20th RoboCup International Symposium, Leipzig, Germany, July 2016.

Kick Motions for the NAO Robot using Dynamic Movement Primitives

Arne Böckmann and Tim Laue

Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
E-Mail: {arneboe,tlaue}@informatik.uni-bremen.de

Abstract. In this paper, we present the probably first application of the popular *Dynamic Movement Primitives (DMP)* approach to the domain of soccer-playing humanoid robots. DMPs are known for their ability to imitate previously demonstrated motions as well as to flexibly adapt to unforeseen changes to the desired trajectory with respect to speed and direction. As demonstrated in this paper, this makes them a useful approach for describing kick motions. Furthermore, we present a mathematical motor model that compensates for the NAO robot's motor control delay as well as a novel minor extension to the DMP formulation. The motor model is used in the calculation of the Zero Moment Point (ZMP), which is needed to keep the robot in balance while kicking. All approaches have been evaluated on real NAO robots.

1 Introduction

Kick motions are an essential part of robot soccer. In recent years, the speed of the game has increased a lot with most teams now being able to stably walk at high speeds. Thus, fights for the ball are more common. A flexible kick motion that is able to adapt to different and changing ball locations as well as to different kick speeds on the fly while keeping the robot in balance during pushes from other robots is a huge advantage in such situations.

Several methods to design and execute flexible kick motions have already been developed. For instance, Müller et al. [8] model the kick foot trajectory using hand-crafted piecewise Bézier curves, which are modified on the fly to adapt to different ball positions. However, handcrafting Bézier curves is a complex and time consuming task. Wenk et al. [18] tackle this problem by automatically inferring trajectories based on the ball position, kick velocity, and kick direction. While this method works, it does not allow the user to influence the resulting trajectory, i. e. creating special purpose kicks like backward kicks is not possible.

In this paper, we present a middle ground between the two above-mentioned approaches: A kick motion that can be hand-crafted easily by using kinematic teach-in or be created by a multitude of optimization algorithms while retaining the ability to adapt to different ball positions and kick velocities. This is done by using a modified version of Dynamic Movement Primitives (DMPs) [4] to describe the kick trajectory. During the kick, the robot is dynamically balanced using

a Linear Quadratic Regulator (LQR) with previews to keep the Zero Moment Point (ZMP) inside the support polygon. This involves a new way of estimating the ZMP based on a model of the motor behavior of the NAO.

The remainder of the paper is organized as follows: Section 2 introduces the motor model that is the basis of the ZMP calculation, Sect. 3 explains the ZMP estimation and introduces the balancing algorithm while Sect. 4 introduces DMPs and explains how we use them to model a kick trajectory. Sections 5 and 6 wrap up the paper with an evaluation of the kick motion and a conclusion.

2 Model-Based Motor Position Prediction

As described below, the NAO's motor response delay is usually 30 ms. Thus, the ZMP balancer needs to take into account that the motor will not be at the currently measured position when the current command reaches the motor. Our solution to this problem is to predict the current motor position using a mathematical model and to use this prediction in our control algorithms.

2.1 Determining the NAO's Motor Response Delay

The NAO's motors are position-controlled using the proprietary NaoQi software. It processes the commands and relays them to an ARM-7 micro controller at 100 Hz. The controller distributes the commands over RS-485 to several dsPIC micro controllers which are responsible for controlling the actual motors. Measured motor positions travel back the same chain [3]. This chain together with the slow control rate of 100 Hz induces a delay between sending a command and being able to measure a reaction of the motor.

To determine the actual delay, a motor is moved from a resting position into a random direction and the time between sending the command and measuring a movement is recorded. A movement is registered as soon as the measured motor position deviates from the position that the motor was in when the command was issued. No threshold is used. For measuring, the internal sensor is used. This is done 100 times for each leg motor of four different NAOs, thus, we get 4400 measurements in total. As shown in Fig. 1, the vast majority of motor reactions occurs at 30 milliseconds. The measured average distance of the reactions at 10 and 20 ms is 0.087°. This is below the maximum accuracy of the motors, which is 0.1°. Therefore, we can assume that the measurements at 10 and 20 ms are due to sensor noise. However, the measurements at 40 ms cannot be discarded as noise. Taking a closer look, it seems that some joints in some robots are more prone to responding after 40 ms than others, suggesting that hardware wear or defects might cause a delayed measurement.

