b) *The optimal gait of steady-state locomotion is symmetrical about the left and the right leg.*

Based on Assumption 3.2(a), we determine the rough arrangement of waypoints and transition times for steady-state locomotion as illustrated in Fig. 3. The upper part of the figure shows the trajectory of the state in steady-state locomotion, while the lower part shows the posture of the robot on the corresponding time instant.

Since the inter-waypoint control is based on an assumption that the mode remains unchanged while the state travels between the two waypoints, in this graph design procedure, this assumption must be treated as a constraint condition. For this reason, it is necessary that at least one waypoint is placed on each subset triggering a discrete event. The waypoints x_c and x_f are placed for this purpose; therefore, $x_c \in \mathcal{S}_{LR}$ and $x_f \in \mathcal{S}_{RL}$. The waypoints x_d and

$\mathbf{x}_a$ refer to the state just after the discrete events are triggered at $\mathbf{x}_c$ and $\mathbf{x}_f$, respectively. Thus,

$$\mathbf{q}_d = \mathbf{q}_c, \quad \mathbf{v}_d = H_{LR}(\mathbf{q}_c)\mathbf{v}_c, \quad \mathbf{q}_a = \mathbf{q}_f, \quad \mathbf{v}_a = H_{RL}(\mathbf{q}_f)\mathbf{v}_f. \quad (15)$$

Here, $\mathbf{x}_i = [\mathbf{q}_i; \mathbf{v}_i]$ ($i \in \{a, b, \dots, f\}$).

On the other hand, the waypoints $\mathbf{x}_b \in \mathcal{S}_L$ and $\mathbf{x}_e \in \mathcal{S}_R$ are placed to ensure a certain amount of clearance between the swing foot and the ground to avoid an unexpected discrete event, which in this case physically means a *ground scuffing*. From Assumption 3.2(b), the following relations hold.

$$\begin{aligned} \mathbf{q}_d &= C\mathbf{q}_a, & \mathbf{v}_d &= C\mathbf{v}_a, \\ \mathbf{q}_e &= C\mathbf{q}_b, & \mathbf{v}_e &= C\mathbf{v}_b, \\ \mathbf{q}_f &= C\mathbf{q}_c, & \mathbf{v}_f &= C\mathbf{v}_c, \end{aligned} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (16)$$

Consequently, the waypoints $\mathbf{x}_b$, $\mathbf{x}_c$, and the time intervals T_1 and T are independent variables. However, numerically optimizing these variables at the same time is still a difficult task. Instead, we manually specify the posture variables $\mathbf{q}_b$, $\mathbf{q}_c$ and the walking period T , and determine the rest ($\mathbf{v}_b$, $\mathbf{v}_c$ and T_1) by numerical optimization. In this way, one can directly specify the stride, the swing foot clearance, and the walking speed of the gait. The objective function for the optimization of $\mathbf{v}_b$, $\mathbf{v}_c$ and T_1 is given by energy consumption, which is approximately expressed as $\tilde{J}^*(\mathbf{x}_a, \mathbf{x}_b, T_1) + \tilde{J}^*(\mathbf{x}_b, \mathbf{x}_c, T - T_1)$. We employ Downhill-Simplex method [11] for optimization. In general, the Downhill-Simplex method shows slower convergence compared to gradient-based methods. Nevertheless, it has an advantage that it does not require gradient information, and it is applicable to optimization problems with non-smooth cost functions. Regarding precision, like gradient-based methods, it produces a solution arbitrarily close to a local optimum according to a termination criteria.

At this point, there still remains a possibility that an unexpected mode change will occur during a sampling time interval; for instance, there might exist $t \in (0, T_1)$ such that $\mathbf{x}^*(t, \mathbf{x}_a, \mathbf{x}_b) \in \mathcal{S}_{LR}$. In order to avoid such a situation, each time a candidate $(\mathbf{x}_b, \mathbf{x}_c, T_1, T)$ is evaluated in the iteration of Downhill-Simplex method, we assign a sufficiently large penalty when the corresponding trajectory triggers a discrete event in the meantime. This check can be efficiently computed since the intermediate trajectory is expressed as an explicit function of time.

