ijrr

The International Journal of Robotics Research
31(10) 1117–1133

DOI: 10.1177/0278364912452762
ijr.sagepub.com
SAGE

Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid

Jerry Pratt[1], Twan Koolen[1,2], Tomas de Boer[2], John Rebula[3], Sebastien Cotton[1], John Carff[1], Matthew Johnson[1] and Peter Neuhaus[1]

Abstract

This two-part paper discusses the analysis and control of legged locomotion in terms of N-step capturability: the ability of a legged system to come to a stop without falling by taking N or fewer steps. We consider this ability to be crucial to legged locomotion and a useful, yet not overly restrictive criterion for stability. Part 1 introduced the N-step capturability framework and showed how to obtain capture regions and control sequences for simplified gait models. In Part 2, we describe an algorithm that uses these results as approximations to control a humanoid robot. The main contributions of this part are (1) step location adjustment using the 1*-step capture region, (2) novel instantaneous capture point control strategies, and 3) an experimental evaluation of the* 1*-step capturability margin. The presented algorithm was tested using M2V2, a 3D force-controlled bipedal robot with 12 actuated degrees of freedom in the legs, both in simulation and in physical experiments. The physical robot was able to recover from forward and sideways pushes of up to 21 Ns while balancing on one leg and stepping to regain balance. The simulated robot was able to recover from sideways pushes of up to 15 Ns while walking, and walked across randomly placed stepping stones.*

Keywords

Legged robots, push recovery, viability, capturability, capture points.

1. Introduction

Making humanoid robots useful in complex environments requires attaining good disturbance rejection properties while performing other tasks, such as walking. Current robots have not sufficiently demonstrated this ability. In Part 1 (Koolen et al. 2012), we proposed to approach this problem using the concept of *N*-step capturability, informally defined as the ability to come to a stop in *N* steps or fewer. We provided a capturability analysis of three simplified gait models. This part presents capturability-based control algorithms for two control tasks: (1) push recovery while balancing on one leg; and (2) walking. The presented algorithms are robust to pushes and unexpected ground variations. These control algorithms were implemented on M2V2, a force controlled 3D lower-body humanoid with two six-degree-of-freedom (6-DoF) legs.

The three main areas of contribution of this part are:

1. step location adjustment using an approximated 1-step capture region, based on the 3D Linear Inverted Pendulum Model (3D-LIPM) with finite-sized foot;
2. novel instantaneous capture point control strategies using placement of the center of pressure (CoP);
3. an experimental evaluation of the control algorithms using the *N-step capturability margin* introduced in Part 1.

Furthermore, the current part extends some of the control strategies from previous work (Pratt et al. 2001) to 3D.

The presented control algorithms allow the physical M2V2 to recover from forward and sideways pushes of up to 21 Ns. The simulated version of the robot was able to recover from sideways pushes of up to 15 Ns while walking on level ground. In simulation, we have also achieved push recovery while walking over stepping stones. These results demonstrate that a capturability analysis for a simplified model can be used to develop bipedal walking algorithms that are robust to disturbances.

The remainder of this paper is structured as follows. Section 2 presents related literature. In Section 3 we

[1]Institute for Human and Machine Cognition, Pensacola, FL, USA
[2]Delft University of Technology, the Netherlands
[3]University of Michigan, Ann Arbor, MI, USA

Corresponding author:
Jerry Pratt, Institute for Human and Machine Cognition, 40 South Alcaniz Street Pensacola, FL 32502, USA.
Email: jpratt@ihmc.us

describe the M2V2 robot. Section 4 presents implementation details of our control algorithms. Section 5 presents results. In Section 6, we discuss capturability-based analysis and control and suggest future work. Finally, we conclude the paper in Section 7.

2. Background

The literature on control algorithms for humanoid robots is extensive. Here we provide a brief survey of some widely used control techniques and focus on their disturbance rejection properties.

2.1. ZMP-based trajectory tracking control

The zero moment point (ZMP) is the point about which the resultant ground reaction torque has no horizontal component (Vukobratovic and Borovac 2004). ZMP-based trajectory tracking control usually encompasses choosing a desired ZMP trajectory based on available footholds and desired gait properties, and calculating the center of mass (CoM) motion that results in that ZMP trajectory (Kagami et al. 2002; Kajita et al. 2003). It is widely used in legged robot control since maintaining the ZMP strictly inside the support polygon at all times guarantees that it is physically possible to track the reference joint trajectories using conventional control tools. The distance from the ZMP to the edge of the support polygon can be used as a measure of robustness. Modifying the reference ZMP trajectory online has also been explored, including both small local ZMP changes and larger step placement changes (Nishiwaki and Kagami 2010). Another ZMP control approach treats the ZMP as a control input, which is manipulated to produce a desired motion of the CoM. For example, central pattern generators have been used to calculate the reference ZMP trajectory to produce walking (Sugihara 2010).

Typical ZMP-based gait generation techniques cannot be used to generate gaits for which the stance foot rolls from heel to toe, as observed in fast human walking. This limitation is due to the ZMP needing to be inside the support polygon to meet the ZMP stability criterion (Vukobratovic and Borovac 2004). However, when the robot rotates about an edge of its support foot, the area of the support polygon decreases to too small of a value to provide robustness against small modeling errors. Also, the reference joint trajectories themselves might lead the robot to a fall by design, even if the ZMP is kept inside the support polygon at all times. Hence, the ZMP criterion is not a necessary condition to avoid falling (Pratt and Tedrake 2006).

2.2. Passive dynamics-based control

Another approach to walking control explicitly relies upon the passive behavior of the robot's mechanical components. Straight-line periodic walking has been demonstrated for purely passive devices walking down a slope (McGeer 1990; Collins et al. 2001). Adding limited actuation to machines designed for passive walking can yield a controlled, efficient gait (Collins et al. 2005). Rejection of small disturbances has been shown for planar walkers under limited control (Kuo 1999; Tan et al. 2010), as well as locally stable gaits with purely reflexive control (Geng et al. 2006). A major focus of our current work is endowing robots with the ability to recover from disturbances large enough to require significant actuation, so we cannot rely on passive dynamics alone to avoid falling.

2.3. Hybrid zero dynamics

Another approach to locomotion control identifies relationships between the degrees of freedom of a robot that lead to a steady gait (Westervelt et al. 2003). These relationships are then enforced by a feedback controller, yielding a locally stable, periodic gait. This method has been shown to be able to cope with moderately rough terrain (Westervelt et al. 2004; Sreenath et al. 2012). More recently, this method has been used to generate three-dimensional walking (Chevallereau et al. 2009). However, it requires off-line computation of a repetitive gait, and therefore it currently has no mechanism for explicitly handling rough terrain with impassable areas. Also, it is still unclear how a robot using this method will handle large pushes that significantly disturb the state of the machine from the preplanned gait.

2.4. Compliant strategies for force controllable robots

Force controllable robots have lead to the development of compliant control strategies. Virtual model control was applied to a range of legged robots, including the 2D walking robot Spring Flamingo (Pratt et al. 1997; Hu et al. 1998, 1999; Chen et al. 2001; Pratt et al. 2001). Coros et al. (2010) combine Jacobian transpose control, joint space PD control, and gravity compensation with step planning based on an inverted pendulum model to obtain a walking controller that works for a range of simulated characters, while performing secondary tasks. Stephens and Atkeson (2010) introduced an algorithm that combines joint PD control, virtual model control and dynamic balance force control. Dynamic balance force control is an inverse dynamics approach based on the contact forces obtained from a CoM planner. Hyon and Cheng (2007) achieved disturbance rejection using a passivity-based controller, later complemented by CoM control using a dynamic balancer (Hyon et al. 2009).

Compliant strategies can enhance the robustness of a walking algorithm since they focus on interaction forces with the environment to achieve higher level goals, instead of relying on high gain position control and extremely accurate ground models to achieve perfect kinematic trajectories. The presented work also uses a compliant control strategy.

2.5. Instantaneous capture point-based control

In 2000, Pratt (2000b) achieved lateral stability for a simulated 3D robot using the capture angle, which is the azimuth of the capture point when written in polar coordinates. Capture point-based push recovery on a physical robot was first shown in Pratt et al. (2009). Walking controllers based on the instantaneous capture point have recently started to become more popular. Control algorithms for Honda's ASIMO are based on the *divergent component of motion* (Takenaka et al. 2009), which is the dynamics of the instantaneous capture point. Englsberger et al. (2011) introduced a walking controller based on capture point dynamics. Stephens (2011) used the instantaneous capture point in an optimization-based force control scheme.

In this paper we use the instantaneous capture point location in feedback control and for state transition conditions. We estimate the one-step capture region using the 3-D Linear Inverted Pendulum Model with finite-sized foot, as described and analyzed in Part 1. We use this capture region as a feasible foot placement area, thereby increasing the robustness of walking.

3. Description of M2V2 robot

M2V2 (see Figure 1) is a lower-body humanoid with two 6-DoF legs (Pratt and Krupp 2008). There are three DoFs at each hip, one at each knee, and two at each ankle. See Appendix A for approximate inertia parameters and joint offsets, which we use in the simulation model. M2V2 is a second-generation redesign of M2, a robot developed at the MIT Leg Laboratory (Pratt 2000a; Paluska 2000).

Each DoF is driven by an identical force controllable Series Elastic Actuator (SEA) (Pratt and Williamson 1995; Pratt and Pratt 2002). These actuators use a spring in series between the drive train and the load. By measuring the spring deflection, the force exerted by the actuator can be measured. Using a feedback controller, the actuator force can be controlled accurately. For M2V2, each actuator can produce a force of up to 1.3 kN, with a smallest resolvable force of approximately 4.4 N, giving it a 300 : 1 dynamic range. The low-force control bandwidth of each actuator is approximately 40 Hz. M2V2 has two U.S. Digital EM1-0-500 linear encoders and LIN-500 encoder strips at each Series Elastic Actuator, one to measure position and one to measure spring deflection.

Onboard computation is provided by a PC104 with a dual core Pentium-M processor. Sensor reading is done by several AccesIO 104-Quad8 encoder input boards. Desired current is output as a PWM signal through two Real Time Devices 6816 PWM boards. Body orientation and angular rate are measured using a MicroStrain 3DM-GX3-25 inertial measurement unit. Current control is provided through 12 Copley Controls Accelnet module ACM-180-20 amplifiers. The PC104, I/O cards, and current amplifiers are all located in the body of the robot.

