IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Footstep Modification Including Step Time and Angular Momentum Under Disturbances on Sparse Footholds

Yuta Kojio, Yuki Omori, Kunio Kojima, Fumihito Sugai, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba

***Abstract*— Maintaining dynamic balance is an important requirement for bipedal robots. To deal with large disturbances, the footsteps need to be modified depending on the disturbance. Currently, there are few methods that determine footsteps by considering foothold constraints and the balance of the robot. In this paper, we propose a footstep modification method that considers the steppable region. In certain situations, robots cannot maintain balance due to the limitations of the landing position on sparse footholds, such as stepping stones. Therefore, our proposed method modifies not only the step position, but also the step timing and the angular momentum, and balance can be maintained even on the footholds where the steppable region is strictly limited. These walking parameters are analytically calculated by representing the steppable region as convex hulls and applying our previously utilized method. We verified the effectiveness of the proposed method in an experiment where a life-sized humanoid robot walked on stepping stones consisting of unsteady blocks and was able to recover when pushed.**

I. Introduction

A humanoid robot will easily fall over because it is not fixed to the environment and the height of its center of gravity (COG) is high for its support area. Robot's dynamic balance is, therefore, important especially for practical applications, and thus balance control is widely being researched [1], [2]. In addition, there are many studies on footstep modification methods [3], [4] to withstand larger disturbances and the control method [5], [6] that integrates three strategies: ankle strategy, hip strategy, and stepping strategy [7]. However, most studies assume that the robot's surroundings are almost flat, and they consider only the robot's state. Therefore, these studies are not comparable to real environments that have steps, ditches, etc. For example, the robot will fall if it steps into a gap between stepping stones as shown in Fig.1.

As a robot cannot walk considering only the robot's state in a real environment, it has to determine its surrounding by using external sensors. Thus, many studies have also been conducted on footstep planning by using visual information [8]–[10]. However, these methods could not immediately change the position where the current swing foot lands, and did not consider robot's balance in real-time. Moreover, they used a stabilization controller, which was unable to adjust the footsteps. Specifically, Griffin *et al.* [10] stated that the robot could traverse on stepping stones in their experiments, but with a low success rate due to balancing errors.

In this study, we combine footstep determination considering foothold information and balance control including

Y. Kojio, Y. Omori, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada and M. Inaba are with Department of Mechano-Informatics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan `kojio@jsk.imi.i.u-tokyo.ac.jp`

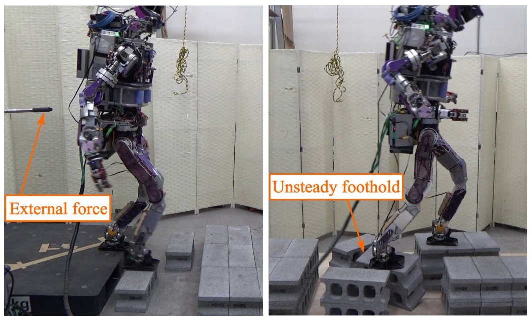


Fig. 1. Robot walking under disturbances on stepping stones.

modifications of the landing position, timing, and centroidal angular momentum. Thus, a humanoid robot is able to walk stably on areas where the steppable region is restricted. In the next section, we further compare our work with previous studies, as well as summarize our method. Our proposed method is validated through an experiment where a humanoid robot walks on stepping stones.

II. Integration of Real-time Balance Control and Vision Recognition

Recently, vision-based online footstep modification has been studied [11]–[13]. In this section, we outline previous works, and describe the specific approach and contributions of our method.

A. Related Works

Hildebrandt *et al.* [11] proposed a footstep modification method using model predictive control (MPC) to minimize the inclination of the robot. The researchers also added penalty term to the optimization to avoid collision with obstacles during push recovery. However, this solution does not necessarily satisfy the steppable region since this consideration is not a constraint but rather a penalty. In addition, their method only modified the landing position. Other strategies are required in such situations where visual information is needed as the robot cannot freely select the position.

Yamamoto *et al.* [12] presented a landing position determination method using a terrain map, which finds the position that satisfies dynamic balance and steppability by a breadth first search around the virtual rail [14]. The researchers performed a push recovery simulation on rough terrain. Additionally, there are some studies on acquiring not only the geometries but also the physical properties of the surroundings using vision or force sensors [15], [16]. It makes the robot more stable to determine footsteps by utilizing the property information obtained from the database or online estimation. In addition, adjusting gait depending on the footholds can be beneficial. Therefore, it is desired to determine footsteps considering steppable polygons rather than to sequentially check whether the position is steppable per grid. Furthermore, compensation due to a lack of the landing position was not addressed in [12].

There are many researches on an online walking parameters modification considering inequality constraints by using an MPC framework [17], [18]. However, these approaches are difficult to apply to a real-time step timing adjustment because changing the step timing affects the state in the other axis and makes the problem nonlinear. Jeong *et al.* [6] enabled to numerically calculate the landing position, timing, and angular momentum by linearizing the problem with some assumptions. In addition, the method was extended to include the upper and lower limits of the steppable region as the landing position constraint [13]. However, their method still assumes that the sagittal and lateral planes of the steppable region are independent. Although they imposed a conservative constraint so that the stepping constraint was not violated even near concave obstacles, this was equivalent to only dealing with rectangular obstacles. Moreover, it is difficult to determine the landing position on sparse footholds.

B. Overview of the Proposed Method

In this paper, we propose a footstep modification considering the steppable region, which is expressed as a series of polygons. We proposed a real-time modification method that analytically solves the step position, time, and centroidal angular momentum [19]. We extend our method so that the robot can calculate all the walking parameters considering the steppable region at high speeds. Thus, the robot can handle large disturbances even on sparse footholds where there are stringent constraints.

III. Walking Pattern Generation That Ensures Dynamic Balance at Every Step with Double Support Phase

A. Walking Pattern Generation

This subsection gives a brief description of our walking pattern generation method [19] that extended the controller by Sugihara *et al.* [20].

