# Dynamic Kick Motion Design with Implementation of Whole-Body Operational Space Control on the NAO

Yuchen Cui

*Abstract*— **Dynamic, or closed-loop, kicking motion is more flexible with ball position, hence theoretically more accurate than key-frame based open-loop kicking motions. This improved accuracy can be beneficial to teams participating in the RoboCup Standard Platform League, where the Aldebaran NAO robot is used as the platform. Implementing dynamic kicks is not trivial since it also requires employing dynamic balancing. Many of the existing balance controllers for NAO use inverse-kinematics-based control and center-of-mass- (CoM) or zero-moment-point- (ZMP) based balancing techniques. These approaches are unable to handle conflicting control scenarios in nature and often require compensation of the kick speed for balancing. Therefore, our team propose to implement a Whole-Body Operational Space Controller (WBOSC) on the NAO, where the control is hierarchical and tasks can be prioritized. WBOSC allows us to develop behaviors of the robot while complying environment and body constraints. With WBOSC, we can implement tasks such as 'foot trajectory task', 'balance control task' and 'joint position task' with desired hierarchy, so that tasks with higher priority will not compensate for tasks with lower priorities. This idea has been proved successful in a NAO simulator we built. However, with limited computational resource and sensor accuracy, there are some challenges implementing the complete WBOSC on a real NAO robot.**

## I. INTRODUCTION

A good kick motion for the NAO is desired for teams participating in the Standard Platform League of RoboCup. Kicking plays an important, if not determining, part in helping the team to win the game. Quality of kick motion depends on its robustness and flexibility. The kick motion we implemented on our NAO (for the course assignment) was an open-loop kick, where a sequence of joint angle combinations are tuned manually ahead of time, recorded as key-frames and executed/replayed when the robot needs to kick. The performance of such open-loop kicks relies heavily on how accurately the robot is able to walk to a desired kick position, where its designated kick foot is nicely lined up with the ball. With sensing and actuating errors and limitations, adjusting the position of the robot takes a rather long time while it may still end up at some undesired position. In a game setting, the robot also needs to cope with the situation when the ball is moved (probably by an opponent) after it already started the kick sequence. Therefore, we propose to design a closed-loop/dynamic kick motion, integrating feedback from vision and gyroscope measurements in order to improve the performance of our NAO's kick. We are going to implement a Whole-Body Operational Space Control (WBOSC) [11] system on the NAO to achieve this purpose.

## II. LITERATURE REVIEW

### A. Dynamic Kick Design

Since Aldebaran Robotics' NAO was selected to be the platform for the Standard Platform League of RoboCup, there has been a lot research effort put into designing good kicks, especially dynamic and omni-directional kicks.

Xu and Mellmann [16] model the NAO's motion in Cartesian space and generate the joint trajectories using inverse kinematics. Their approach divides kick motion into four phases: preparation, retraction, execution and wrap-up. The retraction phase adapts visual inputs and calculates desired trajectory of the kick foot. To avoid collision and also be able to detect impossible kick condition, the authors defined reachability of points and a reachable space for constraining the trajectories. The execution phase simply moves the kick foot along the shortest path in the reachability grid.

Muller et al. [7] extended the motion engine of team B-Human's work in 2009 [10], where a motion is defined as a set of phases and a phase is a set of six trajectories. The authors model the trajectories of NAO's limbs in Cartesian space using piecewise Bezier curves. The Bezier curves are modified so that the ball gains velocity at the desired direction after the kick. Such modification is also constrained so that the curve is continuously differentiable in order to always get smooth trajectories.

Ferreira et al. [5] proposed an omni-directional kick for NAO adapting the inverse kinematics model from B-Human's work in 2011. The authors divide kick behavior into three modules: inverse kinematics module, path planning module and stability module. The inverse kinematics module implements B-Human's geometric approach, which computes exact desired angles for each joint in constant time, where the hip yaw angles are corrected due to the fact that the joints are mechanically connected together among two legs. The path planning module models the trajectory of the kick foot using Bezier cubic curve. The results obtained from simulation tests show that the NAO was able to perform desired motion accurately in different situations.