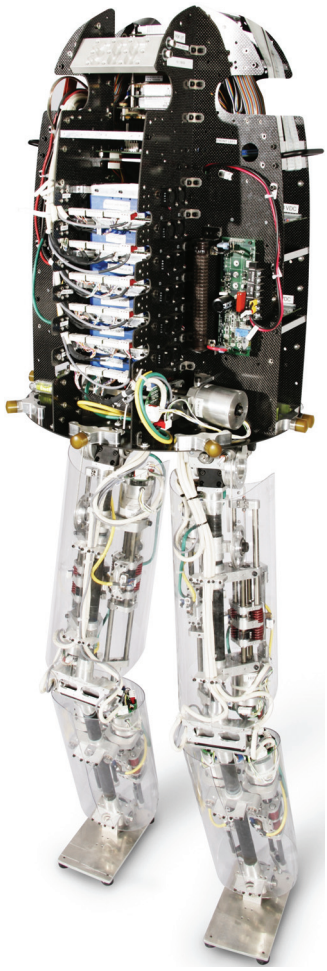

Fig. 1. M2V2, a 3D force-controllable humanoid robot with 12 actuated degrees of freedom.

A custom designed push stick equipped with a digital Loadstar ILoad Pro load cell was constructed for measuring pushing forces applied to the robot. This load cell is connected to the robot to eliminate data synchronization issues, but the control algorithm does not have access to its output.

4. Controller implementation

We now present a detailed description of the balancing and walking controller. The controller's input is composed of joint angles and angular velocities, and the orientation and orientation rates of the upper body. The controller's output is composed of desired torques at each joint.

Out of the three simplified gait models described in Part 1, we have chosen to base our controller on the 3D-LIPM with finite-sized foot, which does not include a reaction mass. See Section 6.1 for the motivation for this choice. In addition, the controller will always strive to keep the robot 1-step capturable. Section 6.2 provides a discussion on why we chose to base the controller on 1-step capturability, as opposed to using N-step capturability with $N > 1$.

The controller consists of five main parts:

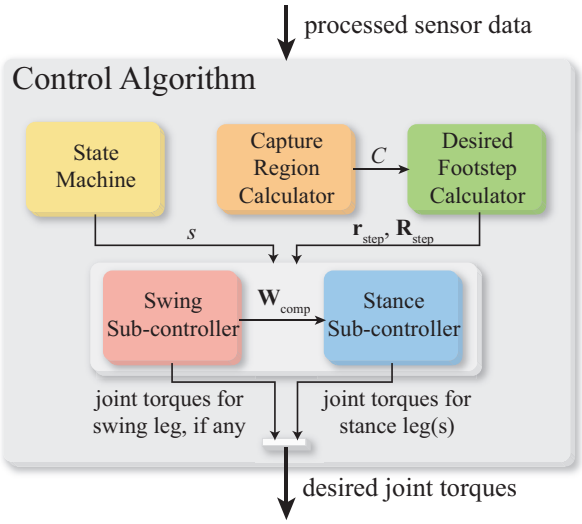

Fig. 2. Overview of the control architecture. Arrows represent data flow. Arrows that point to the edge of a block signify that the corresponding information is available to all sub-blocks. The state machine produces the controller state s. The capture region calculator approximates the 1-step capture region C. The desired footstep calculator uses the capture region to compute a step location $\mathbf{r}_{\text{step}}$ and orientation $\mathbf{R}_{\text{step}}$. The swing sub-controller computes torques for the swing leg if the robot is in a single support state. It also produces a wrench $\mathbf{W}_{\text{comp}}$ that the stance sub-controller uses to compensate the swing motion. The stance sub-controller computes the torques for both legs in double support, or just the stance leg in single support.

1. the state machine, which keeps track of the gait phase for each leg and acts as a supervisory system that calls the appropriate lower-level routines (see Section 4.1);
2. the capture region calculator, which determines the instantaneous capture point and the 1-step capture region (see Section 4.2);
3. the desired footstep calculator, which determines where to step to next (see Section 4.3);
4. the swing sub-controller, which computes the torques for the swing leg joints, if any (see Section 4.4); and
5. the stance sub-controller, which computes the torques for the stance leg joints (see Section 4.5).

See Figure 2 for an overview of the control architecture. The same state machine structure is used for both control tasks (balancing and walking), but with different control actions and transition conditions for each task. While the swing and stance sub-controllers are different for each task, they share many of the underlying control modules. The desired footstep calculator is also task specific.

The capture region calculator is task independent and is the module most linked with capturability-based analysis and control. This module may also be useful in other legged robot control architectures and with other walking control techniques.

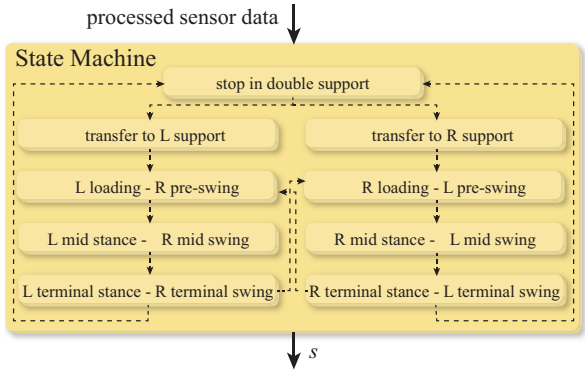

Fig. 3. The state machine, which produces the controller state s. Blocks represent states and dashed lines represent available state transitions. When the control algorithm is started, the robot is in the 'stop in double support' state.

4.1. State machine

The state machine structure, shown in Figure 3, is based on the gait phases that a single leg goes through during human walking, as described in the biomechanics literature (Perry 1992). See Figure 4 for a graphical depiction of these gait phases. The gait phases can be grouped into stance phases and swing phases. Below we provide a short description of each gait phase.

4.1.1. Stance phases The loading phase is the only double support phase, in which the robot uses both legs to control the instantaneous capture point towards a desired location. Once the instantaneous capture point is moved into the forward half of the front foot polygon, the opposite leg is lifted and the leg transitions into mid stance. This is the first single support phase, in which the instantaneous capture point is guided toward the expected location of the next support foot. The robot transfers into terminal stance when the swing leg is nearly finished swinging. During this phase, the CoP is constrained to lie on the front edge of the foot, so that the heel may rise.

4.1.2. Swing phases The first swing phase is pre-swing. In this phase, a constant hip torque is applied so that the knee is naturally unlocked and swing is initiated, while all other swing leg torques are set to zero. Mid swing begins after a short, fixed duration. In this phase, trajectory tracking is used to achieve ground clearance and foot placement. Finally, when the swing foot trajectory is completed, the robot transitions into terminal swing, in which position control gains for the ankle are reduced to prepare for impact.

Transitions between gait phases for the right leg are directly coupled to those for the left leg. The states shown in Figure 3 were hence created by combining one gait phase for the left leg and one gait phase for the right leg, e.g. 'left mid stance–right mid swing'. In addition to the six walking

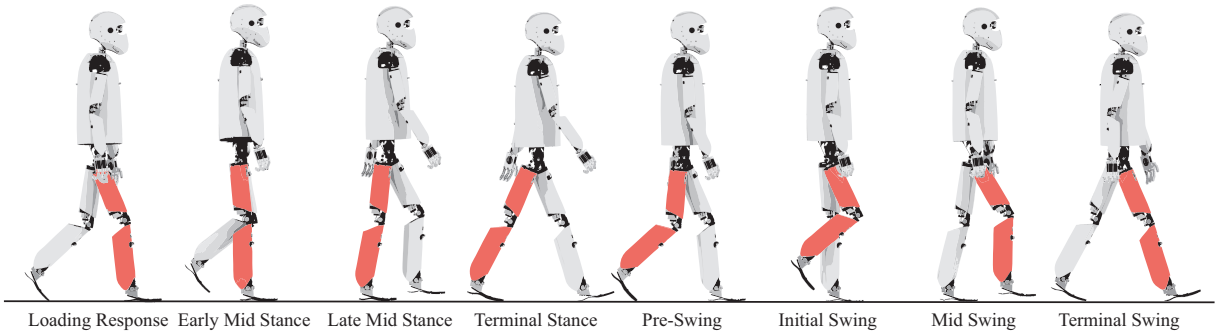

Fig. 4. The gait phases upon which the state machine is based. The labels correspond to the phase of the highlighted right leg. Adapted from Perry (1992), which describes the phases of gait for human walking.

states, there is also a state in which the robot is stopped in double support and states in which weight is transferred to one leg, allowing the robot to start from a stop.

While this state machine is based on walking, it is easily adapted to one-legged balance, through appropriate selection of control actions in each state and transition conditions between states.

4.2. Capture region calculator

The capture region calculator approximates the 1-step capture region C, which is the set of all reachable contact reference point placements that allow the robot to come to a captured state. The contact reference point is an arbitrarily chosen point on the swing foot. The algorithm is based on the capturability analysis of the 3D-LIPM with finite-sized foot, as presented in Part 1, and we refer to that part for the underlying theory; here we only present the implementation details. See Algorithm 1 for pseudocode. Figure 5 describes the algorithm graphically.

The capture region calculator is able to handle limitations on where the CoP may be placed, represented as a set of polygons called the allowable CoP regions and denoted $A = \{A_0, A_1, \dots\}$. These allowable CoP regions may be used to model stepping stones, as well as to avoid cross-over steps and overlapping footsteps.

The output of the algorithm is a tuple $(C_P, r_{\max}, l_{\max})$, consisting of a set of polygons C_P, a maximum offset distance to these polygons $r_{\max}$, and a maximum distance to the contact reference point $l_{\max}$. This tuple completely describes the capture region C, as shown in Figure 6.

The basic idea is to determine a set of polygons, $C_P = \{C_0, C_1, \dots\}$, with $C_i \subseteq A_i$, containing all allowable CoP locations where the instantaneous capture point could be located at a time when it is possible to take a step. The robot can reach a captured state by directing the instantaneous capture point to any such point, and then placing the CoP there after taking a suitable step. Given C_P, the capture region C can be found by determining allowable contact reference point locations that enable the CoP to be placed inside C_P. Such contact reference point locations are easy to find if it is assumed that the orientation of the swing foot may be chosen freely. In that case, we can define the maximum distance $r_{\max}$ from the contact reference point to the edge of the foot polygon (see Figure 6), and state that for any swing foot contact reference point location that has a distance of at most $r_{\max}$ to a given point $\mathbf{p} \in \cup_i C_i$, there exists at least one foot orientation that will allow the CoP to be placed at $\mathbf{p}$. The capture region is finally bounded by the maximum step length $l_{\max}$, defined as the maximum distance between subsequent contact reference point locations.