The motion of COG of the linear inverted pendulum model (LIPM) [21] in the sagittal plane is obtained as follows:

$$\ddot{x}_G = \omega^2(x_G - x_Z) \tag{1}$$

$$\omega := \sqrt{\frac{g}{z_G - z_Z}} \tag{2}$$

where x, y, z are the components of the position vector and subscripts G, Z denote the COG and Zero Moment Point (ZMP), respectively. g is the acceleration of gravity. Although the equations in this subsection are formulated for the x component, they apply to the y component as well with the same step time. Pratt *et al.* [22] presented the Capture Point (CP), which is a point where the ZMP should be placed for the COG to stop. In our method, we ensure the walking balance by controlling the ZMP so that the middle of the supporting foot corresponds to the CP at every step.

By letting T_s be the time when the current swinging foot lands and T_d be the time when the current supporting foot lifts off the ground, we define the ZMP reference $x_Z^{ref}(t)$ in the double support phase as the following linear function:

$$x_Z^{ref}(t) = at + b \tag{3}$$

$$a := \frac{x_Z^{next} - x_Z^{cur}}{T_d - T_s}, \qquad b := x_Z^{cur} - \frac{x_Z^{next} - x_Z^{cur}}{T_d - T_s} T_s \tag{4}$$

where x_Z^{cur}, x_Z^{next} are the constant ZMP references of the current and next steps respectively. We generate the ZMP so that the next supporting foot position corresponds to the CP when the current supporting foot is lifted. The optimization problem for keeping the ZMP as close as possible to the ZMP reference is expressed as follows:

$$x_Z(t) = \arg\min_{x_Z} \frac{1}{2}\int_t^{T_d} (x_Z(\tau) - x_Z^{ref}(\tau))^2 d\tau \tag{5}$$

$$\text{s.t.} \quad \ddot{x}_G(t) = \omega^2(x_G(t) - x_Z(t)) \tag{6}$$

$$x_{CP}(T_d) = x_f \tag{7}$$

where

$$x_{CP}(t) = x_G(t) + \frac{\dot{x}_G(t)}{\omega} \tag{8}$$

$$x_Z^{ref}(t) = \begin{cases} x_Z^{cur} & (t \le T_s) \\ at + b & (otherwise) \end{cases} \tag{9}$$

x_f is the landing position. In the case of more dynamic walking, the landing position can be set to be behind the CP depending on the maximum step length and minimum step time. The problem stated above can be solved as follows:

$$x_Z(t) = \begin{cases} x_Z^{ref}(t) + \dfrac{2(x_{CP}^Z(t) - x_f^Z e^{-\omega(T_d - t)})}{1 - e^{-2\omega(T_d - t)}} \\ \quad + \dfrac{\frac{2a}{\omega}(e^{-\omega(T_d - t)} - e^{-\omega(T_s - t)})}{1 - e^{-2\omega(T_d - t)}} & (t \le T_s) \\ x_Z^{ref}(t) + \dfrac{2(x_{CP}^Z(t) - x_f^Z e^{-\omega(T_d - t)})}{1 - e^{-2\omega(T_d - t)}} \\ \quad + \dfrac{\frac{2a}{\omega}(e^{-\omega(T_d - t)} - 1)}{1 - e^{-2\omega(T_d - t)}} & (otherwise) \end{cases} \tag{10}$$

$$x_{CP}^Z(t) := x_{CP}(t) - x_Z^{ref}(t) \tag{11}$$

$$x_f^Z := x_f - x_Z^{ref}(T_d) = x_f - x_Z^{next} \tag{12}$$

Thus the walking pattern to maintain balance is instantly calculated from the nominal landing position, timing, and measured CP.

The differential of CP is obtained from Eq.(1) and Eq.(8).

$$\dot{x}_{CP} = \omega(x_{CP} - x_Z) \tag{13}$$

Assuming that the ZMP reaches the edge of the support polygon and is constant, CP at T_d is solved by using Eq.(13).

$$x_{CP}(T_d) = e^{\omega(T_d - t)}(x_{CP}(t) - x_Z) + x_Z \tag{14}$$

We utilized this relational expression to derive a real-time modification for step position, time, and angular momentum.

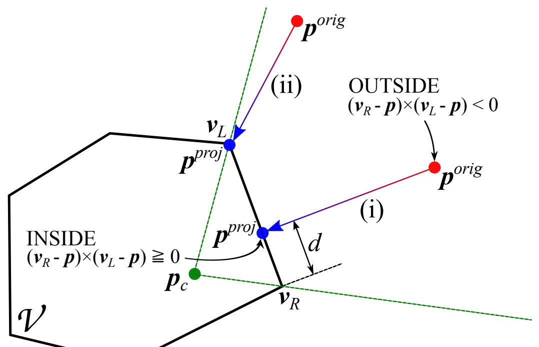


Fig. 2. Target point projection to convex hull. $\boldsymbol{p}_c$ is an arbitrary point contained in $\boldsymbol{\mathcal{V}}$, which can be calculated from the center of the vertices. Inside/outside determination is conducted by calculating a cross product from $\boldsymbol{p}$ and the two adjacent vertices $\boldsymbol{v}_R, \boldsymbol{v}_L$ obtained by using a binary search. We defined a symbol $\times$ as $\boldsymbol{a} \times \boldsymbol{b} = a_x b_y - a_y b_x$ for 2D vectors $\boldsymbol{a} = (a_x \;\; a_y)^T, \boldsymbol{b} = (b_x \;\; b_y)^T$.

B. ZMP Projection to Support Polygon

As Eq.(5) does not include a constraint for the support polygon, the ZMP generated from Eq.(10) can be outside this region. If the ZMP is outside the region, we project it onto the nearest point at the edge of the region so that the robot can keep walking without tilting. We define the support polygon $\boldsymbol{\mathcal{V}}^{sup}$ by using the horizontal vertices $\boldsymbol{v}^{sup}$.

$$\boldsymbol{\mathcal{V}}^{sup} = \left\{\boldsymbol{v}_1^{sup}, \boldsymbol{v}_2^{sup}, \ldots, \boldsymbol{v}_n^{sup} \mid \boldsymbol{v}_k^{sup} \in \mathbb{R}^2\right\} \tag{15}$$

We check whether or not the point is outside the convex hull based upon an algorithm by Preparata *et al.* [23]. The algorithm binary searches the two adjacent vertices so that the target point is included in the region formed by half line created from the vertices and any inclusive point (Fig.2). Therefore, the nearest neighbor edge to the target point is obtained secondarily using this algorithm.

