Step Adjustment for a Robust Humanoid Walk

Philip Reichenberg[1] and Thomas Röfer[2,1]

[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
{s_ksfo6n,roefer}@uni-bremen.de

Abstract. A stable and fast walk for humanoid robots is a challenging problem. In particular for multiple robots with different degrees of wear, it is necessary to tune several control parameters, which consumes a lot of time and wears down the hardware even more. In this paper, we present the latest improvements of our current walk, which enable a significant boost in the walking speed and stability, basically independent of the robot's hardware state. Based on a linear inverted pendulum model and thresholds determined statistically, the feet positions are adapted to follow the center of mass' direction of movement and let the robot walk in the direction of a potential fall if necessary. In addition, we ensure that the movement of the swing foot is parallel to the ground with a closed-loop feedback controller over the measured rotation errors of the support foot, without the direct use of the inertial measurement unit. The approaches presented allow for a up to 40% faster walk on different robots of the same type, without the need for a manual calibration.

1 Introduction

In the RoboCup Standard Platform League (SPL), all teams use the same humanoid robot, the NAO manufactured by SoftBank Robotics (currently, most teams use the NAO[6]). Although a number of different walks were developed in the past [2,12,4,9,14,8], in actual competitions, the current walks [13,11,6,3,10,1] all reach similar speeds. In addition, even after 12 years of development, the robots still fall down quite frequently in actual games. Falls often result from colliding with other robots while fighting for the ball, but often enough they also happen when a robot is simply walking from one point to another.

The NAO is equipped with 26 joints that are driven by 25 motors. Each leg contains six joints of which the only one that allows a rotation around the vertical axis is mechanically linked to the same joint in the other leg. All joints are position-controlled by motor controllers that are not accessible by the user. The NAO is also equipped with four pressure sensors under each of its feet and an inertial measurement unit (IMU) in its chest. The robot operates at a cycle time of 12 ms (i. e. 83.33 Hz). However, there is a delay of 3–4 cycles between sending joint commands and measuring the effects of these commands. This control delay

makes the development of a fast and stable walk difficult. In addition, the joints typically also have 1°–3° of play, depending on the wear of the robot hardware.

Since 2017 in the SPL, robots play on artificial grass of 8 mm height, which many walks developed before could not handle. Many teams switched to the walk developed by Hengst [3], because of its high stability under such conditions. The walk, which is often called the *rUNSWift* walk, has four core properties: Firstly, it creates fixed trajectories for both the swing foot (lifted up in the walking direction) and the support foot (on the ground against the walking direction) based on the step size required to walk at the requested speed at the begin of each step. The trajectories are generated in Cartesian space, relative to the robot's torso (i. e. the robot's coordinate system), and converted to the joint space during the execution using inverse kinematics ([10]). Although the trajectories of the support foot and the swing foot are not perfect mirrors of each other (because the first one is linear, the second one parabolic), the torso-relative center of mass (COM) of the robot basically stays above the middle between both feet all the time. Secondly, the walk uses a natural swinging motion to shift the weight between the two legs, i. e. it does not actively move the hip sideways, but simply lets gravity do its work. Thirdly, it uses the pressure sensors under the feet to determine when the weight shifts from one leg to the other and starts the next step in that moment. Fourthly, it uses the measurements of the sagittal gyroscope to press the support foot against the fall direction by increasing the ankle pitch angle proportional to the rotational speed of the torso.

The works in this paper are based on the version of that walk that was adapted by the team B-Human [10]. The paper is organized as follows. In section 2, we discuss some of the related work. In section 3, we present the first approach for a more robust walk, the step adjustment. It uses a tolerance range that is relative to an actual foot support area, the calibration of which is shown in section 4. The second approach for a more stable walk is described in section 5, which is keeping the swing foot parallel to the ground. Both approaches are evaluated in section 6. Finally, we conclude the paper in section 7.

2 Related Work

Tilgner *et al.* [13] use a similar approach as Hengst [3]. However, in their work, the support foot switch is predicted to compensate for the delay of 48 ms. In addition, the steps are adjusted by a stability controller based on the error in the body tilt. For example, if the robot is tilting backwards the step duration is increased and the forward velocity reduced.