Algorithm 1 1-step capture region calculator

Input: S, $\mathbf{r}$, $\dot{\mathbf{r}}$, A, $t_{\text{s,min}}$, t, $r_{\max}$, $l_{\max}$

1: $\mathbf{r}_{\text{ic}} \Leftarrow \mathbf{Pr} + \sqrt{\frac{z_0}{g}}\dot{\mathbf{r}}$
2: $C_P \Leftarrow A$
3: **if** ¬pointInPolygon($S, \mathbf{r}_{\text{ic}}$) **then**
4: $\quad Q = (\mathbf{q}_0, \dots, \mathbf{q}_n) \Leftarrow$ visibleVertices($S, \mathbf{r}_{\text{ic}}$)
5: $\quad$ **for** $i = 0$ to n **do**
6: $\quad\quad \mathbf{p}_i \Leftarrow [\mathbf{r}_{\text{ic}} - \mathbf{q}_i]e^{t_{\text{s,min}} - t} + \mathbf{q}_i$
7: $\quad$ **end for**
8: $\quad C_P \Leftarrow$ bound($C_P, \mathbf{p}_0, \mathbf{r}_{\text{ic}}$)
9: $\quad C_P \Leftarrow$ bound($C_P, \mathbf{r}_{\text{ic}}, \mathbf{p}_n$)
10: $\quad$ **for** $i = 1$ to n **do**
11: $\quad\quad C_P \Leftarrow$ bound($C_P, \mathbf{p}_{i-1}, \mathbf{p}_i$)
12: $\quad$ **end for**
13: **end if**
14: **return** $(C_P, r_{\max}, l_{\max})$

In Algorithm 1, S is the current support polygon represented as a clockwise ordered sequence of vertices, $\mathbf{r}$ is the CoM position, $t_{\text{s,min}}$ is the estimated earliest time at which a step can be taken, t is the current time, $r_{\max}$ is the maximum distance from the contact reference point to the edge of the support polygon, and $l_{\max}$ is the maximum step length, as measured from the contact reference point.

Line 1 in the algorithm is the instantaneous capture point definition, Equation (8) in Part 1. Line 6 describes the instantaneous capture point motion for a constant CoP and originates from Equation (22) in Part 1.

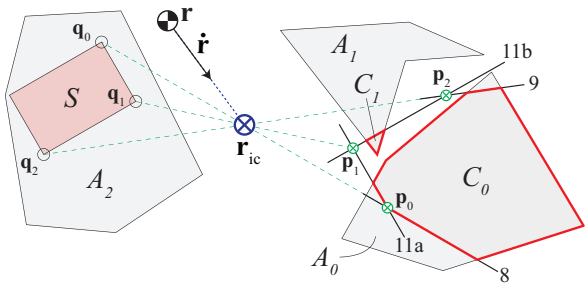

Fig. 5. Graphical depiction of Algorithm 1. The instantaneous capture point $\mathbf{r}_{ic}$ is first calculated (line 1 in Algorithm 1). Next, the ordered set of support polygon vertices that are visible from the instantaneous capture point, $(\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2)$ is determined (line 4), as well as the corresponding possible locations of the instantaneous capture point at $t_{s,min}$, $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ (lines 5–7). The set of polygons C_P is finally determined by bounding the set of allowable CoP regions, A, using the lines denoted 8, 9, 11a and 11b in the figure. The numbers in the figure correspond to line numbers in Algorithm 1.

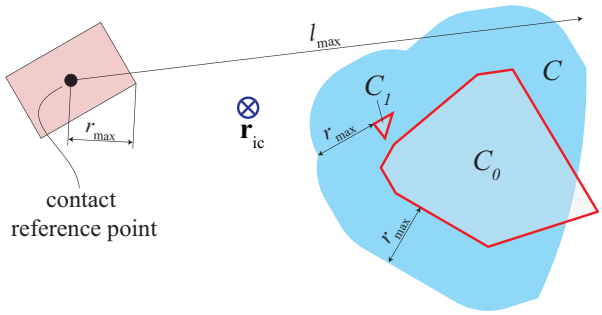

Fig. 6. The 1-step capture region C, as found from (C_P, r_{max}, l_{max}) by (1) offsetting the set of polygons C_P by r_{max}, and (2) intersecting the result with a disk that has radius l_{max} and has the contact reference point as its center, to take the maximum step length constraint into account. Note that for this figure, the centroid of the foot was chosen as the contact reference point.

The function pointInPolygon$(X, \mathbf{p})$ determines whether point $\mathbf{p}$ is contained within the polygon defined by the sequence of vertices X.

Function visibleVertices$(X, \mathbf{p})$ computes a clockwise ordered sequence of vertices $Q = (\mathbf{q}_0, \ldots, \mathbf{q}_n)$ taken from X that 'can be seen' from $\mathbf{p}$, in the sense that a line segment starting from $\mathbf{p}$ and pointing to $\mathbf{q}_i$ does not intersect the polygon spanned by X. The first element of the returned sequence is the rightmost visible vertex as seen from $\mathbf{p}$, and the last element is the leftmost.

The function bound$(P, \mathbf{r}_0, \mathbf{r}_1)$ takes a set of polygons P and returns a modified version of P, where each individual polygon is intersected with a half-plane containing all points to the right of a boundary line through $\mathbf{r}_0$ and $\mathbf{r}_1$. Note that the order of the arguments $\mathbf{r}_0$ and $\mathbf{r}_1$ determines the 'direction' of the boundary line and hence the direction of the outward normal of the half-plane. This function is used to cut off parts of the allowable CoP regions to which the instantaneous capture point may not be directed or which are not reachable given the minimum step time.

Note that if the current support foot is only partially placed inside the allowable CoP regions A, the support polygon S is the intersection of the foot polygon and A, and Algorithm 1 can be used without modification. Furthermore, note that if there are limits on allowable foot orientations, the step of offsetting the set of polygons C_P by r_{max} should be replaced by convolving the allowable configurations of the swing foot polygon about C_P.

4.3. Desired footstep calculator

The desired footstep calculator (see Figure 7) determines the desired position of the swing ankle $\mathbf{r}_{step}$ and orientation of the swing foot $\mathbf{R}_{step}$ at the end of the upcoming step. The desired position and orientation are expressed in a frame fixed to the support foot. Because footstep planning depends greatly on the control task, we have created two separate implementations of this module: one for the balancing task and another for the walking task.

The general pattern used for both implementations is to choose a desired initial footstep $(\mathbf{r}_{step,init}, \mathbf{R}_{step,init})$ at the start of the swing phase and adjust it during the swing phase if necessary, for example if the robot is significantly perturbed, to obtain the final output $(\mathbf{r}_{step}, \mathbf{R}_{step})$. If, during the swing phase, $\mathbf{r}_{step,init}$ ceases to be within the capture region or comes close to its edge, the adjusted desired step location $\mathbf{r}_{step}$ is computed by projecting $\mathbf{r}_{step,init}$ inside the 1-step capture region C by a specified margin. See Figure 20 for an example of step location adjustment. The orientation is not adjusted.

4.3.1. Balancing The desired footstep calculator for the balancing task is a minimal implementation. The initial desired step location is computed using a fixed step length l_{step} in the direction of the instantaneous capture point:

$$\mathbf{r}_{step,init} = \mathbf{r}_{ankle} + l_{step} \frac{\mathbf{r}_{ic} - \mathbf{r}_{ankle}}{\|\mathbf{r}_{ic} - \mathbf{r}_{ankle}\|}, \qquad (1)$$

where $\mathbf{r}_{ankle}$ is the position of the stance foot ankle. The initial foot orientation is chosen to be the same as the stance foot orientation and is never adjusted. The direction in which the robot steps is adjusted by recomputing (1) during the first 0.1 s after the instantaneous capture point has left the foot polygon. After that it remains fixed for the remainder of swing.

4.3.2. Walking The desired footstep for the walking task is chosen in such a way that forward progression is made, while the robot remains 1-step capturable. The initial step length is determined by multiplying the desired walking velocity by a proportional gain, while the initial step width is set to a constant value. On flat ground, the step height is set to zero but may be changed to any feasible desired value

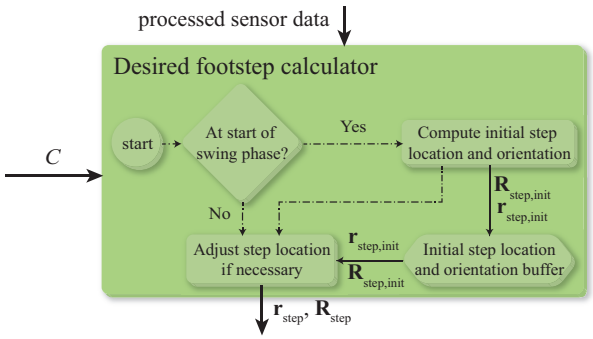

Fig. 7. Basic structure of the desired footstep calculator. The desired footstep calculator determines the desired position $\mathbf{r}_{\text{step}}$ and orientation $\mathbf{R}_{\text{step}}$ of the foot for the upcoming step. Solid lines represent data flow and dash-dotted lines represent flow of control.

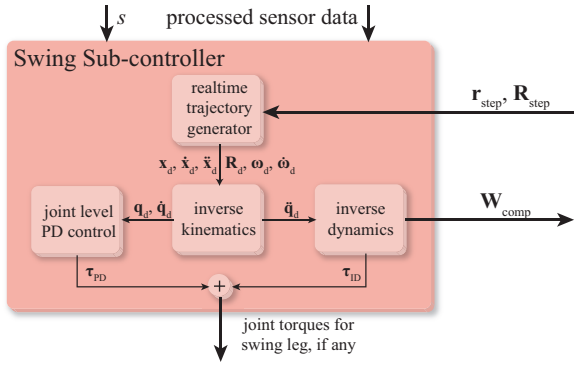

Fig. 8. Swing sub-controller. The output of a realtime trajectory generator is used to compute desired joint positions, velocities and accelerations ($\mathbf{q}_{\text{d}}$, $\dot{\mathbf{q}}_{\text{d}}$ and $\ddot{\mathbf{q}}_{\text{d}}$). This information is used by a PD + inverse dynamics algorithm, which computes joint torques $\boldsymbol{\tau}_{\text{id}}$ and an upper-body compensation wrench $\mathbf{W}_{\text{comp}}$, used by the stance sub-controller. The output $\boldsymbol{\tau}_{\text{PD}}$ of a joint-space PD controller is added to $\boldsymbol{\tau}_{\text{ID}}$ to obtain the swing leg joint torques.

when necessary, for instance to climb a slope. Footstep yaw is set equal to the desired walking direction. Footstep pitch and roll are set to zero on flat ground but may be used to walk on rough terrain to reduce the need for compliance in the ankle joint.

4.4. Swing sub-controller

The task of the swing sub-controller (see Figure 8) is to compute the desired torques for the swing leg joints. It accommodates changes in the desired step location, obtained from the desired step location calculator, on the fly.

The swing sub-controller contains a realtime Cartesian space trajectory generator, which outputs the desired position $\mathbf{x}_{\text{d}}$, velocity $\dot{\mathbf{x}}_{\text{d}}$ and acceleration $\ddot{\mathbf{x}}_{\text{d}}$ of the swing foot, as well as the desired orientation, angular velocity and angular acceleration ($\mathbf{R}_{\text{d}}$, $\boldsymbol{\omega}_{\text{d}}$, $\dot{\boldsymbol{\omega}}_{\text{d}}$). We chose a bang–zero–bang scheme in the horizontal plane, consisting of a maximum acceleration phase, a cruise phase, and a maximum deceleration phase, to smoothly move the reference position of the swing foot from the initial actual position towards the time-variant desired step location $\mathbf{r}_{\text{step}}$. The desired height trajectory is determined by a specified take-off slope, cruise height, and landing slope. For the orientation trajectory of the swing foot, we use a quintic spline interpolation between the measured orientation at the start of the swing phase and the desired final foot orientation, $\mathbf{R}_{\text{step}}$.

