# Graph-based Model Predictive Control of a Planar Bipedal Robot

Yuichi Tazaki and Jun-ichi Imura

*Abstract*— **In this paper, we propose a new offline model predictive control for a planar bipedal robot, which we call here Graph-based Model Predictive Control. This method consists of two phases : the graph construction phase and the real-time control phase. The directed graph is constructed off line by i) placing a certain number of nodes on the state space of the robot, ii) computing the optimal path starting from each of the graph nodes with respect to a given cost function, and iii) creating a set of directed edges by taking the first edge of each of the optimal paths. This means that one can achieve receding horizon control in some sense by simply tracing the edges of the directed graph and therefore the real-time computational cost is dramatically reduced compared with the ordinary MPC. In addition, by constructing multiple directed graphs based on different cost functions, one can design multiple motions and switching trajectories among them in a uniform way. The proposed method is applied to the speed changing control problem of a bipedal walker on a two-dimensional plane and its effectiveness is verified by numerical simulation.**

## I. Introduction

Bipedal robots need to realize various motions for adopting to a real environments; 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. More precisely, when the robot is commanded to change its motion, it has to generate a transient trajectory which starts from its posture and velocity at that point of time, and converges to the new motion.

For this purpose, a number of locomotion controllers and motion planners has been proposed in the previous literatures [1][2][3][4]. However, these do not take into account optimality such as minimization of energy consumption.

Recently, a great deal of research interest has been paid on Model Predictive Control (MPC for short) since it can effectively cope with nonlinear dynamics and constraints on the state and the input. The control law of MPC is quite simple and therefore it is applicable to a wide range of systems including bipedal robots. However, the usual MPC requires to solve the finite-time optimal control problem in real time and this requirement prevents MPC from being applied to relatively high-dimensional systems.

Imura et al.[5][6] has proposed a new MPC strategy in an obstacle avoidance problem of a vehicle. In their method, the time axis and state space are discretized and therefore the optimal control problem to be solved at each sampling time in MPC is reduced to an optimal path-finding problem over the candidates of waypoints. However, since discretization and optimal path-finding are computed in real time, there still remain computational difficulties in the case of high-dimesional systems.

Y. Tazaki is with the Department of Mechanical and Environmental Informatics, Tokyo Institute of Technology, Tokyo, Japan tazaki@cyb.mei.titech.ac.jp

J. Imura is with the Department of Mechanical and Environmental Informatics, Tokyo Institute of Technology, Tokyo, Japan imura@mei.titech.ac.jp

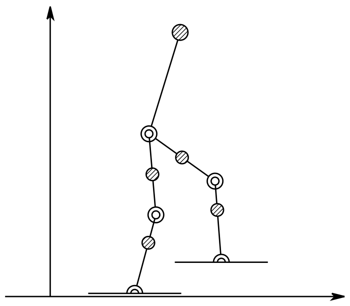


Fig. 1. Model of a planar bipedal robot.

In this paper, we propose a new method, which we call Graph-based MPC. In this method, the discretization and the optimal path-finding are moved offline, which dramatically decreases the real-time computational cost compared with ordinary MPC and hence enables application to high-dimensional systems such as bipedal robots. Furthermore, by using multiple cost functions, this method provides a uniform way to design multiple stationary motions and transient motions among them. The effectiveness of Graph-based MPC is presented in a variable speed control of a planar bipedal walker.

This paper is organized as follows. In Section II, we formalize a planar bipedal robot as a hybrid system model. In Section III, the Graph-based MPC is discribed in detail. Section IV shows numerial simulations. Concluding remarks are made in section V.

## II. A Hybrid System Model of a Planar Bipedal Robot

Fig. 1 shows a planar bipedal robot. 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 collision between the swing foot and the ground is completely inelastic; the velocity of the swing foot always becomes $0$ immediately after the collision.

We define $\boldsymbol{x} = \left(\boldsymbol{q}^T, \frac{d}{dt}\boldsymbol{q}^T\right)^T$ as the state of the robot, where the vector $\boldsymbol{q}$ denotes the robot's posture given as $\boldsymbol{q} = (\theta_{\text{body}}, \theta_{\text{hip}_\text{l}}, \theta_{\text{hip}_\text{r}}, \theta_{\text{kneel}}, \theta_{\text{kneer}})^T$. There exists an actuator on each joint of the robot, and hence the input of the robot is defined as a vector of all actuator torques written as $\boldsymbol{u} = (\tau_{\text{hip}_\text{l}}, \tau_{\text{hip}_\text{r}}, \tau_{\text{kneel}}, \tau_{\text{kneer}}, \tau_{\text{anklel}}, \tau_{\text{ankler}})^T$. The components of $\boldsymbol{q}$ and $\boldsymbol{u}$ are as follows.

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

$(\cdot)_\text{l}$ and $(\cdot)_\text{r}$ denote symbols related to the left and right leg of the robot, respectively.

