Model-based Fall Detection and Fall Prevention for Humanoid Robots

Thomas Muender[1], Thomas Röfer[1,2]

[1] Universität Bremen, Fachbereich 3 – Mathematik und Informatik, Postfach 330 440, 28334 Bremen, Germany
[2] Deutsches Forschungszentrum für Künstliche Intelligenz, Cyber-Physical Systems Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
E-Mail: `thomas.muender@uni-bremen.de, thomas.roefer@dfki.de`

Abstract. Fall detection and fall prevention are crucial for humanoid robots when operating in natural environments. Early fall detection is important to have sufficient time for making a stabilizing movement. Existing approaches mostly analyze the sensor data to detect an ongoing fall. In this paper, we use a physical model of the robot to detect whether the measured sensor data indicates a fall in the near future. A trajectory for the foot is calculated to compensate the rotational velocity and acceleration of the fall. In an evaluation with the humanoid robot NAO, we demonstrate that falls can be detected significantly earlier than with traditional sensor classification with little false-positive detections during staggering. Falls due to small to medium impacts can be prevented.

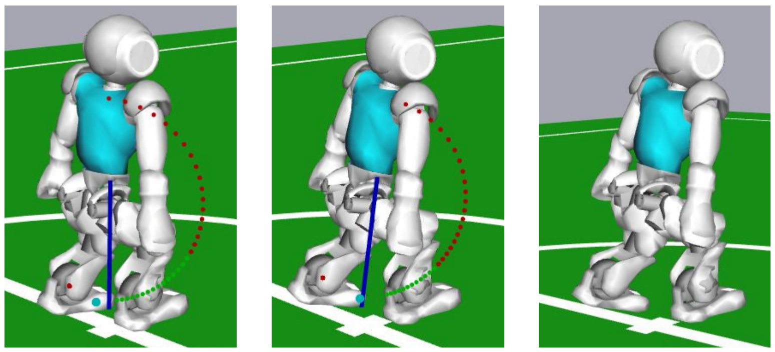

Fig. 1: Stepping motion to prevent a fall using the SSIP model.

1 Introduction

Humanoid robots are inherently unstable due to their small footprint compared to the height of their center of mass (CoM) [24]. This is a problem widely encountered in locomotion generation for robots such as humanoid walking and kicking. Even though there are many improvements in fall prevention and controlled walking [7,26] robots are still at risk to fall due to obstacles, impacts, or uneven/soft ground.

In order to prevent a humanoid robot from falling, two steps are necessary: First, the fall has to be detected. Fall detection must be fast in order to have sufficient time left to perform a counter-action. Furthermore it has to be robust to false-positive detections to prevent the robot from initiating a counter action when it is not necessary. To this time, fall detection is mostly achieved using a form of sensor classification. In contrast to other existing methods, the approach presented in this paper does not only classify sensor data to detect an ongoing fall, but it rather determines whether the sensor data can lead to a fall in the near future. A physical model is used to simulate the effect of the forces acting on the robot. The model can be used to calculate whether the robot will be in a stable or an unstable state. In case of an unstable state, the model calculates how the robot will fall. This can be used to prevent the robot from falling over.

The main contribution of this paper is the development of a physical model for a humanoid robot capable of describing the robot during a fall. We extend the commonly used inverted pendulum model by a stand space and body dynamics of the robot. Analyzing the common approach to use linearization for the non-linear inverted pendulum shows that it is not feasible to describe a falling motion. Therefore, we use an iterative model to simulate the dynamics of the robot. This model is used to detect imminent falls and perform a counter motion. We demonstrate that a fall can be detected significantly earlier in comparison to traditional sensor classification methods. Falls due to small to medium impacts to the robot can mostly be prevented using a lunge.

The paper is organized as follows: In Sect. 2, we analyze related work, followed by the development of the physical model and functions in order to detect and prevent a fall in Sect. 3. In Sect. 4, the evaluation of the developed approach is described. A comprehensive discussion of the results is given in Sect. 5, followed by the conclusion in Sect. 6.

