Applied Intelligence (2019) 49:3801–3814
https://doi.org/10.1007/s10489-019-01446-z

Closed-loop push recovery for inexpensive humanoid robots

Amirhossein Hosseinmemar[1] · Jacky Baltes[2] · John Anderson[1] · Meng Cheng Lau[1] · Chi Fung Lun[1] · Ziang Wang[1]

Published online: 22 March 2019

Abstract

Active balancing in autonomous humanoid robots is a challenging task due to the complexity of combining a walking gait with dynamic balancing, vision and high-level behaviors. Humans not only walk successfully over even and uneven terrain, but can recover from the interaction of external forces such as impacts with obstacles and active pushes. While push recovery has been demonstrated successfully in, expensive robots, it is more challenging with robots that are inexpensive, with limited power in actuators and less accurate sensing. This work describes a closed-loop feedback control method that uses an accelerometer and gyroscope to allow an inexpensive humanoid robot to actively balance while walking and recover from pushes. Three common balancing strategies: center of pressure, centroidal moment pivot, and step-out, for biped robots are studied. An experiment is performed to test three hand-tuned closed-loop feedback control configurations; using only the gyroscope, only the accelerometer, and a combination of both sensors to recover from pushes. Each of the sensors is discretized into four discrete domains in order to categorize pushes with different strengths. Experimental results show that the combination of gyroscope and accelerometer outperforms the other methods with 100% recovery from a light push and 70% recovery from a strong push. The proposed closed-loop feedback control is examined in both simulation and real-world.

Keywords Push recovery · Humanoid robot · Autonomous active balancing · Centroidal moment pivot · Stepping

1 Introduction

Robots have been used for decades, but moving from factory floors to the everyday lives of humans is a challenging task. Traditional wheeled robots are more stable and balanced than humanoid robots [1]. Also, they can avoid obstacles easily and generate a new path [2]. However, humanoid robots are closer to a human's body shape, and as a result they can potentially function better in environments structured for humans, such as offices and homes. One of the main difficulties with humanoid robots is balancing: falls occur easily, even when walking on flat surfaces [3]. A typical humanoid robot has two arms, two legs, a head and a torso and it locomotes on its two legs. However, there are semi-humanoid robots, such as Robovie [4] and PR2 [5] that have two arms and a head but are not able to walk like a human, instead using wheels to locomote.

Most adult humans not only walk well on both flat and uneven surfaces, they also recover successfully from external forces introduced while walking, such as low-impact obstacle collisions or small pushes in any dimension. Push recovery in robotics involves dealing with these external forces: negotiating and recovering from an abnormal status to a normal situation when either walking or standing [6, 7].

The majority of basic walking algorithms are designed to walk on flat surfaces at a predefined static angle to the ground and with feet parallel to it, and do not account for external forces [8]. While a number of push recovery approaches have been implemented in humanoid robots, these are generally intended to operate on platforms that are currently very expensive: in the hundreds of thousands or even millions of dollars (e.g. Atlas [9]). Such platforms have very powerful servos, allowing significant force to be used to correct aberrations, strong power supplies (e.g. hydraulic), highly precise machining, and very refined, rich sensors. In our work, we deal with sophisticated problems in artificial intelligence and robotics using much less expensive equipment. Working with less expensive equipment means that all elements of software, from vision

✉ Amirhossein Hosseinmemar
memar@cs.umanitoba.ca

[1] Autonomous Agents Laboratory, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada

[2] Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan

to control, must be much more robust. For example, push recovery on an inexpensive robot must be achieved with lower torque servos, less precise machining, and limited sensing and computation. Limb and servo damage are also much more likely with inexpensive robots and must be taken into account.

This paper describes work on push recovery implemented on an autonomous humanoid robot meeting adult-size standards for the FIRA HuroCup robotics competition [10]: *Polaris*, a 95 cm tall humanoid based on readily-available Robotis Dynamixel servos. Polaris uses an inertial measurement unit (IMU) as an input sensor for balancing, incorporating both a gyroscope and accelerometer. We present a closed-loop control method that allows Polaris to actively balance while walking as well as to recover from pushes. To evaluate this approach, we isolate the means of perception to examine the effect on this control method. We tested this approach using three sensor configurations for feedback: using only the gyroscope, using only the accelerometer, and finally using a combination of both sensors. These mechanisms were all used to recover from pushes in the form of a suspended 2 kg mass released from varying distances. This approach has also been demonstrated successfully in the field, resulting in several competition awards including a first place award in the RoboCup 2016 and 2018 humanoid technical challenge for push recovery. It has also been embedded as a part of our robotics code in award-winning FIRA HuroCup entries in 2016, 2017 and 2018.

The remainder of this paper is as follows: Section 2 describes background on push recovery and related work. Section 3 describes the hardware employed and the closed-loop control method for push recovery. Section 4 describes the evaluation of this work through experimental testing. We then discuss the results and directions for future research.

2 Related work

There are two common models that are used for walking in humanoid robots, either on their own or in combination with other methods. One of these is the Linear Inverted Pendulum model [11, 12], used in robots such as [13]. This method is one of the simplest biped locomotion strategies that used in human kinetic to describe the change in potential energy and kinematic. The inverted pendulum motion model provides predictable trajectories of the Center Of Mass (COM) of the body. The other is Zero Moment Point (ZMP) [14, 15]. Humanoid robots such as ASIMO [16], HRP-2 [17], HUBO [18] generate their trajectory based on the ZMP method, and they are indeed the leading humanoid robots in walking. ZMP takes both static and dynamic forces into consideration. There are two main components for the ZMP walking control method, 1) a walking pattern generator 2) stabilization around it. Therefore, the walking trajectories cannot be predictable, unlike the linear inverted pendulum method. This method is more costly in terms of computational power in compare to linear inverted Pendulum model. These humanoids walk very stably, and their walk is reliable on predefined or flat surfaces. However, this method is not practical when it comes to collisions or any other type of disturbance and would lead to a fall in almost all such cases [19]. Balancing in a humanoid robot involves maintaining the robot's center of mass (COM) within the support polygon provided by the robot's feet in its current pose. This is illustrated for Polaris in Fig. 1. Any push or other external force moving the COM outside this support polygon will cause a fall. Such a situation may happen for two main reasons. The first is that of applying direct external forces such as pushing the robot's chest, causing the robot to fall. Such a push may come from any direction. The second reason would be stepping on uneven terrain, including bumping into obstacles. To solve the stability issue of the robot, the Center of Pressure (COP) [20] needs to be in the support polygon region. One of the practical ways to address this problem is predicting the footstep from capturing walking motions [21] and using a gait generator such as capture steps [19]. By adjusting the footsteps in the x or y plane, or both, the area of the support polygon can be made large enough to balance the robot.

Push recovery refers to negotiating and recovering from an abnormal status to a normal situation either in walking or standing [6, 7]; this would happen when the robot is subjected to a large disturbance from external forces. These forces apply to the robot for a short period, and they destabilise the walking rhythm and cause the robot to fall over. Generating a smooth and stable walking path in an uneven terrain that could be an unknown surface with obstacles is a very complex task: The robot needs to learn and generate footsteps based on the environment and external forces.

Over decades many solutions have been introduced to solve the balancing problem of humanoid robots when they are subjected to considerable disturbance, external forces and walking on uneven terrain. Among them the three common strategies are [6]: 1) Centre of Pressure, 2) Balancing Centroidal Moment Pivot, and 3) Step-out and capture step.