The dynamics of the robot consists of four different modes (i.e., discrete states) : L, R, LR, and RL. Mode L corresponds to the case when the robot stands on its left foot, and mode R to the case of the right foot. When the system is in mode L and the swing foot (which is the right foot) hits the ground, an impulsive force acts on the right foot and the velocity of the robot changes instantaneously. Consequently, the stance leg switches from the left leg to the right leg, which indicates the mode that is about to change to R. We call this event mode LR. Similarly, mode RL corresponds to the case from mode R to mode L.

The behavior of the system is discribed by the following ordinary differential equations and algebraic equations.

$$\begin{cases} \begin{array}{ll} \ddot{\boldsymbol{q}}(t) = F_\text{L}(\boldsymbol{q}(t))\boldsymbol{u}_\text{L}(t) + G_\text{L}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) & \\ I(t) = \text{L} & \text{if } (\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t)) \in \mathcal{S}_\text{L} \\ \ddot{\boldsymbol{q}}(t) = F_\text{R}(\boldsymbol{q}(t))\boldsymbol{u}_\text{R}(t) + G_\text{R}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) & \\ I(t) = \text{R} & \text{if } (\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \boldsymbol{u}(t)) \in \mathcal{S}_\text{R} \\ \dot{\boldsymbol{q}}(t)^+ = H_\text{LR}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) & \\ \quad I(t) = \text{LR} & \text{if } (\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) \in \mathcal{S}_\text{LR} \\ \dot{\boldsymbol{q}}(t)^+ = H_\text{RL}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t) & \\ \quad I(t) = \text{RL} & \text{if } (\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) \in \mathcal{S}_\text{RL} \end{array} \end{cases} \tag{1}$$

The symbol $\dot{\boldsymbol{q}}(t)^+$ denotes the value of $\dot{\boldsymbol{q}}(t)$ immediately after the instantaneous jump. The symbol $\boldsymbol{u}_\text{L}(\boldsymbol{u}_\text{R})$ is obtained by removing the ankle joint torque of the swing leg (which is always 0 from the assumption) from $\boldsymbol{u}$. The symbol $I(t)$ is the mode of the system in time $t$. The set of $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$ whose modes are L and R are denoted by $\mathcal{S}_\text{L}$ and $\mathcal{S}_\text{R}$, respectively. Similarly, the set of $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ whose modes are LR and RL are denoted by $\mathcal{S}_\text{LR}$ and $\mathcal{S}_\text{RL}$, respectively.

## III. Graph-based Model Predictive Control

### A. Brief Discription of the Method

As shown in Fig. 2, the architecture of Graph-based MPC consists of a directed graph constructed on the state space and a real-time controller. The directed graph is constructed offline, which we call graph construction phase, while the real-time controller performs MPC in real time, based on the directed graph. In the graph construction phase, each node of the directed graph expresses a state

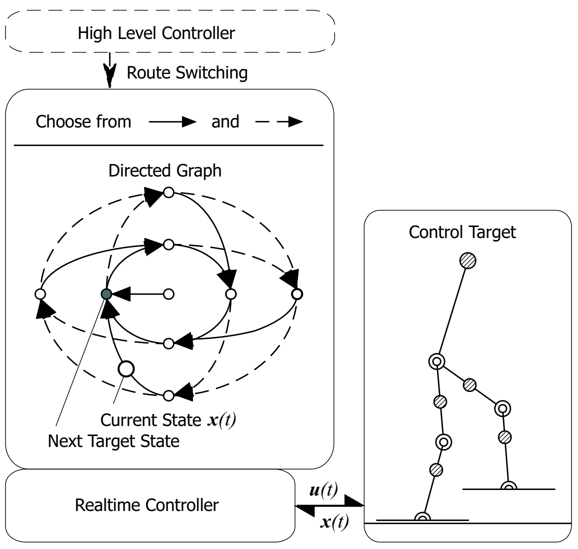


Fig. 2. Controller architecture.

(a posture and a velocity) of the robot. By considering sequences of nodes (i.e., a path) of the graph, one can compose a variety of motion trajectories that pass through each node as a waypoint. Besides, the directed graph is composed of multiple subgraphs that share the nodes. Each subgraph is responsible of realizing a certain motion of the robot; for instance, the subgraph drawn in solid lines in Fig. 2 may correspond to low-speed walking while the ones drawn in dashed lines may correspond to high-speed walking. Observe that each subgraph consists of a cyclic path, which will generate a cyclic motion, and of switching paths, which will bring the state onto the cyclic path. On the other hand, one subgraph is selected to be "active" at any time instance in the real-time control phase. Then, the real-time controller drives the state of the robot so as to trace the directed edges of the active subgraph. The desired motion is realized by simply switching the active subgraph.

As mentioned in Section I, we shall design two diffirent kinds of motions: principal motions and switching motions. The principal motion is a cyclic motion such as steady-state walking. The switching motion is a transient motion describing a transition from one principal motion to another. Taking this in mind, the procedure for the design of the directed graph is decomposed into the following steps.

Step 1: Design path cost functions for each motion to be realized.

