# A Robust Closed-Loop Gait for the Standard Platform League Humanoid

Colin Graf [+1], Alexander Härtl [+2], Thomas Röfer [#3], Tim Laue [#4]

[+]*Fachbereich 3 – Mathematik und Informatik, Universität Bremen,*
*Postfach 330 440, 28334 Bremen, Germany*
[1]cgraf@informatik.uni-bremen.de, [2]allli@informatik.uni-bremen.de

[#]*DFKI Bremen, Safe and Secure Cognitive Systems*
*Enrique-Schmidt-Str. 5, 28359 Bremen, Germany*
[3]Thomas.Roefer@dfki.de [4]Tim.Laue@dfki.de,

***Abstract*— In this paper, we present a robust closed-loop gait for the Nao, the humanoid robot used in the RoboCup Standard Platform League. The active balancing used in the approach is based on the pose of the torso of the robot, the estimation of which we also describe. In addition, we present an analytical solution to the inverse kinematics of the Nao, solving the problems introduced by the special hip joint of the Nao.**

## I. Introduction

Since 2008, the humanoid robot *Nao* [1] that is manufactured by the French company Aldebaran Robotics is the robot used in the RoboCup Standard Platform League (cf. Fig. 1). The Nao has 21 degrees of freedom (cf. Fig. 2 left). It is equipped with a 500 MHz processor, two cameras, an inertial measuring unit, sonar sensors in its chest, and force-sensitive resistors under its feet.

Aldebaran Robotics provides a gait for the Nao [1] that is based on keeping the center of mass of the robot above the area supported by the feet. It is completely open-loop resulting in a low robustness on surfaces such as the carpet usually used in RoboCup. In addition, it only allows the user to choose between a pre-defined set of steps, i. e., it is not omni-directional. This makes it hard to be used in RoboCup, where precise and fast alignment behind the ball is a must. The maximum speed reached is approximately 10 cm/s. For

Fig. 1. Naos on a soccer field at RoboCup 2009 in Graz.

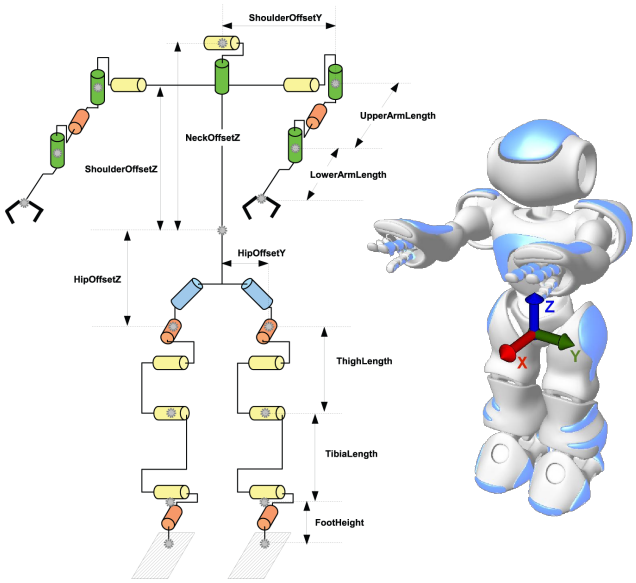


Fig. 2. The joints of the Nao [1] (left). The robot coordinate system used in this paper (right).

RoboCup 2008, Kulk and Welch designed an open-loop walk that keeps the stiffness of the joints as low as possible to both conserve energy and to increase the stability of the walk [2]. The gait reached 14 cm/s. However, since it is still based on the walking module provided by Aldebaran Robotics, it still shares the major drawback of not being omni-directional. Two groups worked on walks that keep the Zero Moment Point (ZMP) [3] above the support area using preview controllers. Both implement real omni-directional gaits. Czarnetzki *et al.* [4] reached speeds up to 20 cm/s with their approach. In their paper, this was only done in simulation. However, at RoboCup 2009 their robots reached similar speeds on the actual field, but they seemed to be hard to control and there was a certain lack in robustness, i. e., the robot fell down quite often. Strom *et. al* [5] modeled the robot as an inverted pendulum in their ZMP-based method. They reached speeds of around 10 cm/s.

The main contributions of this paper are: we describe a

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

CoM-based gait for the Nao that is supported by strong balancing methods resulting in a robust system, as proven in actual RoboCup games. We also describe an analytical solution to the inverse kinematics of the Nao. To our knowledge, it is the first one that was published so far, because in the work described above, only iterative approaches are mentioned. Finally, we provide some practical insights in how the pose of the robot's torso can be estimated using the readings from the inertial board and we compare them to the pose estimation provided by the inertia board itself.

The structure of this paper is as follows: first, we present the analytical solution to the inverse kinematics of the Nao. Then, our method for walking is presented, followed by our approach to active balancing, which makes the gait a closed-loop solution. Since balancing is based on the pose of the torso of the robot, the estimation pitch and roll of the torso is described afterwards. The paper closes with a short presentation of the results.

## II. Inverse Kinematics

Solving the inverse kinematics problem analytically for the Nao is not straightforward because of two special circumstances:

- The axes of the hip yaw joints are rotated by 45 degrees (cf. Fig. 2).
- These joints are also mechanically connected among both legs, i. e., they are driven by a single servo motor.