The original target point is defined by $\boldsymbol{p}^{orig}$, the projected point is $\boldsymbol{p}^{proj}$, and the nearest neighbor vertices are $\boldsymbol{v}_R, \boldsymbol{v}_L$. If the perpendicular line to the nearest neighbor edge intersects the edge as shown in Fig.2 (i), the intersection point is the projected point. Otherwise, as illustrated in Fig.2 (ii), the nearest neighbor vertex is the projected point. Thus, when $\boldsymbol{p}^{orig}$ is outside the convex hull, $\boldsymbol{p}^{proj}$ is obtained as follows:

$$\boldsymbol{p}^{proj} = \begin{cases} \boldsymbol{v}_R & (d < 0) \\ \boldsymbol{v}_L & (d > 1) \\ \boldsymbol{v}_R + d(\boldsymbol{v}_L - \boldsymbol{v}_R) & (otherwise) \end{cases} \tag{16}$$

$$d := \frac{(\boldsymbol{v}_L - \boldsymbol{v}_R) \cdot (\boldsymbol{p}^{orig} - \boldsymbol{v}_R)}{\|\boldsymbol{v}_L - \boldsymbol{v}_R\|^2} \tag{17}$$

IV. Generation of Steppable Region

It is possible to represent any complex footholds as a set of steppable polygons. Additionally, any polygon can be divided into a set of convex hulls. We reduce the computational time of a real-time walking parameters modification by processing the surrounding footholds to be a set of convex hulls in advance. In this section, we assume that the steppable object has already been recognized as the 2D polygon obtained by projecting the 3D steppable object upon a horizontal plane.

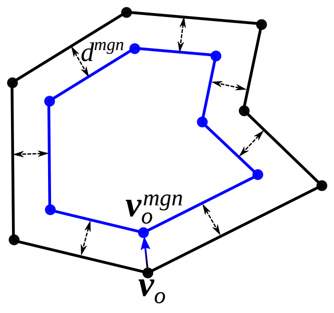


Fig. 3. Black area indicates a recognized steppable object. Blue area indicates the shrunk region used for a footstep determination.

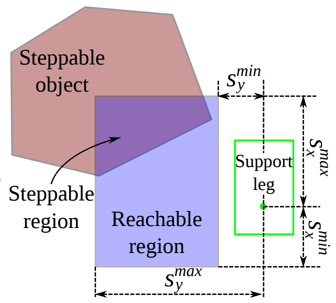


Fig. 4. Steppable region is obtained by calculating the intersection of the steppable object and the reachable region.

The recognition method of steppable object will be reported in our future paper.

A. Convex Partition of Steppable Object

In this study, we treat the landing position as a point, and we determine whether the position satisfies a stepping constraint by using this point. However, since the ZMP is changed within the sole by stabilization controller, it is desired that not only the point but also the whole sole are inside the region as much as possible. Therefore, we shrink the region of the steppable object for a footstep determination as shown in Fig.3. Then the shrunk region is divided into a set of convex hulls. In our implementation, we divide the shrunk region into triangle meshes and then combine the meshes to be a minimum number of convex hulls. This process is executed online at approximately 15 Hz.

B. Generation of Steppable Region by Calculating Convex Hull Intersections

The steppable region is defined as the intersection of the range where the foot is geometrically reachable and the region obtained by IV-A. Fig.4 illustrates an example of the steppable region. In this paper, we simply set the reachable region as a rectangle relative to the supporting foot, and the intersection is also convex.

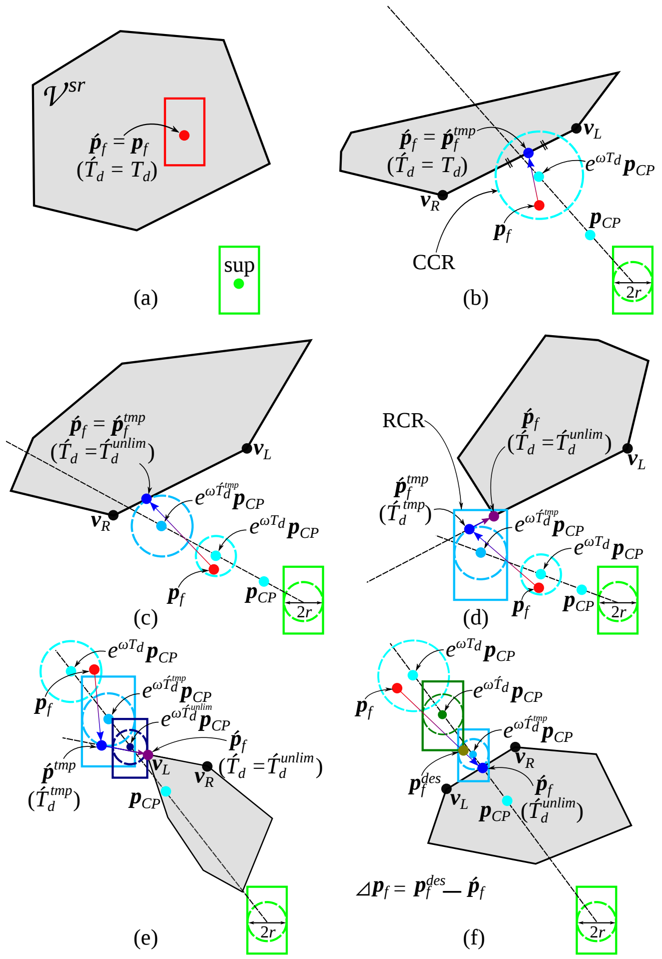
V. Modification of Walking Parameters Considering Steppable Regions

A. Modification of Walking Parameters for Single Region

The algorithm described in III-B gives the following solutions according to time complexity $\mathcal{O}(\log n)$: (i) whether the point is outside the region or not, (ii) the edge of its closest neighbor, and (iii) the projected point to the region. Making use of this, we derive a fast modification method considering the steppable region. In this subsection, we present the modification method when only one steppable region is given. First, we determine whether or not the original reference landing position $\boldsymbol{p}_f = (x_f \;\; y_f)^T$ is outside the steppable region $\boldsymbol{\mathcal{V}}^{sr}$:

$$\boldsymbol{p}_f \in \boldsymbol{\mathcal{V}}^{sr} \tag{18}$$

If $\boldsymbol{p}_f$ is given to maintain balance and Eq.(18) is satisfied, the footstep does not need to be changed (Fig.5 (a)). Otherwise, the footstep is recalculated.