Step 2: Design a set of waypoints and a set of transition times

Step 3: Construct sets of directed edges.

In Step 1, we design a cost function of a path (which we call a path cost function) for each principal motion. The path cost function measures the desirability of a path according to the corresponding motion. In Step 2, we design a set of waypoints and a set of transition times for each principal motion by optimization with respect to the corresponding path cost function. In Step 3, using each path cost function, we create a set of directed edges by means of the dynamic programming and the receding horizon policy. In this step, not only the directed edges for principal motions but also those for switching motions are obtained at the same time. The above three steps are detailed in Section III-C, III-D, and III-E, respectively. Beforehand, in the next section, we will discuss how to generate a continuous-time control input which drives the state from a initial waypoint to a target waypoint in certain amount of time in the real-time control.

## *B. Inter-waypoint Control*

One of the building blocks of Graph-based MPC is a control law, which generates a trajectory between given waypoints $\boldsymbol{x}_k = \left(\boldsymbol{q}_k^T, \dot{\boldsymbol{q}}_k^T\right)^T$ and $\boldsymbol{x}_{k+1} = \left(\boldsymbol{q}_{k+1}^T, \dot{\boldsymbol{q}}_{k+1}^T\right)^T$. If $\boldsymbol{x}_k$ is an element of $\mathcal{S}_{\mathrm{LR}}$ or $\mathcal{S}_{\mathrm{RL}}$, it jumps instantaniously to the next state independent of control inputs according to equation (1). Hence, we concentrate on the case that $\boldsymbol{x}_k$ is an element of $\mathcal{S}_{\mathrm{L}}$ or $\mathcal{S}_{\mathrm{R}}$. Then, the problem is formalized as follows.

**[Problem 1]**

*Suppose the initial state $\boldsymbol{x}(0) = \boldsymbol{x}_k$, the final state $\boldsymbol{x}_{k+1}$, $h_k > 0$, and $R > 0$ are given. Then for the system (1), find a control input $\boldsymbol{u}$ that minimizes the cost function*

$$J(\boldsymbol{u}\,;\,h_k) = \int_0^{h_k} \boldsymbol{u}(\tau)^T R \boldsymbol{u}(\tau) d\tau \tag{2}$$

*satisfying $\boldsymbol{x}(h_k) = \boldsymbol{x}_{k+1}$.* This problem, which is an optimal control problem of a hybrid system, is difficult to solve in a straightforward manner. For this reason, we consider solving this problem approximately. We assume the following condition, which will be guaranteed in III-D.

**[Mode Invariance Condition]**

*No mode change occurs during each sampling time interval.*

While the above condition is satisfied, the behavior of the system during each sampling time interval is described by the following equation of motion.

$$\ddot{\boldsymbol{q}} = F_{\mathrm{I}}(\boldsymbol{q})\boldsymbol{u}_{\mathrm{I}} + G_{\mathrm{I}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{3}$$

In the above equation of motion, I is one of $\{\mathrm{L}, \mathrm{R}\}$ and it is determined by the initial state and the target state according to the following rules.

$$\mathrm{I} = \begin{cases} L & \text{if } \ p_{\mathrm{l}_y}(\boldsymbol{q}_k) < p_{\mathrm{r}_y}(\boldsymbol{q}_k),\ p_{\mathrm{l}_y}(\boldsymbol{q}_{k+1}) \le p_{\mathrm{r}_y}(\boldsymbol{q}_{k+1}) \\ R & \text{if } \ p_{\mathrm{l}_y}(\boldsymbol{q}_k) > p_{\mathrm{r}_y}(\boldsymbol{q}_k),\ p_{\mathrm{l}_y}(\boldsymbol{q}_{k+1}) \ge p_{\mathrm{r}_y}(\boldsymbol{q}_{k+1}) \end{cases} \tag{4}$$

Here, $p_{\mathrm{l}_y}$ and $p_{\mathrm{r}_y}$ indicate the vertical components of position vectors from the center of the robot's body to the left and right ankles, respectively.

**Remark.** For the pair $(\boldsymbol{q}_k, \boldsymbol{q}_{k+1})$ such that neither of the conditions (4) holds, the mode invariance condition may not in general be satisfied with any input. In this sense, such a pair is said to be uncontrollable and ignored in this subsection.

Let us introduce an input transformation as follows.

$$\boldsymbol{u}_{\mathrm{I}} = F_{\mathrm{I}}(\boldsymbol{q})^{-1}(\boldsymbol{v} - G_{\mathrm{I}}(\boldsymbol{q}, \dot{\boldsymbol{q}})) \tag{5}$$

Then, one obtains the following double integrator system.