Wenk and Rofer [15] approach the problem without prior modeling of trajectories of limbs. The authors calculate the trajectory of the kick foot by interpolating between a number of reference poses inferred from the ball position, the kick direction and the kick strength. The reference points of the kick foot are calculated by modeling the contour of the front of the kick foot as a cubic Bezir curve and calculating its tangent line at the touch point of the ball. The interpolation of reference points uses a spline that is continuously differentiable twice so that the resulting trajectory is always smooth.

Barrett et al. [3] designed a kick engine that mimics a fully dynamic kick engine with much less computation. The authors' design of kick engine follows the key-frame scheme. However, instead of repeating the same kick every time or selecting from a large set of static kicks, their kick engine keeps a small set of parameterized kicks. The parameters are selected based on ball position and target position. Their design is able to handle desired kick angles from 10 degree to the inside of the kicking leg out to 30 degree to the outside, and distances from 0.5 to 3.5 meters.

### B. Balance Control

Dynamic kick motion usually involves balancing the robot, although there are ways to go around as in [3]. Since only on leg and its corresponding foot is used to gain velocity and hit the ball, the rest of the robot body is used to balance itself. Actively maintaining balance of the body is also one requirement of gait design so that research work in bipedal walking suggests solution to active balancing during dynamic kicking as well. Zero Moment Point (ZMP), which is also the point of pressure, is commonly used in many approaches in order to measure the stability of the robot. Gyroscope feedback is also commonly used in balancing a robot.

Ferreira et al. [5] implemented a simple stability module. The stability module calculates the center of mass and its ground projection to see if it is inside the support polygon. If not, the module will open the arm on the same side of the supporting foot in an effort to correct the ground projection of center of mass. In cases where the are movement is not enough to compensate the error, the hip and ankle roll angles of the supporting foot will tilt to correct the error.

Xu and Mellmann [16] used a similar approach that they named as Body Inclination Control. The control algorithm computes the error between center of mass and the center of support polygon, which is fed to a proportional(P) controller to adjust the body inclination so that the error is minimized.

Faber and Behnke [4] proposed a method for balancing bipedal walking with two feedback mechanisms: a P-controller that regulates the angular velocity of the robot body, and a phase resetting system which resets when supporting foot alternates. Building on Faber and Behnke's method, Muller et al. [7] used a PID controller to balance the robot during a kick. They calculate desired angular velocity using commanded angles, estimate the next possible angular velocity using previous measurements with exponential smoothing, and then calculates the error in angular velocities as input to the PID-controller. To balance the walking for a NAO, Shafii et al. [13] also use a PID-controller to keep the trunk of NAO in an upright position, orthogonal to the ground all the time.

Wenk and Rofer [15] implemented an estimation of ZMP and compared two balancing model: linear quadratic regulation and cart-table controller. It turned out that the cart-table controller responds more quickly to the ZMP changes. However, the cart-table controller assumes that the center of mass is at a fixed a height, which may always be the case.

There are many other sophisticated solutions for balancing bipedal walking as well, including [1] [2] and [6]. [1] combines the commonly used ZMP criterion with angular momentum suppression in order to address the common false assumption that the supporting foot (soles of the robot) is always firmly on the ground. [2] uses two layers of contorl, of which the first generate center of mass trajectories and the second extends a preview controller with recovery strategies. [6] proposed an online learning approach for optimizing bipedal walking. With an inverted pendulum-like model, their bipedal robot was able to learn how to balance it self and recover from push very quickly.