In the approach by Schwarz *et al.* [11], a step size is calculated based on the requested walking speed. With the resulting foot positions, a trajectory for the zero moment point (ZMP) is determined. With a flexible linear inverted pendulum (FLIP) [15], a COM trajectory is calculated, which shall follow the ZMP one. The FLIP is extended with a second COM, which in theory models the flexibility of the limbs and pulls on the first COM. In addition, several PID controllers are used to compensate various errors. The COM is shifted based on

body angle errors. The hip and ankle joints are adjusted based on gyroscope and body angle errors. The legs are rotated based on body angle errors to keep the feet parallel to the ground.

Mellmann *et al.* [6] calculate a sequence of ZMPs based on a requested walking direction, one for each execution cycle of the following step. Afterwards with a linear inverted pendulum model [5] (LIPM), the COM trajectory is determined and with that the 3D trajectory of the feet and the corresponding joint angles for each motion cycle. For a higher stability, a PD controller is used in the step target generation based on the COM. In addition, if the average COM error exceeds a threshold, the robot must come to a full stop as long as the error does not drop below a certain threshold. During the step execution, three stabilization mechanisms are applied. The height and the roll angle of the legs are adjusting in the moment a foot is lifted. Another controller balances the body to keep it upright. Thirdly, the ankles are adjusted based on the body orientation.

Similar to the walk used in the paper, the one developed by Missura *et al.* [7] first determines the foot target positions for each step and then interpolates between the current positions and the target ones. With a LIPM, those foot target positions are modified instantly when responding to a disturbance. As a result, the robot starts to walk backwards when pushed from the front and is walking forward when pushed from the back. The difference to the approach presented in this paper is that we do not calculate new target positions for the feet but just let the swing foot follow the COM without the use of the ZMP and move the support foot against the swing foot's direction.

A general approach to reduce the risk of falling when fighting for the ball is to move the arms to the back of the robot and shifting or tilting the upper body forward. This reduces the overall footprint of the robot and reduces the likelihood of colliding with another robot. Such an approach is used by most teams. Basler *et al.* [1] are tilting the upper body even more to the front in a tackling situation, which makes the robot more stable, as disturbances, such as collisions with other robots, from the front are less harmful.

3 Step Adjustment

Although the two stability mechanism of the *rUNSWift* walk, i. e. detecting the change of the support foot and gyro-based balancing, can counter a certain amount of disturbances while walking, there are situations which require to change the steps itself to prevent falls. For instance, the gyro-based balancing has its limits, in particular for preventing falls to the back, because the soles of the feet extend less to the back than to the front and therefore, they cannot exert that much force against the ground to keep the body upright. In addition, the available torque of the motors is limited, so even some swings to the front cannot be compensated although the foot might be long enough to do so. Therefore, the goal of the step adjustment is to modify the step trajectories themselves instead of just pitching the ankle joints. However, these adjustments are only applied when the robot reaches a state the original walk cannot cope with.

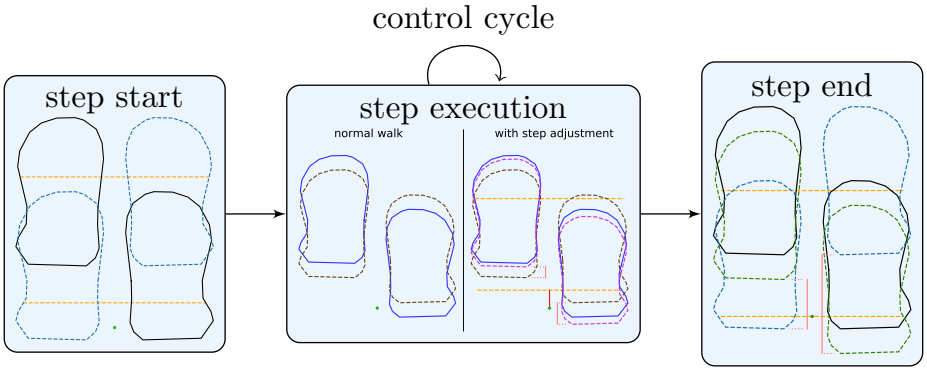

Fig. 1. The step adjustment. The planned step (cyan) is to swing the right foot forward, while the left (support) foot is moved backward. The projected COM (green) is outside the tolerance range (yellow) by an, in this case negative, offset δ (red). In each control cycle, the feet (dark blue) are moved closer to their planned end positions (brown). Meanwhile, the step adjustment adds δ to the position of the swing foot and thereby moves the foot slowly backwards in the direction of the COM, and half of the offset is subtracted from the position of the support foot (violet). At the end of the step, the robot effectively walked backwards for a small step (dark green). At the start of the step, the tolerance range is shown relative to the initial foot positions (black). At the end of the step, it is shown relative to the new foot positions (dark green).