Based on the Cartesian trajectory, desired joint positions, velocities and accelerations are computed using inverse kinematics. Problems due to the singularity that occurs when the knee is stretched are circumvented using a damped least-squares scheme (Buss 2009). An inverse dynamics algorithm (Featherstone 2008), augmented by PD position control in joint space is used to compute the desired torques for the swing leg joints. We exclude the stance leg joints in computing the inverse dynamics. The desired spatial acceleration of the upper body is set to zero. In addition to the torques across the swing leg pin joints, the inverse dynamics algorithm also returns a wrench $\mathbf{W}_{\text{comp}}$ that should be exerted across the 'floating joint' that connects the upper body to the world to achieve the desired zero spatial acceleration. This wrench will be used in the stance sub-controller as a feed-forward term to compensate the swing leg torques and reduce upper-body oscillations.

4.5. Stance sub-controller

The stance sub-controller controls balance by computing desired torques for the stance leg(s). The goals of the stance sub-controller are to control (1) instantaneous capture point location, (2) upper-body orientation, and (3) upper-body height. The stance sub-controller is an implementation of virtual model control. For each stance leg, it computes a desired wrench on the upper body, to be exerted by the leg, that satisfies these control goals. Jacobian transpose control is then used to find corresponding desired leg torques. The stance sub-controller consists of multiple control modules, as shown in Figure 9. The following sections will describe these modules in more detail.

4.5.1. Upper-body height control module The upper-body height control module regulates the upper-body height by determining the net vertical force, f_z, that the legs should exert on the upper body. In double support, f_z is set to a constant value that slightly overcompensates the estimated gravitational force acting on the entire robot, simulating a virtual constant force spring. In single support, f_z is set to the weight of the stance leg and the body plus the z-component of the force from the swing leg compensation wrench $\mathbf{W}_{\text{comp}}$, cancelling out some of the dynamic effects due to the swing leg motion.

Note that with virtual model control, the system will dynamically act as if the virtual forces are being applied

through virtual actuators. At kinematic singularities, components of the forces applied by the virtual actuators would be taken up by the robot structure. For example, when the knees are straight, the mechanical advantage is infinite, and the component of virtual forces along the leg have no effect on the knee torque. Therefore, while the desired vertical force f_z achieves the same dynamic effect that it would had the virtual actuator really existed, it may not be fully applied between the body and the ground. However, there are no numerical problems at kinematic singularities, since virtual model control does not require the inversion of a Jacobian that spans the leg joints.

4.5.2. Upper-body orientation control module The upper-body orientation control module determines a desired torque $\boldsymbol{\tau}$ that the legs should exert on the upper body. It is used to control the orientation of the upper body with respect to the world (as perceived by the inertial measurement unit). The desired pitch of the upper body is constant and set to zero. Both the desired yaw and the desired roll depend on the gait phase and state of the support leg. Yawing and rolling are used to obtain a longer reach for the swing leg and to make the gait look more human-like. In addition, the desired yaw also depends on the desired walking direction.

The desired upper-body torque is computed using PD control on the roll, pitch and yaw corresponding to the rotation matrix that describes the orientation of the actual upper body with respect to the desired orientation. In single support, the torque part of $\mathbf{W}_{\text{comp}}$ is added to the result to compensate the swing leg motion.

A lunging strategy based on the results of the 3D-LIPM with finite-sized foot and reaction mass has not yet been implemented and is part of future work.

4.5.3. Instantaneous capture point control module The goal of the instantaneous capture point control module is to regulate the location of the instantaneous capture point. Its output is the desired location of the CoP, $\mathbf{r}_{\text{CoP,des}}$. This control module switches between two modes of operation, depending on whether the instantaneous capture point is inside the support polygon.

Instantaneous capture point inside support polygon See Figure 10. When the instantaneous capture point is inside the support polygon, a *desired instantaneous capture point*, $\mathbf{r}_{\text{ic,des}}$, is determined. Given the current and desired location of the instantaneous capture point, the following simple control law is used to obtain the tentative location of the desired CoP, $\bar{\mathbf{r}}_{\text{CoP,des}}$:

$$\bar{\mathbf{r}}_{\text{CoP,des}} = \mathbf{r}_{\text{ic}} + k_{\text{ic}}(\mathbf{r}_{\text{ic}} - \mathbf{r}_{\text{ic,des}}), \tag{2}$$

where k_{ic} is the proportional gain. This proportional control law is motivated by the linear instantaneous capture point dynamics for the 3D-LIPM with finite-sized foot described

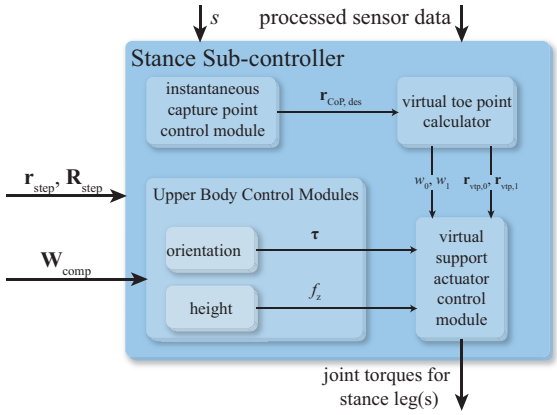

Fig. 9. Stance sub-controller. The instantaneous capture point control module computes the desired CoP ($\mathbf{r}_{\text{CoP,des}}$) within the base of support. This desired CoP is used by the virtual toe point calculator to compute leg support fractions (w_0, w_1) and virtual toe points ($\mathbf{r}_{\text{vtp,0}}$, $\mathbf{r}_{\text{vtp,1}}$), which determine each leg's contribution to supporting the upper body. The orientation and height upper-body control modules determine the torque $\boldsymbol{\tau}$ and the vertical force f_z to be exerted on the upper body. In single support, these modules use the swing leg compensation wrench $\mathbf{W}_{\text{comp}}$ to compensate swing leg motion. Finally, the virtual support actuator control module computes joint torques for the stance legs which result in the desired virtual toe points, leg strengths, upper-body torque and force.

in Part 1. If $\bar{\mathbf{r}}_{\text{CoP,des}}$ lies inside the support polygon, then the final output of this control module is $\mathbf{r}_{\text{CoP,des}} = \bar{\mathbf{r}}_{\text{CoP,des}}$. Otherwise, $\mathbf{r}_{\text{CoP,des}}$ is obtained by projecting $\bar{\mathbf{r}}_{\text{CoP,des}}$ to the edge of the support polygon along a line through $\mathbf{r}_{\text{ic}}$ and $\mathbf{r}_{\text{ic,des}}$, as shown in Figure 10b.[1] The idea behind this control law is that the instantaneous capture point is always pushed away from the CoP, and hence towards the desired instantaneous capture point.

The desired instantaneous capture point is determined as a function of the state and the control task. For the balancing task, the desired instantaneous capture point coincides with the centroid of the support polygon during the double support state. When the robot is commanded to start balancing on one leg, the desired instantaneous capture point is moved to the centroid of the upcoming support foot, where it remains as long as the robot is able to maintain its balance without taking a step. This location maximizes robustness against external disturbances from unknown directions.

For the walking task, the desired instantaneous capture point is located near the toes of the leading foot during the double support states, promoting forward motion. At the start of the swing phase, the desired capture point is moved outside the stance foot in the direction of the upcoming step location.

Instantaneous capture point outside support polygon See Figure 11. The instantaneous capture point will move outside the support polygon if the robot is pushed significantly, or because it has been driven outside the stance foot

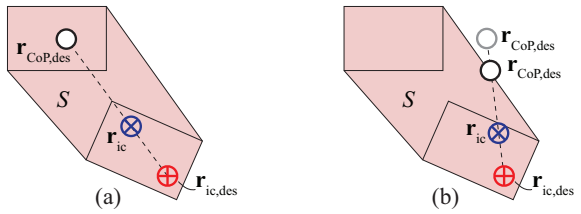

Fig. 10. Action of the instantaneous capture point control module when $\mathbf{r}_{ic}$ is inside the support polygon S. (a) The tentative desired CoP $\bar{\mathbf{r}}_{CoP,des}$, as determined using Equation (2), is inside the support polygon, so it coincides with the final output $\mathbf{r}_{CoP,des}$. (b) $\bar{\mathbf{r}}_{CoP,des}$ is outside the support polygon and is projected onto its edge to obtain $\mathbf{r}_{CoP,des}$.

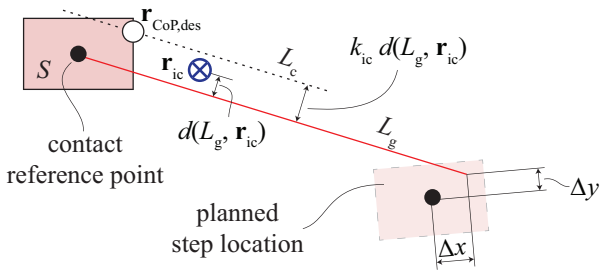

Fig. 11. Action of the instantaneous capture point control module when $\mathbf{r}_{ic}$ is outside the support polygon S. The guide line L_g passes through the contact reference point on the support foot and a point Δx in front and Δy inside of the contact reference point for the planned footstep location, see Equation (3). The control line L_c is parallel to the guide line; the distance between the two is determined by Equation (4). The desired CoP is chosen as the intersection of L_c and S.

polygon during the walking task. When $\mathbf{r}_{ic}$ is outside the support polygon, it is not possible to track a desired instantaneous capture point location, since the instantaneous capture point will always exponentially diverge away from the stance foot. We therefore only control the direction in which it diverges away from the foot. This is done by specifying a guide line, L_g, and choosing the desired CoP such that the instantaneous capture point is kept along this guide line.

We define the guide line by two points. The first is the contact reference point on the support foot. The second is a point Δx in front and Δy to the inside of the contact reference point for the planned footstep, with

$$\Delta x = k_{xx} v_{des,x}, \tag{3a}$$

$$\Delta y = k_{xy} |v_{des,x}|, \tag{3b}$$

where k_{xx} and k_{xy} are positive gains and $v_{des,x}$ is the desired average velocity of the robot in the forward direction, in a frame oriented to match the planned step location. The effect of this simple control law is that the instantaneous capture point is pushed forward and to the inside of the upcoming support foot as desired forward velocity is increased.