$\mathcal{V}^{sr}$
$\acute{p}_f = p_f$
$(\acute{T}_d = T_d)$
sup
(a)
$\acute{p}_f = \acute{p}_f^{tmp}$
$(\acute{T}_d = T_d)$
v_L
v_R
$e^{\omega T_d} p_{CP}$
p_f
p_{CP}
CCR
2r
(b)
$\acute{p}_f = \acute{p}_f^{tmp}$
$(\acute{T}_d = \acute{T}_d^{unlim})$
$e^{\omega \acute{T}_d^{tmp}} p_{CP}$
$e^{\omega T_d} p_{CP}$
(c)
RCR
$\acute{p}_f$
$(\acute{T}_d = \acute{T}_d^{unlim})$
$\acute{p}_f^{tmp}$
$(\acute{T}_d^{tmp})$
(d)
$e^{\omega \acute{T}_d^{unlim}} p_{CP}$
(e)
$\acute{p}_f^{des}$
$\Delta p_f = p_f^{des} - \acute{p}_f$
$e^{\omega \acute{T}_d} p_{CP}$
(f)

Fig. 5. Modification of footstep with landing timing and angular momentum considering one steppable region. The following parameters are modified. (a) : Nothing. (b) : Landing position. (c), (d), (e) : Landing position and timing. (f) : Landing position, timing, and angular momentum.

1) Modification Using CCR: To analytically determine the footstep considering the steppable region, we roughly calculate it by temporarily approximating the supporting sole as a circle with a radius of r. We call this *Circular Capture Region (CCR)*, which is the Capture Region [22] using the circular sole, as shown in Fig.6. For simplicity, we regarded the current time as $t = 0$. The superscript min/max of ZMP represents the edges of the supporting sole, and the variables in this subsection are expressed in the foot coordinate frame. CCR forms a circle with a radius of $(e^{\omega T_d} - 1)r$ and a center at $e^{\omega T_d} \boldsymbol{p}_{CP}$, which is derived from Eq.(14). If the landing position is inside CCR and steppable region, dynamic balance is guaranteed when landing. Therefore, letting the vertices obtained by III-B be $\boldsymbol{v}_R = (x_R \quad y_R)^T, \boldsymbol{v}_L = (x_L \quad y_L)^T$, we can deal with conditions of the landing position $\{x_f, \ y_f\}$ and timing T_d considering the region $\boldsymbol{\mathcal{V}}^{sr}$ by using the following equation:

$$\begin{cases} \left(x_f - e^{\omega T_d} x_{CP}\right)^2 + \left(y_f - e^{\omega T_d} y_{CP}\right)^2 = \left(e^{\omega T_d} - 1\right)^2 r^2 \\ y_f = a x_f + b \end{cases} \tag{19}$$

$$a := \frac{y_L - y_R}{x_L - x_R}, \qquad b := y_R - \frac{y_L - y_R}{x_L - x_R} x_R \tag{20}$$

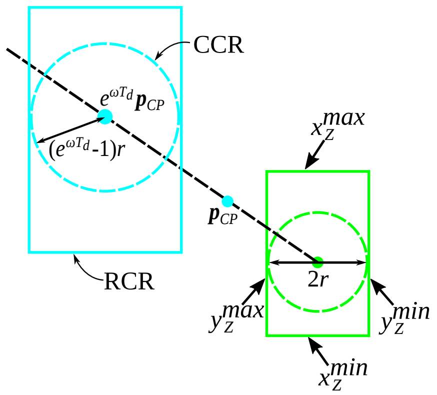


Fig. 6. CCR and RCR. $\boldsymbol{p}_{CP}$ represents the current CP and the green area indicates the supporting sole. CCR is the region where the robot can recover balance with one step when approximating the sole as a circle, and RCR is the region when approximating it as a rectangle.

x_f, y_f, and T_d can be changed from the original values according to the procedure below.

When Eq.(19) has a real number solution regarding the original step position and time, namely:

$$A e^{2\omega T_d} + B e^{\omega T_d} + C \geq 0 \tag{21}$$

$$A := \left(a^2 + 1\right) r^2 - \left(a x_{CP} - y_{CP}\right)^2 \tag{22}$$

$$B := -2\left[a b x_{CP} - b y_{CP} + \left(a^2 + 1\right) r^2\right] \tag{23}$$

$$C := \left(a^2 + 1\right) r^2 - b^2 \tag{24}$$

the equation is satisfied, CCR already intersects the steppable region without changing the step time. Therefore, the projected point of $e^{\omega T_d} \boldsymbol{p}_{CP}$ to the region is the temporarily modified landing position $\acute{\boldsymbol{p}}_f^{tmp}$ and the step time is not adjusted (Fig.5 (b)). On the other hand, when Eq.(21) is not satisfied, CCR is transferred so that it just touches the line of its nearest neighbor by changing the step time. The condition where the step time is calculated is as follows:

$$D := B^2 - 4AC \geq 0 \tag{25}$$

Then the landing timing and position are modified as follows (Fig.5 (c)):

$$\acute{T}_d^{tmp} = \begin{cases} \frac{1}{\omega} \log \frac{-B + \sqrt{D}}{2A} & \left(|(a^2+1)r > |a x_{CP} - y_{CP}|\right) \\ \frac{1}{\omega} \log \frac{-B - \sqrt{D}}{2A} & (otherwise) \end{cases} \tag{26}$$

$$\acute{x}_f^{tmp} = \frac{(x_{CP} + a y_{CP}) e^{\omega \acute{T}_d^{tmp}} - ab}{a^2 + 1} \tag{27}$$

$$\acute{y}_f^{tmp} = \frac{a(x_{CP} + a y_{CP}) e^{\omega \acute{T}_d^{tmp}} + b}{a^2 + 1} \tag{28}$$

When Eq.(25) is not satisfied, the robot cannot recover its balance by only modifying the position and timing under a step condition. At such times, we consider them to be as follows, and proceed to the next procedure:

$$\acute{T}_d^{tmp} = T_d, \qquad \acute{x}_f^{tmp} = x_f, \qquad \acute{y}_f^{tmp} = y_f \tag{29}$$

In the procedure above, it is not necessarily guaranteed that the obtained step position will be inside the region because we considered the point of tangency not with the nearest neighbor edge but with the line. Hence, we move the position to be inside the region by limiting it (Fig.5 (d)).

$$\acute{x}_f = \begin{cases} \max\{x_R,\ x_L\} & (\acute{x}_f^{tmp} > \max\{x_R,\ x_L\}) \\ \min\{x_R,\ x_L\} & (\acute{x}_f^{tmp} < \min\{x_R,\ x_L\}) \end{cases} \tag{30}$$

The y component of the point is limited as well.