2 Related Work

Fall detection is reliant on sensors such as inertia sensors and pressure sensors. A classification of the sensor data into stable states and unstable states is a common approach used in several works [17,18]. Such approaches are very robust but detect falls rather late. A related approach by Höhn et al. uses pattern recognition on the sensor data to detect specific patterns only present during a fall [8]. Other approaches use a model to predict the expected sensor values. The deviation between the expected sensor values and the real sensor values is used for fall detection. Renner and Behnke model the expected sensor data using sinusoidal functions [15]. In order to detect falls during layered motions, Tay *et al.* [20] use interpolated sensor models based on body angles. Also machine learning can be utilized for training a sensor model [10]. Latest approaches use two-level classification neural networks in combination with regression to achieve high levels of accuracy in fall detection and posture monitoring [1].

A general concept to evaluate the stability of humanoid robots is the Zero-Moment-Point (ZMP) [21]. The ZMP is widely used to stabilize humanoid walking [7] and kicking [22]. However, Renner and Behnke [15] as well as Höhn et al. [8] evaluated the ZMP as not being suited for fall detection. Also, the Foot-Rotation Indicator (FRI) [6], was not applied to fall detection yet.

A common approach for the fall prevention is the integration into the generation of the walking motion [2,7]. Dynamic adaptations of the gait phases leading to different step sizes and step durations are able to compensate small impacts and irregularities such as an uneven floor. An extension to this approach is the calculation of a specific point for the foot placement. The Capture Point (CP) [14] and the Foot-Placement-Estimator (FPE) [23] both rely on this approach. Most of these approaches use an inverted pendulum model as a basis. Zhao et al. [27] argue that this model does not model important dynamics of the hip and the ankle as well as the ground reaction force. They developed a hip-ankle and bent-knee strategy to deal with instabilities [11]. A combination of both strategies with machine-learned weights is used by Yi et al. [25].

Mao et al. argue that the Capture Point approach is not feasible for practical use. They developed a continuous step controller [13] that counters the instability with a series of steps. In order to deal with uneven ground, Lee and Goswami [12] developed a moment-based controller that uses the ground reaction force and the center of pressure minimizing the torque on the foot. An extension is the reactive stepping controller [26] modeling the robot as a turning wheel. In contrast to the CP and FPE, a specific target point for the foot can be calculated and it is not necessary to continuously update the target point to a position further away from the origin. This is especially helpful performing a controlled motion on a pre-calculated trajectory.

3 Approach

In order to prevent a humanoid robot from falling, a fall detection method is required, capable of robustly detecting a fall as early as possible. This is necessary to have enough time left to perform a counter-motion. All related work uses some kind of classification of sensor data in order to detect specific patterns only present during a fall. These approaches only detect a fall when it is already in progress. Analyzing a falling motion of a humanoid robot, multiple stages can be identified. An event that brings the robot into instability accelerates the upper body of the robot. The stand space (convex hull of the foot/feet on the ground) generates a counteracting force reducing the acceleration of the upper body. In this first stage, it is not clear whether the robot will only stagger or fall down. If the upper body acceleration is too strong, the robot will fall. Otherwise it will only stagger, which should not be detected as a fall. Therefore, all other approaches do not classify this stage as falling, because it is not clear yet whether a fall will happen or not.

To detect a fall as early as possible, it is already necessary to evaluate whether a fall will happen later on in this first stage. In order to make a statement whether the measured sensor data will lead to a fall in the future, it is necessary to calculate how they will effect the robot. This task is done using a physical model

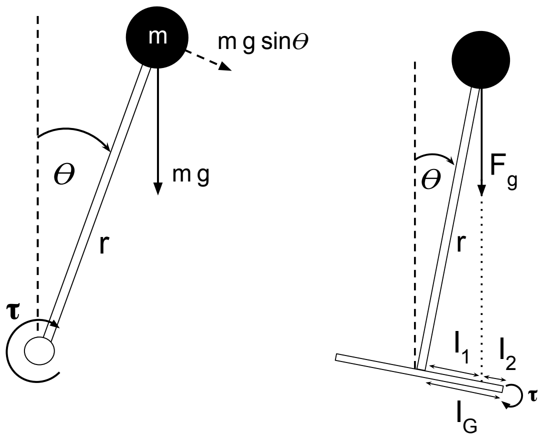

Fig. 2: Schematics for the inverted pendulum model (left) and the extended Stand Space Inverted Pendulum Model (right).