### C. Our Approach

While kicking mostly involves one leg of the robot, the rest of the body has to move to balance itself. However, many of the inverse kinematic control techniques used to generate kick trajectory complicate balancing for the robot. Ideally, if the robot is able to balance itself while walking, it should also be able to balance itself while kicking without explicit change to the balancing module. Therefore, instead of designing many task-specific modules, we propose to implement a Whole-Body Operational Space Controller [11] on the NAO, where controls are layered and tasks are prioritized. The idea behind Whole-Body Operational Space Control is to have a whole-body control framework implementing torque control strategies for advanced contact and non-contact interactions as to address the limitations posed by inverse kinematic techniques. WBOSC is an advanced control framework that synthesizes operational space controllers at multiple levels. WBOSC has been used in the control of Honda humanoid robot Asimo [12]. Using WBOSC to approach dynamic kicking motion for NAO is very different from existing methods from a high-level point of view. At the same time, we implemented a balance controller similar to the work of Orin and Goswami [8], which employs a Centroidal-Momentum-based method and is more sophisticated than the existing models used in balancing NAO.

Our design of the WBOSC system for NAO is described in the following sections of this report. We also present the results obtained using a NAO simulator as well as some initial measurements of the incomplete system on the real robot. We will then conclude with the lessons learned, the challenges we met implementing WBOSC on NAO hardware, as well as potential future work of this project. Last, we also documented how we cooperated throughout the project.

## III. SYSTEM DESIGN

### A. Whole Body Operational Space Controller

The WBOSC models a robot as a free floating system in contact with the ground, and it creates kinematic and dynamic representations of the system by analyzing the impact of associated constraints and reaction forces. It takes the

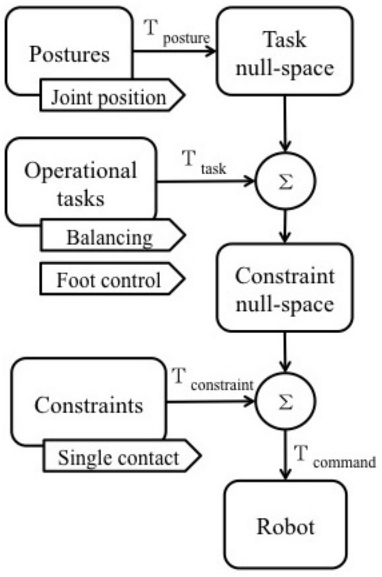


Fig. 1: Structure of WBOSC Hierarchical Control. Torque commands are synthesized through null-space projections.

state of the robot, including states of each joint, as input and output a desired torque command for each joint. A WBOSC is designed to allow the robot simultaneously accomplish multiple low-level tasks as part of the whole-body behavior. As described in [12] [11], low-level tasks of a typical WBOSC are divided into three main categories: constraint-handling tasks, operational tasks, and postures. Constraint-handling tasks are designed to deal with physical and environmental limitations, typically supporting constraints. Operational tasks are designed to provide manipulation and locomotion skills, normally involves control of robot limbs or sometimes center of mass. Postures are designed to control the additional redundancy and used for different purposes including mimicking human-like postures and minimizing torque effort.

In our project, we have four different low-level tasks: contact constraint task, balance control task, foot trajectory task, and joint position task. The contact constraint task is a constraint-handling task that describes whether the robot is supported by one of its feet or both. The balance control task is an operational task where we implemented our balance controller. The foot trajectory task is another operational task that actually performs the dynamic kick with calculated foot trajectory. The joint position task is the posture that is used to minimized the torque effort, basically keeping previous joint positions as much as possible. The priority of the tasks is determined by the order they are pushed onto the task array. Constraint tasks always have the highest priority since they are the physical limitations to the robot motion. Orders of foot trajectory task and balance control task can be switched but will generate different results. The hierarchical control of WBOSC is achieved by kinematic projection of lower-priority Jacobians into the null-space of higher priority tasks. The torque command generated by each task is synthesized to a unique update command sent to the robot. Figure 1 shows the structure of the hierarchical control.

### B. NAO Interface Design

The NAO robot takes joint angle values as command input to actuators, while the WBOSC outputs torque commands. An interface is needed to translate torque commands to joint angles between the control system and NAO. Such translation needs to be as accurate as possible or the joints can start oscillate with command errors. We tried two different approaches converting torque command into joint position command. The first approach is based on the assumption of the NAO's joint position controller model, which can be incorrect, so we call it the "black box conversion". The second approach use the dynamics of NAO to find the relationship between torque commands and joint positions.