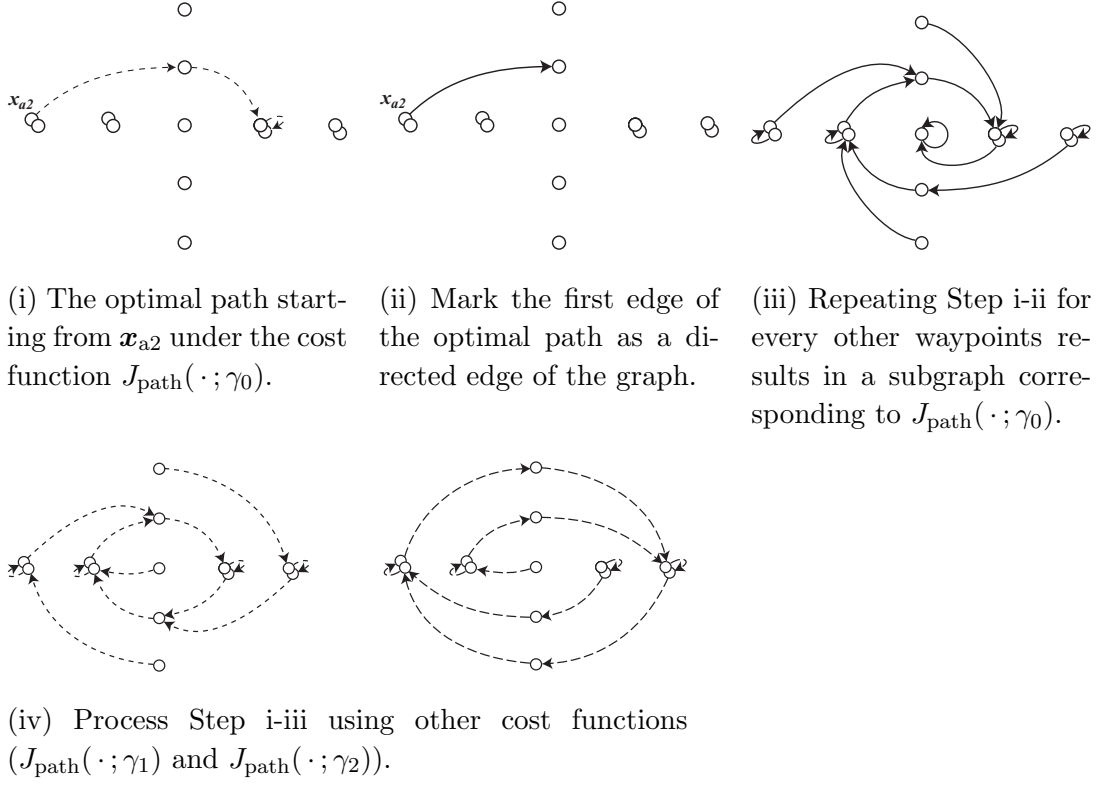


Fig. 5. Procedures of the design of subgraphs ($N = 3$)

So far, we have discussed the case of steady-state walking. The stand-still motion, on the other hand, is realized by a periodic trajectory that passes through a single waypoint. Here, we manually specify the waypoint for the stand-still motion with no special optimization.

We denote by $\mathcal{X}_i$ and $\mathcal{T}_i$ the set of waypoints and the set of transition times designed for the i -th periodic motion. In addition, we define $\mathcal{X} := \cup \mathcal{X}_i$, $\mathcal{T} := \cup \mathcal{T}_i$.

Fig. 4 shows $\mathcal{X}$ projected and plotted onto a two-dimensional plane. The waypoint set for the low-speed walking ($\mathbf{x}_{a1}, \dots, \mathbf{x}_{f1}$) and that of the high-speed walking ($\mathbf{x}_{a2}, \dots, \mathbf{x}_{f2}$), and the single waypoint for the stand-still motion ($\mathbf{x}_{a0}$) are drawn in the figure. The projection from the state space to the two-dimensional space is done by calculating a vector pointing from the left ankle to the right ankle in a given state. In this way, $\mathbf{x}_d$ and $\mathbf{x}_a$ come to the same position of $\mathbf{x}_c$ and $\mathbf{x}_f$, respectively, so $\mathbf{x}_d$ and $\mathbf{x}_a$ are plotted in a slightly shifted position to avoid confusion.

Next, we consider how to construct a set of directed edges over the waypoint set $\mathcal{X}$. The procedure of this step is illustrated in Fig. 5. First, we define the edge candidate set $\mathcal{E}$ as the set of tuples $(\mathbf{x}_0 \in \mathcal{X}, \mathbf{x}_1 \in \mathcal{X}, h \in \mathcal{T})$ in which the aforementioned mode unchanging condition is satisfied. Let us consider an

N -step path starting from a certain waypoint $\mathbf{x}_0$;

$$(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N, h_0, h_1, \dots, h_{N-1}), \quad (17)$$

where h_k is a transition time between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. We introduce the following *path cost function* as a measure of the desirability of a N -step path.

$$J_{\text{path}} : \mathcal{X}^{N+1} \times \mathcal{T}^N \mapsto \mathbb{R}^1 \quad (18)$$

Each motion the robot should perform is associated with a path cost function. In this research, we focus on realizing walking motion in variable speeds, and for this purpose we consider the following path cost function.