of the robot applying the measured accelerations and velocities. The Linear Inverted Pendulum model (LIP) models the movement of the robot's CoM as a 3-dimensional linear inverted pendulum [9], see Fig. 2. An inverted pendulum is suited to describe a falling robot as falling is a rotational motion around the edge of the foot. However, the simplified LIP model is not suited to model the robot during a fall for two reasons: First, the stand space of the foot, which generates a counteracting force to the fall, is not considered. In addition, all forces generated by the joints of the robot are also not integrated into this model. Second, the non-linear pendulum equations cannot be calculated directly. Therefore, they are linearized in this model. Linearization is done by the assumption that either $sin(\alpha) \approx \alpha$ for small values of α or the height of the pendulum mass from the ground is constant [7]. When generating upright motions such as walking, these assumptions can be made. In the case of modeling a fall however, they are not valid. As linearization is not an option and precise modeling is necessary, we develop an iterative model using a numerical approximation of the non-linear differential equation with the Euler integration.

3.1 Stand Space Inverted Pendulum Model (SSIP)

In order to design an iterative inverted pendulum model, we develop a function for the angular acceleration $\ddot{\theta} \in \mathbb{R}^3$ of the CoM. Such a function can be defined using the torque $\tau \in \mathbb{R}^3$ acting on the base of the pendulum. It uses the vector $r \in \mathbb{R}^3$ of the lever arm of force, the linear force $F \in \mathbb{R}^3$ produced by the mass

m, and the gravity $g \in \mathbb{R}^3$ as well as the inertia tensor $I \in \mathbb{R}^{3x3}$.

$$\tau = r \times F \tag{1}$$

$$\tau = I\,\ddot{\theta} \tag{2}$$

$$I\,\ddot{\theta} = r \times (m\ g\ sin(\theta)) \tag{3}$$

$$\ddot{\theta} = (r \times (m\ g\ sin(\theta))) \cdot I^{-1} \tag{4}$$

Only the tangential part of the gravity is acting to rotate the mass, which is dependent on the deflection $\theta \in \mathbb{R}^3$ of the pendulum, see Fig. 2. An iterative term for the angular velocity $\dot{\theta} \in \mathbb{R}^3$ and the deflection can be defined using Euler integration.

$$\dot{\theta}_{n+1} = \dot{\theta}_n + \triangle t \cdot (r \times (m\ g\ sin(\theta))) \cdot I^{-1} \tag{5}$$

$$\theta_{n+1} = \theta_n + \triangle t \cdot \dot{\theta}_{n+1} \tag{6}$$

Iterating Equations 5 and 6 with a small $\triangle t$ give a precise approximation of the rotational motion of the pendulum. However, this model cannot assert whether the robot will fall anyway. Therefore, we integrate the stand space of the robot given by its feet. The pendulum has a plate attached to the bottom and the point of rotation moves to the edge of this plate, see Fig. 2. The plate generates a stabilizing torque counteracting the tangential part of the gravity which is responsible for the angular motion of the pendulum and the respective fall of the robot. We can define this counteracting torque τ_{stand} using the vector from the origin of the foot to the point of rotation on the edge of the foot $l \in \mathbb{R}^3$ and the gravity. But the lever arm of force l is dependent on the deflection of the pendulum.

$$\tau_{stand} = (l_G - l_1) \times g \tag{7}$$

$$\tau_{net} = \sum_i \tau_i \tag{8}$$

$$\ddot{\theta} = (\tau_{fall} + \tau_{stand}) \cdot I^{-1} \tag{9}$$

The length between the projection of the gravitational force into the stand space and the edge of the stand space defines the acting part of the the stabilizing torque's lever arm of force, l_2. Using the length r and the deflection of the pendulum, we can calculate the inactive part l_1. By subtracting l_1 from the whole length of the stand space in the direction of the fall, we can obtain the acting part l_2, see Eq. 7.