Center of Pressure (COP): This approach usually is based on controlling ankles on both feet, and it is also known as the *Ankle strategy* [6]. This method is often used when there is a small disturbance and by just shifting the centre of pressure to relocate the COM between the support polygon. Figure 2a demonstrates COP. In this figure, L is the height, F_x the applied force to the body in the x direction, F_z is the

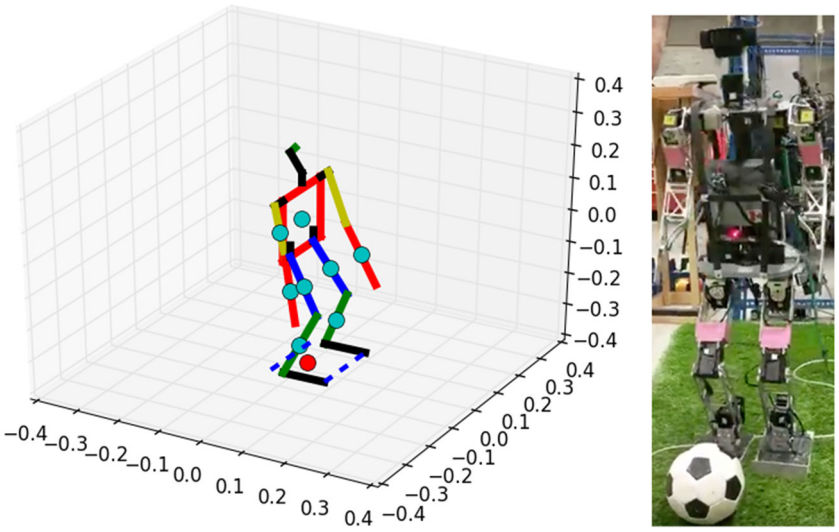


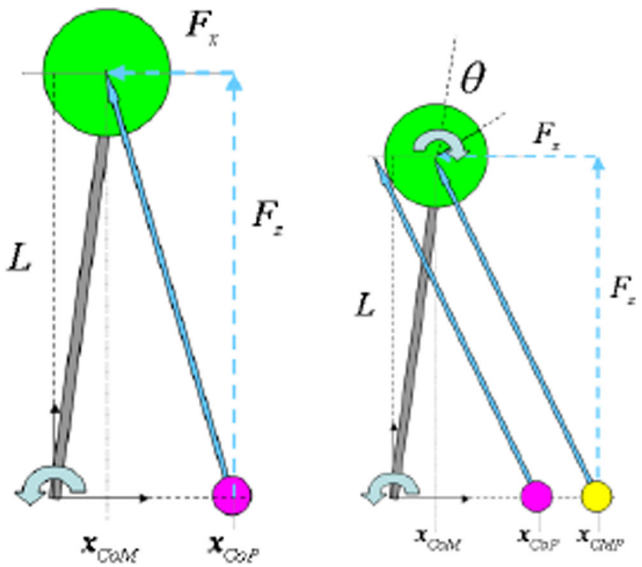
Fig. 1 Polaris (right) and model showing center of mass (red ball at ground level) during normal walking gait. (Color figure online)

ground force. The arrow next to the x_{CoM} is the adjustment that is applied to the ankle's degree for shifting the COM backward.

Centroidal Moment Pivot: Figure 2b shows this strategy, also know as the *hip strategy* [6]. This can be used for small and medium disturbances, and is thus an improvement over COP. In this method, the hip servos play a significant role in recovering from the push. In Fig. 2b, θ is the angle that the hip servo will change from the position before the force is applied. The change in the hip servo helps to absorb most of the external force and balance the COM. Since CMP can also modify the ankle position in the opposite direction of the external force, it subsumes COP. For example, Iverach-Brereton [22] worked on active balancing of a kid-size humanoid robot for his master's thesis. His primary research focus was on balancing a kid size robot on a Bongo-board. He used three different control algorithms that derived from the cart-and-pole inverted pendulum problem: PID control, fuzzy logic and Always-on Artificial Neural Networks. He used two different approaches for applying these algorithms. The first approach that he used is *Do the Shake* that the robot reacts to any external forces and it didn't produce any new trajectories other than the recovery. The second approach was called *Let's Sway* that the robot created a new path to promote the dynamics stability. The robot were able to recover its balance by adjusting the ankle and hip joint to relocate the COM in the frontal plane. Iverach-Brereton et al. [23] used a very similar approach to balance a small alpine skiing humanoid robot (DaRWin), actively on the snow. The approach was based on *Do the Shake* that was previously discussed.

Step-out: Is the last practical strategy that can handle small, medium, and large disturbances in many cases. This strategy is also known as *capture step* [6]. This involves taking another step to relocate the COM within the support polygon area. Figure 3 illustrates these three balancing strategies, with the COM shown as a light blue circle, the COP shown as a red circle, and a brown arrow indicating the external force. The yellow arrow indicates one dimension of the support polygon.

The step-out strategy can be divided into 3 main categories. First is the *control interface* that is the high-level layer, and it uses the Omni-walk (omnidirectional) to send the feet or joints in a particular location on *XYZ* axis. V denotes the velocity vector, and it is $\in \mathbb{R}^3$ in the three sagittal, lateral and rotational directions. The frequency is



(a) COP (b) CMP

Fig. 2 Centre of pressure and Centroidal Moment Pivot [6]

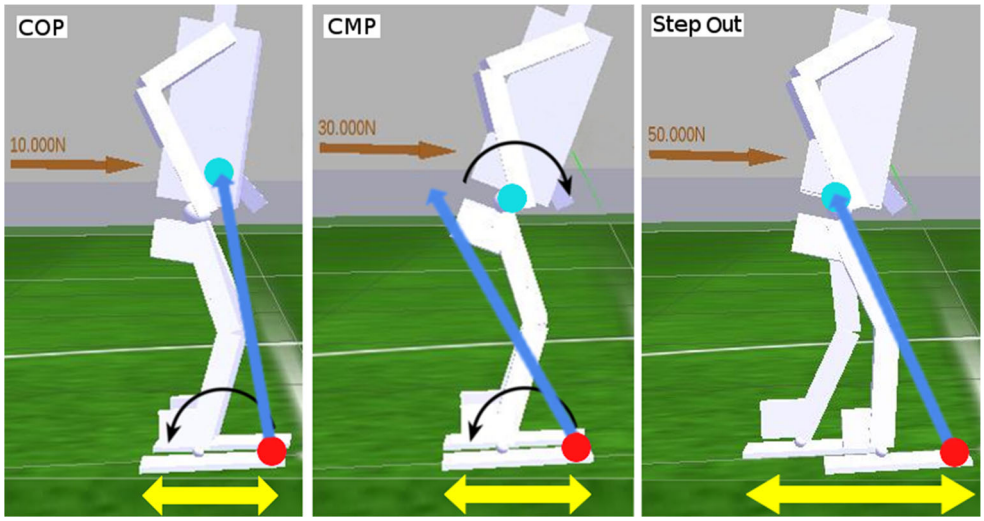


Fig. 3 COP, CMP, and Step-out, using a model of Polaris (after [6])

indicated by Φ^*, and the desired step parameters are denoted by S^* and:

$$S^* = (S_x^*, S_y^*, S_z^*) \in \mathbb{R}^2 \times [-\pi, \pi] \quad (1)$$

S_x^* and S_y^* refer to the sagittal and lateral Cartesian coordinates of the footstep respectively, and the S_z^* is the rotation of the feet. The second is the *foot placement control*. The inputs of this stage are the desired step parameters Φ^* and S^*. However, apart from the desired step parameters the forward kinematics and COM of the robot play a significant role in here. The distances in x and y plane calculated, and a 4D COM created that is:

$$COM_{state} = (x, \dot{x}, y, \dot{y}) \quad (2)$$

x and y denote the location of sagittal and lateral COM areas respectively and $\dot{x}$ and $\dot{y}$ denote sagittal and lateral velocity of the COM. The last is the *Motion Generator* and in this stage, the trajectory for the next step will be generated. The step length S that was given to the second stage (foot placement control) has the direct impact on the swing amplitude. Additionally, the walking gait frequency Φ that discussed in the second phase decides how fast the stride will be executed.