- Black Box Conversion

  Let $\tau$ denote the desired torque and $q, q_d$ denote current and desired joint positions respectively. $K$ is some constant. Assuming the joint position controller of NAO uses proportional feedback control, the following equation holds:

  $$\tau = K(q_d - q)$$

  Therefore, we can calculate the desired joint position $q_d$ given calculated torque command and current joint position $q$ using:

  $$q_d = \frac{\tau}{K} + q$$

  The only thing we need to do is to find some constant $\frac{1}{K}$ for each and every joint of NAO.

- Conversion by Integration

  With the dynamic model of NAO, we can calculate the joint positions with torque commands. Let $U$ be the under-actuation action matrix that maps the global joint vector to the subspace of actuated joints. Let $A, b, g$ be the inertia matrix, coriolis forces and gravitational forces respectively. Let $\tau$ denote the torque command and $\ddot{q}$ denote the second derivative of current joint position $q$. The dynamics of the robot's joints is described as the following equation:

  $$A\ddot{q} + b + g = U^T\tau$$

  With contact constraints, we will have the support Jacobian $J_s$ that maps joint velocity to the velocity of the constrained foot in Cartesian space, and we also have the associated constraint space reaction forces $F_r$. The dynamics of the robot's joints become:

  $$N_s = I - A^{-1}J_s^T(J_sA^{-1}J_s^T)^{-1}J_s$$
  $$A\ddot{q} + N_s^T(b + g) = (UN_s)^T\tau$$

  Hence, we can calculate $\ddot{q}$ using following equation:

  $$\ddot{q} = A^{-1}((UN_s)^T\tau - N_s^T(b + g))$$

From $\ddot{q}$, we can get $q$ and $\dot{q}$ by integration. Let $q_n$ denote the joint position at time $n$ and the time difference between two updates is $\delta t$:

$$q_n = q_{n-1} + \dot{q}_{n-1}\delta t + \ddot{q}\delta t^2$$

$$\dot{q_n} = \dot{q}_{n-1} + \ddot{q}\delta t$$

### C. *Balance Controller Design*

We implemented our balance controller as the balance control task in the WBOSC. For this project, we tried two different types of balance controller. One is the Capturability-based balance controller as described in the work of Pratt et al. [9], which is also very similar to the existing ZMP-based methods for balancing NAO. The other one is the Centroidal-Momentum-based balance controller, same as the work of Orin and Goswami [8], which does not approximate the body to be a simplified model. The ideas of the two balance controller are described below.

- Capturability-based balance controller

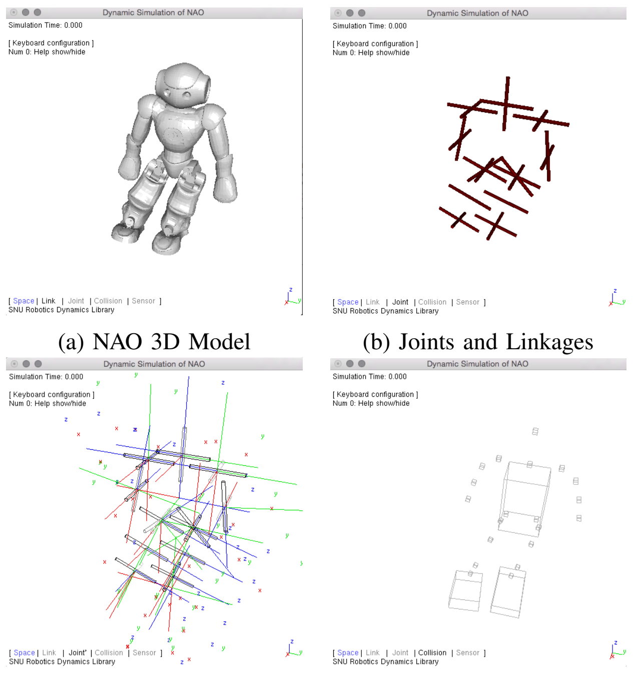
  N-step capturability is defined as the ability of a legged system to come to a stop without falling by taking N or fewer steps. The Capturability-based balance controller uses the 3D Linear Inverted Pendulum Model (3D-LIMP) to approximate a bipedal robot. Considering a 3D-LIMP with finite-sized foot, the instantaneous capture point (also the 1-step capturable point) is the point on the ground where the Center of Pressure (CoP, equivalent to ZMP) should be placed instantaneously and maintained to come to a stop with the Center of Mass (CoM) directly above the CoP. Instantaneous capture points are used to describe the desired CoM trajectory. The linear dynamics of the capture points allows us to find a desired CoP location within the base of support foot that 'pushes' the instantaneous capture point along the desired path. It is essentially a proportional feedback controller with damping parameter.
- Centroidal-Momentum-based balance controller