Utilizing the fact that the net torque can be defined as the sum of all individual torques acting around an axis, see Eq. 8, we can sum up the stabilizing torque and the one responsible for the fall and integrate them into Eq. 9.

This extended model could already give an indication whether the robot will fall, when initialized with the measured sensor data and then iterated. But the robot might perform a motion while getting into instability. Such motion generates an additional torque that can amplify or counteract the falling motion.

Therefore the torque generated by the motion of all joints of the robot in respect to the rotation point at the edge of the foot has to be calculated.

The inverse rigid body dynamics [5] describes the problem of calculating the torques based on the kinematics (motion) of a body and the body's inertial properties. A model of the robot's rigid body given by the mass and moment of inertia of all body parts and how they are connected is needed. In addition with the generalized angular positions (q), velocities ($\dot{q}$), and accelerations ($\ddot{q}$) of all joints, the inverse dynamics can be defined as $\tau = ID(model,\, q,\, \dot{q},\, \ddot{q})$. In order to calculate the inverse dynamics, the recursive Newton-Euler algorithm is used in this work.

$$\ddot{\theta} = (\tau_{fall} + \tau_{stand} + \sum_{i \in N} \tau_i) \cdot I^{-1} \tag{10}$$

$$\dot{\theta}_{n+1} = \dot{\theta}_n + \triangle t \cdot (\tau_{fall} + \tau_{stand} + \sum_{i \in N} \tau_i) \cdot I^{-1} \tag{11}$$

The obtained torques for each joint i around the axis orthogonal to the fall direction can be summed up and added to Eq. 9, resulting in Eq. 10. The sum of torques around the fall axis can be used to calculate the complete rotational acceleration acting on the robot. This can be used to form the final iterative model (Eq. 11) using Eq. 5 and 6.

Initializing this model with the measured angle and angular velocity of the robot and iterating it will either provide an upright stabilizing pendulum motion or a rotational pendulum motion. We can decide whether the robot will fall directly based on this simulation. When the Stand Space Inverted Pendulum model will fall, also the robot will fall down approximately following the simulated motion.

3.2 Fall Prevention

In order to prevent the robot from falling, one or more steps are necessary [19]. Orienting towards a human model, a dedicated long-range step is most promising. Performing such a step requires one foot to keep contact to the ground, and one to perform the stepping motion. Deciding which foot performs which task is dependent on the direction of the fall and the current position of the feet. If one foot is already in mid-air, it is used to perform the step. If both feet are on or close to the ground, the foot in the direction of the fall has to keep contact with the ground as the point of rotation is at its edge.

Such a step should meet two requirements in order to prevent the fall. The rotational acceleration as well as the rotational velocity have to be compensated to stop the rotational motion of the fall. Regarding the first requirement, we can refer to the inverted pendulum model, see Fig. 2. The rotational acceleration is caused by the tangential part of gravity, which depends on the deflection of the pendulum. It is no longer accelerated if the base is positioned directly under the CoM. This can be achieved by placing the stepping foot directly under the CoM, which will then become the base of the pendulum.

Calculating the target position for the foot can be done using the motion of the robot's CoM over time calculated by the model. For convenience, we can define the iterative SSIP model as a closed function $\theta_t = SSIP(\theta_0, \dot{\theta_0}, t)$. This can be used to calculate the position of the CoM $\mathcal{P}$ at time t using the Rodrigues formula $Rot()$ [3] and the CoM in upright position CoM_0, see Eq. 12.

$$\mathcal{P}(t) = Rot(SSIP(\theta_0, \dot{\theta_0}, t)) \cdot CoM_0 \tag{12}$$

$$\mathcal{T}(t) = T_{POR \rightarrow Robot} \cdot \begin{bmatrix} \mathcal{P}(t)_x \\ \mathcal{P}(t)_y \\ footheight \end{bmatrix} \tag{13}$$

Equation 12 gives us the position of the CoM at time t with respect to the point of rotation (POR). As we know the point of rotation at the edge of the foot, we can transform the function into the robot coordinate system in order to actuate the joints using inverse kinematics [17]. As the foot should be positioned on the ground with its sole, the height of the last joint from the sole $footheight$ is used as the z-coordinate.