The target of the feet is given as homogenous transformation matrices, i. e., matrices containing the rotation and the translation of the foot in the coordinate system of the torso. To explain our solution we use the following convention: A transformation matrix that transforms a point $p_A$ given in coordinates of coordinate system $A$ to the same point $p_B$ in coordinate system $B$ is named $A2B$, so that $p_B = A2B \cdot p_A$. Hence the transformation matrix that describes the foot position relative to the torso is $Foot2Torso$ that is given as input. The coordinate frames used are depicted in Fig. 3.

The position is given relative to the torso, i. e., more specifically relative to the center point between the intersection points of the axes of the hip joints. So first of all the position relative to the hip is needed[1]. This is a simple translation along the $y$-axis[2]

$$Foot2Hip = Trans_y\left(\frac{l_{dist}}{2}\right) \cdot Foot2Torso \tag{1}$$

with $l_{dist}$ = distance between legs. Now the first problem is solved by describing the position in a coordinate system rotated by 45 degrees, so that the axes of the hip joints can be seen as orthogonal. This is achieved by a rotation around the $x$-axis of the hip by 45 degrees or $\frac{\pi}{4}$ radians.

$$Foot2HipOrth = Rot_x(\frac{\pi}{4}) \cdot Foot2Hip \tag{2}$$

[1]The computation is described for one leg. Of course, it can be applied to the other leg as well.

[2]The elementary homogenous transformation matrices for rotation and translation are noted as $Rot_{<axis>}(angle)$ resp. $Trans_{<axis>}(translation)$.

Because of the nature of the kinematic chain, this transformation is inverted. Then the translational part of the transformation is solely determined by the last three joints and hence they can be computed directly.

$$HipOrth2Foot = Foot2HipOrth^{-1} \tag{3}$$

The limbs of the leg and the knee form a triangle, in which an edge equals the length of the translation vector of $HipOrth2Foot$ ($l_{trans}$). Because all three edges of this triangle are known (the other two edges, the lengths of the limbs, are fix properties of the Nao) the angles of the triangle can be computed using the law of cosines (4). Knowing that the angle enclosed by the limbs corresponds to the knee joint, that joint angle is computed by equation (5).

$$c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos\gamma \tag{4}$$

$$\gamma = \arccos\frac{{l_{upperLeg}}^2 + {l_{lowerLeg}}^2 - {l_{trans}}^2}{2 \cdot l_{upperLeg} \cdot l_{lowerLeg}} \tag{5}$$

Because $\gamma$ represents an interior angle and the knee joint is being streched in the zero-position, the resulting angle is computed by

$$\delta_{knee} = \pi - \gamma \tag{6}$$

Additionally the angle opposite to the upper leg has to be computed, because it corresponds to the foot pitch joint:

$$\delta_{footPitch1} = \arccos\frac{{l_{lowerLeg}}^2 + {l_{trans}}^2 - {l_{upperLeg}}^2}{2 \cdot l_{lowerLeg} \cdot l_{trans}} \tag{7}$$

Now the foot pitch and roll joints combined with the triangle form a kind of pan-tilt-unit. Their joints can be computed from the translation vector using atan2.[3]

$$\delta_{footPitch2} = \text{atan2}(x, \sqrt{y^2 + z^2}) \tag{8}$$

$$\delta_{footRoll} = \text{atan2}(y, z) \tag{9}$$

where $x, y, z$ are the components of the translation of $Foot2HipOrth$. As the foot pitch angle is composed by two parts it is computed as the sum of its parts.

$$\delta_{footPitch} = \delta_{footPitch1} + \delta_{footPitch2} \tag{10}$$

After the last three joints of the kinematic chain (viewed from the torso) are determined, the remaining three joints that form the hip can be computed. The joint angles can be extracted from the rotation matrix of the hip that can be computed by multiplications of transformation matrices. For this purpose another coordinate frame $Thigh$ is introduced that is located at the end of the upper leg, viewed from the foot. The rotation matrix for extracting the joint angles is contained in $HipOrth2Thigh$ that can be computed by

$$HipOrth2Thigh = Thigh2Foot^{-1} \cdot HipOrth2Foot \tag{11}$$

[3]atan2$(y, x)$ is defined as in the C standard library, returning the angle between the $x$-axis and the point $(x, y)$.

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

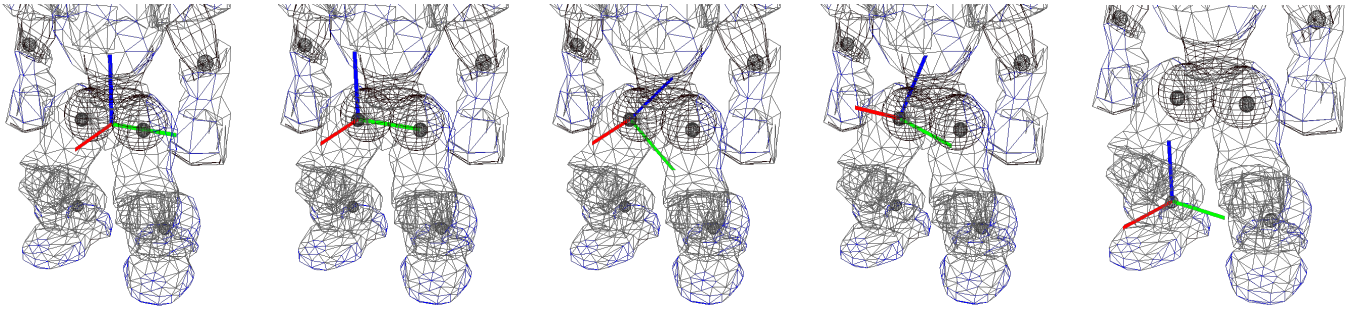

Fig. 3. Visualization of coordinate frames used in the inverse kinematic. From left to right: $Torso$, $Hip$, $HipOrth$, $Thigh$, $Foot$. The $x$-axis is shown in red, the $y$-axis in green, and the $z$-axis in blue.