Thus, for a fully repaired robot it is safe to assume that the motor response delay is 30 ms. Actually, the delay might be anywhere between 20 and 30 ms, but due to the 100 Hz duty cycle, more precise measurements are not possible.

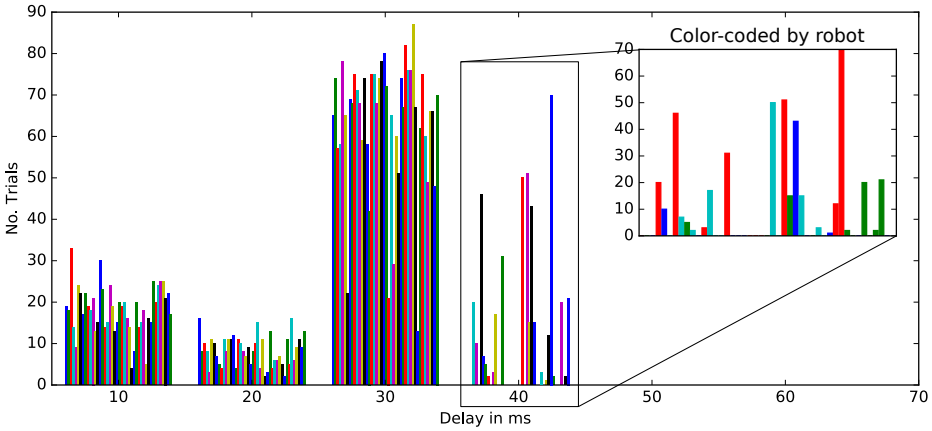


Fig. 1. Histograms of motor response delay. Each bar is one motor of one robot. The zoomed part shows the motors that responded after 40ms, color-coded by robot. The experiment was conducted using the leg motors of four robots and repeating each trial 100 times. Thus, 4400 responses were measured in total.

2.2 A Model to Estimate the Motor Position

We propose to model the behavior of a motor as second order dynamical system based on a mass spring damper:

$$T^2\ddot{y}(t) + 2DTmin(\dot{y}(t), V_{max}) + y(t) = u(t), \quad T, D, y, u \in \mathbb{R} \tag{1}$$

T is the time constant, $y(t)$ is the motor position at time t, D is a dampening constant, V_{max} is the maximum motor velocity that is used to limit $\dot{y}(t)$, and $u(t)$ is the requested motor position at time t.

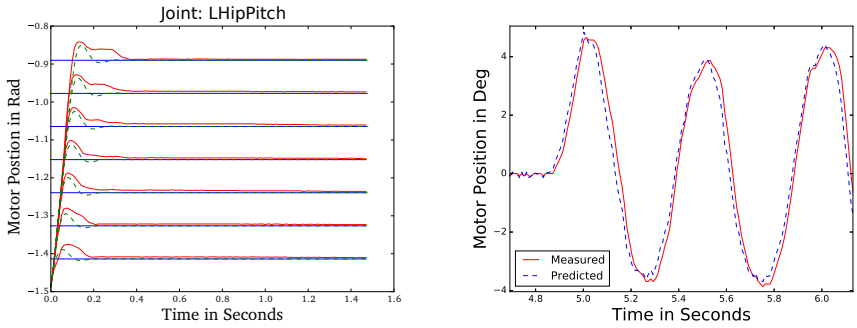
The parameters (T, D, V_{max}) need to be set in a way that the model optimally mimics a motor. This can be achieved by minimizing the error function J:

$$J = \sum_{s \in S} \sum_{i=0}^{|s|} d(i)(m(i) - s(i))^2 \tag{2}$$

S is a set of step responses for a given motor, $|s|$ the number of measurements in step response s, $m(i)$ the position of the model at the i-th step, $s(i)$ the actual motor position at the i-th step and $d(i) = 0.85^i$ a decay function.