Given the guide line and the instantaneous capture point location, a second line, called the control line, is defined. The control line is parallel to the guide line. The distance $d(L_g, L_c)$ between the guide line L_g and the control line L_c is set to be proportional to the distance between the guide line and the instantaneous capture point, $d(L_g, \mathbf{r}_{ic})$:

$$d(L_g, L_c) = k_{ic} d(L_g, \mathbf{r}_{ic}), \tag{4}$$

where k_{ic} is a positive gain.

Finally, the desired CoP is computed by finding the intersection of the stance foot polygon and the control line that is closest to $\mathbf{r}_{ic}$. If there are no intersections, the support polygon vertex closest to the control line is used. The control law (4) causes the instantaneous capture point to be pushed back onto the guide line after a deviation.

4.5.4. Virtual toe point calculator The virtual toe point (VTP) calculator uses the desired CoP to compute a VTP and a leg support fraction for each leg (Pratt and Pratt 2002). Controlling VTP locations and leg support fractions results in approximate control of the overall CoP of the robot.

The VTP of a foot is the point about which no torque is commanded in the horizontal plane. VTPs are similar to the CoPs for each foot, except that a VTP is a commanded quantity, not a measured one, and is only based on a static analysis. Details on how the VTPs are used are given in Section 4.5.5.

In single support, the VTP for the stance leg is placed at the location of the desired CoP. In double support, we use a heuristic based on geometric relations, depicted and explained in Figure 12, to determine VTPs $\mathbf{r}_{vtp,0}$ and $\mathbf{r}_{vtp,1}$ in such a way that the desired CoP and both VTPs lie on one line.

The leg support fractions are two scalars, denoted by w_0 and w_1, such that $w_0 + w_1 = 1$ and $w_0, w_1 \in [0, 1]$. A leg support fraction w_i describes the fraction of the desired torque $\boldsymbol{\tau}$ and vertical force f_z to be exerted by stance leg $i \in \{0, 1\}$. The distances between the VTPs and the overall desired CoP determine the leg support fractions:

$$w_i = \frac{\left\| \mathbf{r}_{CoP,des} - \mathbf{r}_{vtp,1-i} \right\|}{\left\| \mathbf{r}_{vtp,0} - \mathbf{r}_{vtp,1} \right\|}, \quad i \in \{0, 1\}. \tag{5}$$

This equation stems from a moment balance around the desired CoP: if a VTP is far removed from the desired CoP, a force exerted at this VTP has a large moment arm, and hence the associated leg support fraction should be small. Support is gradually transferred from one leg to the other in double support due to continuously changing leg support fractions, associated with a continuous desired CoP trajectory.

4.5.5. Virtual support actuator control module The virtual support actuator control module distributes the torque $\boldsymbol{\tau}$ and

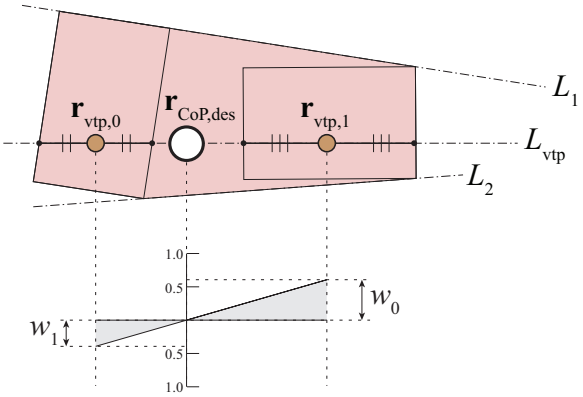

Fig. 12. Implementation of the virtual toe point calculator during double support. Line L_{vtp} is first constructed. It passes through the desired CoP and the intersection of lines L_1 and L_2, the edges of the support polygon that connect the foot polygons. The intersections of line L_{vtp} with the foot polygons define two line segments. The VTPs $\mathbf{r}_{\text{vtp},0}$ and $\mathbf{r}_{\text{vtp},1}$ are found as the center points of these line segments. The leg support fractions are then found based on the distances between the VTPs and the desired CoP using Equation (5), which also has a clear geometric interpretation as shown in the figure. If the desired CoP lies on the outside of a foot, so that it lies farther to the edge of the support polygon than the center of the line segment for that foot, then the VTP for that foot is chosen to be equal to the desired CoP and the leg is assigned a leg support fraction of 1 (this case is not shown in the figure).

the vertical force f_z over the support leg(s) using the leg support fractions w_i as weighting factors:

$$\begin{aligned} f_{z,i} &= w_i f_z \qquad i \in \{0, 1\}, \\ \boldsymbol{\tau}_i &= w_i \boldsymbol{\tau}, \end{aligned} \tag{6}$$

where $f_{z,i}$ and $\boldsymbol{\tau}_i$ are the z-component of the force and the torque to be exerted by leg i, respectively. We use these partial wrenches to compute a complete desired wrench $\mathbf{W}_i$ for each leg, where

$$\mathbf{W}_i = \begin{pmatrix} \mathbf{f}_i \\ \boldsymbol{\tau}_i \end{pmatrix} \text{ with } \begin{aligned} \mathbf{f}_i &= \left(f_{x,i}, f_{y,i}, f_{z,i}\right)^{\mathrm{T}} \\ \boldsymbol{\tau}_i &= \left(\tau_{x,i}, \tau_{y,i}, \tau_{z,i}\right)^{\mathrm{T}}. \end{aligned} \tag{7}$$

The remaining x- and y-components of the force $\mathbf{f}_i$ for each leg are computed using the VTPs. The VTP constraint, which requires that no torque should be applied about either horizontal axis at the VTP, can be enforced as follows. We consider the VTP for a foot to be the intersection of the axes of two virtual pin joints, located on the sole of the foot. Their orthogonal axes of rotation lie in the plane of the foot. The virtual pin joints do not exist on the physical robot, but provide a simple way of computing $f_{x,i}$ and $f_{y,i}$ since the torques across these joints should be zero. The virtual pin joints come after the real joints of the robot in the kinematic chain from upper body to foot. Their rotation angles are set to zero, but their location on the foot changes in time, depending on the location of the VTP. We use $\boldsymbol{\tau}_{\text{vtp},i} \in \mathbb{R}^2$ to denote the vector of virtual joint torques exerted at the virtual pin joints for stance leg i. A static analysis results in

$$\boldsymbol{\tau}_{\text{vtp},i} = \mathbf{J}_{\text{vtp},i}^{\mathrm{T}} \mathbf{W}_i. \tag{8}$$

In this equation, $\mathbf{J}_{\text{vtp},i} \in \mathbb{R}^{6\times 2}$ is the Jacobian that maps the joint velocities of the virtual pin joints to the twist of the foot with respect to a virtual body attached 'after' the virtual pin joints in the kinematic chain, expressed in an upper-body-fixed frame.

Splitting the Jacobian $\mathbf{J}_{\text{vtp},i}$ into a 2×2 block $\mathbf{J}_{\text{vtp},i,2\times 2}$ and a 4×2 block $\mathbf{J}_{\text{vtp},i,4\times 2}$ and using Equation (7), we can rewrite Equation (8) as

$$\boldsymbol{\tau}_{\text{vtp},i} = \mathbf{J}_{\text{vtp},i,2\times 2}^{\mathrm{T}} \begin{pmatrix} f_{x,i} \\ f_{y,i} \end{pmatrix} + \mathbf{J}_{\text{vtp},i,4\times 2}^{\mathrm{T}} \begin{pmatrix} f_{z,i} \\ \boldsymbol{\tau}_i \end{pmatrix}. \tag{9}$$

We require that the torques at a leg's VTP be zero. We can hence solve Equation (9) to find the values of $f_{x,i}$ and $f_{y,i}$:

$$\begin{pmatrix} f_{x,i} \\ f_{y,i} \end{pmatrix} = -\mathbf{J}_{\text{vtp},i,2\times 2}^{-T} \mathbf{J}_{\text{vtp},i,2\times 4}^{\mathrm{T}} \begin{pmatrix} f_z \\ \boldsymbol{\tau}_i \end{pmatrix} \tag{10}$$

The matrix $\mathbf{J}_{\text{vtp},i,2\times 2}^{\mathrm{T}}$ is invertible as long as the following two conditions hold: (1) the z-coordinate of the origin of the upper body, expressed in the foot-fixed frame, is not zero; and (2) the z-axis of the upper-body-fixed frame does not lie in the horizontal plane of the foot-fixed frame. These conditions are always satisfied during normal operation.

Now that $f_{x,i}$ and $f_{y,i}$ are also known, we know the complete wrench $\mathbf{W}_i$ to be exerted on the upper body by stance leg i, and we can use a different Jacobian, $\mathbf{J}_{\text{leg},i} \in \mathbb{R}^{6\times 6}$ to find the joint torques $\boldsymbol{\tau}_{\text{leg},i}$:

$$\boldsymbol{\tau}_{\text{leg},i} = \mathbf{J}_{\text{leg},i}^{\mathrm{T}} \mathbf{W}_i, \tag{11}$$

where $\mathbf{J}_{\text{leg},i}$ maps the joint velocities of the real joints of leg i to the twist of the upper body with respect to the foot, expressed in the upper-body-fixed frame.

Computing $f_{x,i}$ and $f_{y,i}$ based on VTPs instead of specifying these forces directly has as an advantage that VTPs are closely related to the CoP, which plays a major role in the instantaneous capture point dynamics described in Part 1. This relation to the CoP also means that limits due to the finite-sized support polygon are straightforward to accommodate. We simply make sure that each foot's VTP lies inside its convex polygon.

5. Results

In this section we present results obtained for both the balancing task and the walking task. Balancing and walking without pushes was achieved on the real M2V2 robot. Walking while recovering from pushes and walking over stepping stones was achieved on the simulated M2V2 robot. For both balancing and walking, we evaluate the N-step capturability margin proposed in Part 1, that is, the area of the N-step capture region. Here we use $N = 1$ since the presented algorithm is based on remaining 1-step capturable.

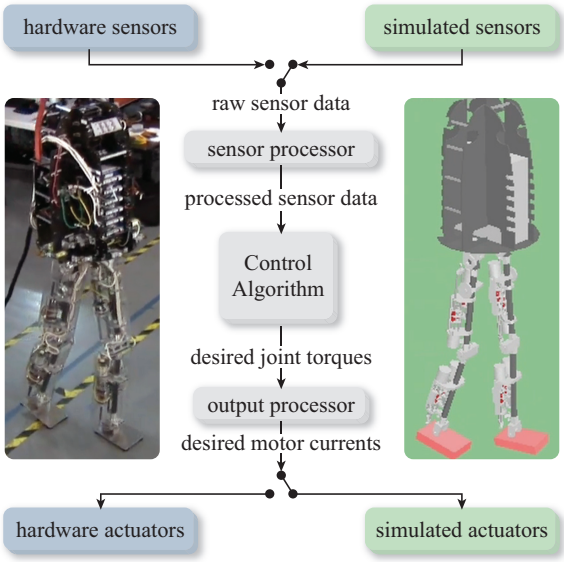

Fig. 13. Overview of the M2V2 software architecture. The majority of the software runs both in simulation and on the real robot, eliminating the need to maintain separate versions, and allowing for significant development and testing in simulation.