$$\begin{aligned} J_{\text{path}}(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N, h_0, h_1, \dots, h_{N-1}; \gamma) \\ = \sum_{k=0}^{N-1} \left\{ \tilde{J}^*(\mathbf{x}_k, \mathbf{x}_{k+1}, h_k) - \gamma \Lambda(\mathbf{x}_k, \mathbf{x}_{k+1}) \right\} \end{aligned} \quad (19)$$

In (19), the term $\tilde{J}^*(\mathbf{x}_k, \mathbf{x}_{k+1}, h_k)$ expresses the transition cost between waypoints. On the other hand, $\Lambda(\mathbf{x}_k, \mathbf{x}_{k+1})$ is a function which returns a horizontal displacement of the body of the robot when the state moves from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$. Although the state variable $\mathbf{x}$ does not include the displacement of the robot's body with respect to the global frame, one can calculate $\Lambda(\mathbf{x}_k, \mathbf{x}_{k+1})$ from the change of the posture in conjunction with the assumption that the stance foot does not slip on the ground; that is,

$$\Lambda(\mathbf{x}_k, \mathbf{x}_{k+1}) = \begin{cases} p_{l_x}(\mathbf{q}_k) - p_{l_x}(\mathbf{q}_{k+1}) & \text{if } I_k = \text{L} \\ p_{r_x}(\mathbf{q}_k) - p_{r_x}(\mathbf{q}_{k+1}) & \text{if } I_k = \text{R} \end{cases}. \quad (20)$$

Here, $\mathbf{q}_k$ denotes the posture corresponding to the state $\mathbf{x}_k$ and I_k is the mode when the state moves from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$, which is determined by

$$I_k = \begin{cases} \text{L} & \text{if } p_{l_y}(\mathbf{q}_k) < p_{r_y}(\mathbf{q}_k), p_{l_y}(\mathbf{q}_{k+1}) \leq p_{r_y}(\mathbf{q}_{k+1}) \\ \text{R} & \text{if } p_{l_y}(\mathbf{q}_k) > p_{r_y}(\mathbf{q}_k), p_{l_y}(\mathbf{q}_{k+1}) \geq p_{r_y}(\mathbf{q}_{k+1}) \end{cases}. \quad (21)$$

The functions p_{l_*} and p_{r_*} express the displacement of the left ankle and the right ankle with respect to the body center, respectively. The subscript x denotes the horizontal component, while y denotes the vertical component. Thus, the objective represented by the cost function (19) is to increase the walking distance with as small energy consumption as possible. Note that in general, the maximization of the walking distance and the minimization of the energy consumption cannot be achieved at the same time. The trade-off between these two quantities is determined by the scalar parameter $\gamma > 0$. In an extreme case, when γ is set to 0, the walking distance is no longer taken into account and this setting is utilized to obtain a stand-still motion. There larger value of γ we choose, the more weight we put on walking distance.

Using the above path cost function, we process the following procedures for each i . At first, we calculate the optimal N -step path starting from each waypoint $\mathbf{x} \in \mathcal{X}$ with respect to $J_{\text{path}}(\cdot; \gamma_i)$ (Fig. 5(i, ii)). Next, we create a set of directed edges composed of the first edge of each N -step optimal path (Fig. 5(iii)). We denote this set by $\mathcal{E}_i$. Repeating the above procedure for every path cost function, we obtain multiple sets of directed edges (Fig. 5(iv)).

It is worth pointing out that the N -step optimal path has the following recursive characteristics:

$$J_{\text{path}}^*(\mathbf{x}_0; N, \gamma) = \min_{(\mathbf{x}_0, \mathbf{x}_1, h) \in \mathcal{E}} \left\{ J_{\text{path}}(\mathbf{x}_0, \mathbf{x}_1, h; \gamma) + J_{\text{path}}^*(\mathbf{x}_1; N-1, \gamma) \right\}. \quad (22)$$

Here, $J_{\text{path}}^*(\mathbf{x}_0; N, \gamma)$ is the cost of the N -step optimal path starting from the waypoint $\mathbf{x}_0$. Taking advantage of this recursive characteristics, one can compute the N -step optimal path starting from each waypoint efficiently by means of dynamic programming. First, we compute the 1-step optimal path for every waypoint. This takes the computational time proportional to $|\mathcal{E}|$ (the number of elements in $\mathcal{E}$). Next, we compute a 2-step optimal path for every waypoint. Since the 1-step optimal paths have already been obtained in the preceding step, this step also takes the computational time proportional to $|\mathcal{E}|$ using (22). Repeating this procedure until the length of the paths reaches N amounts to the total computational time proportional to $N|\mathcal{E}|$.