The decay function d emphasizes the short term model quality over the long term, i. e. we prefer parameters that provide a better short term prediction over parameters that provide an overall good prediction. This is done because in our use case the model is only used to predict a short amount of time.

Sets of step responses can be generated by applying step functions, which jump from zero to their respective values instantly, with different step heights to the motor. Fig. 2(a) shows a set of step responses that has been recorded and the respective optimal model response. For fitting, the first three samples of the step response should be ignored to make up for the motor response delay.



(a) (b)

Fig. 2. (a) Step responses and their optimal fit. The blue lines are the step functions, red are the step responses, and green the response of the best fitting model. The model of the depicted LHipPitch joint has an error that is slightly above average (see Tab. 1). (b) Prediction and actual motor response of a motor while walking

Joint	Avg. error	Variance
LAnklePitch	0.196°	0.04
LAnkleRoll	0.153°	0.02
LHipPitch	0.216°	0.053
LHipRoll	0.111°	0.009
LHipYawPitch	0.048°	0.003
LKneePitch	0.363°	0.145
RAnklePitch	0.216°	0.054
RAnkleRoll	0.170°	0.025
RHipPitch	0.307°	0.092
RHipRoll	0.097°	0.007
RKneePitch	0.297°	0.118

Table 1. Error in leg motor predictions

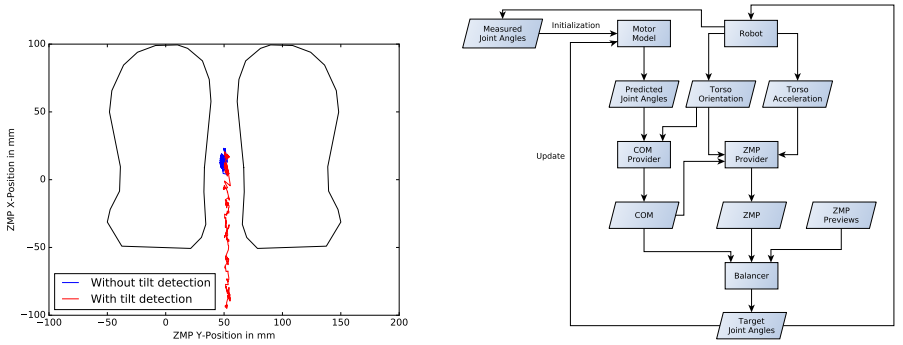
Robot	Avg. error over all leg joints	Variance
Original	0.185°	0.057
NAO 1	0.159°	0.055
NAO 2	0.183°	0.065
NAO 3	0.179°	0.056
NAO 4	0.181°	0.071
NAO 5	0.197°	0.061

Table 2. Errors when applying model parameters that have been optimized for one NAO to five different NAOs

2.3 Model Evaluation

To show that the model can be used to predict the real world motor positions of the NAO, we compared the model response and the actual motor position while executing a five second walking motion. For the comparison, the real motor values have been shifted in time to remove the measurement delay. Fig. 2(b) shows an excerpt of the experiment of the LHipRoll motor. The average absolute error over all leg joints is 0.185° with a variance of 0.0573. Thus, the model seems to be able to predict the motor behavior with sufficient accuracy. Detailed results for each motor can be seen in Tab. 1.

We were also interested in the portability of the model parameters. As documented in Tab. 2, a repetition of the experiment on different robots (always using the same model again) was successful.



(a) (b)

Fig. 3. (a) The red and blue lines show the estimated ZMP position with and without tilt detection while the robot is being tilted backwards. (b) The balancing process.

3 ZMP-based Balancing

The robot needs to be kept in balance while kicking. A good measure for the balance of a robot is the Zero Moment Point (ZMP). A robot is said to be dynamically stable if the ZMP is inside the support polygon [17].