For example in a situation in which the robot is walking forward and its body starts tilting backwards, the torso does not seem to keep up with the legs. Therefore, the step size is reduced (up to a full backward step), which allows the upper body to "catch up". This is achieved with the approach that is shown in Figure 1. The COM of the robot is projected onto the plane of the feet (m). The robot is considered to be stable if the projected COM is located inside a tolerance range ($T_{\min}$, $T_{\max}$) that is spanned by fixed regions on both feet. If the COM leaves this area, the swing foot is moved in that direction by the amount that the COM left the area (δ) and the support foot is moved in the opposite direction by half that amount, until the COM is once again inside the tolerance range. This adjustment is only applied in the sagittal direction and the adjustment size is clipped (by $p_{\max}$), to ensure that the feet do not move with too much speed. Also half of δ is applied on the support foot, because in test trials applying nothing or the full value on the support foot resulted in a significantly less stable walk, while the half showed a significant boost in the stabilization. This is due the fact that the full value results in an overcompensation. Not adjusting the position of the support foot results in both feet being placed too far backward or forward. Therefore, it would not adapt the walking direction based on the fall direction. The projected COM is lowpass-filtered to avoid any jerky movements due to measurement noise. As shown in Figure 1, this can lead to a step in the opposite direction of the current walking direction. In this case, the next step will be very small, because the swing foot is already in front and the support foot is in the

Algorithm 1: Step Adjustment

Input: $w_t, w_{t-1}, \bar{w}_{t-1}, u_t, m, p_{\max}, T_{\min}, T_{\max}$
Output: $\bar{w}_t, \bar{u}_t$

$\Delta w_{t-1} = \bar{w}_{t-1} - w_{t-1}$
$\Delta \hat{w}_{t-1} = \max(-p_{\max}, \min(p_{\max}, -\Delta w_{t-1}))$
$\Delta w_t = w_t - w_{t-1}$
$\hat{w}_t = \bar{w}_{t-1} + \Delta w_t$
$\Delta_{min} = \min(\hat{w}_t + \Delta \hat{w}_{t-1}, u_t - \frac{\Delta w_{t-1} + \Delta \hat{w}_{t-1}}{2}) + T_{\min}$
$\Delta_{max} = \max(\hat{w}_t + \Delta \hat{w}_{t-1}, u_t - \frac{\Delta w_{t-1} + \Delta \hat{w}_{t-1}}{2}) + T_{\max}$
$\delta = \min(m - \Delta_{\min}, 0) + \max(m - \Delta_{max}, 0)$
$\hat{p}_{\min} = \min(0, -p_{\max} - \Delta w_t) - \max(\Delta \hat{w}_{t-1}, 0)$
$\hat{p}_{\max} = \max(0, p_{\max} - \Delta w_t) - \min(\Delta \hat{w}_{t-1}, 0)$
$\bar{w}_t = \hat{w}_t + \max(\hat{p}_{\min}, \min(\hat{p}_{\max}, \delta))$
$\bar{u}_t = u_t - \frac{(\bar{w}_t - w_t)}{2}$
return $\bar{w}_t, \bar{u}_t$

back. In combination, this results in two steps on the spot (or even more) that stabilize the robot, after which it can continue to walk normally.