2) Modification Using RCR: We regard the supporting sole as a rectangle, which is similar to the shape of the actual sole. We call the Capture Region using the rectangle sole *Rectangular Capture Region (RCR)*, and the following derivation of the modification method using RCR is based on our previous method [19]. We then check whether the position $\acute{\boldsymbol{p}}_f$ that was calculated by the procedure above is realized according to the original step time T_d. For instance, if the robot is pushed forward, the step time needs to be modified based upon the following condition from Eq.(14):

$$\acute{x}_f < e^{\omega T_d}(x_{CP}(t) - x_Z^{max}) + x_Z^{max} \tag{31}$$

The modified time is derived as follows:

$$\acute{T}_d^{unlim} = \frac{1}{\omega}\ln\frac{s_x^{max} - x_Z^{max}}{x_{CP}(t) - x_Z^{max}} \tag{32}$$

The modified step time has a lower limit, and the limited time of $\acute{T}_d^{unlim}$ is the determined step time $\acute{T}_d$. In the experiments, the lower limit is heuristically set such that joint velocities and torques do not exceed the hardware specification. If the time is modified for both axes, the lower time is used [19].

When the step time is not restricted by a lower limit as shown in Fig.5 (e), the step position is no longer modified, and the determined step position and time are $\acute{\boldsymbol{p}}_f, \acute{T}_d$ respectively. On the other hand, when the step time is limited, the step position can change depending on $\acute{T}_d$. Therefore, we recheck whether $\acute{\boldsymbol{p}}_f$ is realized according to $\acute{T}_d$. If the robot is pushed forward, the desired step position to maintain balance x_f^{des} differs from $\acute{x}_f$ in the following condition as well in Eq.(31):

$$\acute{x}_f < e^{\omega \acute{T}_d}(x_{CP}(t) - x_Z^{max}) + x_Z^{max} \tag{33}$$

In such cases, x_f^{des} is derived as follows:

$$x_f^{des} = e^{\omega \acute{T}_d}(x_{CP}(t) - x_Z^{max}) + x_Z^{max} \tag{34}$$

$x_f^{des} - \acute{x}_f$ is the deficient length for $\acute{T}_d$, and we compensate for it using the torque around the COG τ_y (Fig.5 (f)).

$$\tau_y = \frac{Mg(x_f^{des} - \acute{x}_f)}{1 - e^{\omega \acute{T}_d}} \tag{35}$$

The torque is realized through whole-body inverse kinematics (IK) with constraints on angular momentum [19].

In this way, introducing CCR and RCR enables to fast calculate the step position and timing in both axes considering the steppable region. In addition, the robot can recover its balance through the use of centroidal angular momentum even if modifying just the step position and timing is insufficient for gaining dynamic balance. Therefore, modifying all the walking parameters increases the possibility that the robot will maintain its balance on complex footholds, even when the steppable region is severely restricted.

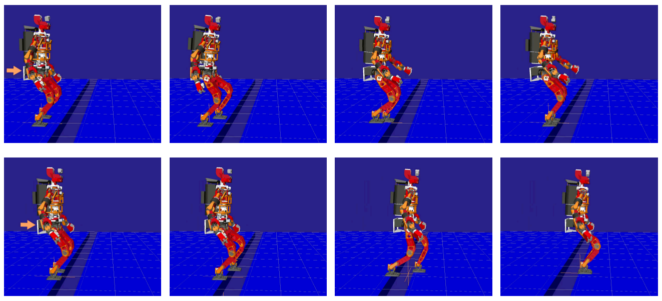


Fig. 7. Push recovery while walking on the spot in the presence of a ditch. For the first push, the robot took a $0.22\,\mathrm{m}$ step and generated the angular momentum to compensate for the deficient step length. For the second push, it took a $0.52\,\mathrm{m}$ step without the angular momentum because the step length was sufficient for dynamic balance.

B. Modification of Walking Parameters for a Set of Regions

In the previous subsection, we described the modification method, which consisted of walking parameters for a single steppable region $\boldsymbol{\mathcal{V}}^{sr}$. In this subsection, we present the method for a set of regions $\boldsymbol{\mathcal{C}}^{sr} = \{\boldsymbol{\mathcal{V}}_1^{sr}, \boldsymbol{\mathcal{V}}_2^{sr}, \ldots, \boldsymbol{\mathcal{V}}_m^{sr}\}$.

We sequentially apply the method described in V-A to the steppable regions $\boldsymbol{\mathcal{V}}_k^{sr}$ $(k = 1, 2, \ldots, m)$. If the region that can satisfy Eq.(18) is found, the process is finished, and the walking parameters do not need to be changed according to visual feedback. Letting the walking parameters according to the $k-1^{\text{th}}$ process be $\boldsymbol{\mathcal{W}} = \{\acute{\boldsymbol{p}}_f^v,\ \acute{T}_d^v,\ \boldsymbol{\tau}^v\}$, the parameters are updated when satisfying the following condition:

$$\acute{\boldsymbol{p}}_f^{ref} \cdot \acute{\boldsymbol{p}}_{f,k} > \acute{\boldsymbol{p}}_f^{ref} \cdot \acute{\boldsymbol{p}}_f^{v} \tag{36}$$

This way, the determined parameters will be as close to the desired walking direction as possible.