where $Thigh2Foot$ can be computed by following the kinematic chain from foot to thigh.

$$\begin{aligned} Thigh2Foot \;=\; & Rot_x(\delta_{footRoll}) \cdot Rot_y(\delta_{footPitch}) \\ & \cdot Trans_z(l_{lowerLeg}) \cdot Rot_y(\delta_{knee}) \\ & \cdot Trans_z(l_{upperLeg}) \end{aligned} \tag{12}$$

To understand the computation of those joint angles, the rotation matrix produced by the known order of hip joints (yaw ($z$), roll ($x$), pitch ($y$)) is constructed (the matrix is noted abbreviated, e. g. $c_x$ means $\cos\delta_x$).

$$\begin{aligned} Rot_{Hip} \;&=\; Rot_z(\delta_z) \cdot Rot_x(\delta_x) \cdot Rot_y(\delta_y) \\ &=\; \begin{pmatrix} c_y c_z - s_x s_y s_z & -c_x s_z & c_z s_y + c_y s_x s_z \\ c_z s_x s_y + c_y s_z & c_x c_z & -c_y c_z s_x + s_y s_z \\ -c_x s_y & s_x & c_x c_y \end{pmatrix} \end{aligned} \tag{13}$$

The angle $\delta_x$ can obviously be computed by $\arcsin r_{32}$.[4] The extraction of $\delta_y$ and $\delta_z$ is more complicated, they must be computed using two entries of the matrix, which can be easily seen by some transformation:

$$\frac{-r_{12}}{r_{22}} = \frac{\cos\delta_x \cdot \sin\delta_z}{\cos\delta_x \cdot \cos\delta_z} = \frac{\sin\delta_z}{\cos\delta_z} = \tan\delta_z \tag{14}$$

Now $\delta_z$ and, using the same approach, $\delta_y$ can be computed by

$$\delta_z = \delta_{hipYaw} = \text{atan2}(-r_{12}, r_{22}) \tag{15}$$

$$\delta_y = \delta_{hipPitch} = \text{atan2}(-r_{31}, r_{33}) \tag{16}$$

At last the rotation by 45 degrees (cf. eq. 2) has to be compensated in joint space.

$$\delta_{hipRoll} = \delta_x - \frac{\pi}{4} \tag{17}$$

Now all joints are computed. This computation is done for both legs, assuming that there is an independent hip yaw joint for each leg. The computation described above can lead to different resulting values for the hip yaw joints of the left and the right leg. Given these two joint values, a single resulting value is determined, the computation of which is dynamically parameterized. This is necessary, because if the values differ, only one leg can realize the desired target, and normally the support leg (cf. sect. III) is supposed to reach the target position exactly. Given this fixed hip joint angle, there are only five variable joints supposed to realize a 6DOF-pose, thus it is impossible to reach the desired pose exactly. This is a typical optimization problem that we solved analytically by introducing a virtual *foot yaw* joint at the end of the kinematic chain. Now we have again six joints to realize a 6DOF-pose which is, as described above, solvable analytically. This modified kinematic chain can be seen as reversed since the universal joint is now located at the foot and not at the torso anymore. Hence this modified inverse kinematic problem can be solved for each leg as described above. The only difference is that the order of joints and the transformation of the foot relative to the torso has to be inverted.

The decision to introduce a foot *yaw* joint was mainly taken because an error in this (virtual) joint has a low impact on the stability of the robot, whereas other joints (e. g. foot pitch or roll) have a huge impact on stability.

[4]The first index denotes the row, the second index denotes the column of the rotation matrix.

## III. Walking

Walking means that the robot moves with desired speeds in forward, sideways, and rotational directions. To accomplish this, the two feet of the robot have to follow three-dimensional trajectories that define the actual steps. In previous work [6], we determined the trajectories of the feet relative to the torso. However, in the approach presented here, the trajectories of the feet are modeled relative to the center of mass (CoM). The foot positions relative to the CoM and the stance of the other body parts are used for determining foot positions relative to the torso that move the CoM to the desired point. Foot positions relative to the torso allow using inverse kinematics (cf. Sect. II) for the generation of joint angles.

Before the creation of walking motions starts, a stance (the stand) that will be used as basis for the foot joint angles during the whole walking motion is required. This stance is called $S$ and it is created from foot positions relative to the torso using inverse kinematics (cf. Sect. II). $S$ is mirror-symmetrically to $x$-$z$-plane of the robot's coordinate system (cf. Fig. 2 right). For the temporal dimension, a phase $\varphi$ that runs from 0 to (excluding) 1 and repeats permanently is used. The phase can be separated into two half-phases, in each of which alternately one leg is the support leg and the other one can be lifted up. The size of each half-step is determined at the beginning of each half-phase.