In addition to the rotational acceleration, the rotational velocity has to be compensated as well. We could actuate a joint in the opposite direction using the acceleration to compensate the velocity. But as motors often cannot be actuated precisely, this could cause another instability. Therefore we propose a different strategy: If the CoM has a deflection in the opposite direction of the fall when the stepping foot makes contact with the ground, the acceleration caused by the deflected pendulum and the rotational velocity of the fall can cancel each other out. This leaves the question of how much the pendulum has to be deflected in order to compensate the rotational velocity completely and coming to a stop in the upright position.

Making the simplifying assumption that the falling motion started in an almost upright position of the CoM, we already know which deflection is necessary to compensate the velocity. It is the current state of the pendulum in the opposing direction. Such a deflection could be achieved by doubling the distance of the target position of the foot from the POR. But when the stepping foot makes contact with the ground, its stand space will also compensate a part of the velocity. We do not have to calculate how much rotational velocity it will compensate, but rather how much we can reduce the deflection while still being in a stable state. As the projection of the gravitational force acting on the CoM has to be within the stand space, see Sect. 3.1 and Fig. 2, in order to still be stable, we can calculate the deflection using $tan^{-1}\frac{l_G}{r}$.

$$\mathcal{D}(t) = Rot(SSIP(\theta_0, \dot{\theta_0}, t) - tan^{-1}\frac{l_G}{r}) \cdot CoM_0 \tag{14}$$

$$\mathcal{T}(t) = T_{POR \rightarrow Robot} \cdot \begin{bmatrix} \mathcal{D}(t)_x + \mathcal{P}(t)_x \\ \mathcal{D}(t)_y + \mathcal{P}(t)_y \\ footheight \end{bmatrix} \tag{15}$$

Instead of doubling the distance between the POR and the target position, we can define a second function for the target position integrating the deflection

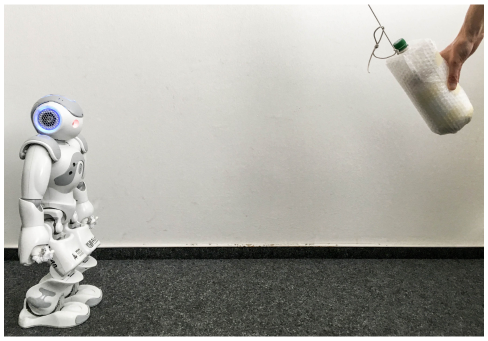

Fig. 3: Experimental setup

that will be compensated by the stand space, see Eq. 14. By adding up both target positions before transforming them into the robot's coordinate system, we can achieve a taget position for the foot that will compensate both rotational acceleration and velocity, see Eq. 15.

A trajectory for the stepping foot can be computed using the function $\mathcal{T}(t)$ from Eq. 15. Figure 1 visualizes the trajectory with the reachable poses in green and non-reachable parts in red. In order to keep the robot stable on the foot after the step, an additional closed-loop PID controller for the hip and knee joints is used. A controlled transition into the stand is used to bring the robot back into a standard operation state.