Likewise, it is possible to determine the landing position considering the foothold safety C_{sf} [15] as an extension of our proposed method:

$$\acute{\boldsymbol{p}}_f^{ref} \cdot \acute{\boldsymbol{p}}_{f,k} + w_{sf}C_{sf,k} > \acute{\boldsymbol{p}}_f^{ref} \cdot \acute{\boldsymbol{p}}_f^{v} + w_{sf}C_{sf}^{v} \tag{37}$$

where w_{sf} is the weight for the foothold safety.

C. Simulation Verification

We validated the proposed method by using Choreonoid [24]. Fig.7 shows the simulation result in which the robot was twice pushed forward in an environment with a ditch. Also, the video is attached to this paper. The landing position, timing, and torque around the COG are shown in Fig.8. The nominal step cycle was set to $0.7\,\mathrm{s}$, and the width of the ditch was $0.35\,\mathrm{m}$. An external force of $400\,\mathrm{N}$ was applied to the upper body for $0.1\,\mathrm{s}$. The actual CP (COG state) is calculated from the body posture estimated by IMU [1]. The foot is controlled in impedance mode.

For the first external force, the robot withstood the disturbance by generating the centroidal angular momentum instead of making a long-distance step as there was a ditch in front of the robot. The robot mainly swung its arms for generating the angular momentum because we set the weight for arms in IK to a small value. However, for the second force, the robot took a big step to clear the ditch without generating the angular momentum. Rather, the step time was

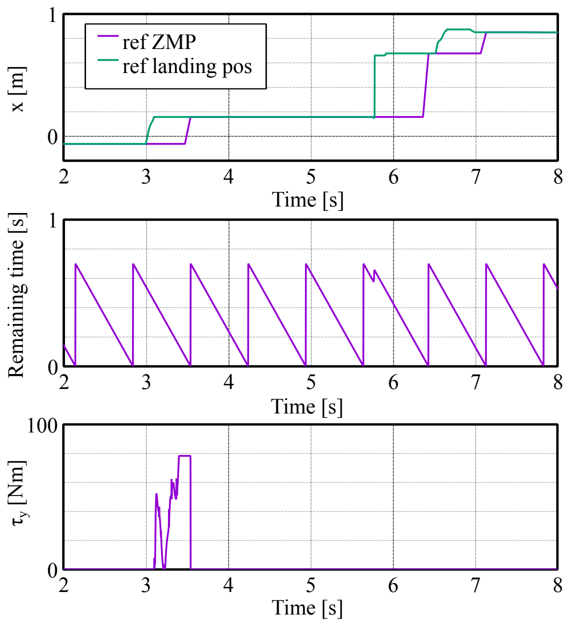


Fig. 8. Landing position, time, and torque around the COG during push recovery considering the steppable region.

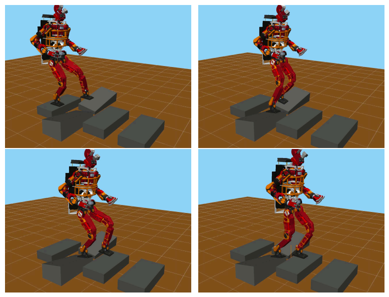
enlarged because the step length was more than adequate for maintaining balance. Thus, our proposed method enables modification of the online walking parameters considering the robot's current balance and the steppable region.

VI. Visual Adaptation to Height and Inclination of Footholds

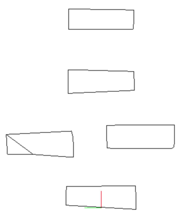
Because the landing position can be modified during each control cycle, it is required to instantaneously deal with the height and inclination of foothold corresponding to the landing position.

We accumulate terrain information using a heightmap. The horizontal landing position is determined by the method presented in V. We calculate the average height and inclination around the horizontal position from the heightmap, and the robot adapts to the road shape by updating the reference height and inclination of the landing foot accordingly.

Fig.9 (a) shows the simulation results, which combined the method in this section with the online footstep modification method detailed in the previous section. Five gray blocks formed the steppable regions, two of which were tilted by $\pm 10\,\mathrm{deg}$ around the roll and pitch axes. The blocks were modeled as static objects and friction coefficients were 0.5. In our implementation of the walking pattern generation, we assumed that the supporting soles contacted with the same horizontal plane and the COG height moved on the constraint plane. Fig.9 (b) shows the steppable regions recognized by using vision sensors. Note that the steppable regions are used for a horizontal footstep modification and not for a height/inclination adaptation. The robot was programmed in advance to walk forward on a flat floor with a step length of $0.15\,\mathrm{m}$, and the footsteps were automatically adjusted online based upon vision recognition. Thus, the robot was able to traverse across complex footholds, including those with an inclination, while considering the robot's real-time state.

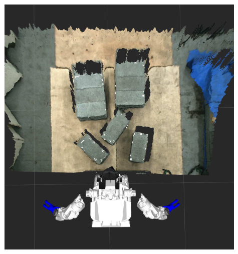


(a) Snapshot of simulation

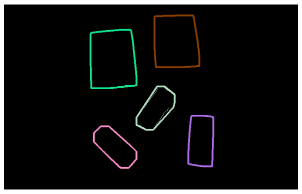


(b) 2D convex steppable region

Fig. 9. Visual adaptation to height and inclination of footholds. (a) : Gray objects can be stepped upon. (b) : Regions are generated through shrinking and convex partition of the recognized footholds.



(a) Drawing of footholds in point cloud



(b) Projection of 3D footholds in horizontal plane

Fig. 10. Identification of steppable objects as stepping stones.

VII. Experimental Results

A. Real Hardware

We verified our proposed method by using the life-sized humanoid robot JAXON [25]. The robot is $188\,\mathrm{cm}$ tall, weighs $127\,\mathrm{kg}$, and has 33 DOF. It is equipped with IMU at its waist. Additionally, it has a LiDAR and stereo camera at its head. The footstep modification is executed at $500\,\mathrm{Hz}$.

B. Vision-Aided Walking on Unsteady Stepping Stones

1) Experimental Set Up and Recognition of Steppable Region: We conducted a series of experiments in which the robot traversed on stepping stones consisting of two layers of concrete blocks. The blocks wobble when the robot steps on them as they are not fixed to the ground. Therefore, the robot needs to walk considering real-time balance.