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

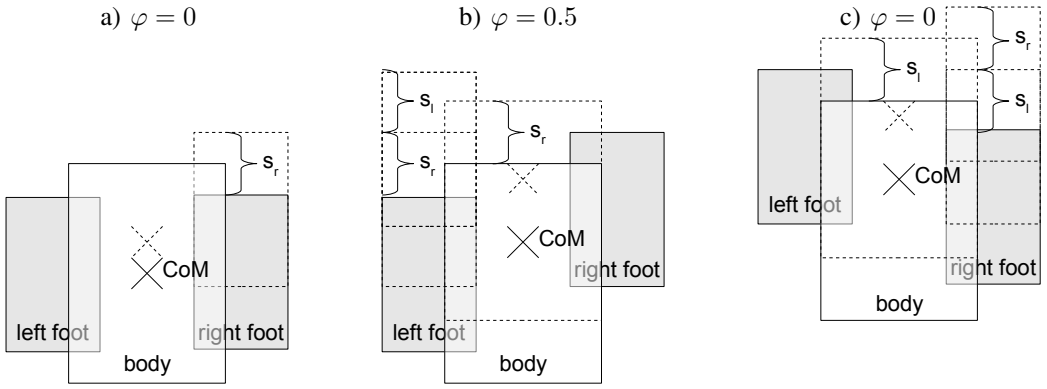


Fig. 4. Illustration of the foot shifting within three half-phases from a top-down view. At first, the robot stands (a) and then it walks two half-steps to the front (b, c). The support leg (left leg in (a)) changes in (b) and (c). Between (a) and (b), the step size $s_r$ is added to the right foot position. Between (b) and (c), $s_r$ is subtracted from the right and left foot positions, so that the body moves forwards. Additionally, the new offset $s_r + s_l$ is added to the left foot position. The CoM visualized is the projection of the desired CoM position on the ground, that differs significantly from the respective projection of the "center of body". Therefore an additional offset has to be added to all translational components of both foot positions, to move the CoM to the desired position.

The shifting of the feet is separated into the "shift"- and "transfer"-phases. Both phases last a half-phase, run sequentially, and are transposed for both feet. Within the "shift"-phase, the foot is lifted ("lift"-phase) and moved ("move"-phase) to another place on the ground, so that the foot is completely shifted at the end of the phase. The foot-shifting is subtracted from the foot position within the "transfer"-phase and also subtracted from the other foot position until that foot is shifted (cf. Fig. 4). This alone creates a motion that already looks like a walking motion. But the desired CoM movement is still missing.

So the foot positions relative to the torso ($p_{lRel}$ and $p_{rRel}$) are calculated as follows:

$$p_{lRel} = \begin{cases} o_l + s_{lLift} \cdot t_{lLift} \\ \quad + s_l \cdot t_{lMove} \\ \quad - s_r \cdot (1 - t_{lMove}) \cdot t_l & \text{if } \varphi < 0.5 \\ o_l + s_{lLift} \cdot t_{lLift} \\ \quad + s_l \cdot (1 - t_r) & \text{otherwise} \end{cases} \quad (18)$$

$$p_{rRel} = \begin{cases} o_r + s_{rLift} \cdot t_{rLift} \\ \quad + s_r \cdot t_{rMove} \\ \quad - s_l \cdot (1 - t_{rMove}) \cdot t_r & \text{if } \varphi \geq 0.5 \\ o_r + s_{rLift} \cdot t_{rLift} \\ \quad + s_r \cdot (1 - t_l) & \text{otherwise} \end{cases} \quad (19)$$

$o_l$ and $o_r$ are the foot origin positions. $s_{lLift}$ and $s_{rLift}$ are the total offsets used for lifting either the left or the right leg. $s_l$ and $s_r$ are the current step sizes. $t_{lLift}$ and $t_{rLift}$ are parameterized trajectories that are used for the foot lifting. They are parameterized with the beginning ($x_l$) and the duration ($y_l$) (cf. Fig. 5) of the "lift"-phase. $t_{lMove}$ and $t_{rMove}$ are used for adding the step sizes. They are parameterized with the beginning ($x_m$) and the duration ($y_m$) of the "move"-phase (cf. Fig. 6). $t_l$ and $t_r$ are shifted cosine shapes used for subtracting the step sizes (cf. Fig. 7). The trajectories are defined as follows ($\widehat{\varphi} = \varphi - 1$):

$$t_{lLift} = \begin{cases} \frac{1-\cos(2\pi(2\varphi - x_l)/y_l)}{2} & \text{if } 2\varphi \in ]x_l \ldots x_l + y_l[ \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

$$t_{rLift} = \begin{cases} \frac{1-\cos(2\pi(2\widehat{\varphi} - x_l)/y_l)}{2} & \text{if } 2\widehat{\varphi} \in ]x_l \ldots x_l + y_l[ \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

$$t_{lMove} = \begin{cases} \frac{1-\cos(\pi(2\varphi - x_m)/y_m)}{2} & \text{if } 2\varphi \in \\ & \quad ]x_m \ldots x_m + y_m[ \\ 1 & \text{if } 2\varphi \in \\ & \quad [x_m + y_m \ldots 1[ \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

$$t_{rMove} = \begin{cases} \frac{1-\cos(\pi(2\widehat{\varphi} - x_m)/y_m)}{2} & \text{if } 2\widehat{\varphi} \in \\ & \quad ]x_m \ldots x_m + y_m[ \\ 1 & \text{if } 2\widehat{\varphi} \in \\ & \quad [x_m + y_m \ldots 1[ \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

$$t_l = \begin{cases} \frac{1-\cos(2\pi\varphi)}{2} & \text{if } \varphi < 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

$$t_r = \begin{cases} \frac{1-\cos(2\pi\varphi)}{2} & \text{if } \varphi \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