The step adjustment algorithm is shown in algorithm 1. All variables are in the robot's system of coordinates. w and u describe the forward components of the swing foot's and support foot's position, respectively, as computed by the original walk. Variables with a hat represent temporary variables. Variables with a bar describe the adjusted forward components of both feet. t addresses the current control frame, $t-1$ the previous one. The tolerance range was determined beforehand by statistically analyzing the projected COM of one of our robots from the log files of a competition game, with the calibration that will be discussed in section 4 applied. Both values are distances relative to the feet sole origins in the sagittal direction. Since the relative positions of the feet towards each other always change while walking, we distinguish between three different regions in the joint area of both soles of the feet that are separated by the origins (i. e. the points below the ankles) of both feet (cf. Figure 2). The COM positions in the middle region are considered to be stable. Therefore, this region is ignored in the analysis. The two remaining regions together always extent the length of a single foot, independent of the relative positions of the two feet. The heatmap of the COM positions is shown in Figure 4. For the tolerance range, using the range $[-2\sigma \dots 2\sigma]$ seems to give good results.

To compensate for the sensor delay of up to 48 ms (i. e. four control cycles), the behavior of the COM is predicted using a LIPM. Thus, the relation between the COM and tolerance range is estimated based on the commanded joint angles and the estimated current position of the COM, reducing the reaction time significantly. In addition, we apply an automatic per-robot calibration procedure for the support area of the feet to compensate for variances between different robots. This is described in the next section.

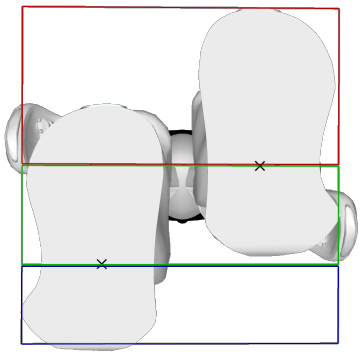

Fig. 2. The three supporting areas. The blue area is defined from the heel of the furthest backward foot to its origin point. The green one is defined between the origin points of both feet. If they are parallel to each other then the green area does not exist. The red area is defined from the origin of the furthest forward foot to its tip.

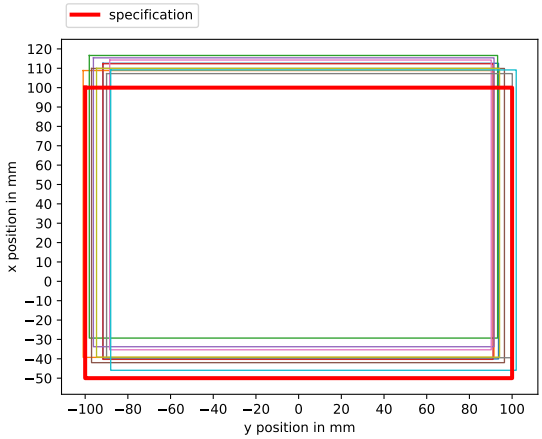

Fig. 3. The calibrated support rectangles, relative to the origin of both feet, of several NAO[6] robots. The support polygon based on the specification of the robot is shown in red.

4 Foot Support Rectangle Calibration

The NAO is equipped with joint position sensors and an IMU. Both are used to determine the COM position projected to the foot area. The joints can have play and IMUs in different robots might return slightly different results. This can result in deviations in the computation of the projected COM in relation to the tolerance range between different robots. Instead of calibrating each joint and calibrating the IMU, we use an abstraction that is sufficient for walking: the support rectangle. This is the rectangle in the foot plane that spans all the projected COM values of a standing robot that do not result in a fall. To determine the support rectangle, the robot executes a semi-automatic calibration procedure, in which it determines, how far it can shift its torso in all four directions while standing normally on both feet, before it topples over. The beginning of the fall is detected using the gyroscopes. At this moment, the COM projected onto the foot plane defines a border of the support rectangle of both feet. The robot is caught manually to prevent it from actually falling down. The calibrated support rectangle is used to interpret the boundaries of the tolerance range (which are given as percentages of the foot length) as described in section 3. In addition, they are used as tipping edges for the LIPM used to predict the current COM.

Figure 3 shows the calibrated foot areas of several NAO[6] robots, in comparison to the foot area resulting from the hardware specification of the robot.