In these experiments, we simply used color recognition to determine which objects could be stepped upon. To begin with, the robot visualized the 2D steppable objects by using an RGB camera, and then it mapped the vertices of these objects into point cloud by using data acquired from LiDAR. The obtained footholds are shown in Fig.10 (a). By projecting the points in 3D point cloud onto a horizontal plane, the regions could be obtained, as shown in Fig.10 (b). Lastly, by applying the method in IV-A to the horizontal regions, they were converted into the steppable regions as a set of convex hulls.

2) Experiment of Online Walking Parameters Generation on Stepping Stones: Fig.11 shows a snapshot of the experiment. Although the robot was nominally commanded to walk forward at a constant step length, the step position was changed online considering vision recognition and dynamic

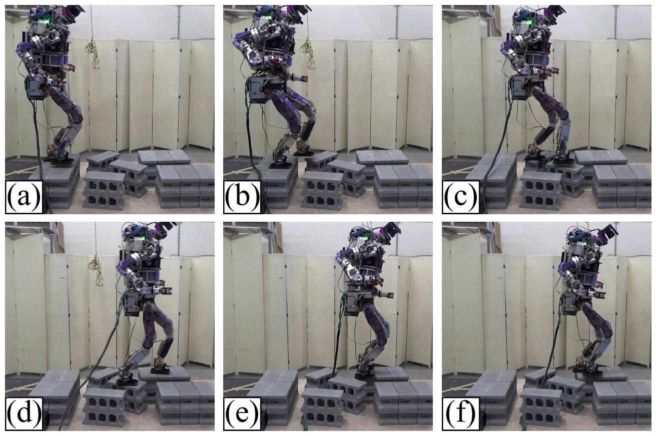


Fig. 11. Walking on unsteady stepping stones using vision recognition.

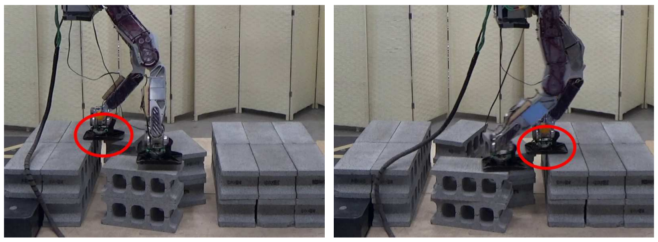


Fig. 12. Walking on closely placed stepping stones.

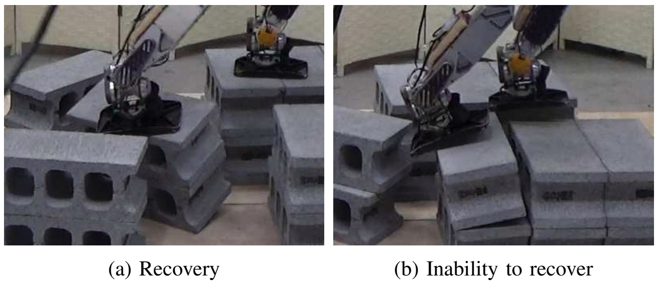


Fig. 13. Limitation of walking stabilization using only geometrical information.

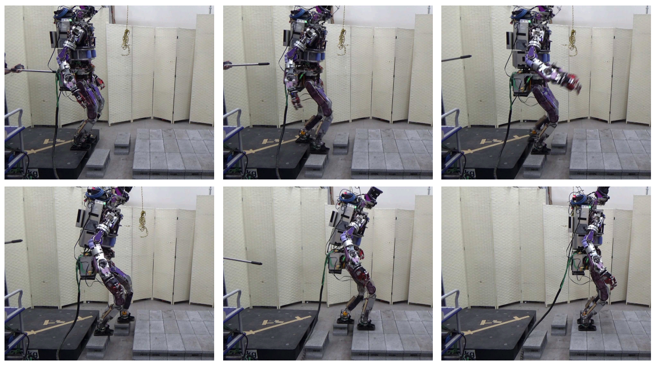


Fig. 14. Push recovery on stepping stones.

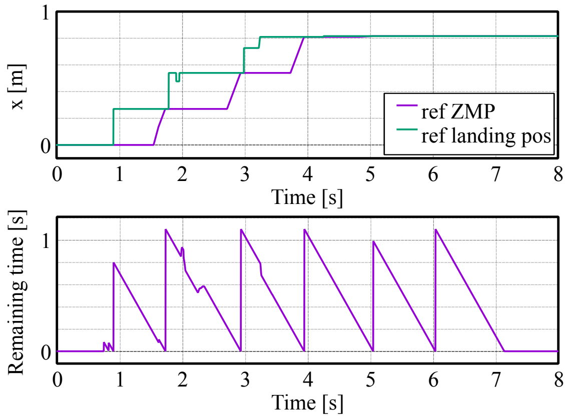


Fig. 15. Step position and timing on stepping stones during push recovery.

balance. In this way, the robot was able to walk on unsteady sparse footholds. In particular, even though the blocks under the supporting foot were tilted as shown in Fig.11 (d) and (e), the robot dealt with the influence by modifying footsteps.

Fig.12 shows the experiment in which the placement of the blocks was different than in Fig.11. By connecting the near footholds compared with the size of the foot sole by using the process such as closing when generating the steppable region, it is possible for the robot to land on two or more footholds at the same time.

Fig.13 shows a failure case, similar to the previous experiment as shown in Fig.11. In Fig.13 (a), although the block under the right foot was significantly tilted, the robot could still maintain its balance by modifying the next landing position backward. On the other hand, in Fig.13 (b), the left swinging foot accidentally touched the tilted block under the right foot. As a result, the robot fell along with the block, and thus it could not recover its balance. Therefore, not only geometrical information, but also the physical properties of the footholds are needed for the robot to walk in various real environments stably. For example, setting limits of the horizontal floor reaction forces may be effective for this type of situation.

C. Push Recovery on Stepping Stones

We carried out the push recovery experiment on stepping stones. We pushed the back of the robot with a bar while it was standing still. The robot started stepping based on the measured CP and support polygon, and kept walking until it could regain dynamic balance. Fig.15 shows the generated landing position and timing. The parameters were instantly adjusted according to the robot's current state. In addition, the robot changed its whole-body motion so that constraints on the steppable region and step time were not violated, as can be seen in Fig.14. Thus, our proposed method is effective even on sparse footholds during push recovery situations, where immediate footstep modification is required.