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{v} \quad A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{6}$$

Substituting (5) into (2) yields

$$\begin{aligned} J_{\boldsymbol{v}}(\boldsymbol{v}\,;\,h_k) &= \int_0^{h_k} ||\boldsymbol{v}(\tau) - Y_{\mathrm{I}}(\boldsymbol{q}(\tau), \dot{\boldsymbol{q}}(\tau))||^2_{R_{\mathrm{I}}(\boldsymbol{q})} d\tau \\ R_{\mathrm{I}}(\boldsymbol{q}) &= X_{\mathrm{I}}(\boldsymbol{q})^{-T} \hat{R} X_{\mathrm{I}}(\boldsymbol{q})^{-1} \end{aligned} \tag{7}$$

which is a cost function of $\boldsymbol{v}$. Here, $||x||^2_M = x^T M x$ and $\hat{R}$ is obtained by removing a row and a column corresponding to the ankle torque of the swing leg from $R$. Since the cost function (7) has nonlinear terms on $\boldsymbol{q}(\tau)$ and $\dot{\boldsymbol{q}}(\tau)$, we approximate the cost function by replacing $\boldsymbol{q}(\tau)$ and $\dot{\boldsymbol{q}}(\tau)$ with $(\boldsymbol{q}_k + \boldsymbol{q}_{k+1})/2$ and $(\dot{\boldsymbol{q}}_k + \dot{\boldsymbol{q}}_{k+1})/2$, respectively.

$$\begin{aligned} \tilde{J}_{\boldsymbol{v}}(\boldsymbol{v}\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) &= \int_0^{h_k} ||\boldsymbol{v}(\tau) - \bar{Y}_{\mathrm{I}}||^2_{\bar{R}_{\mathrm{I}}} d\tau \\ \bar{Y}_{\mathrm{I}} &= Y_{\mathrm{I}}((\boldsymbol{q}_k + \boldsymbol{q}_{k+1})/2, (\dot{\boldsymbol{q}}_k + \dot{\boldsymbol{q}}_{k+1})/2) \\ \bar{R}_{\mathrm{I}} &= X_{\mathrm{I}}((\boldsymbol{q}_k + \boldsymbol{q}_{k+1})/2)^{-T} \hat{R} X_{\mathrm{I}}((\boldsymbol{q}_k + \boldsymbol{q}_{k+1})/2)^{-1} \end{aligned} \tag{8}$$

Thus, Problem 1 is approximately transformed into the following fixed-terminal optimal control problem of a double integrator system.

**[Problem 2]**

*For the system (6), find a control input $\boldsymbol{v}$ that minimizes the cost function (8) subject to $\boldsymbol{x}(0) = \boldsymbol{x}_k$, $\boldsymbol{x}(h_k) = \boldsymbol{x}_{k+1}$.*

**[Theorem]**

*Problem 2 has an analytical solution and the optimal input $\boldsymbol{v}^*$, the optimal state trajectory $\boldsymbol{x}^*$ and the minimum value of the cost function $\tilde{J}_{\boldsymbol{v}}{}^*$ are given as follows.*

$$\begin{aligned} &\boldsymbol{x}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) = \\ &\quad \left(\boldsymbol{q}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k)^T \quad \tfrac{d}{dt}\boldsymbol{q}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k)^T\right)^T \end{aligned} \tag{9}$$

$$\boldsymbol{v}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) = \frac{d^2}{dt^2}\boldsymbol{q}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) \tag{10}$$

$$\begin{aligned} &\boldsymbol{q}^*(t\,;\,\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) = \\ &\quad \begin{bmatrix} \boldsymbol{q}_k & \dot{\boldsymbol{q}}_k & \boldsymbol{q}_{k+1} & \dot{\boldsymbol{q}}_{k+1} \end{bmatrix} \begin{bmatrix} 1 & 0 & -\frac{3}{h_k^2} & \frac{2}{h_k^3} \\ 0 & 1 & -\frac{2}{h_k} & \frac{1}{h_k^2} \\ 0 & 0 & \frac{3}{h_k^2} & -\frac{2}{h_k^3} \\ 0 & 0 & -\frac{1}{h_k} & \frac{1}{h_k^2} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \end{aligned} \tag{11}$$

$$\begin{aligned} &\tilde{J}_{\boldsymbol{v}}{}^*(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) = \\ &\begin{bmatrix} \boldsymbol{q}_k \\ \dot{\boldsymbol{q}}_k \\ \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \\ \bar{Y}_{\mathrm{I}} \end{bmatrix}^T \begin{bmatrix} \bar{R}_{\mathrm{I}}\frac{12}{h_k^3} & \bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & -\bar{R}_{\mathrm{I}}\frac{12}{h_k^3} & \bar{R}_{\mathrm{I}}\frac{6}{h_k} & 0 \\ \bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & \bar{R}_{\mathrm{I}}\frac{4}{h_k} & -\bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & \bar{R}_{\mathrm{I}}\frac{2}{h_k} & \bar{R}_{\mathrm{I}} \\ -\bar{R}_{\mathrm{I}}\frac{12}{h_k^3} & -\bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & \bar{R}_{\mathrm{I}}\frac{12}{h_k^3} & -\bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & 0 \\ \bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & \bar{R}_{\mathrm{I}}\frac{2}{h_k} & -\bar{R}_{\mathrm{I}}\frac{6}{h_k^2} & \bar{R}_{\mathrm{I}}\frac{4}{h_k} & -\bar{R}_{\mathrm{I}} \\ 0 & \bar{R}_{\mathrm{I}} & 0 & -\bar{R}_{\mathrm{I}} & \bar{R}_{\mathrm{I}}h \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_k \\ \dot{\boldsymbol{q}}_k \\ \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \\ \bar{Y}_{\mathrm{I}} \end{bmatrix} \end{aligned} \tag{12}$$