There have been various implementations within each of these three balancing strategies. For example, Toyota's running robot (130 cm high, 50 kg) generates a new trajectory after a push based on the position of its COM and the support foot, and successfully recovers from pushes against the chest during hopping and running [24]. MABEL [25] also demonstrated the ability to stabilise the walk after a push by generating a new trajectory. However, the former is a highly expensive robot that would not compare to the comparatively low-torque servos in Polaris, and the latter is a planar bipedal robot mounted on a boom of radius 2.25 m, and so can only walk around in a circle.

Yun et al. [26] introduced a momentum-based stepping controller that tested an adult-sized humanoid robot in a simulation called Locomote, a software package based on Webots. In their solution, the simulator checks the maximum threshold of the angles as well as the torque for each joint of the two legs. If one of its joints passed their threshold, its step trigger function will be called and it will take a step for fall prevention.

Lee et al. [27] presented another stepping approach that was specially for non-stationary and non-continuous grounds, also in a simulation. Their solution was very costly because of calculating COP, COM, and the linear and angular momentum of the robot in real time. Since this approach is non-continuous, it cannot be applied to a normal walking gait.

Hofmann [28] studied humanoid robot balance control. He argued that taking a step for recovering from an external push is the solution for recovering from a large disturbances by moving the COP. Missura et al. [19] also studied push recovery for a simulated humanoid robot that uses capture steps for recovering. In their approach, the simulated robot calculates a desired ZMP location for every step with respect to the COM.

Many of these and other related works are examined only in simulation, or in restricted settings such as walking in a circle while suspended from a pivoting boom. Those that are physically implemented tend to require highly expensive platforms. Our approach is intended to function for small, medium and strong size pushes on an inexpensive platform, which can only rely on lower-torque servos, low computational power, less precise body machining, and less accurate sensors.

3 Approach

Our approach consists of a closed-loop control mechanism implemented on an autonomous humanoid robot. We begin with an overview of the hardware of *Polaris*, the humanoid robot used to evaluate this work, and then describe the control design and implementation.

3.1 Hardware

Polaris is a 95 cm humanoid robot with 20 degrees of freedom (DOF), weighing 7.5 kilograms. Polaris makes use of comparatively inexpensive ($200-$500) Dynamixel servos from Robotis. All the servos use the TTL serial communication protocol with three pins that share one line for sending and receiving data. Each hip can rotate in the sagittal, frontal, and transversal planes. There are 6 MX-106 servos in each leg (Hip transversal, Hip sagittal, Hip frontal, Knee sagittal, Ankle frontal, and Ankle sagittal). The two arms rotate in the sagittal and frontal planes, and each arm has 3 MX-64 servos (Shoulder sagittal, Shoulder frontal, and Elbow sagittal). The neck is made up of 2 MX-28 servos and rotates in the sagittal and transversal planes, moving an attached webcam. Figure 4 demonstrates these three types of servo motors.

We employed a USB2Dynamixel [30] board from Robotis that uses a USB port to control the servo motors through Polaris's computer. All Polaris's servo motors use TTL (3 pins) network connection to communicate with the other servo motors or the computer. Figure 5 shows the USB2Dynamixel device and Fig. 6 illustrates its connection's diagram.

Sensory information is processed and servo adjustments driven by a *QutePC-3000* mini-PC, with a 1.1Ghz Celeron dual core processor. Sensors for balance consist of a single-chip *InvenSense MPU-6050* IMU containing a 3-axis (x, y, z) accelerometer and a 3-axis (x, y, z) gyroscope. The MPU-6050 is connected to the mini-PC using an Arduino Nano micro-controller board.

3.2 Closed-loop control: using CMP

A closed-loop control system is a control system that uses one or more feedback loops. A humanoid robot using a closed-loop control system can receive feedback from its sensors (vision, inertia) describing changes to the environment (targets, pushes, uneven terrain) and correct its trajectory to adapt to these changes. A closed-loop control system is more computationally expensive compared to an open-loop control system due to the enormous amount of sensory data being processed by the controller. However, it is a more robust control system because of this sensory feedback.

Polaris uses a walking engine based on the linear inverted pendulum model [12, 32], which generates appropriate robot motions based on a description provided by the inverse kinematics of the robot, i.e. sets motion vectors for all servos over time. Our closed-loop control mechanism sets inputs to the walking engine, allowing it in turn to adjust the robot's COM, dynamically altering this as the environment changes. As stated in Section 3.1, we use the sensory feedback from the MPU-6050's gyroscope and accelerometer as the inputs to measure the control output. The walking engine will adapt the robot's trajectory (control output) as necessary to deal with changes in the environment, from varying terrain to external forces.

We describe our control approach through three stages illustrated in Fig. 7, showing a sample push recovery using this approach. In Stage 1, the robot is pushed by hand at a point in its walking gait on a concrete floor, and must recognize that it is in a falling state. Stage 2 represents a brief window in which the robot can calculate a reaction to the push by calculating control changes based on the angular velocity and the linear velocity of the robot's torso. Stage 3 illustrates recovery, where these control changes alter parameters in the robot's walking engine, and these in turn adjust servos accordingly to prevent the robot from falling.

In **Stage 1**, the closed-loop controller takes as input values from the gyroscope and/or accelerometer, and from these must detect a falling state. In practice this can be computationally expensive because of the range of potential values. To allow a fast response, we discretize values. The control methodology categorizes angular velocity in $50 degree/s$ intervals, allowing a definition of constant values for light, medium and strong pushes. This interval was chosen based on three, 5-minute robot walk tests to find the maximum angular velocity that Polaris encounters while using different walking parameters for each test. We similarly experimented with linear velocity thresholds by hitting the robot with a weight approximately a quarter of its body weight, and defined light, medium, and strong pushes as linear velocities of 0.9 m/s, 1.2 m/s and 1.4 m/s, respectively.

Our closed-loop control collects 1000 gyroscope and accelerometer readings per second from the IMU, and uses these to continually check if the robot is in a falling state. Based on the three thresholds for linear and angular velocities, the robot can very quickly determine whether it is in a stable state ($0\ m/s >$ linear velocity $\leq 0.2\ m/s$, with this range necessary to deal with sensor noise), or when it is in a falling state by exceeding angular velocities or linear velocity thresholds for light, medium, or strong pushes.

Figure 8 demonstrates the IMU reading of linear velocity and its discretization. The maximum value of linear

(a) MX-28 (b) MX-64 (c) MX-106

Fig. 4 The three different types of servo motors used in Polaris [29]

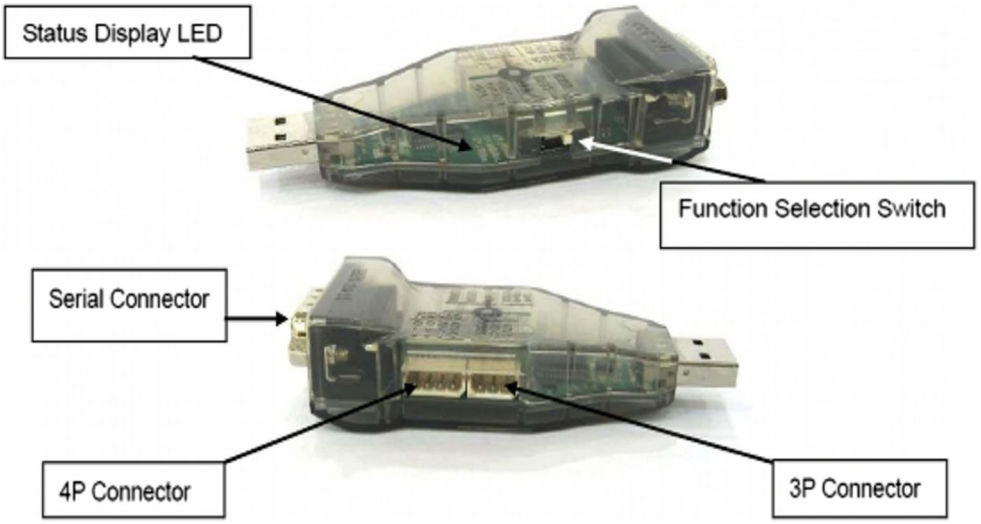


Fig. 5 USB2Dynamixel connection diagram [31]