VIII. Conclusions

In this paper, we proposed a real-time footstep modification method considering footholds. We expanded upon our previous methods which allowed for the robot to instantaneously control the landing position, timing, and centroidal angular momentum, depending on the disturbance. The method in this paper allowed us to determine the parameters under constraints of the steppable region. The landing position to satisfy the steppability is analytically calculated by using a set of convex steppable regions and CCR. Following this, we calculated the corresponding step time for the determined landing position using RCR, which enabled the robot to adequately utilize ankle strategy. Additionally, the robot compensated for an insufficient landing position due to the limit for step time by using the centroidal angular momentum. We demonstrated that the robot was able to walk in the presence of 3D footholds by updating the height and inclination of the landing foot according to information acquired from a heightmap.

This method can instantly determine the walking parameters in an environment, even when there are many footholds, as the computational cost is very low. Moreover, this method means that the robot can maintain its balance on sparse footholds, as it can change all of its walking parameters, including step time and angular momentum. We realized push recovery by using a real humanoid robot on stepping stones, which has not been achieved before.

Our proposed method can potentially determine footsteps considering the physical properties of footholds, as this method explicitly deals with steppable polygons. Besides, the conventional locomotion planner for humanoids needs to strictly plan future footsteps whereas our planner generates only a rough path and steppable regions by using the proposed method, which makes the locomotion system more robust. We are going to report on the path planner and the integration of object transportation in another paper.

REFERENCES

[1] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi. Biped Walking Stabilization Based on Linear Inverted Pendulum Tracking. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4489–4496, 2010.

[2] J. Englsberger, C. Ott, and A. Albu-Schäffer. Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, Vol. 31, No. 2, pp. 355–368, 2015.

[3] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3411–3416, 2012.

[4] T. Kamioka, H. Kaneko, T. Takenaka, and T. Yoshiike. Simultaneous optimization of ZMP and footsteps based on the analytical solution of divergent component of motion. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, pp. 1763–1770, 2018.

[5] Z. Aftab, T. Robert, and P. Wieber. Ankle, hip and stepping strategies for humanoid balance recovery with a single Model Predictive Control scheme. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots*, pp. 159–164, 2012.

[6] H. Jeong, I. Lee, J. Oh, K. K. Lee, and J. Oh. A Robust Walking Controller Based on Online Optimization of Ankle, Hip, and Stepping Strategies. *IEEE Transactions on Robotics*, Vol. 35, No. 6, pp. 1367–1386, 2019.

[7] B. Stephens. Humanoid push recovery. In *Proceedings of the 2007 IEEE-RAS International Conference on Humanoid Robots*, pp. 589–595, 2007.

[8] R. Ueda, S. Nozawa, K. Okada, and M. Inaba. Biped humanoid navigation system supervised through interruptible user-interface with asynchronous vision and foot sensor monitoring. In *Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 273–278, 2014.

[9] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *Proceedings of the 2007 IEEE-RAS International Conference on Humanoid Robots*, pp. 196–202, 2007.

[10] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt. Footstep Planning for Autonomous Walking Over Rough Terrain. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots*, pp. 9–16, 2019.

[11] A. C. Hildebrandt, R. Wittmann, F. Sygulla, D. Wahrmann, D. Rixen, and T. Buschmann. Versatile and robust bipedal walking in unknown environments: real-time collision avoidance and disturbance rejection. *Autonomous Robots*, Vol. 43, No. 8, pp. 1957–1976, 2019.

[12] T. Yamamoto and T. Sugihara. Instantaneous Landing Position Determination for Biped Robots Taking Into Account Terrain, Kinematics and Capturability (in Japanese). In *2018 JSME Conference on Robotics and Mechatronics.* 2A2–H07, 2018.

[13] H. Jeong, J. H. Kim, O. Sim, and J. H. Oh. Avoiding Obstacles During Push Recovery Using Real-Time Vision Feedback. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 483–490, 2019.

[14] H. Kobayashi and T. Sugihara. Self-consistent automatic navigation of com and feet for realtime humanoid robot steering. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3525–3530, 2009.

[15] Y. Omori, Y. Kojio, T. Ishikawa, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba. Autonomous Safe Locomotion System for Bipedal Robot Applying Vision and Sole Reaction Force to Footstep Planning. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4891–4898, 2019.

[16] M. Brando, K. Hashimoto, and A. Takanishi. Friction from vision: A study of algorithmic and human performance with consequences for robot perception and teleoperation. In *Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots*, pp. 428–435, 2016.

[17] M. Shafiee-Ashtiani, A. Yousefi-Koma, and M. Shariat-Panahi. Robust bipedal locomotion control based on model predictive control and divergent component of motion. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation*, pp. 3505–3510, 2017.

[18] K. Guan, K. Yamamoto, and Y. Nakamura. Push Recovery by Angular Momentum Control during 3D Bipedal Walking based on Virtual-mass-ellipsoid Inverted Pendulum Model. In *Proceedings of the 2019 IEEE-RAS International Conference on Humanoid Robots*, pp. 120–125, 2019.

[19] Y. Kojio, Y. Ishiguro, K.-N.-K. Nguyen, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba. Unified Balance Control for Biped Robots Including Modification of Footsteps with Angular Momentum and Falling Detection Based on Capturability. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 497–504, 2019.

[20] T. Sugihara and T. Yamamoto. Foot-guided agile control of a biped robot through zmp manipulation. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4546–4551, 2017.

[21] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 239–246, 2001.

[22] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture Point: A Step toward Humanoid Push Recovery. In *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robots*, pp. 200–207, 2006.

[23] F. P. Preparata and M. I. Shamos. Computational geometry: an introduction. *Springer-Verlag*, 1985.

[24] S. Nakaoka. Choreonoid: Extensible virtual robot environment built on an integrated gui framework. In *Proceedings of the 2012 IEEE/SICE International Symposium*, pp. 79–85, 2012.

[25] K. Kojima, T. Karasawa, T. Kozuki, E. Kuroiwa, S. Yukizaki, S. Iwaishi, T. Ishikawa, R. Koyama, S. Noda, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Development of Life-Sized High-Power Humanoid Robot JAXON for Real-World Use. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, pp. 838–343, 2015.