One might argue that variable speed locomotion could be achieved by simply memorizing periodic trajectories of several different walking speeds as reference trajectories. However, the use of graph structure is crucial for creating not only periodic motions but also switching motions at the same time. Observe that each edge set $\mathcal{E}_i$ is composed of a cyclic path and paths that connect to the cyclic path. The cyclic path generates a periodic motion; i.e., a steady-state locomotion. On the other hand, the paths connecting to the cyclic path generate switching motions, which enables smooth transitions from other periodic motions. Note that these switching paths are determined based on the optimality with respect to the path cost function and the receding horizon policy.

The cyclic path of the $\mathcal{E}_i$ is expected to be formed on the waypoint set $\mathcal{X}_i$. It is not apparent what range of γ results in a cyclic path constructed on the desired waypoint set. This time, the value of γ for each motion is chosen manually (the actual values are listed in Table. 3).

Fig. 6 summarizes the entire flow of the graph design procedure described in this subsection.

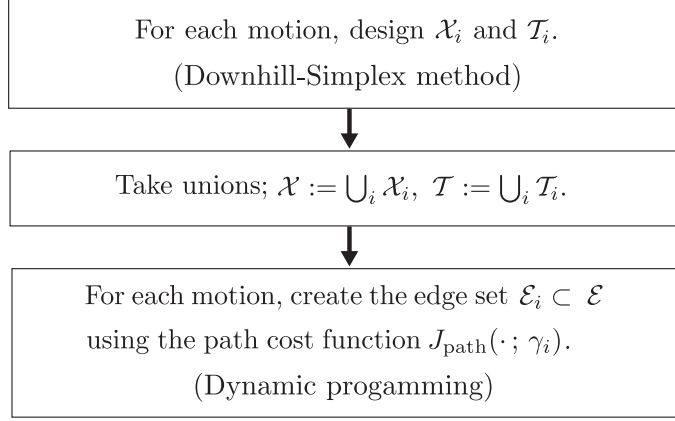


Fig. 6. Off-line graph design procedure

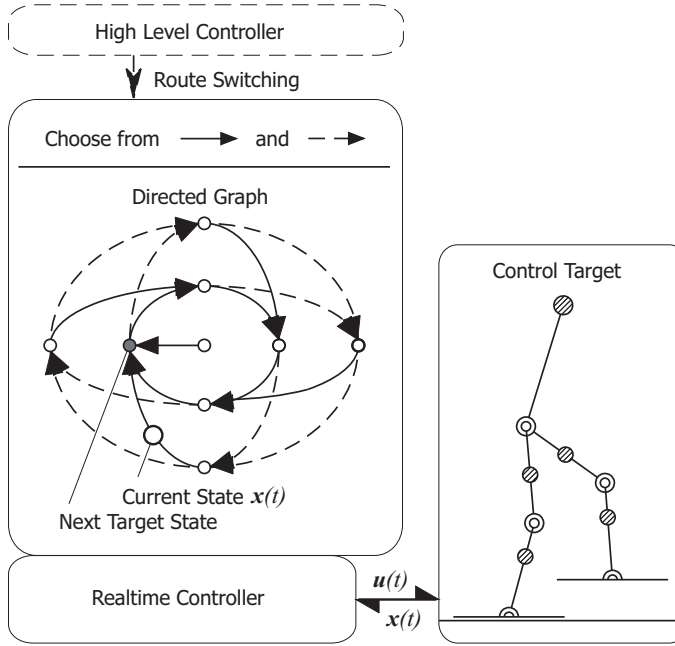


Fig. 7. Controller architecture

3.4 Controller Architecture

Fig. 7 shows the entire controller architecture. The controller stores directed graphs in its memory. These directed graphs are pre-designed by the procedure described in the previous subsection, and each directed graph is responsible for realizing a certain motion of the robot. At the initial time t_0 , one arbitrary directed graph is selected as *active* and the state of the robot is set to one arbitrary waypoint (say $\mathbf{x}_0$). Then, at each sampling time t_k , the controller looks up the directed edge $(\mathbf{x}_k, \mathbf{x}_{k+1}, h_k)$ of the activated graph to determine the next target waypoint. In other words, each time the state arrives at a waypoint, the next target waypoint is the one that is pointed to by the N -

step optimal path starting from the current waypoint. This corresponds to the receding horizon policy of the conventional model predictive control, and thus we call the method Graph-based Model Predictive Control.

Now, let us discuss the robustness of the method. In general, feedback mechanism is needed to make a controller robust against uncertainties. Here, we point out that the following three different levels of feedback can be considered:

1. Inter-waypoint control

During the time interval $t \in [t_k, t_{k+1}]$, the controller applies a continuous-time feed-forward input signal by which the state will arrive at the waypoint $\mathbf{x}_{k+1}$ at the next sampling time instant t_{k+1} . Such an input is obtained by (13) in conjunction with real-time inverse dynamics computation given by (7). To make the inter-waypoint control a feedback control law, we substitute $\mathbf{x}(t_k)$ (the actual state at time t_k) into $\mathbf{x}_s$ of $\hat{\mathbf{u}}^*$ (13) instead of $\mathbf{x}_k$ (the waypoint). Under this new control law, the error between the actual state and the target waypoint is corrected at each sampling time instant. This modification can be implemented with no increase of computational effort, since the relation between the control input (13) and $\mathbf{x}(t_k)$ is explicit. However, one should note that this level of feedback can cope with relatively small disturbances, under which the mode unchanging constraint will not be violated.