The solution to Problem 1 given by the above theorem combined with the input transformation (5) is suboptimal since the cost function is approximated in (8). However, it has the following useful aspects: i) it guarantees the terminal condition $\boldsymbol{x}(h_k) = \boldsymbol{x}_{k+1}$, and ii) the solution is given in a explicit form. These aspects plays an fundamental role in the optimization of the waypoints and the transition times discussed in III-D. It will also be utilized to improve the robustness of the real-time controller, which is discussed in III-F.

## C. Design of Path Cost Functions

For Step 1, let us consider a $N$-step path starting from a certain waypoint $\boldsymbol{x}_0$

$$(\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N, h_0, h_1, \cdots, h_{N-1}) \tag{13}$$

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

$$J_{\mathrm{path}}(\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N, h_0, h_1, \cdots, h_{N-1}; \gamma) = \sum_{k=0}^{N-1} \left\{ \tilde{J_{\boldsymbol{v}}}^*(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k) - \gamma Pr^x(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) \right\} \tag{14}$$

In (14), $\tilde{J_{\boldsymbol{v}}}^*(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k)$ expresses the transition cost between waypoints. On the other hand, $Pr^x(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ is a function which returns a horizontal displacement of the body of the robot when the state moves from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$. Although the state variable $\boldsymbol{x}$ does not include the displacement of the robot's body with respect to the origin of the inertial frame, one can calulate $Pr^x(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ from the change of the posture in conjunction with the assumption that no slipping occurs between the stance foot and the ground; that is,

$$Pr^x(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \begin{cases} p_{\mathrm{l}_x}(\boldsymbol{q}_k) - p_{\mathrm{l}_x}(\boldsymbol{q}_{k+1}) & \text{if } I_k = \mathrm{L} \\ p_{\mathrm{r}_x}(\boldsymbol{q}_k) - p_{\mathrm{r}_x}(\boldsymbol{q}_{k+1}) & \text{if } I_k = \mathrm{R} \end{cases}. \tag{15}$$

Here, $\boldsymbol{q}_i$ represents the posture corresponding to the state $\boldsymbol{x}_i$ and $I_k$ is the mode when the state moves from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$, which is determined by (4). The functions $p_{\mathrm{l}_x}$ and $p_{\mathrm{r}_x}$ expresses the horizontal components of position vectors from the body center to the left and right ankle, respectively. Hence, the objective represented by the cost function (14) is to increase the walking distance while decreasing the energy consumption. Note that so long as $N$ is fixed, the walking distance and the energy consumption is in a trade-off. The point of balance between these quantities is determined by the scalar parameter $\gamma > 0$; as $\gamma$ is larger, the walking distance is made more important. In particular, when $\gamma$ is set to 0, the walking distance is no longer taken into account and this setting is utilized to obtain a stand-still motion.

## D. Design of Waypoints and Transition times

In Step 1, we have related a path cost function to each of the motions that is to be realized. In Step 2, we design a set of waypoints and a set of transition times, which form the cyclic path of each principal motion, based on the corresponding path cost function.

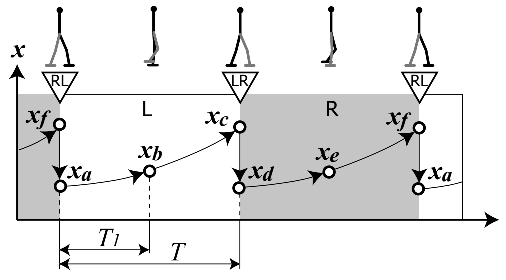
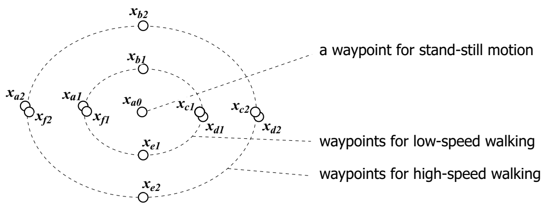


Fig. 3. Arrangement of waypoints and transition times.



xb2
xb1
xa2
xf2
xa1
xf1
xa0
xc1
xd1
xc2
xd2
xe1
xe2
a waypoint for stand-still motion
waypoints for low-speed walking
waypoints for high-speed walking

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

At first, we shall discuss the case of steady-state walking, where $\gamma > 0$. The following assumptions are made for simplicity of discussion.

**[Assumption]**

(a) The sequence of the mode in a steady-state walking is L $\rightarrow$ LR $\rightarrow$ R $\rightarrow$ RL $\rightarrow$ L $\rightarrow \cdots$.