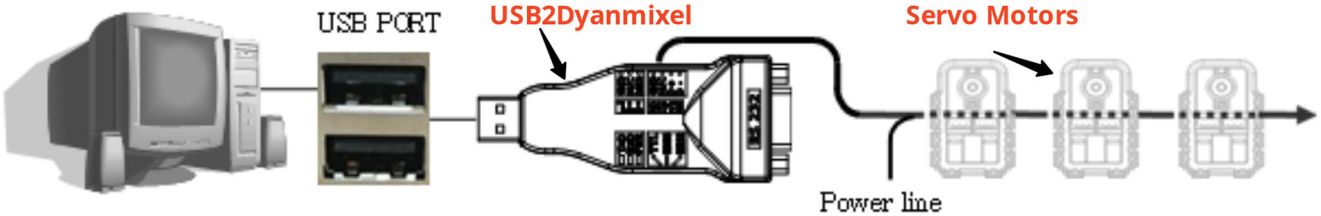
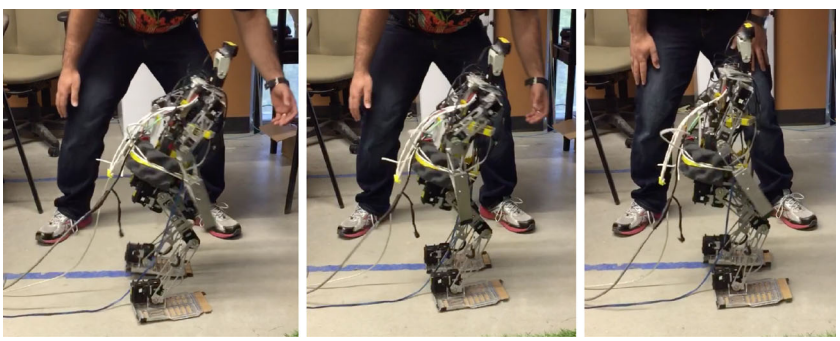


Fig. 6 USB2Dynamixel connection diagram [31]



(a) Stage1 (b) Stage2 (c) Stage3

Fig. 7 Push, Reaction, and Recovery (using CMP)

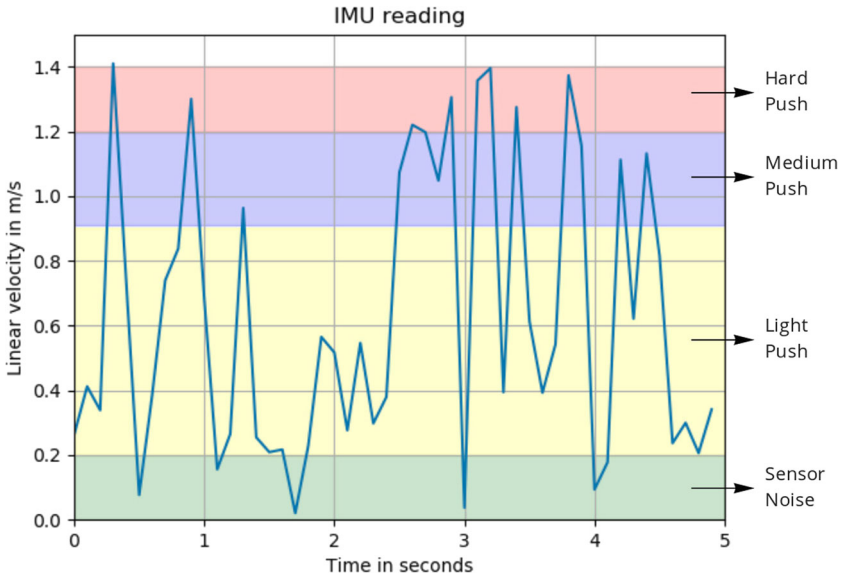


Fig. 8 Discretization of IMU reading for linear acceleration. The Green region is the sensor noise area that is ignored. The Yellow region is a light push. The Purple region is a medium push. The Pink region is a strong push

acceleration was collected every 10 milliseconds over a duration of five seconds during Polaris' walking gait.

In the brief window between when a falling state is recognized and when the fall would be irreversible (**Stage 2**), the robot must make appropriate response. Knowing which threshold (light, medium, or strong push) has been exceeded allows for a quick response look-up. Our control approach implements CMP active balancing (Section 2), which incorporates COM if only the ankles are moved, and similarly discretizes potential responses to support a fast reaction.

For testing the accuracy of this control system, we employed a new set of tools: Robot Operating System *(ROS)* [33] and *Gazebo* [34]. ROS is a set of software libraries and tools that helps to develop a robot software. Gazebo is a physics enabled, high quality graphics simulation that is compatible with ROS. We had designed a 3-dimensional model of Polaris in Gazebo that has the same weight and height to the real-world. Figure 9 demonstrates this 3d model in Gazebo simulation during a jumping test.

As it was stated previously, Polaris is able to measure acceleration and velocity in real-world. To simulate the outside world in the simulation accordingly, we implemented the gyroscope and accelerometer in Gazebo to read the feedback and use it in our closed-loop control system. For reading the sensory feedback in real-time in ROS and Gazebo, we used *Rviz* [35], and adapted our code to this ROS package.

Action Discretization: Our closed-loop controller produces nine outputs that are used to alter parameters in the robot's walking engine through modifying hip and/or ankle positions, and are illustrated in Table 1. *Step-x* is the step length on the x-axis of the robot frame (forward-backward). *Step-y* is the step length on the y-axis of the robot frame (left-right). *Step-height* is the foot height from the ground

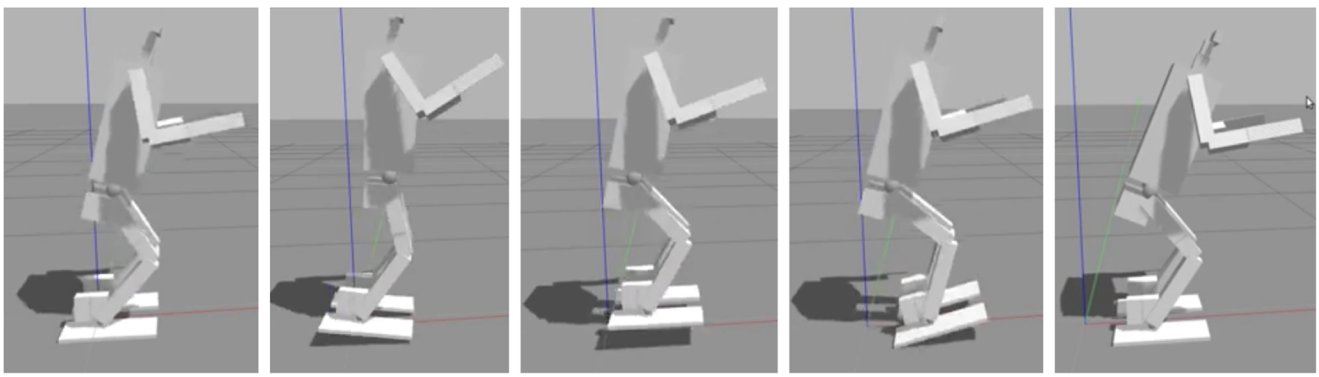


Fig. 9 Polaris is doing a jumping test in Gazebo for testing the accuracy of our control loop system