2. Online search of the target waypoint

As disturbance becomes larger, it will be more likely that the precomputed target waypoint is not reachable from the current state without violating the mode unchanging constraint. When this is the case, the target waypoint should be searched on line. In this level the target waypoint is not fully recomputed but it is chosen from a precomputed waypoint set $\mathcal{X}$. Similarly, a new transition timing is chosen from $\mathcal{T}$.

3. Online sampling of additional waypoints

It could still be the case that no waypoint in $\mathcal{X}$ is reachable from the current state with a transition period chosen from $\mathcal{T}$. Then, one should sample additional waypoints and transition times until one finds an executable path.

In this paper, we have implemented the level 1 only. The implementation of the level 2 and 3 are still an open issue.

4 Numerical Experiments

In this section, we test the proposed method by numerical simulations. Table 1 shows the values of physical parameters.

Table 1
Physical parameters

body mass	20.00 [kg]		
shank mass	10.00 [kg]	thigh mass	10.00 [kg]
body length	0.50 [m]		
shank length	0.40 [m]	thigh length	0.40 [m]
ankle to toe	0.15 [m]	ankle to heel	0.15 [m]

Table 2
Waypoint parameters

	stride[m]	period[s]	clearance[m]
low speed	0.20	0.40	0.03
high speed	0.40	0.50	0.05

Table 3
Weight parameters

steady state	γ_0	0
low speed	γ_1	6000
high speed	γ_2	12000

4.1 Variable-Speed Walking

We designed directed graphs for realizing three kinds of motions : stand-still, low-speed walking and high-speed walking. Parameters describing the gaits of low-speed and high-speed walking are set as shown in Table 2, and posture variables ($\mathbf{q}_b, \mathbf{q}_c$) for low-speed and high-speed walking are determined according to these parameters. The weight parameters of the path cost functions are listed in Table 3. The path step length N is given as 100. Since the computational complexity of dynamic programming is proportional to N , the required computation time is still small enough even if N is as large as 100.

The stick diagram in Fig. 8 shows the gait of the robot. The robot is initially standing still. Then, it switches to the low-speed walking motion and next to the high-speed walking motion in every three strides. After that, it switches to the low-speed walking motion again and finally returns to the stand-still motion. The transition of motion is achieved by simply switching the active graph. Fig. 9 shows the plot of the corresponding state trajectory projected to three dimensional space. In this figure, the vertical coordinate depicts the kinetic energy of the robot, while the axes labeled “foot_x” and “foot_y” depict the x and y coordinates of the displacement vector from the left ankle