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

Based on Assumption (a), Fig. 3 illustrates an arrangement of waypoints and transition times for steady-state walking. The lower part of the figure shows the trajectory of the state in the steady-state walking. The upper part shows the posture of the robot on the corresponding time instant. For guaranteeing the mode invariance condition, when a mode boundary is crossed, the state must pass a waypoint on the boundary. The waypoint $\boldsymbol{x}_a, \boldsymbol{x}_c, \boldsymbol{x}_d$, and $\boldsymbol{x}_f$ are placed for this purpose. On the other hand, $\boldsymbol{x}_b$ and $\boldsymbol{x}_e$ are placed to ensure a certain amount of clearance between the swing foot and the ground to avoid "ground scuffing". From Assumption (b), the following relations hold.

$$\begin{aligned} \boldsymbol{q}_d &= C\boldsymbol{q}_a \\ \dot{\boldsymbol{q}}_d &= C\dot{\boldsymbol{q}}_a \\ \boldsymbol{q}_e &= C\boldsymbol{q}_b \\ \dot{\boldsymbol{q}}_e &= C\dot{\boldsymbol{q}}_b \\ \boldsymbol{q}_f &= C\boldsymbol{q}_c \\ \dot{\boldsymbol{q}}_f &= C\dot{\boldsymbol{q}}_c \end{aligned} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{16}$$

Here, $\boldsymbol{x}_i = (\boldsymbol{q}_i^T, \dot{\boldsymbol{q}}_i^T)^T \quad i \in \{a, b, \cdots, f\}$. In addition, from (1), the following relations hold.

$$\boldsymbol{q}_d = \boldsymbol{q}_c, \ \ \dot{\boldsymbol{q}}_d = H_{\mathrm{LR}}(\boldsymbol{q}_c)\dot{\boldsymbol{q}}_c$$
$$\boldsymbol{q}_a = \boldsymbol{q}_f, \ \ \dot{\boldsymbol{q}}_a = H_{\mathrm{RL}}(\boldsymbol{q}_f)\dot{\boldsymbol{q}}_f$$

Consequently, the waypoints $\boldsymbol{x}_b$, $\boldsymbol{x}_c$, and the length of time interval $T_1$ and $T$ are the independent variables. Here, it is natural to measure the desirability of the gait realized by the above set of variables with the path cost function in one cycle of the gait. Moreover, by Assumption (b), one cycle of the gait costs exactly twice as much as half a cycle. Hence, we shall define the following cost function for the optimization:

$$J_{\mathrm{cycle}}(\boldsymbol{x}_b, \boldsymbol{x}_c, T_1, T; \gamma_i) = J_{\mathrm{path}}(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c, T_1, T - T_1; \gamma_i) \tag{17}$$

where $\boldsymbol{x}_a = ((C\boldsymbol{q}_c)^T, (H_{\mathrm{RL}}(C\boldsymbol{q}_c)C\dot{\boldsymbol{q}}_c)^T)^T$. We employ Downhill-Simplex method for optimization [7]. 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 $\boldsymbol{x}^*(t, \boldsymbol{x}_a, \boldsymbol{x}_b) \in \mathcal{S}_{\mathrm{LR}}$, which physically implies a "ground scuffing". In order to get rid of this case, each time a candidate $(\boldsymbol{x}_b, \boldsymbol{x}_c, T_1, T)$ is evaluated in the iteration of Downhill-Simplex method, we give an sufficiently large penalty when the corresponding trajectory breaks the mode invariance condition. This check can be efficiently computed since the intermediate trajectory is available in the explicit form expressed in (11).

On the other hand, the stand-still motion is generated by a cyclic trajectory that passes through a single waypoint. Here, no special optimization is taken for the waypoint of the stand-still motion. Instead, it is specified by the designer.

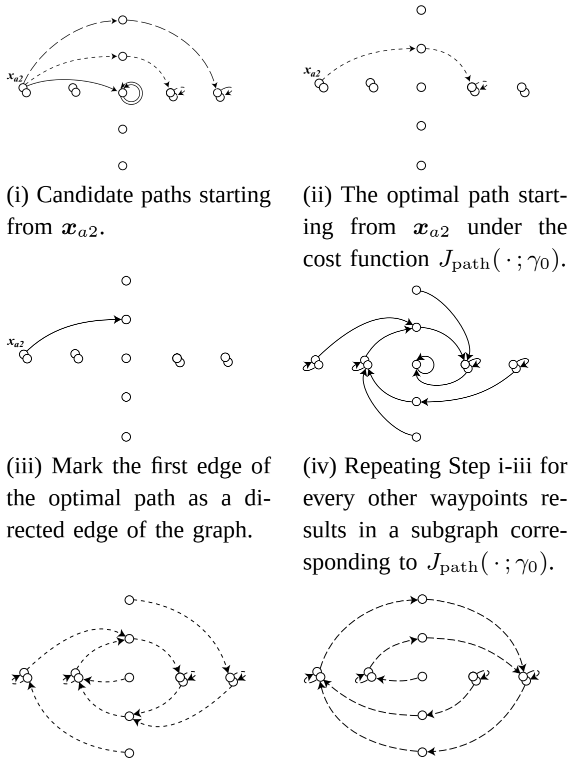
Let us denote by $\mathcal{X}_i$ and $\mathcal{T}_i$ the set of waypoints and the set of transition times optimized with cost function $J_{\mathrm{cycle}}(\cdot; \gamma_i)$, respectively. In addition, we define $\mathcal{X} := \cup\mathcal{X}_i, \mathcal{T} := \cup\mathcal{T}_i$.