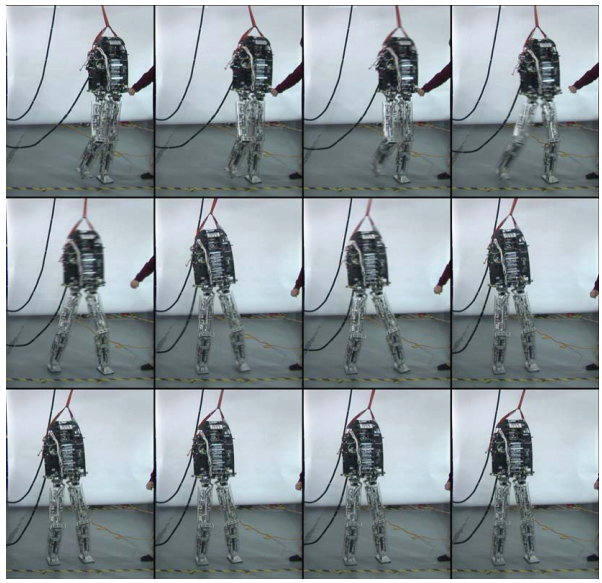

Fig. 14. [REAL] M2V2 recovering from a push while standing on one leg. Images are from left to right starting at the top left.

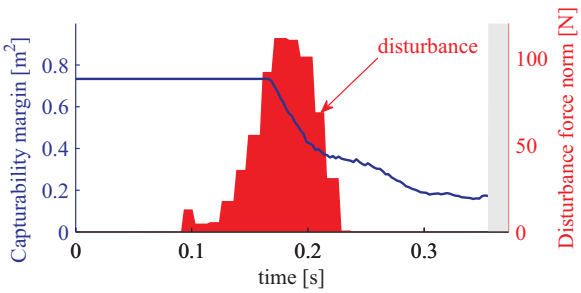

Fig. 15. [REAL] Norm of disturbance force and 1-step capturability margin of M2V2 recovering from a push while standing on one leg. The capturability margin is not shown after transition to the 'stop in double support' state (gray area) to avoid cluttering.

Note that figures are labeled either [REAL] if the data is from the physical robot or [SIM] if from the simulated robot. See Extension 1 for a video of results from both real-world experiments and simulations.

Simulations were performed using the Yobotics Simulation Construction Set (Yobotics, Inc. 2011). Rigid body dynamics were computed using the Articulated Body Algorithm (Featherstone 1987; Mirtich 1996) and simulated using a fourth-order Runge–Kutta integrator with a 0.1 ms integration step size. Ground contact forces were determined using spring-damper ground models. Pushing disturbances were modeled as high-intensity forces of constant magnitude that are applied for a short duration. The disturbance force was applied to the midpoint between the hip joints.

A common control algorithm was used in both simulation and on the physical robot (see Figure 13). The simulated sensors include sensor noise models. The simulated actuators include maximum output force limits and low-pass filters to model the force-control bandwidth of the series elastic actuators.

5.1. Balancing task

On the real M2V2 robot we achieved balancing on one leg and recovering from sideways and forward pushes. Figure 14 shows time-elapsed images of M2V2 recovering from a push. Figure 15 shows the norm of the disturbance force, as recorded from the push stick, and the 1-step capturability margin. We see that the robot was able to recover from pushes of approximately 21 Ns.

5.2. Walking task

On the real M2V2 robot we achieved flat ground walking without disturbances. On the simulated M2V2 robot we achieved walking while recovering from pushes and walking over stepping stones.

Figure 16 shows time elapsed images of M2V2 walking on flat ground. For this walk, the robot uses a constant step length and width. During the walk, the capture region is computed, but since there are no pushes, the robot does not have to change where it steps.

Figures 17–20 show plots of a single data set obtained from simulation for walking on flat ground while recovering from pushes. Three different pushes to the left occur at approximately 4, 8.5, and 13 seconds. These pushes are modeled as forces applied to the midpoint between the hip joints and are 300 N in magnitude for a duration of 0.05 seconds. This corresponds to an impulse of 15 Ns. Note that pushes were to the left while the left foot was swinging. Pushes to the opposite side would require either a cross-over

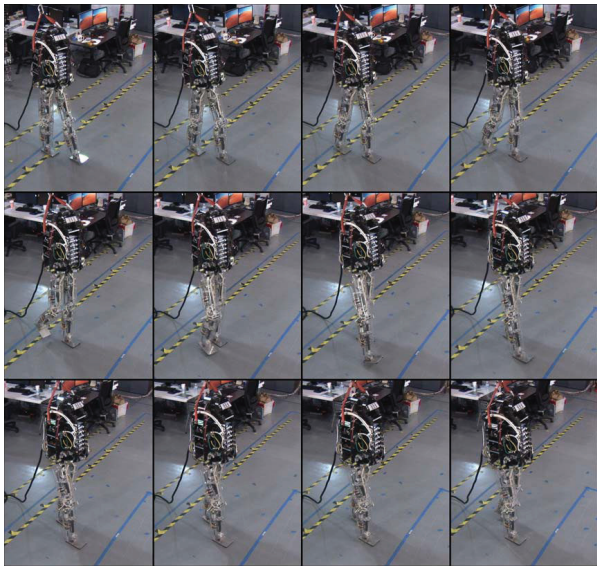

Fig. 16. [REAL] M2V2 robot walking on flat ground. Images are from left to right starting at the top left. In this walk, the robot uses a constant step length of 0.35 m and a constant step width of 0.20 m. The average walking velocity is 0.21 m/s.

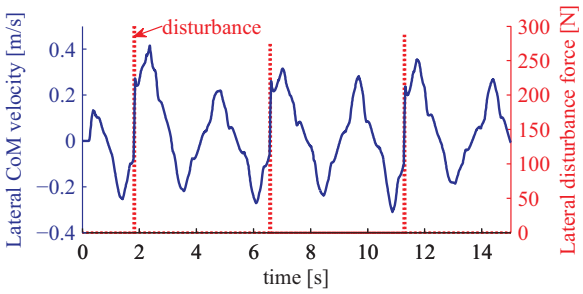

Fig. 17. [SIM] Lateral velocity and disturbances of the M2V2 simulation while recovering from pushes during walking. After each push, the lateral CoM velocity increases by up to 0.44 m/s.

step or two quick steps, both of which are more difficult to achieve and are an area of future work.

Figure 17 shows the sideways pushes applied to the M2V2 simulation, and the resulting change in velocity while recovering to pushes during walking. Since the pushes were mostly to the side, the change in lateral velocity is more prominent than forward velocity. After each push we see that the robot recovers with one step.

Figure 18 shows side and overhead views of the robot. Each time a push occurs, the robot steps to the left to recover from the push. Also plotted in the overhead view are the instantaneous capture point trajectory and the trajectory of each ankle.

Figure 19 shows the 1-step capturability margin during walking of the M2V2 simulation while recovering from pushes. We see that the capturability margin significantly decreases after each of the three pushes, corresponding to the decrease in area of the capture region as seen in Figure 20.

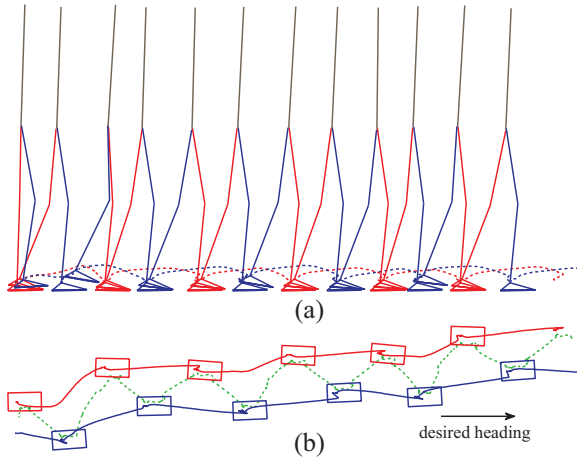

Fig. 18. [SIM] Robot as it walks while being pushed laterally to the left every second step. (a) Side view. Trajectories of the ankles are indicated with dashed lines. (b) Overhead view. Actual ankle trajectories connect the sequence of footprints for each foot. The instantaneous capture point trajectory is indicated with a dashed line.

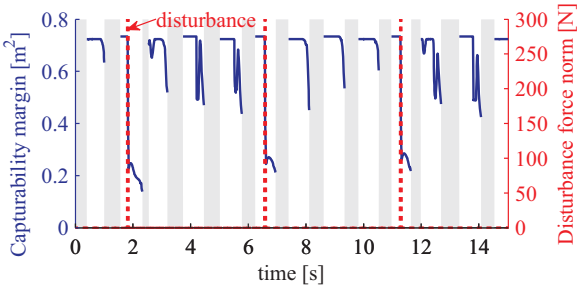

Fig. 19. [SIM] 1-step capturability margin during walking while recovering from pushes. After each push, the 1-step capturability margin significantly decreases showing that the robot is in danger of falling. After each recovery step, the capturability margin recovers. Note that during double support and periods during single support when the instantaneous capture point is inside the support polygon the capturability margin is not plotted (gray areas).

Figure 20 shows a time-lapsed overhead view of the robot. In the first two frames the capture region is relatively large during the beginning of swing. The robot is then pushed between the first and second frame, decreasing the size of the capture region and requiring the robot to choose a different place to step. In frames 3–6 the robot steps further to the left than originally intended, landing in the 1-step capture region, and successfully recovering from the push.

Walking over various stepping stones was achieved on the simulated M2V2 robot. Figure 21 shows an example of walking over stepping stones that are clustered in groups of three. The robot was given exact knowledge of the stepping stones. To ensure that the entire foot rested on each stone, the allowable CoP region for each stepping stone was computed by shrinking the stepping stone based on the size of the foot. However, due to inaccuracies in foot placement, the foot would sometimes slightly overhang the edge of a stone.