The CoM movement has to be performed along the $y$-axis as well as in walking direction. The CoM is already moving in walking direction by foot shifting, but this CoM movement does not allow walking with a speed that meets our needs. Also, a CoM movement along the $z$-axis is useful. First of all, the foot positions relative to the CoM are determined by using the foot positions relative to the CoM of stance $S$ and adding the value of a trajectory to the $y$-coordinate of these positions. Since the calculated foot positions are relative to the CoM, the rotation of the body that can be added with another trajectory has to be considered for the calculation of the foot positions. The CoM movement in $x$-direction is calculated with the help of the step sizes of the currently performed half-steps.

If any kind of body and foot rotations and CoM-lifting along the $z$-axis are ignored, the desired foot positions relative to the

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

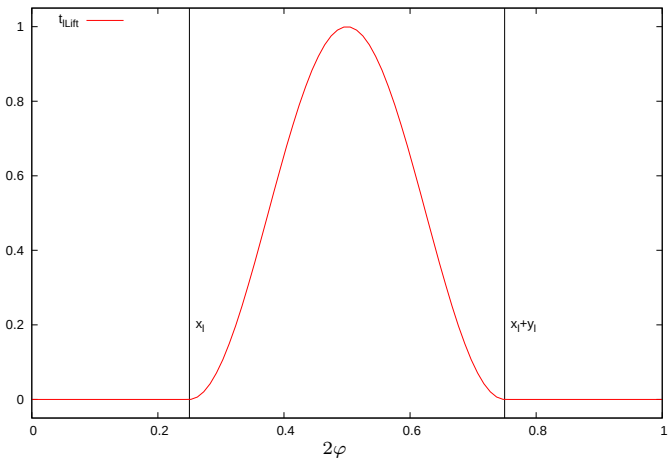


Fig. 5. The trajectory $t_{lLift}$ that is used for foot lifting. $t_{rLift}$ is similar to $t_{lLift}$ except that it is shifted one half-phase to the right.

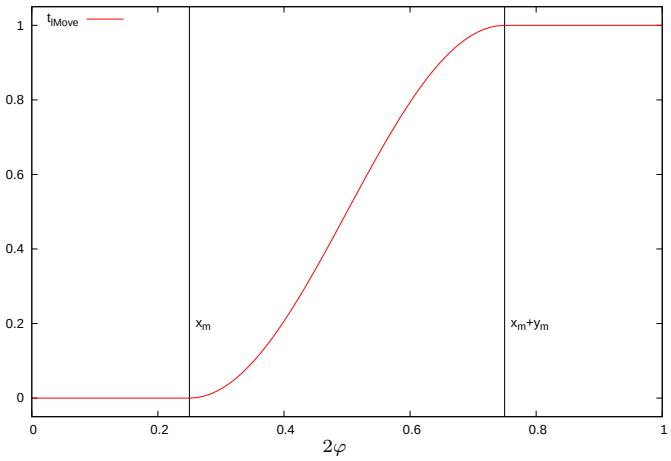


Fig. 6. The trajectory $t_{lMove}$ that is used for adding step sizes. $t_{rMove}$ is similar to $t_{lMove}$ except that it is shifted one half-phase to the right.

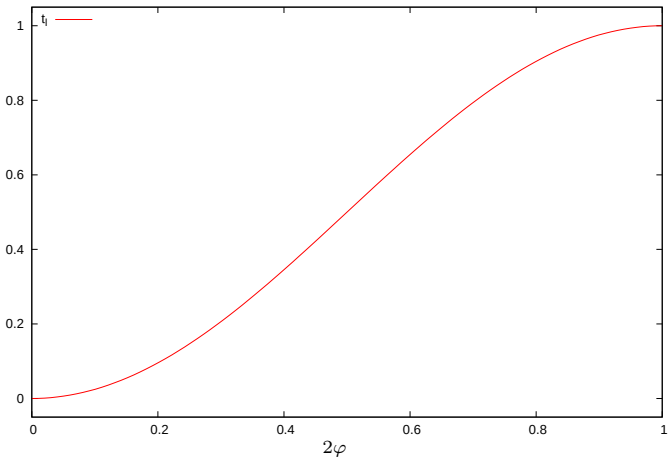


Fig. 7. The trajectory $t_l$ that is used for subtracting step sizes. $t_r$ is similar to $t_l$ except that it is shifted one half-phase to the right.