4 Evaluation

A series of experiments was performed to evaluate the method developed. The algorithms described were implemented for the humanoid robot NAO [16] using the B-Human code release [17]. All experiments used this robot and shared the same setup (cf. Fig. 3). A controlled impact to the upper body of the robot was generated using a padded weight of 800g attached to a cord of 1m, which was fixated above the robot. The weight was deflected to different angles to produce impacts with varying strength. For each deflection set, impacts from four sides (left, right, front, and back side of the robot) were tested with five repetitions each. For the evaluation of the fall detection, two types of experiments were performed: A general functionality test (a) evaluating the correct detection of the state the robot is in. The second experiment (b) compared a traditional sensor classification method used by the team B-Human [17] with the method

Table 1: Results of the fall detection functionality experiment (5 trials each).

impact	walking (trivial)				small, no fall				medium, no fall				strong, fall				very strong, fall			
impact direction	f	b	l	r	f	b	l	r	f	b	l	r	f	b	l	r	f	b	l	r
detection rate in %	100	100	100	100	100	100	100	100	0	20	60	40	100	100	100	100	100	100	100	100

Table 2: Results of the fall detection comparative experiment (5 trials each)

impact	45° (strong impact, fall)				60° (very strong impact, fall)			
impact direction	f	b	l	r	f	b	l	r
traditional sensor classification ($\o\pm\sigma$ in ms)	625.2 ± 11.1	729.2 ± 15.5	639.2 ± 8.1	636.2 ± 5.4	667.2 ± 8.7	355.0 ± 17.6	349.0 ± 12.3	338.3 ± 12.4
model-based fall detection ($\o\pm\sigma$ in ms)	153.0 ± 5.9	201.4 ± 7.0	226.8 ± 6.7	228.6 ± 4.9	100.8 ± 4.9	104.6 ± 4.7	126.0 ± 5.8	111.8 ± 5.7
mean difference in ms	472.2	527.8	412.4	408.0	266.4	248.4	223.0	226.5

presented in this paper. For the fall prevention, an additional functionality test (c) was done. The results of all experiments can be reviewed in Tab. 1–3. All experiments were performed with an impact of different strength indicated by the displacement of the pendulum and from the different directions front (f), back (b), left (l), and right (r).

For the functionality test of the fall detection (cf. Tab. 1), four different impacts with increasing strength (two led to staggering, two led to a fall) were tested in addition to the trivial case of normal walk without an impact. 80 passes were performed in total. In the cases of trivial walking and staggering due to small impacts, the state was correctly classified as stable. Also the cases that led to a fall due to strong and very strong impacts were correctly classified as a fall. Only in the case of strong staggering due to a medium impact, the detection rate dropped to a value between 60% and 0%.

The comparative experiment for the fall detection was performed with two types of impacts (45° and 60° weight deflection), both led to a fall. The time between the impact and the detection of a fall was measured. The results are presented in Tab. 2. For all eight types of impacts (four sides times two different strengths, 40 passes in total), the method presented in this paper was able to detect the fall significantly earlier (t-Test with $p < 0.001$). This resulted in earlier detection between 408ms to 527ms for the strong impacts and between 223ms to 266ms for very strong impacts.

The evaluation of the fall prevention was performed with three different strengths of impact (45°, 55° and 65° weight deflection). All impacts would have led to a fall without a counter-motion. Table 3 shows the results for all 60 passes of the experiment with the rate of successfully achieved fall preventions. For a weight deflection of 45°, the fall could be prevented for almost all passes (18 of 20). With increasing strength of impacts, the successful fall preventions dropped to 7 of 20 for the deflection of 55° and to 3 of 20 for the deflection of 65°. It is to note that impacts from the front, which led to a backwards fall of the robot, could be generally prevented much less than from the other sides whereas impacts from the sides could be prevented better.

Table 3: Results of the fall prevention functionality experiment (5 trials each)

impact	45°				55°				65°			
impact direction	f	b	l	r	f	b	l	r	f	b	l	r
fall prevention rate in %	60	100	100	100	0	20	60	60	0	0	40	20

5 Discussion

The approach developed in this paper uses a physical model of the humanoid robot in order to calculate the effects of sensor data towards the future motion of the robot and if this motion represents a fall. We argued that this approach should be able to detect a fall earlier than traditional classification of sensor data. The results of the comparative study (b) confirm this assumption, detecting a fall between a quarter to half a second earlier than the method used by the team B-Human in the RoboCup. It is to note that this is a comparison to a single specific approach and the findings cannot be generalized without further evaluation. However, Renner and Behnke [15] define the forewarning time as a criterion in their evaluation. The forewarning time is the duration between fall detection and the robot's body reaching 25° deviation from the upright posture. This deviation is also the criterion detecting forward and backward falling in the B-Human system. Therefore, the *mean difference* values from Tab. 2 are comparable to the forewarning times reported by Renner and Behnke. The times their method achieved is 500–600ms, but under the assumption that the robot is executing a regular walk movement, which is exploited by their approach. In addition, most of their attempts would not have brought the robot to a fall even without a fall prevention, so they were definitely not comparable to the *very strong impact* category here, where the advantage of the new approach over the old one is smaller.