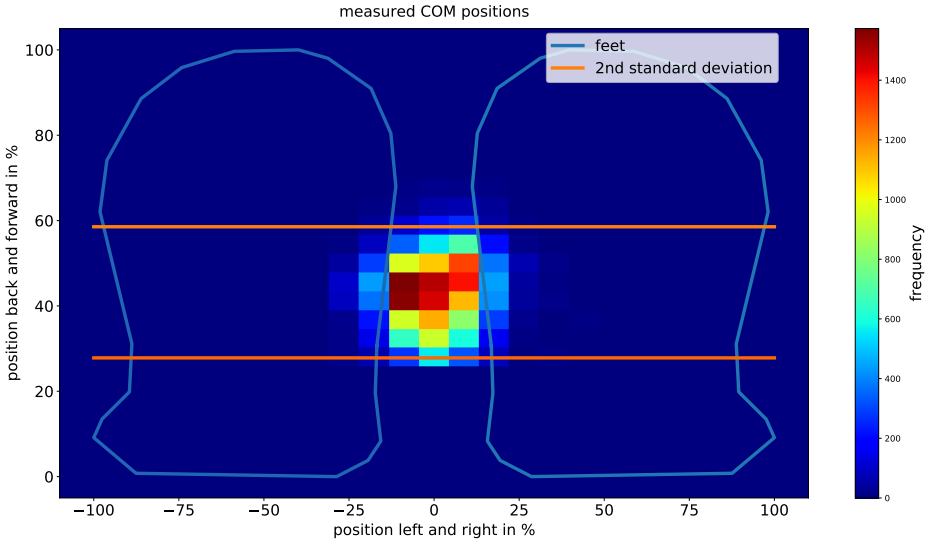

Fig. 4. The COM positions relative to the foot area during a of a robot during a half of a competition game. The actual dimensions of the feet are abstracted to percentages. The positions of two times the standard deviation σ are marked at 27.8% and 58.6%. The average is located at 43.2%.

There seems to be a shift of between one and two cm in the forward and backward direction and a small shift to the sides.

To get a fully automatic calibration procedure without the need of a person to catch the robot, the robot can approximate the forward and backward edges while it walks. From a calibrated robot we can measure the average COM position in the foot plane (see Figure 4). As long as the robots do not walk too fast, they can walk stable without much deviation in the torso tilt. Therefore, a non-calibrated robot can measure its average COM position for a short period of time while walking slowly. Afterwards, the difference of the desired and measured average is used to shift the calibrated forward and backward foot support edges. This procedure is currently used during the preparation for remote soccer games. A robot automatically walks to certain positions on the field to perform an automatic extrinsic camera calibration. While walking, also its foot support rectangle is calibrated.

5 Swing Foot Leveling

A problem of the NAO robots is the limited strength of the joint motors. A typical observation when a robot falls to the front without colliding with another robot is shown in Figure 5. Here the robot is walking forward, with the right foot being the supporting one and the left foot being the swing foot. The swing foot is moving parallel to the theoretical ground (red line), as if the robot would have