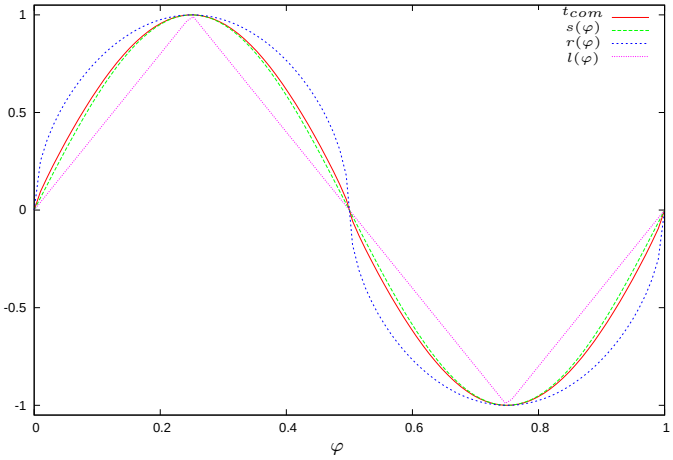


Fig. 8. The trajectory $t_{com}$ that is used for the CoM movement along the $y$-axis. It is a composition of $s(\varphi)$, $r(\varphi)$ and $l(\varphi)$.

CoM ($p_{lCom}$ and $p_{rCom}$) can be calculated as follows:

$$p_{lCom} = \begin{cases} -c_S + s_s \cdot t_{com} + o_l \\ -\frac{s_l \cdot t_{lin} - s_r \cdot (1 - t_{lin})}{2} & \text{if } \varphi < 0.5 \\ p_{rCom} - p_{rRel} + p_{lRel} & \text{otherwise} \end{cases} \quad (26)$$

$$p_{rCom} = \begin{cases} -c_S + s_s \cdot t_{com} + o_r \\ -\frac{s_r \cdot t_{lin} - s_l \cdot (1 - t_{lin})}{2} & \text{if } \varphi \geq 0.5 \\ p_{lCom} - p_{lRel} + p_{rRel} & \text{otherwise} \end{cases} \quad (27)$$

$c_S$ is the offset to the CoM relative to the torso of the stance $S$. $s_s$ is the vector that describes the amplitude of the CoM movement along the y-axis. $t_{com}$ is the parameterized trajectory of the CoM movement. Therefore, a sine ($s(p)$), square root of sine ($r(p)$) and a linear ($l(p)$) component are merged according to the ratios $x_c$ (for $s(p)$), $y_c$ (for $r(p)$) and $z_c$ (for $l(p)$) (cf. Fig. 8). $t_{com}$ is defined as:

$$t_{com} = \frac{x_c \cdot s(\varphi) + y_c \cdot r(\varphi) + z_c \cdot l(\varphi)}{x_c + y_c + z_c} \quad (28)$$

$$s(p) = sin(2\pi p) \quad (29)$$

$$r(p) = \sqrt{|sin(2\pi p)|} \cdot sgn(sin(2\pi p)) \quad (30)$$

$$l(p) = \begin{cases} 4p & \text{if } p < 0.25 \\ 2 - 4p & \text{if } p \geq 0.25 \wedge p < 0.75 \\ 4p - 4 & \text{if } p \geq 0.75 \end{cases} \quad (31)$$

$t_{lin}$ is a simple linear trajectory used for moving the CoM into the walking direction.

$$t_{lin} = \begin{cases} 2\varphi & \text{if } \varphi < 0.5 \\ 2\varphi - 1 & \text{if } \varphi \geq 0.5 \end{cases} \quad (32)$$

Based on the desired foot positions relative to the CoM, another offset is calculated and added to the foot positions relative to the torso ($p_{lRel}$ and $p_{rRel}$) to achieve the desired foot positions relative to the CoM with coordinates relative to the torso. For the calculation of the additional offset, the fact that the desired CoM position does not change significantly

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

between two cycles is exploited, because the offset that was determined in the previous frame is used first. So, given the current leg, arm, and head stance and the offset of the previous frame, foot positions relative to the CoM are determined. The difference between these foot positions and the desired foot positions is added to the old offset to get the new one. The resulting foot positions relative to the CoM are not precise, but the error is negligible.

The additional offset ($e_{new}$) can be calculated using the old offset ($e_{old}$) as follows:

$$e_{new} = \frac{p_{lRel} - p_{lCom} + p_{rRel} - p_{rCom}}{2} - c_{e_{old}, p_{lRel}, p_{rRel}} \quad (33)$$

$c_{e_{old}, p_{lRel}, p_{rRel}}$ is the CoM offset relative to the torso given the old offset, the new foot stance, and the stance of the other limbs.

Finally, $p_{lRel} - e_{new}$ and $p_{rRel} - e_{new}$ are the positions used for creating the joint angles, because:

$$p_{lRel} - c_{e_{new}, p_{lRel}, p_{rRel}} - e_{new} \approx p_{lCom} \quad (34)$$

$$p_{rRel} - c_{e_{new}, p_{lRel}, p_{rRel}} - e_{new} \approx p_{rCom} \quad (35)$$

## IV. Balancing

To react on unexpected events and for stabilizing the walk in general, balancing is required. Therefore, several balancing methods are supported. Three of them are simple p-controllers. Only the step-size balancing also has an integral component. The error that is used as input for the controllers is solely determined from the actual pose of the robot torso, i. e., the pitch and roll angles of the torso relative to the ground, and the expected pose. The delay between the measured and the desired CoM positions is taken into account by buffering and restoring the desired foot positions and torso poses. The different kinds of balancing are:

### A. CoM Balancing

The CoM balancer works by adding an offset to the desired foot positions ($p_{lCom}$ and $p_{rCom}$). Therefore, the error between the measured and desired foot positions has to be determined, so that the controller can add an offset to these positions according to the determined error. The error is calculated by taking the difference between the desired foot positions and the same positions rotated according to rotation error.