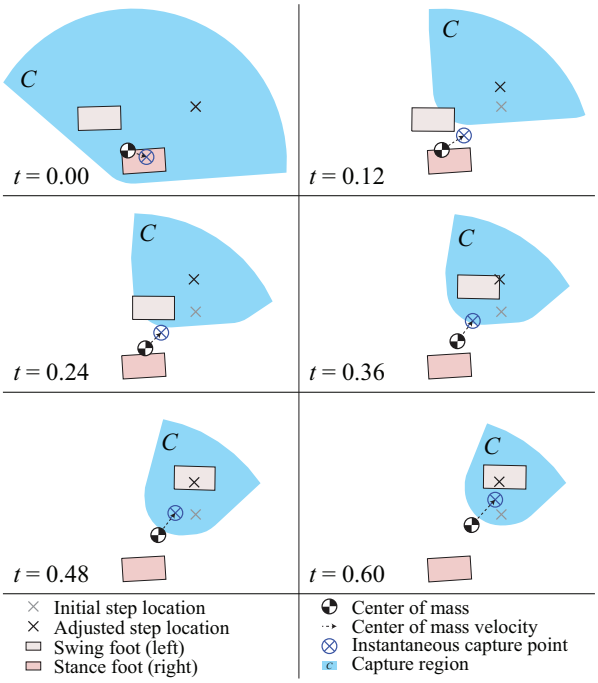

Fig. 20. [SIM] Time-lapse overhead view of the robot during the walking task, showing one step. The robot is perturbed laterally at the start of the step ($t = 0$).

6. Discussion and future work

6.1. Using simple models for complex robots

In this part we used the 3D-LIPM with finite-sized foot, described in Part 1, to estimate the 1-step capture region, which was then used to help control balancing and walking in M2V2, a 3D lower-body humanoid. This simple model was sufficient for controlling balancing on one foot, walking, and recovering from pushes while walking. The simplified model was sufficient for three reasons:

- The model accounts for the dynamics of the CoM with respect to the CoP, which are the key dynamics of walking.
- The model allows for the use of feet, and the modulation of the CoP in real time. This adds robustness, as opposed to an algorithm that predetermines a CoP trajectory.
- The 1-step capture region is relatively large for moderate speed walking. Therefore, there is a large degree of robustness to modeling errors.

The presented control algorithms did not exploit angular momentum of the upper body as a means of control. As we start to address more challenging tasks, such as walking over rough terrain and over narrow beams, we will likely need to use more complex models for computing capture regions and developing control strategies. In Part 1, we analyzed the 3D-LIPM with finite-sized foot and reaction mass. This model should be useful in control when upper-body angular momentum is used to prevent falling. In future work we will use this model and will investigate other strategies for using multi-joint upper-body angular momentum in walking and disturbance recovery.

6.2. 1-step versus N-step capture regions

In this part we developed controllers that always step into the 1-step capture region. This restriction is overly cautious in general, and it is likely the case that fast walking may require periods when the legged system is only 2-step, or perhaps only 3-step capturable. However, M2V2 currently has a long swing time, making the 2-step capture region not much larger than the 1-step capture region.

For M2V2, the falling time constant is $1/\omega_0 = \sqrt{z_0/g} \approx 0.32$ seconds, where $g = 9.81 m/s^2$ is the gravitational acceleration and $z_0 \approx 1.0$ m is the CoM height. The minimum step time Δt_s is currently approximately 0.6 seconds for M2V2. The geometric ratio governing the diminishing returns for the N-step capture region as N increases is $\exp(-\sqrt{z_0/g}\Delta t_s) \approx 0.15$ (see Equation (16) in Part 1). This means that the radius of the 2-step capture region is only about 15% larger than the radius of the 1-step capture region. Therefore, there is not much to be gained in considering 2-step capturability over 1-step capturability with M2V2, until we can get the robot to swing its legs more quickly.

With human walking, on the other hand, the minimum swing time is approximately 0.3 seconds. This results in a geometric ratio of approximately 0.4. Hence, the 2-step capture region for human walking should be relatively large, and even 3-step capturability should be considered.

6.3. Capturability margin

We presented an experimental evaluation of the N-step capturability margin, where we used $N = 1$. Figure 15 and Figure 19 showed a significant decrease in this capturability margin when the robot was perturbed, as expected. For both the balancing task and the walking task, the capturability margin recovered completely after taking one step.

According to the theoretical considerations presented in Part 1, the capturability margin should never increase unless a step is taken or an external force is exerted on the robot. Any other increases in capturability margin are due to modeling errors, sensor noise and lack of exact knowledge about when the swing foot hits the ground.

6.4. Estimation of CoM position and velocity

The algorithm presented, like most feedback control algorithms for bipedal walking, relies on a good estimation of the CoM location and velocity, particularly in the horizontal plane. Obtaining such an estimate on a physical robot is difficult for several reasons.

Knowing the CoM projection on the ground requires knowing which way is down. A small error in the perceived orientation of the body can result in a significant error in

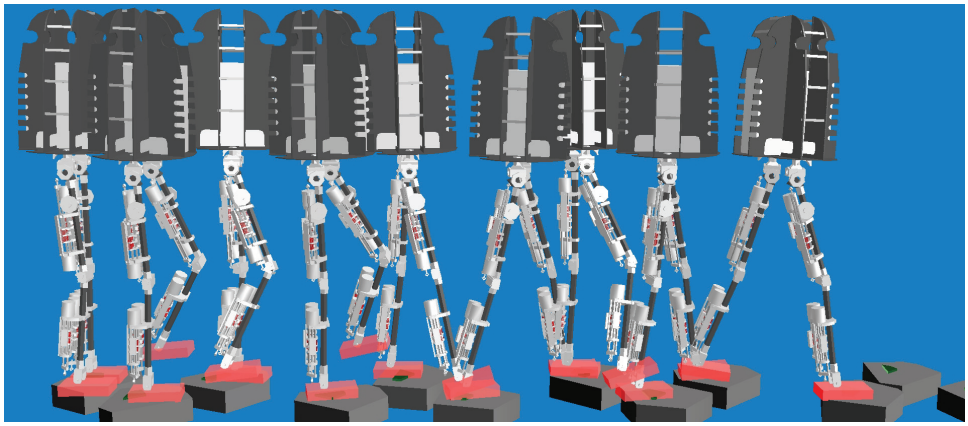

Fig. 21. [SIM] M2V2 walking over stepping stones that are clustered in groups of three. Snapshots are taken at 1.5-second intervals.

the CoM projection. For example, if the CoM is at a height of 1 m and there is a 0.01 rad error in the body orientation, that will result in approximately a 1 cm error of the CoM on the ground. For 3D robots, orientation is typically determined using an inertial measurement unit (IMU), and therefore having a good IMU and related sensor processing is important.

Using leg kinematics to estimate the CoM velocity has the problem that one must assume that the foot is not moving. However, if the foot is slipping, this assumption will make the CoM seem to be moving in the opposite direction. On the other hand, integrating an accelerometer to estimate velocity has the problem of error accumulation. For M2V2 we have used a combination of leg kinematics and accelerometer measurements. We have not yet determined how accurately our CoM velocity estimate is, but we believe we can do much better and therefore improve performance on the physical robot.

6.5. Uneven ground

Both the models presented in Part 1 and the control algorithm presented in Part 2 assumed flat ground. We have performed some preliminary simulations with moderate slopes and relatively small steps, and so far it appears that the same models are applicable to these types of terrain. However, for large slopes and steps, and very rough ground, the models will likely need to be expanded and further control strategies will need to be developed. In addition, we still need to develop capturability-based models and control strategies for situations where hands can push against walls or hold handrails, or where feet can be on different slopes. In these cases concepts such as force closure will need to be taken into account, rather than only using the CoP.

6.6. Controlling velocity versus coming to a stop

Capturability measures the ability of a legged system to come to a stop in N steps or fewer. However, we are not usually interested in coming to a stop, but rather maintaining an average speed. Even though capturability is based on the ability to come to a stop, using tools based on capturability does not require the legged system to come to a stop. Instead, capturability only specifies bounds on what is permissible. Within these bounds, there is ample room for such things as controlling velocity, for example by CoP or foot placement.

6.7. Cross-over steps

In this study, we only considered pushes during single support which did not require a cross-over step. Cross-over steps are challenging for a number of reasons. The swinging leg needs to make sure to not contact the support leg. To do that, the path of the leg may be longer, requiring longer swing time. Also, the length of the step will be smaller than it can be when the leg swings to the outside. An alternative is to quickly step straight down with the currently swinging leg and then quickly swing the other leg to prevent a fall. This two-step recovery strategy requires extra time to execute and for significant pushes will likely only be successful for robots with a relatively short swing time, on the order of how fast humans can swing their legs.

6.8. VTPs and CoP

The presented control algorithm relies on the use of VTPs to track a desired CoP. VTPs can be interpreted as the attachment points of virtual actuators on the feet, and each VTP is used to gain approximate control of each individual foot's CoP. The VTP and a foot's CoP will be theoretically identical if the robot is in single support, the vertical force of the virtual actuator equals the weight of the robot, and the vertical acceleration is zero. In simulation, the two points always remain close (within a few centimeters) during single support. During double support, there can be a large error between the desired and actual CoP, particularly when one of the legs is stretched. For example, if the hind leg is completely stretched, and the desired CoP is on the heel of the hind leg, then the virtual actuator on the hind leg will be assigned a large leg support fraction. However, since the leg is straight, the actual joint torques that the virtual actuator

produces will be small, and the CoP will be located more forward than desired.

One way to obtain a better match between the desired and actual CoP is to keep the knees of the robot bent to avoid losing kinematic range. However, we wish to avoid that solution since human walking does not rely on bent knees and because it requires unnecessarily high torques at the knees. Another solution, which we will investigate in future work, is to use toe off on the rear leg to better control the CoP during double support. Currently, some toe off occurs at the end of the stride, but it is simply the result of the dynamics of the walk, rather than the result of an intentional control action.

6.9. Foot placement speed and accuracy

In this paper, we showed how foot placement can be used to regain balance after a push. Doing so requires a fast swing that is accurate enough to make the foot land in the capture region. However, due to the use of SEAs with very compliant springs, we have had difficulty in quickly and accurately swinging the leg. Although SEAs enable compliant control, they can make joint position trajectory tracking challenging. We believe that we can achieve the same favorable compliant control characteristics and better tracking by increasing the stiffness of the series springs. On the other hand, we also believe that swing can be performed in a more compliant manner, determined mostly by the passive pendulum dynamics of the leg, as opposed to using traditional high-gain trajectory tracking. Determining swing strategies that allow for fast and accurate steps while exploiting the natural dynamics of the leg is an area of future work.

6.10. Application to other robots

We believe that capturability concepts can be applied to the analysis and control of other legged systems. Estimating capture regions and determining capturability-based robustness metrics should be possible for all legged systems. While we advocate compliant force control for legged robots, most of the techniques described in this paper should also apply to high-gain position trajectory tracking robots. Stepping strategies that take the capture region into account should be applicable to any robot that can change where it steps on the fly. Control of the instantaneous capture point should be applicable to any robot that can control its CoP location on the ground. We are currently expanding the algorithms presented in this paper and working toward their application on several different humanoid robot platforms.

7. Conclusion

In this paper we have shown an application of the N-step capturability framework to a 12-DoF bipedal robot. The main contributions of this part were step location adjustment using the 1-step capture region and instantaneous capture point control by CoP placement, which are key ingredients to the presented control algorithm. In addition, the 1-step capturability margin was experimentally evaluated.