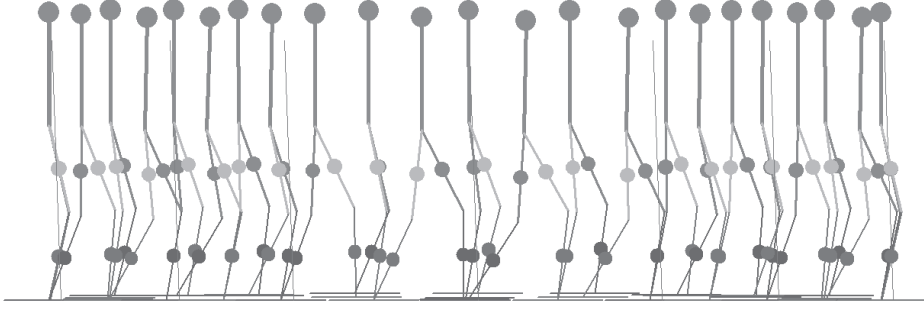


Fig. 8. Gait of bipedal walker

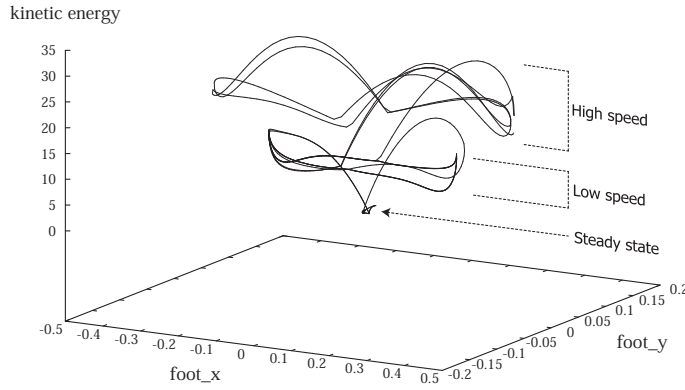
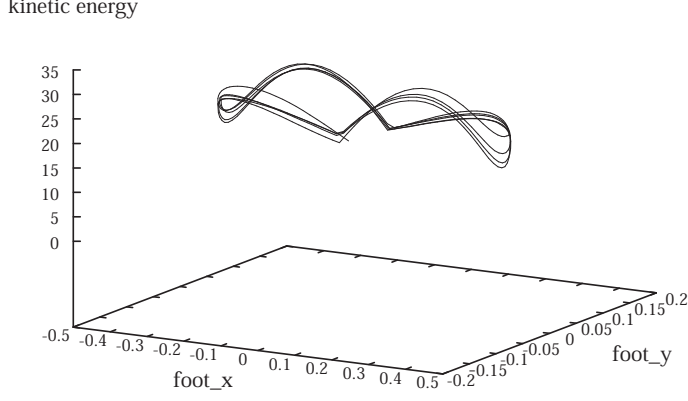


Fig. 9. A phase curve in variable speed locomotion

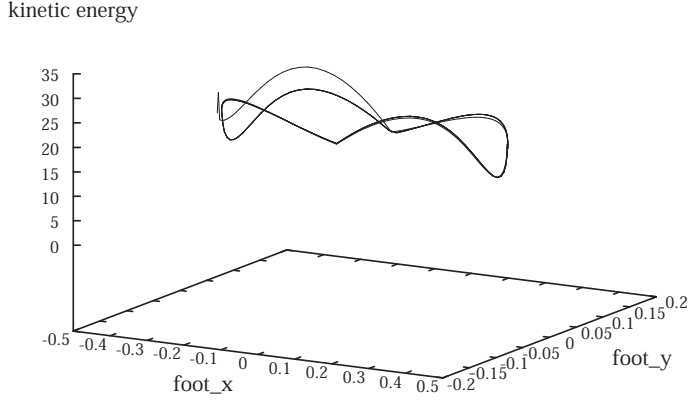
to the right ankle, respectively.

4.2 Verification of Robustness

In this subsection we qualitatively test the robustness of the proposed method against disturbances and model errors. Fig. 10 shows the state trajectories of the robot under disturbances and model errors, plotted in the same way as Fig. 9. In Fig. 10(a), noise is added to the signal line connecting the robot and the controller. The noise is white and its range is $\pm 1\%$ of the norm of the nominal signal. On the other hand, in Fig. 10(b), the physical parameters (listed in Table 1) of the model used by the controller (both for the off-line graph design and the real-time control) are deviated from those of the model used for the simulation by the amount of $\pm 3\%$. It has been verified that our controller is able to maintain stability under a slight amount of disturbances and model errors.



(a) with noises



(b) with model errors

Fig. 10. Phase curves under noises and errors

5 Conclusion

In this paper, we have addressed the locomotion control of a planar bipedal robot. In the proposed method, the original hybrid system model of the robot is reduced to a simple directed graph by means of discrete approximation. By pre-designing directed graphs and storing them in the controller's memory, model predictive control is achieved by simply tracing the directed edge of the graph. Moreover, using multiple graphs enables the realization of multiple motions, including smooth transitions between one motion to another.

To realize a stable locomotion, local stability in the neighborhood of the steady-state gait is achieved by a feed-back law that corrects the error between the target waypoint and the actual state at every sampling time. However, this mechanism works only if the disturbance is sufficiently small. The stability basin of the periodic trajectory should be expanded further in order for this method to be applicable to real environments. To cope with a larger deviation between the waypoint and the actual state, a mechanism for select-

ing the target state on line will be needed. This should be investigated in the future research.