Table 1 Walking engine parameters and value ranges

Walking engine parameters	Threshold	
	Minimum	Maximum
Step-x	−5 cm back	+5 cm front
Step-y	−5 cm right	+5 cm left
Step-height	1 cm	8 cm
x-offset	−5 cm back	+5 cm front
y-offset	+4 cm	+8 cm
z-offset	34 cm	44 cm
Step-pace	125,000 μs	500,000 μs
Hip-pitch	−15° tilt back	+15° tilt front
Ankle-pitch	−10° tilt back	+10° tilt front

of each step. The *x-offset* is the offset distance of the feet from the origin/centre of the robot (centre point of the torso) on the x-axis. *y-offset* is the offset distance on the y-axis from the centre of the torso to each foot. *z-offset* is related to the height of the robot, i.e. standing fully vs. at a crouch. *Step-pace* is the robot's speed in terms of the time it takes to take a single step (not a full walking cycle). The final two parameters are *hip-pitch* and *ankle-pitch* for the lateral motion at the hip and ankles, respectively.

Each of these walking engine parameters has minimum and maximum values, also indicated in Fig. 1. Any setting acts as an offset value for the current robot pose. For example, if the robot's torso is leaning too much to the front, the robot will fall over. To encounter this problem, *hip-pitch* can be tuned to adjust the torso's lateral motion to prevent the robot from falling.

All nine of these responses have particular values based on the strength and direction of the push that has been recognized. Values can be zero, indicating no change. For example, a light push on the right side results in detecting a fall to the left, and modifies the *step-y* value by 2 cm and the *step-pace* value by 0.5 per second, leaving all other values unchanged. These values and their mapping to discretized angular and linear velocities (fall states) have been tuned over several years of robotics competitions as well as testing the robot specifically under push recovery conditions. These have proven themselves in the field to be a very fast method of adapting control to changing conditions (e.g. carpet vs. solid surfaces) in addition to push recovery.

Once the appropriate response has been mapped, the parameters to the walking engine are changed (**Stage 3**), and servos are collectively altered in the time window that remains to correct for the disturbance. Central to all of this is making Stage 2 as short as possible, leaving time for the servos to be adjusted to recover from the fall.

Domain Discretization: Based on the strength and direction of the push, we have discretized the states of the robot into 14 states that can be mapped with a look-up table. Figure 10 shows all the 14 states. For every direction {front, right, left and back} there are four distinct *discrete states* plus two common discrete states. These states are: 1) Stable state: the robot is stable in this state, and this is one of the common discrete states 2) Light push state: ω, which represents angular velocity, is $\leq \theta_l$, where θ_l denotes the maximum ω that a 2 kg object suspended from a 1 meter rope causes after being released from a 30 cm distance. 3) Medium push state: ω is $\leq \theta_m$, where θ_m denotes the maximum ω that a 2 kg object causes after being released from a 40 cm distance. 4) Strong push state: ω, is $\leq \theta_s$, where θ_s denotes the maximum ω that a 2 kg object causes after being released from a 50 cm distance. 5) Fall state: the robot could not recover successfully and fell. This state is the other common discrete state.

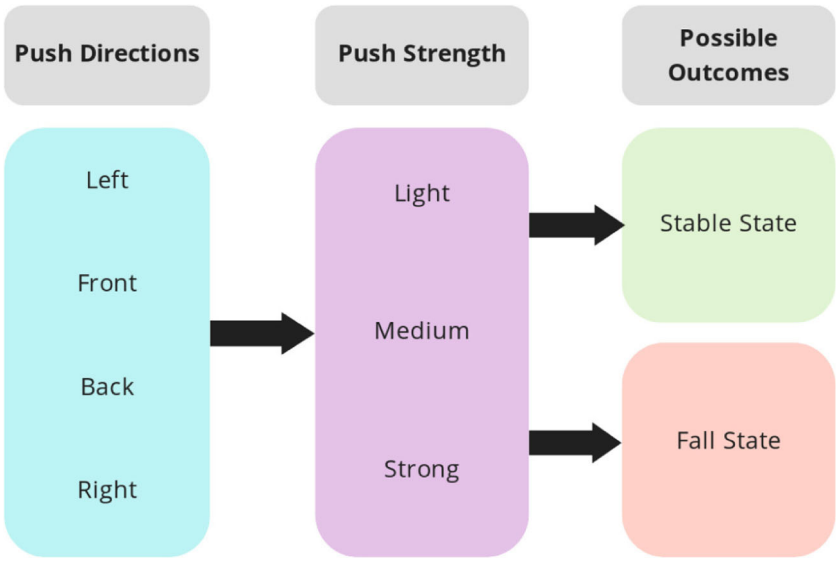


Fig. 10 Discretization of the robot states

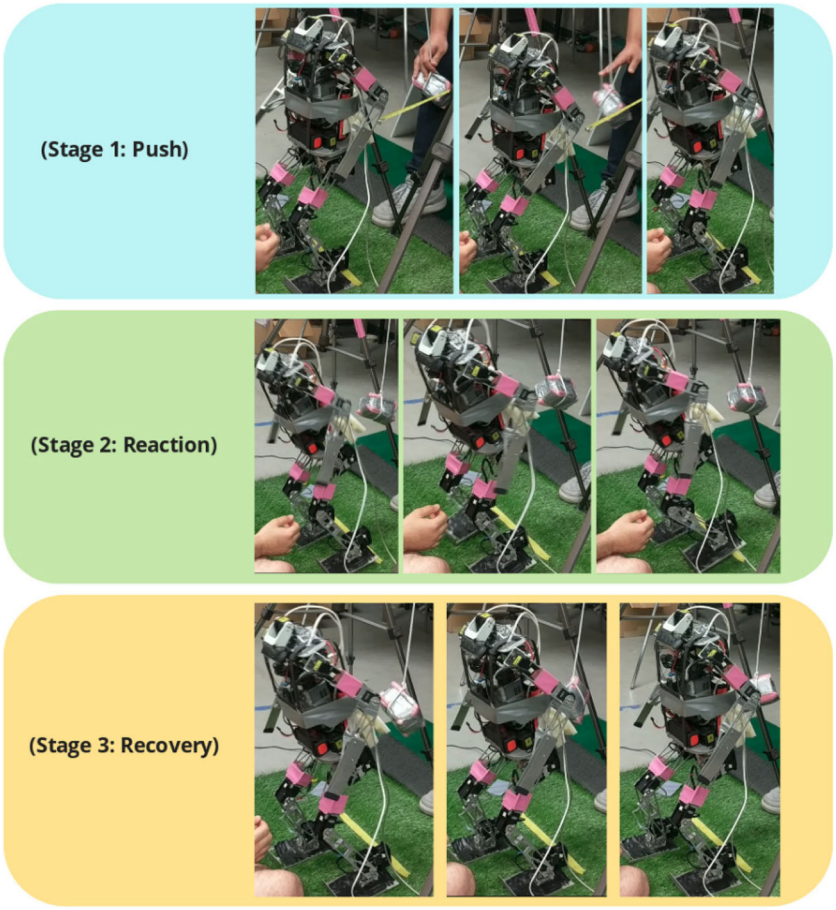


Fig. 11 Push, Reaction, and Recovery (using step-out strategy, sequence starts from left)

3.3 Closed-loop control: using step-out

Even though Polaris was able to recover from many different light and medium pushes from different directions and on different surfaces (e.g. concrete and carpet), Polaris was not able to recover from a strong push (Fig. 10). To give the robot the ability to recover from even stronger external forces, we implemented a step-out strategy to the robot's push recovery module. By taking extra steps, the robot can handle the external pushes more easily and less pressure will be applied to the servo motors. Therefore, the walking engine was modified to take an individual step (half of a cycle) for faster reactions to the external disturbances. This means the robot can take only one step that could be a right or left step, not a stride. We implemented the step-out in such a way that every step has its own walking engine parameters as it is shown in Table 1. Based on the different walking situations, these parameters can be altered. Our step-out approach is an extension of the CMP model [3] that was discussed in Section 3.2. Similar to our CMP model, that has three stages (Stage1, Stage2, and Stage3) Fig. 7, our step-out closed-loop control model also follow the exact structure.