  The Centroidal Momentum of a humanoid robot is sum of the individual momentum of each link projected the robot's CoM. Centroidal Momentum is a linear function of the robot's velocities and the Centroidal Momentum Matrix is the matrix form of this function. This matrix is a product of a Jacobian and an inertia matrix, which maps the joint velocities to the Centroidal Momentum. Centroidal-Momentum-based balance controller makes no approximation of the robot and uses the Centroidal Momentum Matrix to manipulate CoM with joint positions.

## IV. Results

### A. *Simulation*

With the modified model of NAO (nao.urdf, the original version is provided by ROS), which is also the model used in the real system, we built a simulator of NAO using srLib [14]. In the simulator, NAO is modeled using linkages and collision spaces. We included the twenty two joints we used to buld the WBOSC for NAO in the simulator and simplified the collision spaces to be just the enclosing cuboid of the NAO's torso and two feet. Figure 2a shows the 3D model of NAO in the simulator, figure 2b shows the joints and joint linkages, figure 2c shows the coordinates of each joint, and figure 2d show the collision spaces of the NAO simulator.


Dynamic Simulation of NAO
Simulation Time: 0.000
[ Keyboard configuration ]
Num 0: Help show/hide
[ Space | Link | Joint | Collision | Sensor ]
SNU Robotics Dynamics Library

(a) NAO 3D Model (b) Joints and Linkages

(c) Joint Coordinates (d) Collision Spaces

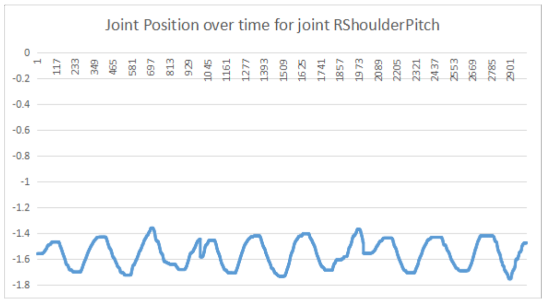
Fig. 2: NAO Simulator.

Using this simulator, we tested the balance controllers by using a virtual ball to hit the NAO at its torso. The NAO was only supported by its left foot. Without any balancing control, the NAO will fall down as expected. Both types of balance controllers described previously were able to balance the NAO and recover from the ball hit. However, we found that the Capturability-based balance controller is not very stable after running for a long while. Therefore, we used the Centroidal-Momentum-based balance controller to design a balanced kick. However, it is still worth trying both balance controller on a real NAO to see how the performance differs since the sensing imperfection may affect the balance control in different ways.

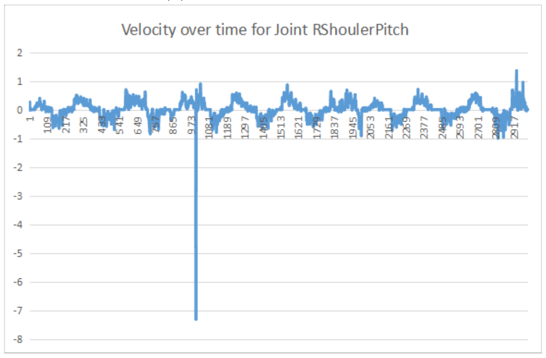
In the foot trajectory control task, desired positions of the foot in global coordinate are used to command the foot motion. Since we did not simulate the NAO vision in the simulator, we did not implement the vision-based dynamic kick. We used a fixed kick trajectory that is not necessarily balanced by itself. We were able to see that the kick was balanced with our balance controller implementation.