Fig. 4 shows $\mathcal{X}$ projected and plotted onto a two-dimensional plane. Two sets of waypoints for low-speed and high-speed steady-state walking and a waypoint for the stand-still motion are drawn in the figure. The projection of a state $\boldsymbol{x}$ to a two-dimensional vector is done by calculating a position vector which points from the left ankle to the right ankle in a given state $\boldsymbol{x}$.

## *E. Construction of Subgraphs*

In Step 3, we generate subgraphs onto the set of waypoints $\mathcal{X}$ designed in Step 2. Each stage in Step 3 is illustrated in Fig. 5. In this step, the following procedures are processed using each path cost function $J_{\mathrm{path}}(\cdot; \gamma_i)$. For each waypoint $\boldsymbol{x} \in \mathcal{X}$, at first, the optimal $N$-step path among the $N$-step paths starting from $\boldsymbol{x}$ is determined (Fig. 5(i, ii)). Next, the first edge of the optimal path is marked as a directed edge, which composes a resultant subgraph (Fig. 5(iii)). Thus we obtain a subgraph composed of directed edges, each of which is the first edge of the optimal path starting from the corresponding waypoint (Fig. 5(iv)). Repeating the above procedures for every path cost function, we obtain multiple subgraphs (Fig. 5(iv, v)).

In the real-time control phase, the system is driven so as to trace the directed edges of a specific set. In other words, it is only the first edge of each of the $N$-step optimal paths



(i) Candidate paths starting from $\boldsymbol{x}_{a2}$.

(ii) The optimal path starting from $\boldsymbol{x}_{a2}$ under the cost function $J_{\mathrm{path}}(\,\cdot\,; \gamma_0)$.

(iii) Mark the first edge of the optimal path as a directed edge of the graph.

(iv) Repeating Step i-iii for every other waypoints results in a subgraph corresponding to $J_{\mathrm{path}}(\,\cdot\,; \gamma_0)$.

(v) Process Step i-iv with respect to other cost functions ($J_{\mathrm{path}}(\,\cdot\,; \gamma_1)$ and $J_{\mathrm{path}}(\,\cdot\,; \gamma_2)$).

Fig. 5. Procedures of the design of subgraphs. $N$ is given by 3.

that is actually traced in real time. This corresponds to the receding horizon policy of the usual MPC.

Observe that each of the directed graphs is composed of a cyclic path and one or more paths that connect to the cyclic path. The cyclic path is constructed onto the set of waypoints that has been designed in Step 2 using the corresponding path cost function. On the other hand, each of the paths connecting to the cyclic path works as a switching path, which connects the waypoint outside the cyclic path to the waypoint on the cyclic path. Note that such switching paths are determined based on the optimality with respect to the path cost function corresponding to the cyclic path they connect to, and the receding horizon policy.

It is noteworthy to point out that the $N$-step optimal path has the following recursive characteristics:

$$J^*_{\mathrm{path}}(\boldsymbol{x}_0; N, \gamma) = \min_{(\boldsymbol{x}_0, \boldsymbol{x}_1, h) \in \mathcal{E}} \left\{ J_{\mathrm{path}}(\boldsymbol{x}_0, \boldsymbol{x}_1, h; \gamma) + J^*_{\mathrm{path}}(\boldsymbol{x}_1; N-1, \gamma) \right\}. \tag{18}$$

Here, $J^*_{\mathrm{path}}(\boldsymbol{x}_0; N, \gamma)$ is a cost of the $N$-step optimal path starting from the waypoint $\boldsymbol{x}_0$. Besides, $\mathcal{E}$ is defined as a set of every tuples $(\boldsymbol{x}_0, \boldsymbol{x}_1, h) \in \mathcal{X} \times \mathcal{X} \times \mathcal{T}$ that the trajectory $\boldsymbol{x}^*(t; \boldsymbol{x}_0, \boldsymbol{x}_1, h)$ fulfills the mode invariance condition. Taking advantage of this recursive characteristics, the computation of the $N$-step optimal path starting from each waypoint can be carried out efficiently by means of

TABLE I
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] |

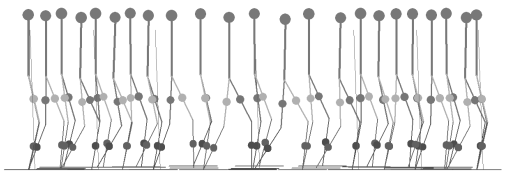


Fig. 6. Gait of bipedal walker.

dynamic programming. First, compute a 1-step optimal path for every waypoint. This takes the computational time propotional to $|\mathcal{E}|$ (the number of elements in $\mathcal{E}$). Next, compute a 2-step optimal path for every waypoint. Since the 1-step optimal paths have been already obtained in the preceding step, this step also takes the computational time propotional to $|\mathcal{E}|$ using (18). Repeating this procedure until the length of the path reaches $N$ results in the total compuational time propotional to $N|\mathcal{E}|$.