Figure 11 demonstrates a successful push recovery by taking extra steps. The 2 kg weight was released from a 50 centimeter distance, causing a push when hitting the robot. For this experiment, to examine the robustness of our closed-loop control, we test the robot on a different floor. Artificial turf is one of the most difficult surfaces for the robot in terms of recovery. The turf's surface is very soft and usually its height is between 2-3 cm, based on our competition experiences. Walking on such a surface alone, makes the robot unstable. Moreover, applying pushes from different directions to the robot with an external object, easily causes a fall. In Fig. 11, during Stage 1 (Push), Polaris reads the value of falling state from its sensory feedback value from gyroscope and accelerometer. Then in Stage 2 (Reaction), Polaris maps the sensory feedback values to the discretized table, and it measures the strength of the force. Finally, in Stage 3 (Recovery), Polaris updates its walking engine parameters to counter the applied push and it recovers successfully.

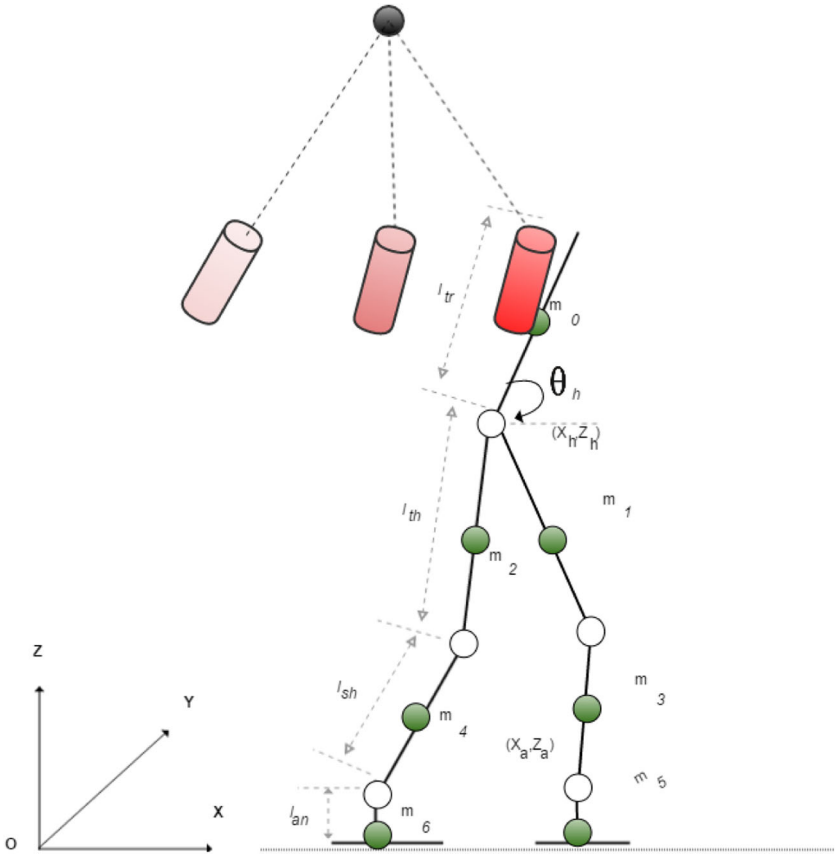
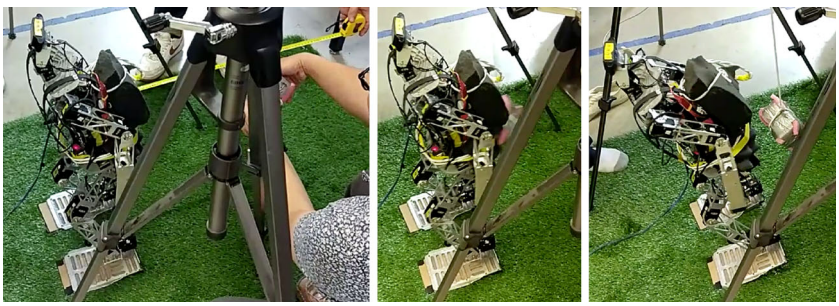


Fig. 12 Humanoid robot with double support contact. The green circles show masses for each individual link. The red gradient cylinder represents a 2 kg mass that is swinging toward the robot, which will create a disturbance upon impact

4 Evaluation

To evaluate our approach quantitatively, we set up an experiment to control the push force applied to the robot, based on the 2015 RoboCup push recovery technical challenge [36]. This is illustrated abstractly in Fig. 12, showing the robot in the double support phase of a walking gait (i.e., both feet on the ground). The red cylinder represents a 2 kg mass (more than a quarter of Polaris' body weight) that is connected to a 1 meter rope . The container is pulled back to a given distance and released naturally, resulting in the mass hitting the robot at torso height, introducing an external force that can be varied depending on the distance at which the mass is released.

For this experiment, the robot was positioned on artificial turf for all trials, with a piece of foam mounted on the robot to protect the electronics from impact. The experiment tested the closed-loop control method using 90 trials divided into three subsets: 30 trials using only the gyroscope as input to the control mechanism, 30 trials using only the accelerometer, and 30 trials using both of these sensors. In each of these divisions, the 2 kg mass was released from a 30 cm distance from the robot (light push) in 10 trials, from a 40 cm distance (medium push) in 10 trials,



(a) stage 1 (b) stage 2 (c) stage 3

Fig. 13 Polaris recovering from a push on artificial turf

Table 2 Results of trials for gyroscope-only (Gyro), accelerometer-only (Acc) and both of these sensors together (Gyro+Acc)

MPU-6050 sensors	Gyro			Acc			Gyro+Acc		
Distance, cm	30	40	50	30	40	50	30	40	50
Trials	1	1	0	0	1	0	1	1	0
	1	1	0	1	0	1	1	1	1
	0	1	0	1	0	0	1	0	1
	1	0	0	1	0	0	1	1	1
	1	1	0	0	0	0	1	1	1
	1	0	0	1	0	0	1	1	1
	1	1	0	1	1	0	1	1	0
	1	1	0	1	1	0	1	1	1
	1	1	0	1	1	0	1	1	0
	1	0	0	0	0	0	1	1	1
Successful attempts	**9**	**7**	**0**	**7**	**4**	**1**	**10**	**9**	**7**

and from a 50 cm distance (strong push) in 10 trials. Figure 13 illustrates Polaris during the course of one of these trials.

4.1 Results

The results of all trials are shown in Table 2. Each case in which the robot successfully recovered from the impact is labeled with a 1, and each case in which the robot did not successfully recover is labeled with a 0.