Note

1. In practice, we use a slightly smaller support polygon when projecting the tentative desired CoP, to prevent the feet from tipping at times when this is not desired. The use of a smaller support polygon is necessary because of unmodeled dynamics, inability to perfectly track the desired CoP, and model uncertainty.

Funding

This work was funded through the Army Tank and Automotive Research and Development Command (W56HZV-04-C-0072), the Defense Advanced Research Projects Agency (FA8650-05-C-7265), the Office of Naval Research (N00014-09-1-0913-01), NASA (NNX10AG54A), and the Honda Research Institute.

Acknowledgment

The authors would like to thank A Goswami and E Westervelt for their helpful comments.

M2V2 is a second-generation version of the M2 robot, developed by G Pratt, D Paluska, J Pratt, and D Robinson at the MIT Leg Laboratory. M2V2 was designed by B Krupp, V Ragusila, I Olaru, T Craig, and J Pratt. M2V2 electronics and interface software was designed by G Watkins, S Emami, T Hutcheson, J Pratt, J Smith, and S Nayak. M2V2 was assembled and maintained by T Craig and J Taylor. The force sensing push stick used in push recovery experiments on M2V2 was designed and constructed by J Joyner.

A team at Bucknell University, led by S Shooter and K Buffinton, have designed improved feet and a vision head for the robot. Students involved include M Kandler, D Snyder, L Markison, J Ricci and C Hubicki.

Various low-level control modules for M2V2 were developed by C Shake, N van Nieuwenhuizen, F Cañas, D Garg, S Tamadoni, R van Doesburgh, T Koolen, J Pratt, M Johnson, and P Neuhaus. A team from NASA JSC, consisting of J Braman, D Gooding, S Tamblyn, M Goza, and A Hulse, has also contributed in this area.

M2V2 filming and video editing was performed by W Howell. Part acquisition and lab management was performed by B Layton.

Simulation Software and user interfaces were improved by J Carff, M Fortenberry, D Reyes Duran, G Barr, B Waxler, and P DeMonaco.

References

Buss S (2009) *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*. Technical report, University of California, San Diego.

Chen J, Cheng J and Yu G (2001) Virtual model control of a quadruped walking robot. *Journal of the Shanghai Jiaotong University* 35: 1771–1775.

Chevallereau C, Grizzle JW and Shih C (2009) Asymptotically stable walking of a five-link underactuated 3-D bipedal robot. *IEEE Transactions on Robotics* 25: 37–50.

Collins SH, Ruina A, Tedrake R and Wisse M (2005) Efficient bipedal robots based on passive-dynamic walkers. *Science* 307: 1082–1085.

Collins SH, Wisse M and Ruina A (2001) A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research* 20: 607–615.

Coros S, Beaudoin P and van de Panne M (2010) Generalized biped walking control. In *Proceedings of SIGGRAPH 2010*. New York, NY: ACM Press, pp. 1–9.

Englsberger J, Ott C, Roa MA, Albu-Schaffer A and Hirzinger G (2011). Bipedal walking control based on Capture Point dynamics. In *Proceedings of 2011 IEEE/RSJ International Conference on Intelligent Robotic Systems*. Piscataway, NJ: IEEE Press, pp. 4420–4427.

Featherstone R (1987) *Robot Dynamics Algorithms*. Norwell, MA: Kluwer Academic Publishers.

Featherstone R (2008) *Rigid Body Dynamics Algorithms*. Boston, MA: Springer.

Geng T, Porr B and Wörgötter F (2006) Coupling of neural computation with physical computation for stable dynamic biped walking control. *Neural Computation* 18: 1156–1196.

Hu JJ, Pratt JE, Chew C-M, Herr HM and Pratt GA (1998) Adaptive virtual model control of a bipedal walking robot. In *Proceedings of 1998 IEEE International Joint Symposium on Intelligent Systems*, pp. 245–251.

Hu JJ, Pratt JE, Chew C-M, Herr HM and Pratt GA (1999) Virtual model based adaptive dynamic control of a biped walking robot. *International Journal of Artificial Intelligence Tools* 8: 337–348.

Hyon S-H and Cheng G (2007) Disturbance rejection for biped humanoids. In *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, pp. 2668–2675.

Hyon S-H, Osu R and Otaka Y (2009) Integration of multi-level postural balancing on humanoid robots. In *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, pp. 1549–1556. Piscataway, NJ: IEEE Press.

Kagami S, Kitagawa T, Nishiwaki K, Sugihara T, Inaba M and Inoue H (2002) A fast dynamically equilibrated walking trajectory generation method of humanoid robot. *Journal of Autonomous Robots* 12: 71–82.

Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K and Yokoi K (2003) Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of 2003 IEEE International Conference on Robotics and Automation*, pp. 1620–1626.

Koolen T, de Boer T, Rebula J, Goswami A and Pratt J. (2012) Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *International Journal of Robotics Research* 31: 1094–1113.

Kuo AD (1999) Stabilization of lateral motion in passive dynamic walking. *The International Journal of Robotics Research* 18: 917–930.

McGeer T (1990) Passive walking with knees. In *Proceedings of 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp. 1640–1645.

Mirtich BV (1996) *Impulse-based Dynamic Simulation of Rigid Body Systems*. Ph.D. thesis, University of California, Berkeley, CA.

Nishiwaki K and Kagami S (2010) Strategies for adjusting the ZMP reference trajectory for maintaining balance in humanoid walking. In *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pp. 4230–4236.

Paluska DJ (2000) *Design of a Humanoid Biped for Walking Research*. M. eng. thesis, Massachusetts Institute of Technology.

Perry J (1992) *Gait Analysis: Normal and Pathological Function*. Thorofare, NJ: Slack.

Pratt, G. A. (2000a). Legged robots at MIT: what's new since Raibert? *IEEE Robotics and Automation Magazine* 7(3): 15–19.

Pratt GA and Williamson MM (1995) Series elastic actuators. In *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robotic Systems*, pp. 399–406.

Pratt JE (2000b) *Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots*. Doctor of philosophy thesis, Massachusetts Institute of Technology.

Pratt JE, Chew C-M, Torres A, Dilworth P and Pratt GA (2001) Virtual model control: an intuitive approach for bipedal locomotion. *The International Journal of Robotics Research* 20: 129–143.

Pratt JE, Dilworth P and Pratt GA (1997) Virtual model control of a bipedal walking robot. In *Proceedings of 1997 IEEE International Conference on Robotics and Automation*, pp. 193–198.

Pratt JE and Krupp BT (2008) Design of a bipedal walking robot. In *Proceedings of SPIE* 69621F.

Pratt JE, Krupp BT, Ragusila V, et al. (2009) The Yobotics–IHMC lower body humanoid robot. *Proceedings of 2009 IEEE/RSJ International Conference on Intelligent Robotic Systems*, pp. 410–411.

Pratt JE and Pratt GA (2002) Intuitive control of a planar bipedal walking robot. In *Proceedings of 1998 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2014–2021.

Pratt JE and Tedrake R (2006) Velocity-based stability margins for fast bipedal walking. In Diehl M and Mombaur K (eds), *Fast Motions in Biomechanics and Robotics* (*Lecture Notes in Control and Information Sciences*, vol. 340). Berlin: Springer, pp. 299–324.

Sreenath K, Park H-W, Poulakakis I and Grizzle JW (2012) A compliant hybrid zero dynamics controller for achieving stable, efficient and fast bipedal walking on MABEL. *The International Journal of Robotics Research*, submitted for publication.

Stephens B (2011) *Push Recovery Control for Force-Controlled Humanoid Robots*. Ph.D. thesis, Carnegie Mellon University.

Stephens BJ and Atkeson CG (2010) Dynamic balance force control for compliant humanoid robots. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robotic Systems*

Sugihara T (2010) Consistent biped step control with COM-ZMP oscillation based on successive phase estimation in dynamics morphing. In *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pp. 4224–4229.

Takenaka T, Matsumoto T and Yoshiike T (2009) Real time motion generation and control for biped robot - 1st report: Walking gait pattern generation. In *Proceedings of 2009 IEEE/RSJ International Conference on Intelligent Robotic Systems*. Piscataway, NJ: IEEE, pp. 1084–1091.

Tan F, Fu C and Chen K (2010) Biped blind walking on changing slope with reflex control system. In *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pp. 1709–1714.

Vukobratovic M and Borovac B (2004) Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics* 1: 157–173.

Westervelt ER, Buche G and Grizzle JW (2004) Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds. *The International Journal of Robotics Research* 23: 559–582.

Westervelt ER, Grizzle JW and Koditschek DE (2003) Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control* 48: 42–56.

Yobotics, Inc. (2011). Legged Robots, Simulation, Actuators, http://www.yobotics.com/.

Appendix A: Robot parameters

Table A1 gives the joint layout and inertia parameters of M2V2. Each row represents a joint and its associated rigid body. We use a coordinate system in which x is forward, y is to the left, and z is up. Note that the bottom six rows represent the left leg joints and masses (marked with the letter 'L'). The right leg is a mirror image of the left leg, and thus is identical to the left leg except for the y values, which are all the additive inverse. For pin-type joints, the letter following 'Pin' refers to the rotational axis to which the joint is aligned.

Appendix B: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Results of control algorithm implementation, both on the physical robot and in simulation.

Table A1. M2V2 simulated robot joint and mass layout. Approximated from CAD models of real M2V2 robot. The joint offsets are with respect to each joint's parent joint and defined when all joint angles are 0. The CoM offsets are with respect to each joint origin. The moments of inertia are with respect to the CoM of each link.

		Offset from parent [m]					COM offset [m]			Moment of inertia [kg m^2]		
Joint	Parent	x	y	z	Joint Type	Mass [kg]	x	y	z	I_{xx}	I_{yy}	I_{zz}
Root	N/A	0.0	0.0	0.0	Floating	24.0	0.0	0.0	0.0	2.756	2.756	0.380
L Hip Yaw	Root	0.0	0.092	−0.381	Pin-Z	0.2	0.0	0.0	0.0	5.3×10^{-4}	5.3×10^{-4}	5.3×10^{-4}
L Hip Roll	L Hip Yaw	0.0	0.0	0.0	Pin-X	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L Hip Pitch	L Hip Roll	0.0	0.0	−0.029	Pin-Y	4.6	−0.0318	0.006	−0.229	0.044	0.044	0.004
L Knee	L Hip Pitch	0.0	0.0	−0.450	Pin-Y	4.3	−0.025	0.0	−0.229	0.054	0.054	0.003
L Ankle Pitch	L Knee	0.0	0.0	−0.467	Pin-Y	0.11	0.0	0.0	0.0	2.6×10^{-4}	2.6×10^{-4}	2.6×10^{-4}
L Ankle Roll	L Ankle Pitch	0.0	0.0	0.0	Pin-X	0.727	0.0127	0.0	−0.051	3.63×10^{-4}	0.002	0.002