References

- [1] D. Hsu, R. Kindel, J.C. Latombe and S. Rock : Randomized Kinodynamic Motion Planning with Moving Obstacles, *The International Journal of Robotics Research*, Vol. 21, No. 3, pp. 233-255, 2002.
- [2] S.M. Lavalley: From dynamic programming to RRTs: Algorithmic design of feasible trajectories, In A. Bicchi, H.I. Christensen and D. Prattichizzo, editors, *Control Problems in Robotics*, pp. 19-37, Springer-Verlag, Berlin, 2002.
- [3] L.E. Kavraki, P. Svestka, J.C. Latombe and M. Overmars: Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, pp. 566-580, 1996.
- [4] J. Imura and H. Matsushima : Simultaneous optimization of continuous control inputs and discrete state waypoints, 9th International Workshop on Hybrid Systems: Computation and Control (HSCC 2006), J.Hespanha and A.Tiwari (Eds.) LNCS 3927 Springer-Verlag, pp302-317, 2006.
- [5] H.L. Hagenaaers, J. Imura and H. Nijmeijer : Approximate continuous-time optimal control in obstacle avoidance by time/space discretization of non-convex state constraints, *Proc. of IEEE Conf. on Control Applications*, pp878-883, 2004.
- [6] J. Denk and G. Schmidt: Synthesis of Walking Primitive Databases for Biped Robots in 3D-Environments, *Proc. of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, September 14-19, 2003.
- [7] T. Inamura, I. Toshima, H. Tanie and Y. Nakamura: Embodied Symbol Emergence based on Mimesis Theory, *International Journal of Robotics Research*, Vol. 23, No. 4, 2004.
- [8] W. Takano, H. Tanie and Y. Nakamura: Key Feature Extraction for Probabilistic Categorization of Human Motion Patterns, 12th International Conference on Advanced Robotics (ICAR'05), 2005.
- [9] M. Okada, K. Tatani and Y. Nakamura : "Polynomial Design of the Nonlinear Dynamics for the Brain-like Information Processing of Whole Body Motion", *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'02)*, pp1410-1415, Washington, DC, 2002.
- [10] L. Kovar, M. Gleicher and F. Pighin, "Motion Graphs", In *Proceeding of SIGGRAPH 02*, 2002, pp. 473-482
- [11] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery: *Numerical Recipes in C The Art of Scientific Computing Second Edition*, Cambridge University Press.

A Derivation of the model

This section shows the derivation of the systems dynamics (1) and (2). First of all, we define the augmented position and velocity vector as

$$\hat{\mathbf{q}} := \begin{bmatrix} \mathbf{q}_{\text{body}} \\ \mathbf{q} \end{bmatrix}, \quad \hat{\mathbf{v}} := \begin{bmatrix} \mathbf{v}_{\text{body}} \\ \mathbf{v} \end{bmatrix},$$

respectively, where $\mathbf{q}_{\text{body}} = [x \ y]^T$ and $\mathbf{v}_{\text{body}} = [v_x \ v_y]^T$ are the displacement and velocity of the robot's body in the absolute frame, respectively. Further, we define $J_{I(t)}(\mathbf{q}(t))$ as the Jacobian matrix that maps $\mathbf{v}(t)$ to the velocity of the support foot relative to the robot's body according to the mode $I(t)$. Then, the equation of motion is given as follows.

$$\hat{M}(\mathbf{q})\dot{\hat{\mathbf{v}}} = \hat{h}(\mathbf{q}, \mathbf{v}) + \hat{g}(\mathbf{q}) + \hat{J}_I(\mathbf{q})^T \boldsymbol{\lambda} + \hat{N}\mathbf{u} \quad (\text{A.1})$$

Here, $\hat{M}(\mathbf{q})$ denotes the inertia matrix, $\hat{h}(\mathbf{q}, \mathbf{v})$ denotes the centrifugal force vector, and $\hat{g}(\mathbf{q})$ denotes the gravitation force vector. The matrix $\hat{N}$ maps the motor torque $\mathbf{u}$ to the generalized force. Moreover, the matrix $\hat{J}_I(\mathbf{q}) = [I_2 \ J_I(\mathbf{q})]$ and the vector $\boldsymbol{\lambda}$ refers to the reaction force acting on the support foot. Since the acceleration of the support foot is 0, this implies

$$\hat{J}_I \dot{\hat{\mathbf{v}}} + \dot{\hat{J}}_I \hat{\mathbf{v}} = 0. \quad (\text{A.2})$$

Equations (A.1) and (A.2) yields

$$\boldsymbol{\lambda} = -(\hat{J}_I \hat{M}^{-1} \dot{\hat{J}}_I^T)^{-1} (\hat{J}_I \hat{M}^{-1} (\hat{h} + \hat{g} + \hat{N}\mathbf{u}) + \dot{\hat{J}}_I \hat{\mathbf{v}}). \quad (\text{A.3})$$

Since $\mathbf{q}_{\text{body}}$ and $\mathbf{v}_{\text{body}}$ is not included in $\dot{\hat{J}}_I \hat{\mathbf{v}}$, $\boldsymbol{\lambda}$ is a function of $\mathbf{q}, \mathbf{v}$, and $\mathbf{u}$. This means that

$$\dot{\mathbf{v}} = [O \ I_5] \dot{\hat{\mathbf{v}}} = [O \ I_5] \hat{M}^{-1} (\hat{h} + \hat{g} + \hat{J}_I^T \boldsymbol{\lambda} + \hat{N}\mathbf{u})$$

is also a function of $\mathbf{q}, \mathbf{v}$, and $\mathbf{u}$, which can be written as

$$\dot{\mathbf{v}} = F_I(\mathbf{q})\mathbf{u} + G_I(\mathbf{q}, \mathbf{v}). \quad (\text{A.4})$$