### B. Rotation Balancing

Besides CoM Balancing, it is also possible to balance with the body and/or foot rotation. Therefore, angles depending on the rotation error can be added to the target foot rotations or can be used for calculating the target foot positions and rotations. This affects the CoM position, so that the offset computation that is used for moving the foot positions relative to the CoM might compensate the balancing impact. So this kind of balancing probably makes only sense when it is combined with CoM Balancing.

### C. Phase Balancing

Balancing by modifying the walking phase is another possibility. Therefore, the measured $x$-angle, the expected $x$-angle and the current walking phase position are used to determine a measured phase position. The measured position allows calculating the error between the measured and actual phase position. This error can be used for adding an offset to the current phase position. When the phase position changes in this manner, the buffered desired foot positions and body rotations have to be adjusted, because the modified phase position affects the desired foot positions and body rotations of the past.

### D. Step-Size Balancing

Step-size balancing is the last supported kind of balancing. It works by increasing or decreasing the step size during the execution of half-steps according to the foot position error that was already used for CoM Balancing. Applied step-size balancing has the disadvantage that the predicted odometry offset becomes imprecise. Therefore, it can be deactivated for critical situations such as when positioning for kicking the ball.

## V. Torso Pose Estimation

Estimating the pose of the torso consists of three different tasks. First, discontinuities in the inertial sensor readings are excluded. Second, the calibration offsets for the two gyroscopes ($x$ and $y$, cf. Fig. 2 right for the robot coordinate system used in this paper) are maintained. Third, the actual torso pose is estimated using an Unscented Kalman filter (UKF) [7].

Excluding discontinuities in the sensor readings is necessary, because some sensor measurements provided by the Nao cannot be explained by the usual sensor noise. This malfunction occurs sporadically and affects most measurements from the inertial measuring unit within a single frame (cf. Fig. 9). The corrupted frames are detected by comparing the difference of each value and its predecessor to a predefined threshold. If a corrupted frame is found that way, all sensor measurements from the inertial measuring unit are ignored. Corrupted accelerometer values are replaced with their predecessors and corrupted gyroscope measurements are simply not used.

Gyroscopes have a bias drift, i. e., the output when the angular velocity is zero drifts over time due to factors such as temperature that cannot be observed. The temperature changes slowly as long as the robot runs, so that it is necessary to redetermine the bias continuously. Therefore, it is hypothesized that the torso of the robot and thereby the inertial measurement unit has the same pose at the beginning and the end of a walking phase (i. e. two steps). Therefore, the average gyro measurement over a whole walking phase should be zero. This should also apply if the robot is standing. So either, the average measurements over a whole walking phase are determined, or the average over 1 sec for a standing robot. These averages are filtered through one-dimensional Kalman filters and used as biases of the gyroscopes. The collection of gyroscope measurements is limited to situations in which the

Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37

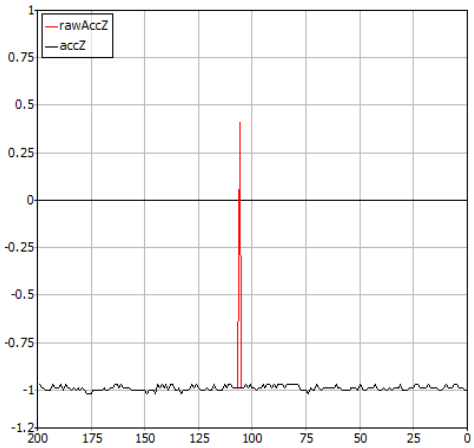


Fig. 9. A typical corrupted inertia sensor reading between the frames 110 and 100. The corrupted data was detected and replaced with its predecessor.

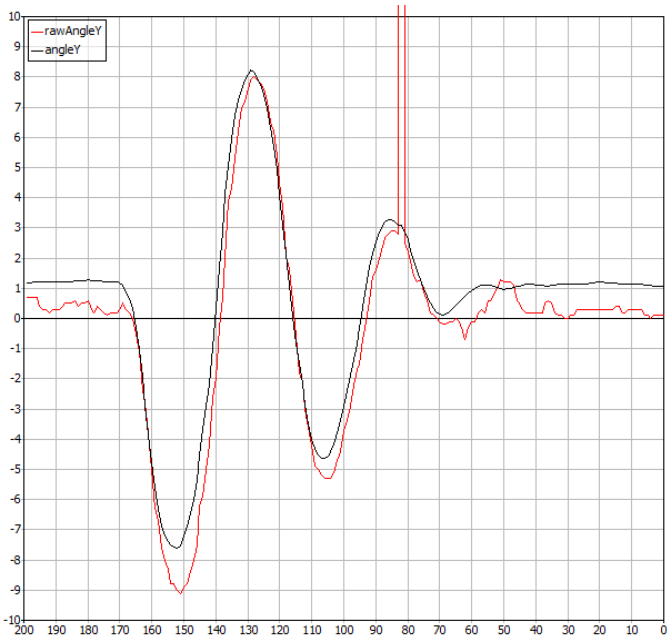


Fig. 10. The difference between the estimated pitch angle *angleY* and the pitch angle *rawAngleY* provided by the inertia board of the Nao.

robot is either standing or walking slowly and has contact to the ground (determined through the force-sensitive resistors in Nao's feet).