The functionality evaluation (a) revealed that when the NAO is heavily staggering (medium impact) but not falling, this approach falsely detects a fall. These false positive detections most likely originate from deviations between the model and the NAO robot. Hardware limitations of the NAO such as imprecise inertia and joint angle sensors and strong joint play in the leg joints are likely to produce a deviation from the model. Furthermore, the NAO does only provide joint angle measurements, but not the joint velocities and accelerations needed to calculate the inverse dynamics used in our model. Joint velocities and accelerations are therefore calculated by a motor model [4] and only give approximated values. These reasons indicate why false-positive measurements are likely.

The results from the evaluation of the fall prevention (c) show a distinct drop in the rate of successful fall preventions between 45° and 55° weight deflection. In the first case almost all falls could be prevented whereas in the latter less than half of of all passes were successful. This can be explained by the fact that the leg motors of the NAO are too slow to reach the position necessary to prevent the fall. During the experiments, we observed that the leg which should be at least positioned under the CoM was not able to reach this position before the fall was no longer preventable. We argue that the rate of successful fall prevention could be improved for strong pushes using a hardware platform with superior motors for the leg joints. In general, pushes from the front could be prevented worse than others. This relates to the fact that the actuation limits of the NAO's legs are more restricted to the back than to the front. Overall, falls to the side could be prevented better, which affiliates to the moment of inertia of the robot. Rotation to the front and back are more accelerated than to the sides.

Concluding our findings, the presented approach is capable of detecting fall earlier than the method we compared to and prevent falls in limitation to the capabilities of the hardware. The main limiting factor is the hardware of the NAO that was used for the evaluation. In future work, we would like to generalize our findings using a superior hardware platform. In addition, we could model the robot in more detail if sufficient computation power is accessible to iterate the model, potentially further improving the results.

6 Conclusion

This paper presents a novel approach for fall detection and prevention for humanoid robots. We developed an iterative model based on the inverted pendulum and extended it with a stand space and the body dynamics of the robot. The model is used to simulate the effects of acting forces on the robot measured by sensors. If the simulated motion represents a fall, a counter-motion is performed. The trajectory for this motion is also based on the future states of the robot's CoM calculated by the model. We demonstrate that this method can detect a fall significantly earlier than traditional methods. The practical use is limited by the hardware of the robot, but this approach is promising for robots with accurate sensors and precise joints. With improved computational power, a more precise model can be used to potentially obtain even better results.

References

1. Abeyruwan, S.W., Sarkar, D., Sikder, F., Visser, U.: Semi-automatic extraction of training examples from sensor readings for fall detection and posture monitoring. IEEE Sensors Journal 16(13), 5406–5415 (July 2016)
2. Adiwahono, A.H., Chew, C.M., Huang, W., Dau, V.H.: Humanoid robot push recovery through walking phase modification. In: Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on. pp. 569–574. IEEE (2010)
3. Altmann, S.L.: Rotations, quaternions, and double groups (1986)
4. Böckmann, A., Laue, T.: Kick motions for the nao robot using dynamic movement primitives. In: RoboCup 2016: Robot Soccer World Cup XIX. Lecture Notes in Artificial Intelligence, Springer
5. Featherstone, R.: Rigid body dynamics algorithms, chap. Inverse Dynamics. Springer (2014)
6. Goswami, A.: Postural stability of biped robots and the foot-rotation indicator (FRI) point. The International Journal of Robotics Research 18(6), 523–533 (1999)
7. Graf, C., Röfer, T.: A center of mass observing 3D-LIPM gait for the RoboCup Standard Platform League humanoid. In: Röfer, T., Meyer, N.M., Savage, J., Saranli, U. (eds.) RoboCup 2011: Robot Soccer World Cup XV. Lecture Notes in Artificial Intelligence, vol. 7416, pp. 101–112. Springer (2012)
8. Höhn, O., Gačnik, J., Gerth, W.: Detection and classification of posture instabilities of bipedal robots. In: Climbing and walking robots, pp. 409–416. Springer (2006)
9. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Yokoi, K., Hirukawa, H.: A realtime pattern generator for biped walking. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002). pp. 31–37. Washington, D.C., USA (2002)