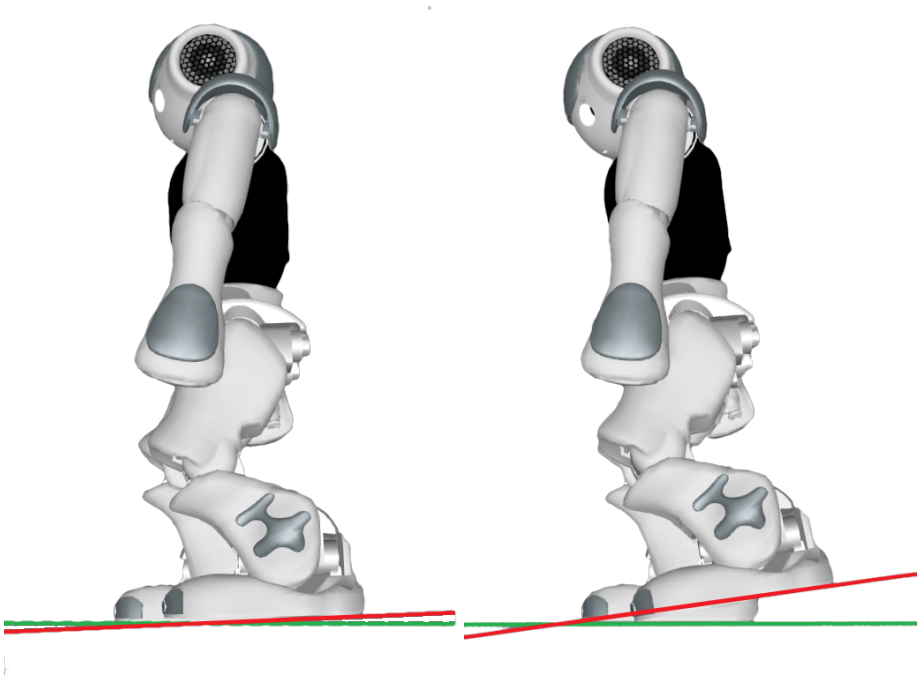

Fig. 5. The feet rotations of a robot while walking. Left: the feet under normal circumstances. Right: The robot's body swings quickly forward and is already tilted a lot. The right foot is the support foot, the left foot the swing foot. The ground is shown in green. The movement direction of the swing foot is shown in red.

a perfectly straight upper body. The supporting foot is assumed to be parallel to the real ground (green line) and has full ground contact. Therefore the rotation of the supporting foot equals the rotation of the ground. To keep the robot upright, the supporting foot must also move parallel to the theoretical ground, i. e. both feet should be parallel to each other. But in situations in which the robot already starts to tilt forward, even with a small angular velocity, the joint motors are too weak to press against the robot's weight. As a result the hip joint of the supporting foot gets stuck, while the other joints continue their movement, because those are not pressing against the robot's weight. This tilts the robot even further forward and the swing foot will inevitably touch the ground earlier than expected. The tip of the swing foot moves really fast into the ground, which is then measured as a support foot switch. Afterwards, the new support foot, in this case the left foot, is really far backward while the robot is still tilted forward a lot. In most cases the robot will fall shortly afterwards in the same step phase.

Since hardware changes are not allowed in the SPL and we also do not have access to the parameters of the motor controllers, the problem can only be changed by adapting the trajectory of the swing foot. The goal is that the

swing foot should reach its intended target position as close as possible, because its support is required in that position to counter a possible fall and to bring the body back up to an upright state. If the foot touches the ground too early, the robot will stumble and topple over.

The adaptation of the swing foot trajectory works as follows: The deviation of the angle of the support foot from its intended pitch angle is determined from the measured joint angles using forward kinematics. Then this deviation is also added to the pitch angle of the swing foot. This keeps the swing foot level, although its trajectory is still tilted. This is intended, because the swing foot should be fully extended at the end of the step cycle. Leveling the foot simply prevents to hit the ground with the tip of the foot too early. However, actually keeping the swing foot level over the whole step cycle would simply transfer the unintended tilt of the robot's body into the next step cycle. Therefore, over the last 25% of the step phase, the swing foot pitch is linearly interpolated back to its normal value. This ensures that the swing foot will not collide with the ground too early but that it also has as little rotation as possible. The 25% are used due the fact that the original *rUNSWift* walk only accepts support foot switches after 75% of the step duration have passed.

The swing foot leveling is only applied if the robot's body is tilted forward by a relevant amount and there is some unintended support foot pitch to be compensated (transitions in and out are smoothed) to keep using the original walk most of the time.

6 Evaluation[1]

We evaluated our adjustments with a test setup (see Figure 6). We let several robots walk over the field with different configurations: The old walk (Old), the old walk with only the walk step adjustment (wsa), the old walk with only the swing foot leveling (sfl), and the new walk with both adjustments (New). Each configuration was tested eight times, distributed over four different NAO[6] robots. We counted how often the robots fell at each obstacle with the default forward walking speed of $250\frac{\text{mm}}{\text{s}}$ used by our team B-Human. The results are shown in Table 1.

All configurations except for the new walk showed similar result. With the first three configurations, the robots fell on average 2.5 times in each try, while the robots with the new walk fell less than once per try, which is only one third compared to the old walk.

We also compared the old and new walk with higher forward walking speeds of $300\frac{\text{mm}}{\text{s}}$ and $350\frac{\text{mm}}{\text{s}}$. Here we tested both configurations only four times each, distributed over two different Nao[6] robots. The results are shown in Table 2. Once again the new walk performed significantly better than the old one, with about 45.5% less falls. Also even with a 40% higher walking speed, the new walk fell about 33% less than the old walk with the default walking speed.