These results are summarized in Fig. 14. Our closed-loop (with threshold) control mechanism is able to recover from majority of pushes (>70%) at the low impact level (30 cm swing) irrespective of the sensor type employed. At the medium impact level (40 cm swing), there is a more distinct difference between use of a gyroscope vs. use of an accelerometer, only 40% of the impacts are recoverable compared to the gyroscope. At the strong impact level (50 cm swing), the gyroscope is of no help, but a small number of pushes can be recovered using the accelerometer. The strongest success by far is using both sensors together, resulting in the strongest push recovery at all impact levels (30 cm = 100%, 40 cm = 90%, 50 cm = 70%). As pushes become stronger, there is a shorter window for the robot to respond and greater servo alterations to be made. Ultimately, neither sensor on its own can recognize a fall early enough to allow a response to a strong push, while fusing both sensors allows this in most cases.

5 Discussion and future work

Despite the great results that we obtained from our experiment as well as different international robotics competitions, there are some limitations with our proposed method.

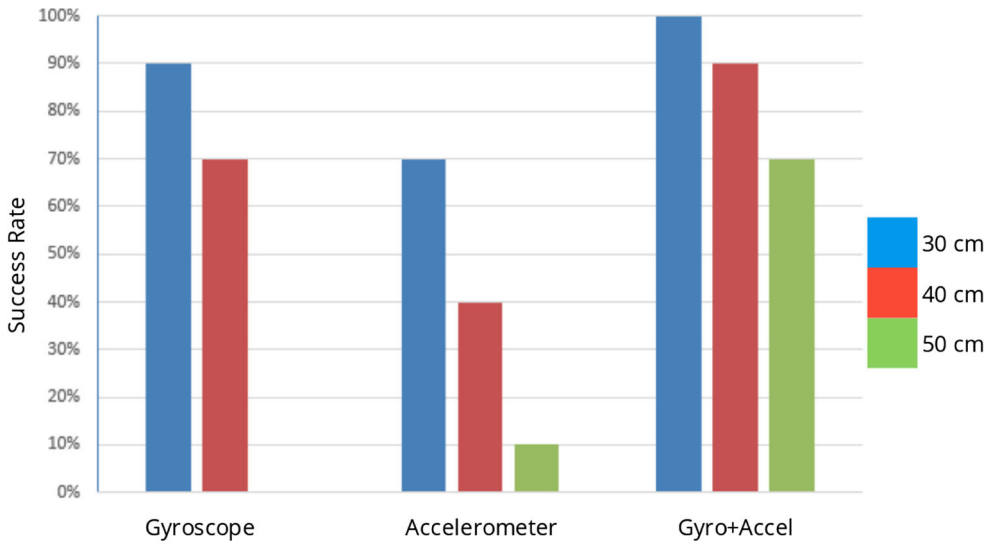


Fig. 14 Summary of successful recovery by distance of mass and type of sensor

The robot's walking engine configures (Table 1) are set based on the human operator's experiences. If a very small factor in the robot's environment changes, it is very difficult to find a set of appropriate walking engine parameters for making the robot actively balanced. One of interesting future work to overcome this limit is to employ machine learning techniques. Reinforcement learning (RL) and Deep Reinforcement learning (DRL) are two possible approaches to achieve this goal.

Whenever the surface that the robot is standing on changes (for example, changing the concrete floor to artificial turf), the robot should have a prior knowledge about it. This way it uses the predefined configuration for that specific surface. As a future work, the robot can learn (using machine learning) different walking trajectories on different surfaces. This learning process should be done in a simulation environment (to avoid damaging the servo motors). After the process of learning is finished, the robot can choose actions on different surfaces based on its previous experiences, unlike, knowing it in advance by the human operator.

We are currently working on a step-out method of push recovery, using reinforcement learning and deep reinforcement learning to automatically generate sensor and reaction discretizations like those used in this work. This will allow the robot to recover from even stronger pushes while minimizing the set of servo motions, to both react quickly and limit wear on servos. We will be using the results described here as a baseline comparison for that work.

6 Conclusion

In this paper, we have described the design, implementation, and evaluation of a closed-loop control approach for push recovery on an inexpensive 20-DOF humanoid robot. In addition to the results shown here, this work has also been demonstrated in the field with strong success. This mechanism for push recovery was our basis for entering the RoboCup Humanoid league technical challenge for push recovery (this event varies by year and normally has more than one challenge with scores totalled). We won this challenge in 2016 and 2018, came second in 2015, and the highest results in the push recovery component in 2017 [37]. The described control loop system also won a best paper award out of 146 accepted publications at the IEA-AIE 2018 conference. Recovery from unexpected impacts is an important part of many physical activities, and this approach also forms a core within our entries to other robotic sporting events such as the 2016 and 2017 FIRA HuroCup. These field evaluations are very important, since each of these events requires robots to work in a range of tasks under novel locations with very limited time for developers to recover from anything unexpected. Such conditions can be expected to be much more challenging than laboratory environments [38]. These events must also go hand in hand with quantitative evaluations however, since while challenging, it is difficult to isolate the performance of one feature in a competitive setting (e.g. how much a push recovery methodology on its own affects playing soccer).

References

1. Huang Q, Yokoi K, Kajita S, Kaneko K, Aral H, Koyachi N, Tanie K (2001) Planning walking patterns for a biped robot. IEEE Trans Robot Autom 17(3):280–289
2. Shojaeipour S, Parhizkar B, Hosseinmemar A, Shojaeipour A, Esfandiari H, Mobasheri E, Gebril B, Mohana Z (2010) Laser-pointer rangefinder between mobile robot and obstacles via webcam based. In: Key Engineering Materials. Trans Tech Publ, Vol 447, pp 609–613
3. Hosseinmemar A, Baltes J, Anderson J, Lau MC, Lun CF, Wang Z (2018) Closed-loop push recovery for an inexpensive humanoid robot. In: Mouhoub M, Sadaoui S, Ait Mohamed O, Ali M (eds) Recent Trends and Future Technology in Applied Intelligence. Springer International Publishing, Cham, pp 233–244
4. Ishiguro H, Ono T, Imai M, Maeda T, Kanda T, Nakatsu R (2001) Robovie: an interactive humanoid robot. Ind Robot: Int J 28(6):498–504
5. Cousins S (2010) ROS on the PR2. IEEE Robot Autom Mag 17(3):23–25
6. Stephens B (2007) Humanoid push recovery. In: Proceedings of Humanoids-2007. IEEE, Pittsburgh, pp 589–595
7. Stephens BJ, Atkeson CG (2010) Push recovery by stepping for humanoid robots with force controlled joints. In: Proceedings of Humanoids-2010, Nashville, pp 52–59
8. Kim JY, Park IW, Oh JH (2007) Walking control algorithm of biped humanoid robot on uneven and inclined floor. J Intell Robot Syst 48:457–484
9. Kuindersma S, Deits R, Fallon M, Valenzuela A, Dai H, Permenter F, Koolen T, Marion P, Tedrake R (2016) Optimization-based locomotion planning, estimation, and control design for Atlas. Auton Robot 40(3):429–455
10. Baltes J, Tu KY, Sadeghnejad S, Anderson J (2017) Hurocup: competition for multi-event humanoid robot athletes. Knowl Eng Rev 32:1–14
11. Pratt J, Carff J, Drakunov S, Goswami A (2006) Capture point: A step toward humanoid push recovery. In: 2006 6th IEEE-RAS International Conference on Humanoid Robots. IEEE, pp 200–207
12. Lee SH, Goswami A (2007) Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In: Proceedings of ICRA-2007, Rome, pp 4667–4672
13. Kajita S, Morisawa M, Miura K, Nakaoka S, Harada K, Kaneko K, Kanehiro F, Yokoi K (2010) Biped walking stabilization based on linear inverted pendulum tracking. In: Proceedings of IROS 2010. IEEE, Taipei, pp 4489–4496
14. Vukobratović M (1973) How to control artificial anthropomorphic systems. IEEE Trans Syst Man Cybern 3(5):497–507
15. Yi SJ, Zhang BT, Hong D, Lee DD (2011) Online learning of a full body push recovery controller for omnidirectional walking. In: IEEE-RAS International conference on humanoid robots
16. Hirai K, Hirose M, Haikawa Y, Takenaka T (1998) The development of Honda humanoid robot. In: Proceedings of ICRA-98, pp 1321–1326