10. Kalyanakrishnan, S., Goswami, A.: Learning to predict humanoid fall. International Journal of Humanoid Robotics 8(02), 245–273 (2011)
11. Kiemel, S.: Balance maintenance of a humanoid robot using the hip-ankle strategy. Ph.D. thesis, TU Delft, Delft University of Technology (2012)
12. Lee, S.H., Goswami, A.: Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. pp. 3157–3162. IEEE (2010)
13. Mao, W., Kim, J.J., Lee, J.J.: Continuous steps toward humanoid push recovery. In: Automation and Logistics, 2009. ICAL'09. IEEE International Conference on. pp. 7–12. IEEE (2009)
14. Pratt, J., Carff, J., Drakunov, S., Goswami, A.: Capture point: A step toward humanoid push recovery. In: Humanoid Robots, 2006 6th IEEE-RAS International Conference on. pp. 200–207. IEEE (2006)
15. Renner, R., Behnke, S.: Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes. In: Intelligent robots and systems, 2006 IEEE/RSJ international conference on. pp. 2967–2973. IEEE (2006)
16. Robotics, A.: NAO Documentation (2016), `http://doc.aldebaran.com`, consulted online on June 7, 2017
17. Röfer, T., Laue, T., Stöwing, A., Thielke, F., Kuball, J., Lübken, A., Maaß, F., Müller, J., Post, L., Richter-Klug, J., Schulz, P., Stolpmann, A.: B-human team report and code release 2016 (2016), only available online: `https://github.com/bhuman/BHumanCodeRelease`
18. Ruiz-del Solar, J., Moya, J., Parra-Tsunekawa, I.: Fall detection and management in biped humanoid robots. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. pp. 3323–3328. IEEE (2010)
19. Stephens, B.: Push recovery control for force-controlled humanoid robots. Ph.D. thesis, Carnegie Mellon University Pittsburgh, Pennsylvania USA (2011)
20. Tay, J., Chen, I.M., Veloso, M.: Fall Prediction for New Sequences of Motions, pp. 849–864. Springer International Publishing, Cham (2016)
21. Vukobratović, M., Borovac, B.: Zero-moment point—thirty five years of its life. International Journal of Humanoid Robotics 1(01), 157–173 (2004)
22. Wenk, F., Röfer, T.: Online generated kick motions for the nao balanced using inverse dynamics. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013: Robot Soccer World Cup XVII. Lecture Notes in Artificial Intelligence, vol. 8371, pp. 25–36. Springer (2014)
23. Wight, D.L., Kubica, E.G., Wang, D.W.: Introduction of the foot placement estimator: A dynamic measure of balance for bipedal robotics. Journal of computational and nonlinear dynamics 3(1), 011009 (2008)
24. Winter, D.A.: Human balance and posture control during standing and walking. Gait & posture 3(4), 193–214 (1995)
25. Yi, S.J., Zhang, B.T., Hong, D., Lee, D.D.: Learning full body push recovery control for small humanoid robots. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. pp. 2047–2052. IEEE (2011)
26. Yun, S.k., Goswami, A.: Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. pp. 3943–3950. IEEE (2011)
27. Zhao, J., Schutz, S., Berns, K.: Biologically motivated push recovery strategies for a 3d bipedal robot walking in complex environments. In: Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on. pp. 1258–1263. IEEE (2013)