### B. *NAO Interface Implementation*

For the black box conversion approach, we tuned the constant $K$ value for each and every one of the twenty two joints of NAO, so that they can follow a joint position trajectory as a torque command describes. However, when



(a) Joint Position Data



(b) Filtered Joint Velocity Data

Fig. 3: NAO Joint Measurement.

we put everything together with gravity compensation of WBOSC, the joints moved in an unpredictable way when torque commands are set to zeros. It seems that there might be some significant terms we ignored in the equation used and the joint position controller could be a full PID controller. We would have to tune a total of 66 parameters by hand if we went on with this approach assuming the joint position controller has the integral and derivative terms. Therefore, we did not use this method and implemented conversion by integration for the interface we currently use in the system.

Conversion by integration has been proved working better than the previous method. The NAO can stand still when commands are set to zeros. However, error accumulates easily as we integrate the commands. Without joint velocity feedback, we observed increasing motion amplitudes of sinusoidal trajectories. Feedback joint velocity data are essential to correct the commands. Unfortunately the joint position measurements are not very accurate so that the velocity data is very noisy. For example, figure 3a and 3b show the joint position and joint velocity over time for the joint RShoulderPitch (right shoulder pitch joint). The command was a sinusoidal trajectory. The velocity data is already filtered with a low-pass filter but sill very noisy. In this case, we had to tune the command gain for joint velocities very carefully so that they do not cause vibration or unexpected jerky motion. We were able to observe stable sinusoidal trajectories of different joints after some parameter tunning.

## V. CONCLUSION

### A. LESSONS LEARNED

From our experience working with a NAO, we think it is better to try and debug in simulation before putting the algorithm onto the real robot or it is very likely to break the robot. Besides, while parameter tuning itself is already a long process, tuning parameters on a real robot takes longer due to long compiling time. For parameters not specific to hardware such as individual actuators, it is easier and safer to test using a simulator. Therefore, we also consider the simulator we made as a contribution of this project that can be beneficial for future projects on motion design.

We also found the NAO have noisy sensors and limited computational resource that can prevent us from implementing a dynamics-based WBOSC. As we mentioned in implementing the interface, we found that the joint velocity data is very noisy while it is critical data for balance control. IMU data is also noisy that we had concerns about whether the Centroidal-Momentum-based balance controller is practical on the NAO since it requires accurate IMU measurements to compute the correct momentum and corresponding response command. We also found the NAO is not meeting the standard update rate required by a WBOSC and the WBOSC can be heavy for the NAO processors. On one hand, a typical WBOSC requires a motion update rate of at least 1K Hz, while the NAO is running at 100 Hz update rate. On the other hand, we observed motion update rate drop the first time we merged the WBOSC into the NAO system. We tried to solve this problem by making the dynamics model updating a thread by itself and drop the update rate of the dynamics model manually. This helped to lower the rate of slow updates but do not solve the problem entirely.

### B. FUTURE WORK

As previously stated in implementation challenges, we did not implement the complete WBOSC on NAO due to the concerns of both the slow update rate and the sensitivity of the system to noise in joint velocity data and IMU data. However, we think it is still worth trying the completed system (with balance controller and foot motion) on a NAO to see how it works. It may require some additional parameter-tuning.

Besides, we did not incorporate vision in the simulator and that is readily feasible. It will be beneficial to future projects if we have a full simulator for NAO. At the same time, a full simulator will help us to simulate a dynamic kick with vision-feedback so that we can test foot motion planning in simulation. We also considered adding some control noise into the simulator to better model the real system. More parameter measurements from the robot are needed for this purpose.