17. Morisawa M, Kanehiro F, Kaneko K, Mansard N, Sola J, Yoshida E, Yokoi K, Laumond JP (2010) Combining suppression of the disturbance and reactive stepping for recovering balance. In: Proceedings of IROS 2010, Taipei, pp 3150–3156
18. Cho BK, Park SS, Oh JH (2010) Stabilization of a hopping humanoid robot for a push. In: Proceedings of IROS 2010, Taipei, pp 60–65
19. Missura M, Behnke S (2013) Omnidirectional capture steps for bipedal walking. In: Proceedings of Humanoids-2013, Atlanta, pp 401–408
20. Collins JJ, De Luca CJ (1993) Open-loop and closed-loop control of posture: a random-walk analysis of center-of-pressure trajectories. Exper Brain Res 95(2):308–318
21. Schmitz A, Missura M, Behnke S (2011) Learning footstep prediction from motion capture. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6556 LNAI, pp 97–108
22. Iverach-Brereton C (2015) Rocking the Bongo Board: Humanoid Robotic Balancing on Dynamic Terrain. Master's thesis, University of Manitoba, Winnipeg, Manitoba
23. Iverach-Brereton C, Postnikoff B, Baltes J, Hosseinmemar A (2017) Active balancing and turning for alpine skiing robots. Knowl Eng Rev 32:e6, 1–10
24. Tajima R, Honda D, Suga K (2009) Fast running experiments involving a humanoid robot. In: Proceedings of ICRA-2009, pp 1571–1576
25. Sreenath K, Park HW, Poulakakis I, Grizzle JW (2011) A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL. Int J Robot Res 30(9):1170–1193
26. Yun SK, Goswami A (2011) Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery. In: Proceedings of IROS-2011, San Francisco
27. Lee SH, Goswami A (2010) Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In: Proceedings of IROS-2010, Taipei, pp 3157–3162
28. Hofmann A (2006) Robust execution of bipedal walking tasks from biomechanical principles. PhD thesis, Massachusetts Institute of Technology
29. Robotis: Dynamixel (2018) http://www.robotis.us/dynamixel/
30. Robotis: Usb2dynamixel (2017) http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm/
31. Trossenrobotics: Robotis usb2dynamixel adapter (2018) https://www.trossenrobotics.com/robotis-bioloid-usb2dynamixel.aspx
32. Pratt J, Carff J, Drakunov S, Goswami A (2006) Capture point: a step toward humanoid push recovery. In: Proceedings of Humanoids-2006, Genoa
33. System RO (2016) Kinetic Kame robot operating system http://www.ros.org/
34. Gazebo: Why gazebo? (1999) http://www.quanmax.com/site/product/qutepc-3000-series/
35. System RO (2017) Rviz. http://wiki.ros.org/rviz
36. RoboCup: Humanoid league technical challenge. http://www.robocuphumanoid.org/wp-content/uploads/HumanoidLeagueRules2015-06-29.pdf
37. Ramezani S, Setaieshi A, Pourmohammadi N, Yarahmadi P, Arvand A, Fallah F, Hosseinmemar A, Santos J, Morris K, Lau MC et al (2017) Autman humanoid teen size team description paper robocup 2017 humanoid robot league. Teen-size Humanoid League, Team Description Paper, Nagoya
38. Anderson J, Baltes J, Tu KY (2009) Improving robotics competitions for real-world evaluation of ai. In: Proceedings of the AAAI Spring Symposium on Experimental Design for Real-World Systems, Stanford

Amirhossein Hosseinmemar received his Bachelor of honours in science in Information Technology from Limkokwing University of Creative technology in 2010. He received his master of science in Computer science from the University of Nottingham in 2012. Mr. Hosseinmemar started his Ph.D. in 2013 at the University of Manitoba. Since 2013, he has been an active member of Autonomous Agent Laboratory at the University of Manitoba. For his Ph.D., he worked on active balancing of a 1 meter tall humanoid robot using reinforcement learning and deep reinforcement learning. Mr. Hosseinmemar has taught variety of computer science courses at the university of Manitoba, International College of Manitoba and St. Boniface Diocesan High School. His research interests include humanoid robotics, walking engines, machine learning, reinforcement learning, and deep reinforcement learning.

Dr. Jacky Baltes received his Ph.D. in 1996 from the University of Calgary, Canada in the area of artificial intelligence and machine learning. For his Ph.D., Dr. Baltes developed a learning multi-strategy planning system called DoLittle. From 1996 to 2002, Dr. Baltes worked as a Senior Lecturer in the department of computer science at the University of Auckland, New Zealand. In 2002, Dr. Baltes moved to the University of Manitoba in Winnipeg, Canada, where he was promoted to Full Professor in 2008. In 2016, Prof. Baltes took up a position as Full Professor at the National Taiwan Normal University (NTNU) in Taipei, Taiwan, where he was awarded an Outstanding Professor supplement and is now the Director of the Educational Robotics Lab. Dr. Baltes also continues to hold an adjunct appointment at the University of Manitoba. He was elected as President of the Federation International of Robot Sports Association (FIRA). He is a Executive Committee member of the International RoboCup Federation since 2010, and a member of the humanoid league organizing committee and the technical committee since 2002. He is also the founder and Chair of the IEEE Conference on Intelligent Robots and Systems (IROS) competition Humanoid Application Challenge - Robot Magic. IROS is the flagship conference for robotics. Dr. Baltes has been founding member of two successful startups Cogmation Inc., Winnipeg, Canada and Taibotics, Kaohsiung, Taiwan. Dr. Baltes also holds 1 U.S.A. patent, 2 Chinese patents, and 1 European patent.

Dr. John Anderson has been a faculty member of the Department of Computer Science at the University of Manitoba since 1995, after completing his Ph.D. at the same institution. He has been running the Autonomous Agents Laboratory in the department since 1999, and fielding teams to major robotics competitions such as RoboCup and FIRA since 2003. He completed his first term as Head of the Department in 2017, and is a member of the Board of Governors of the University. He is also a member of AAAI and a Senior Member of the IEEE.

Meng Cheng Lau received his B. IT (2003) and M. IT (2006) in Industrial Computing from National University of Malaysia. In 2014, He received his Ph. D in Computer Science from the University of Manitoba. Dr. Lau was a lecturer (2006–2008) at Universiti Tunku Abdul Rahman and Senior Lecturer (2008–2016) at National University of Malaysia. He was appointed as the Head of Autonomous Robot and Vision Systems Laboratory (2014–2016) at National University of Malaysia. In 2016. He joined Autonomous Agent Laboratory at the University of Manitoba as a Postdoctoral Fellow and Sessional Instructor. He has been the leader and member for a range of robotics competitions and research groups. Dr. Lau has authored and coauthored over 20 publications in the journals, book chapters and proceedings. His research interests include computer vision, intelligent robotics, and machine learning.

Chi Fung Lun is currently an undergraduate student in Computer Science at the University of Manitoba. Mr. Lun has been an active member of Autonomous Agent Laboratory since 2016. He received the Undergraduate Student Research Awards in 2017.

Ziang Wang received his B. S in Electrical Engineering (2017) from the University of Manitoba. Currently, Mr. Wang is pursuing his M. S in Electrical Engineering at the University of Manitoba. His research interests are microfluidics and robotics. Mr. Wang has been an active member of Autonomous Agent Laboratory since 2016.