[1] A comparison between the old and the new walk is shown here: `https://www.youtube.com/watch?v=N_Q7qLDYqyY`

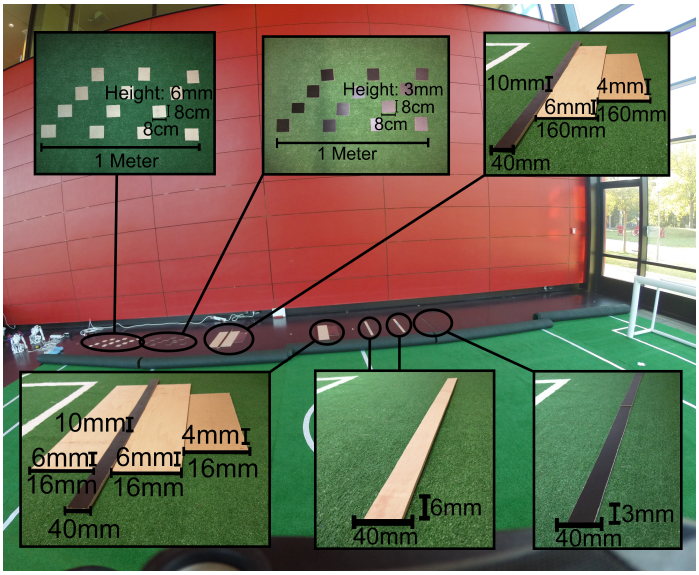

Fig. 6. The construction of the test setup. The robots walked from the right to the left, with the obstacles hidden under the field carpet.

However, there were two main problems with the new walk: On one hand the step adjustment sometimes started too late, while the robots were already falling backwards too fast. On the other hand, especially at the last obstacle, the step duration was a lot longer for a few walking steps. The walk has no balancing for the sideways swinging, therefore the robot can only keep walking until the sideways swinging reduced itself and the step duration becomes normal again. This resulted in a few falls, in which the robots fell diagonally or even sideways.

7 Conclusion and Future Work

In this paper, we presented an approach for stabilizing an existing walk for the NAO robot. It is based on two ideas: On the one hand, the positions of the feet are adapted to ensure that the COM stays above a safe area in the foot plane. On the other hand, the swing foot is kept parallel to the ground to avoid ramming the tip of the foot into the ground when the robot is tilted too far forward. Together, both approaches result in a faster and more stable walk. However, the evaluation also showed that the step adjustment sometimes adapts the foot positions too late and the support foot switch sometimes takes longer than expected, which brings the robot into a diagonal or sideways fall. Unfortunately, the approaches presented in this paper do not reduce the number of falls resulting from collisions with other robots. We are currently investigating how such falls could be prevented as well.

Table 1. The number of fallen robots for the different walk configurations, with the default forward walking speed of $250\frac{\text{mm}}{\text{s}}$. Each configuration was tested eight times.

obstacle configuration	Old	WSA	SFL	New	sum
first three meters	0	0	0	0	0
3 mm edge	0	0	0	0	0
first 6 mm edge	6	4	4	2	16
second 6 mm edge	5	4	4	1	14
first 1 cm ramp	3	4	5	0	12
second 1 cm ramp	3	5	3	0	11
3 mm wooden blocks	0	0	0	0	0
6 mm wooden blocks	4	1	4	4	13
sum	21	18	20	7	66

Table 2. The number of fallen robots, compared between the old and new walk with walking speeds of $300\frac{\text{mm}}{\text{s}}$ and $350\frac{\text{mm}}{\text{s}}$. Both configurations were tested four times.

obstacle configuration	old $300\frac{\text{mm}}{\text{s}}$	new $300\frac{\text{mm}}{\text{s}}$	old $350\frac{\text{mm}}{\text{s}}$	new $350\frac{\text{mm}}{\text{s}}$	sum
first three meters	0	0	0	0	0
3 mm edge	0	0	0	0	0
first 6 mm edge	0	1	2	2	5
second 6 mm edge	0	1	3	0	4
first 1 cm ramp	3	0	4	0	7
second 1 cm ramp	1	0	2	1	4
3 mm wooden blocks	2	0	0	1	3
6 mm wooden blocks	3	3	2	3	11
sum	9	5	13	7	34

Acknowledgements

We thank the team rUNSWift and in particular Bernhard Hengst for releasing the source code of their walking engine. We also thank the past and current members of the team B-Human for developing the software that is the basis for the work presented in this paper. Without them, this work would not have been possible.