The UKF estimates the pose of the robot torso (cf. Fig. 10) that is represented as three-dimensional rotation matrix. The change of the rotation of the feet relative to the torso in each frame is used as process update. The sensor update is derived from the calibrated gyroscope values. Another sensor update is added from a simple absolute measurement realized under the assumption that the longer leg of the robot rests evenly on the ground as long as the robot stands almost upright. In cases in which this assumption is apparently incorrect, the acceleration sensor is used instead.

It is not only possible to get the orientation from the UKF, but also to get a "filtered" version of the gyroscope measurements from the change in orientation, including a calculated $z$-gyroscope value that is actually missing on the Nao.

## VI. Results

The gait described here was used at RoboCup 2009 by the team B-Human [8] in the Standard Platform League – the world champion and winner of the technical challenge. The maximum speeds reached in soccer competitions were 15 cm/s forwards, 10 cm/s backwards, 9 cm/s sideways, and 35°/s rotational speed. An interesting effect is that the theoretical maximum forward speed resulting from the foot trajectories generate is only 12 cm/s. The additional 3 cm/s seem to result from balancing by changing the step size. Therefore it seems that the robot is walking faster because it continuously prevents itself from falling down to the front.

At RoboCup 2009, our walk was the most robust one for the Nao, and it still was among the fastest walks. Its precision made us the team that was able to most quickly align behind the ball for kicking. As a result we scored more goals than all the other 23 teams in the Standard Platform League together. There is a video on the B-Human homepage (`http://www.b-human.de`) showing some scenes from the games in which the gait can be seen. The software used by B-Human at RoboCup 2009 can also be downloaded from that website, including the implementation of the algorithms described in this paper.

## VII. Conclusion and Future Work

In this paper we presented a CoM-based closed-loop gait for the Nao. Four different balancing methods make the gait very robust, as has been proven during the RoboCup 2009 competitions. We also described an analytical solution to the inverse kinematics of the Nao that is used for our gait. Finally, we described the estimation of the pose of the robot's torso, which is the reference for the balancing methods.

Higher speeds can be achieved with this approach, but so far not in a way that would allow walking omni-directionally for more than 10 minutes in a row (a half-time) without falling down. Hence, we will replace the CoM-based core of our gait with a ZMP-based preview controller in the future, carefully keeping the balancing methods in place that are the main reasons for the robustness of the current walk. In addition, we will work on automatically optimizing the parameters of our gait, using Particle Swarm Optimization (PSO) [9] as we have already done for a Kondo robot [10]. Actually, a few parameters of the gait presented in this paper were already optimized using this method.

## Acknowledgements

The authors would like to thank all B-Human team members for providing the software base for this work.

## REFERENCES


[1] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "The NAO humanoid: a combination of performance and affordability," *CoRR*, vol. abs/0807.3223, 2008.

[2] J. A. Kulk and J. S. Welsh, "A low power walk for the NAO robot," in *Proceedings of the 2008 Australasian Conference on Robotics & Automation (ACRA-2008)*, J. Kim and R. Mahony, Eds., 2008.

[3] M. Vukobratovic and B. Borovac, "Zero-moment point – thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 157–173, 2004.

[4] S. Czarnetzki, S. Kerner, and O. Urbann, "Observer-based dynamic walking control for biped robots," *Robotics and Autonomous Systems*, vol. 57, no. 8, pp. 839–845, 2009.

[5] J. Strom, G. Slavov, and E. Chown, "Omnidirectional walking using ZMP and preview control for the nao humanoid robot," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Artificial Intelligence, J. Baltes, M. G. Lagoudakis, T. Naruse, and S. Shiry, Eds. Springer, to appear in 2010.

[6] T. Röfer, T. Laue, A. Burchardt, E. Damrose, K. Gillmann, C. Graf, T. J. de Haas, A. Härtl, A. Rieskamp, A. Schreck, and J.-H. Worch, "B-Human team report and code release 2008," 2008, 72 pages. [Online]. Available: http://www.b-human.de/media/coderelease08/bhuman08_coderelease.pdf

[7] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, 1995. Proceedings of the*, vol. 3, 1995, pp. 1628–1632. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=529783

[8] T. Röfer, T. Laue, O. Bösche, I. Sieverdingbeck, T. Wiedemeyer, and J.-H. Worch, "B-Human team description for robocup 2009," in *RoboCup 2009: Robot Soccer World Cup XII Preproceedings*, J. Baltes, M. Lagoudakis, T. Naruse, and S. Shiry, Eds. RoboCup Federation, 2009.

[9] R. C. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[10] C. Niehaus, T. Röfer, and T. Laue, "Gait optimization on a humanoid robot using particle swarm optimization," in *Proceedings of the Second Workshop on Humanoid Soccer Robots in conjunction with the 2007 IEEE-RAS International Conference on Humanoid Robots*, C. Zhou, E. Pagello, E. Menegatti, and S. Behnke, Eds., 2007.



Proceedings of the 4th Workshop on Humanoid Soccer Robots
A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009),
Paris(France), 2009, December 7-10
ISBN 978-88-95872-03-2
pp. 30-37