## VI. TASK DIVISION

The three of us, Yi-Chu and Donghyun and I, met regularly to discuss what we needed to do and how we plan to do things throughout the project. The first several meetings were devoted to Donghyun explaining WBOSC to me and Yi-Chu. Since Donghyun has rich experience with WBOSC, he primarily worked on migrating the code he used for a different robot onto NAO. Yi-Chu and I worked on checking the model file (nao.urdf, provided by ROS) of NAO, building the interface and tuning parameter values. I also wrote some post-processing scripts to retrieve and plot the data we wanted to monitor as we tested different parameters. Unfortunately, our first robot broke when we were testing the first version of our interface using individually tuned parameters. Therefore, we all worked on building a simulator using the NAO model file (nao.urdf) and 3-D modeling files. After we got our second robot, Yi-Chu and I built the second version of the interface while Donghyun worked on building the balance controller in the simulator.

## VII. Acknowledgment

Thanks Professor Peter Stone for reviewing our proposals, giving suggestions and timely responses to our questions. Thanks our teaching assistant Jake Menashe for answering our questions timely and helping us get a second robot to work with. Thanks Katie Genter for interviewing with us and commenting on our ideas.

## References

[1] J. J. Alcaraz-Jimenez, D. Herrero-Perez, and H. Martinez-Barbera. Robust feedback control of zmp-based gait for the humanoid robot nao. *The International Journal of Robotics Research*, 32(9-10):1074–1088, Aug 2013.

[2] J. J. Alcaraz-Jimenez, M. Missura, H. Martınez-Barbera, and S. Behnke. Lateral disturbance rejection for the nao robot. In X. Chen et al., editors, *RoboCup 2012*, volume 7500 of *LNAI*, page 1–12. Springer, 2013.

[3] S. Barrett, K. Genter, T. Hester, M. Quinlan, and P. Stone. Controlled kicking under uncertainty. Technical Report in, The Fifth Workshop on Humanoid Soccer Robots (HSR-10), Nashville, TN, December, 2010.

[4] F. Faber and S. Behnke. Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In *Proceedings of the IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids2007), Pittsburgh, PA, USA*. IEEE, 2007.

[5] R. Ferreira, L. P. Reis, A. P. Moreira, and N. Lau. Development of an omnidirectional kick for a nao humanoid robot. In *IBERAMIA 2012. LNCS (LNAI), vol. 7637, pp. 571-580.*, 2012.

[6] M. Missura and S. Behnke. Online learning and of bipedal and walking stabilization. In *German Journal on Artificial Intelligence (KI), Springer*, 2015.

[7] J. Mullet, T. Laue, and T. Rofer. Kicking a ball – modeling complex dynamic motions for humanoid and robots. In J. R. del Solar, E. Chown, and P. Ploeger, editors, *RoboCup 2010*, volume 6556 of *LNAI*, page 109–120. Springer, 2011.

[8] D. E. Orin and A. Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[9] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower body humanoid. 2011.

[10] T. Rofer, T. Laue, J. Muller, O. Bosche, A. Burchardt, E. Damrose, K. Gillmann, C. Graf, T. J. de Haas, and A. Hartl. B-human team report and code release. Technical report, 2009.

[11] L. Sentis. Synthesis and control of whole-body behaviors in humanoid systems. 2007.

[12] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *in Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[13] N. Shafii, A. Abdolmaleki, R. Ferreira, N. Lau, and L. P. Reis. Omnidirectional walking and active balance for soccer humanoid robot. In *Progress in Artificial Intelligence, 2013, pp. 283-294*, 2012.

[14] S. N. University. srlib - snu robotics library, Nov. 2008.

[15] F. Wenk and T. Rofer. Online generated kick motions for the nao balanced using inverse dynamics. In *RoboCup 2013*, 2013.

[16] Y. Xu and H. Mellmann. Adaptive motion control: Dynamic kick for a humanoid robot. In *KI 2010: Advances in Artificial Intelligence, ser. Lecture Notes in Computer Science, R. Dillmann, J. Beyerer, U. Hanebeck, and T. Schultz, Eds. Springer Berlin / Heidelberg, 2010, vol. 6359, pp.392–399, 10.1007/978-3-642-16111-745*, 2010.