Next, we derive the equation of state jump at the discrete event $I^- I^+$. The velocity change at the ground impact is given in the following equation.

$$\hat{M}(\mathbf{q}(t^-))(\hat{\mathbf{v}}(t^+) - \hat{\mathbf{v}}(t^-)) = \hat{J}_{I^+}(\mathbf{q}(t^-))^T \boldsymbol{\Lambda} \quad (\text{A.5})$$

Here, the vector $\boldsymbol{\Lambda}$ denotes the impulsive force that acts on the swing foot. After the impact, the velocity of the swing foot becomes 0, which is written

as

$$\hat{J}_{I+}\hat{\mathbf{v}}(t^+) = 0. \quad (\text{A.6})$$

From (A.5) and (A.6) we have

$$\mathbf{\Lambda} = -(\hat{J}_{I+}\hat{M}^{-1}\hat{J}_{I+}^T)^{-1}\hat{J}_{I+}\hat{\mathbf{v}}(t^-).$$

Notice that $\hat{J}_{I+}\hat{\mathbf{v}}(t^-) = (J_{I+}\mathbf{v}(t^-) - J_{I-}\mathbf{v}(t^-)) + \hat{J}_{I-}\hat{\mathbf{v}}(t^-)$ and $\hat{J}_{I-}\hat{\mathbf{v}}(t^-) = \mathbf{0}$. Therefore we have

$$\mathbf{\Lambda} = -(\hat{J}_{I+}\hat{M}^{-1}\hat{J}_{I+}^T)^{-1}(J_{I+} - J_{I-})\mathbf{v}(t^-). \quad (\text{A.7})$$

Therefore, $\mathbf{\Lambda}$ is a function of $\mathbf{q}$ and $\mathbf{v}(t^-)$ and

$$\begin{aligned} \mathbf{v}(t^+) &= [O \ I_5] \hat{\mathbf{v}}(t^+) = [O \ I_5] \left(\hat{M}^{-1}\hat{J}_{I+}^T\mathbf{\Lambda} + \hat{\mathbf{v}}(t^-) \right) \\ &= [O \ I_5] \hat{M}^{-1}\hat{J}_{I+}^T\mathbf{\Lambda} + \mathbf{v}(t^-) \end{aligned}$$

is also a function of $\mathbf{q}$ and $\mathbf{v}(t^-)$, which can be rewritten as

$$\mathbf{v}(t^+) = H_{I-I+}(\mathbf{q})\mathbf{v}(t^-). \quad (\text{A.8})$$

Finally, we derive the region of the continuous state space corresponding to each mode. In the mode L, the vertical component of the ground reaction force should be positive. This condition is written as

$$\lambda_y := [0 \ 1] \mathbf{\lambda} > 0. \quad (\text{A.9})$$

Moreover, the *center of pressure* should be placed inside the contact range of the stance foot, meaning

$$-w_h\lambda_y \leq \tau_{\text{anklel}} \leq w_t\lambda_y. \quad (\text{A.10})$$

Here, the constants w_h and w_t denotes the length between the ankle joint and the heel, and the length between the ankle and the toe, respectively. The above conditions together with geometric configurations determines the region $\mathcal{S}_L$ as follows.

$$\begin{aligned} \mathcal{S}_L = \{ & \mathbf{q}, \mathbf{v}, \mathbf{u} \mid \lambda_y > 0 \wedge (-w_h\lambda_y \leq \tau_{\text{anklel}} \leq w_t\lambda_y) \wedge \\ & (p_{ly} < p_{ry} \vee (p_{ly} = p_{ry} \wedge v_{ly} < v_{ry})) \} \end{aligned} \quad (\text{A.11})$$

The region $\mathcal{S}_{LR}$ is given by a series of conditions meaning; a ground impact should take place, the vertical component of the impact force should be positive, and the vertical component of the left foot should be positive after the

impact.

$$\mathcal{S}_{\text{LR}} = \{\mathbf{q}, \mathbf{v} \mid p_{\text{l}_y} = p_{\text{r}_y} \wedge v_{\text{l}_y}^- > v_{\text{r}_y}^- \wedge \Lambda_y > 0 \wedge v_{\text{l}_y}^+ > v_{\text{r}_y}^+\} \quad (\text{A.12})$$

where $\mathbf{v}_i^-$ and $\mathbf{v}_i^+$ denote the velocity of foot indicated by i relative to the body, before and after the ground impact, respectively. The remaining regions, $\mathcal{S}_{\text{R}}$ and $\mathcal{S}_{\text{RL}}$ are simply the opposite case of $\mathcal{S}_{\text{L}}$ and $\mathcal{S}_{\text{LR}}$, respectively.