### F. Real-time Control

In this paper, we assume that the state at the initial time $t_0$ is set onto one of the waypoints of the directed graph. As mentioned in Section III-A, at any time instance, one of the multiple directed graphs is selected as active. Then, at each sampling time $t_k$ in the real-time control phase, the real-time controller looks up the directed edge $(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k)$ of the selected directed graph in order to determine the next target waypoint. In each sampling time interval $t \in [t_k, t_{k+1}]$, the controller applies a feedforward, continuous-time input by which the state will arrive at the waypoint $\boldsymbol{x}_{k+1}$ at the next sampling time $t_{k+1}$. Such an input is obtained by (10) in conjunction with real-time inverse dynamics computation given in (5).

So far, the real-time control law is a feedforward type of controller; it assumes $\boldsymbol{x}(t_k) = \boldsymbol{x}_k$. However, in practice, it is likely that this assumption will break due to disturbances or model errors; as a consequence, the above control law will become useless. In order to overcome this problem, consider replacing $\boldsymbol{x}_k$ with $\boldsymbol{x}(t_k)$ in $\boldsymbol{v}^*$. Then, using input $\boldsymbol{v}^*(t\,; \boldsymbol{x}(t_k), \boldsymbol{x}_{k+1}, h_k)$ instead of $\boldsymbol{v}^*(t\,; \boldsymbol{x}_k, \boldsymbol{x}_{k+1}, h_k)$, gives some sort of feedback effect to the controller.

## IV. NUMERICAL RESULT

In this section, the effectiveness of the proposed method is verified by numerical simulations. Physical parameters of the bipedal robot are listed in Table I. Values of the weight parameter $\gamma$ of the path cost functions for the

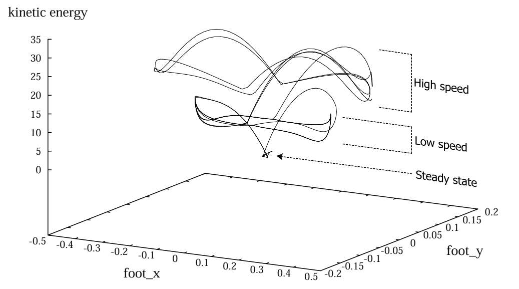


Fig. 7. A phase curve in variable speed locomotion.

stand-still motion, the low-speed walking, and the high-speed walking are set to 0, 6000, and 12000, respectively. Prediction length $N$ is given by 100. Fig. 6 shows the gait of the robot. At the initial time, the state is set onto the waypoint of the standing-still motion and the subgraph of the standing-still motion is selected to be active. Then, the robot begins walking by switching the active subgraph to low-speed-walking subgraph. Afterwards, the active subgraph is switched to high-speed-walking, low-speed-walking, stand-still subgraph in that order on each three step of the walking. Fig. 7 shows a projected phase portrait of the state in the same example. Here, the vertical axis represents the kinetic energy. The horizontal and depth axes represent $x$ and $y$ component of the position vector from the left ankle the right ankle, respectively.

## V. CONCLUSION

In this research, a new offline MPC which we call Graph-based MPC is presented and is applied to a variable speed control of a planar bipedal walker. So far, we have introduced a graph-construction scheme for steady-state walking. For future works, the method should be extended to handle more complex tasks such as obstacle avoidance.

## REFERENCES

[1] M. Okada, K. Osato, and Y. Nakamura : "Motion Emergence of Humanoid Robots by an Attractor Design of a Nonlinear Dynamics" Proc. of the IEEE International Conference on Robotics and Automation (ICRA'05), pp.18-23, 2005.

[2] S. Kajita, F. Kanehiro, K. Kaneko, et al. : "The 3D Linear Inverted Pendulum Mode : A simple modeling for a biped walking pattern generation", Proc. of 2003 IROS, pp.239-246, 2001.

[3] L. Kovar, M. Gleicher, and F. Pighin, "Motion Graphs", In Proceeding of SIGGRAPH 02, 2002, pp. 473-482

[4] K. Yamane, Y. Nakamura : "Dynamics Filter - Concept and Implementation of On-Line Motion Generator for Human Figures", Proc. of IEEE ICRA, pp688-695, 2000

[5] J. Imura : "Optimal control of sampled-data piecewise affine systems", Automatica, Vol.40, No.4, pp.661-669, 2004.

[6] H.L. Hagenaars, 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, pp.878-883, 2004

[7] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: "Numerical Recipes in C The Art of Scientific Computing Second Edition", Cambridge University Press.