References

1. Basler, J., Borchert, M., Essig, A., Gleske, P., Kahlefendt, C., Konda, Y., Kost, R., Nölle, K., Riebesel, N., Schmidt, M., Sharif, M., Wege, F., Wetters, B.: HULKs team research report 2019 (2019), `https://hulks.de/_files/TRR_2019.pdf`
2. Graf, C., Röfer, T.: A center of mass observing 3D-LIPM gait for the RoboCup Standard Platform League humanoid. In: Robot Soccer World Cup. pp. 102–113. Springer (2011)
3. Hengst, B.: rUNSWift Walk2014 report. Tech. rep., School of Computer Science & Engineering University of New South Wales, Sydney 2052, Australia (2014),

http://cgi.cse.unsw.edu.au/~robocup/2014ChampionTeamPaperReports/20140930-Bernhard.Hengst-Walk2014Report.pdf
4. Jahn, K.U., Borkmann, D., Reinhardt, T., Tilgner, R., Rexin, N., Seering, S.: Team research report 2009 (2009), http://robocup.imn.htwk-leipzig.de/documents/TRR_2009.pdf?lang=de
5. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on. vol. 2, pp. 1620–1626 vol.2 (Sept 2003)
6. Mellmann, H., Schlotter, B., Kaden, S., Strobel, P., Krause, T., Couque-Castelnovo, E., Ritter, C.N., Hubner, T., Tofangchi, S.: Berlin United - Nao Team Humboldt team report 2018 (2018), https://github.com/NaoDevils/CodeRelease/blob/master/TeamReport2019.pdf
7. Missura, M., Bennewitz, M., Behnke, S.: Capture steps: Robust walking for humanoid robots. International Journal of Humanoid Robotics 16(06), 1950032 (2019)
8. Qian, Y., He, Y., Han, Q.R., Poudel, S., Small, A., Lee, K.I., Lee, D.: The UPennalizers RoboCup 2015 Standard Platform League team description paper (2015), https://fling.seas.upenn.edu/~robocup/files/2015Report.pdf
9. Riebesel, N., Hasselbring, A., Peters, L., Poppinga, F.: Team research report 2016 (2016), https://hulks.de/_files/TRR_2016.pdf
10. Röfer, T., Laue, T., Baude, A., Blumenkamp, J., Felsch, G., Fiedler, J., Hasselbring, A., Haß, T., Oppermann, J., Reichenberg, P., Schrader, N., Weiß, D.: B-Human team report and code release 2019 (2019), https://github.com/bhuman/BHumanCodeRelease/raw/master/CodeRelease2019.pdf
11. Schwarz, I., Urbann, O., Larisch, A., Brämer, D.: Nao Devils team report 2019 (2019), https://github.com/NaoDevils/CodeRelease/blob/master/TeamReport2019.pdf
12. Tilgner, R., Reinhardt, T., Kalbitz, T., Seering, S., Wünsch, M., Schließer, J., Mewes, F., Haßler, S., Wissing, A., Jagla, T., Dawidowski, K., Göbe, M., Freick, P., Bischoff, S., Burke, T., Eckermann, S., Weiß, D.: Team research report Nao-Team HTWK (2018), http://robocup.imn.htwk-leipzig.de/documents/TRR_2017.pdf?lang=de
13. Tilgner, R., Reinhardt, T., Seering, S., Kalbitz, T., Eckermann, S., Wünsch, M., Mewes, F., Jagla, T., Bischoff, S., Gümpel, C., Jenkel, M., Kluge, A., Wieprich, T., Loos, F.: Team research report Nao-Team HTWK (2020), https://drive.google.com/file/d/13s28gGVsKkxd8ogoNfEf4Ic5Gyysi137/view
14. Tsogias, A.: A zero-moment point preview controlled gait for the NAO-robot (2016)
15. Urbann, O., Schwarz, I., Hofmann, M.: Flexible linear inverted pendulum model for cost-effective biped robots. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids). pp. 128–131. IEEE (2015)