The center of pressure of the support polygon and the Zero Moment Point are coincident [12]. Therefore, it is possible to use the pressure sensors under the NAO's feet to measure the ZMP, if it exists. However, the sensors are quite inaccurate and often faulty. To avoid using these sensors, we estimate the ZMP $(z_x, z_y)^T$ based on the cart-table model proposed by Kajita et al. [5]:

$$\begin{pmatrix} z_x \\ z_y \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \end{pmatrix} - \frac{c_z}{g} \begin{pmatrix} \ddot{c}_x \\ \ddot{c}_y \end{pmatrix} \tag{3}$$

$(c_x, c_y, c_z)^T$ is the center of mass (COM) and $g \approx 9.81$ the gravitational force. Due to the measurement delay and the high sensor noise, we estimate the COM using the motor model and forward kinematics $(c_x^m, c_y^m, c_z^m)^T$. The motor model is initialized using sensor readings at the beginning of the motion. While the kick motion is being executed, the model is not updated from sensor readings.

Tilting the robot over the edges of the supporting foot does not influence the estimated ZMP. To detect such situations, we calculate the scaled difference $P\Delta\Theta = P(\gamma - \phi)$ between the expected torso orientation γ as provided by the motor model and the measured torso orientation ϕ as provided by the IMU and scale the COM accordingly [1]. The unitless constant factor P needs to be adjusted manually. We chose $P = (30, -30)^T$. Fig. 3(a) shows how the ZMP behaves with and without tilt detection.

To be able to measure outside influences, e. g. someone pushing the robot, we replaced the COM acceleration by the acceleration of the torso $\ddot{o}$ as measured by the NAO's IMU. This can be done because the COM is usually inside the torso and thus both accelerations are similar.

Thus, the final ZMP is calculated by:

$$\begin{pmatrix} z_x \\ z_y \end{pmatrix} = P\Delta\Theta \begin{pmatrix} c_x^m \\ c_y^m \end{pmatrix} - \frac{c_z^m}{g} \begin{pmatrix} \ddot{o}_x \\ \ddot{o}_y \end{pmatrix} \tag{4}$$

To finally balance the robot, an LQR preview controller as described in [2, 16, 18] has been implemented. The inputs of the controller are the current ZMP and current COM as well as the next 50 desired ZMP positions. Fig. 3(b) shows an overview of the balancing sub-system.

4 Describing Kicks Using Dynamic Movement Primitives

The kick trajectory is described by using Dynamic Movement Primitives (DMPs) [4, 13]. For the sake of simplicity, this chapter only considers one-dimensional DMPs but they can be easily scaled to n dimensions as long as the dimensions are independent of each other, i. e. for translational movements. For rotational movements, a special DMP formulation has been introduced by Ude et al. [15]. However, for kick motions it is sufficient to simply keep the foot level to the ground, thus no rotational DMP was used in this paper.

DMPs model goal-directed movements as weakly non-linear dynamical systems. They consist of the canonical system and the transformation system.

The *canonical system* s describes the phase of the movement:

$$\tau\dot{s} = -\alpha_s s \tag{5}$$

The phase s replaces the time in the transformation system. Intuitively, it drives the transformation system similar to a clock [9]. s conventionally starts at 1 and monotonically converges to zero. τ is the execution time of the movement. As long as τ remains constant, a closed solution for the canonical system exists [9]:

$$s(t) = exp(\frac{-\alpha_s}{\tau}t) \tag{6}$$

α_s determines how fast s converges. The value of α_s has to be chosen in a way that s is sufficiently close to zero at the end of the execution. We chose α_s by setting $s(\tau) = 0.01$ and solving (6) for α_s.

The *transformation system* is defined by two first order differential equations:

$$\tau\dot{z} = \alpha_z(\beta_z(g - y) - z) + sf(s) \tag{7}$$

$$\tau\dot{y} = z \tag{8}$$

τ is the execution time of the movement, α_z and β_z are damping constants, g is the goal position of the movement, y is the current position of the movement, s is the current phase as defined by eq. 5 and $f(s)$ is called the forcing term.

The dampening constants are set for critical dampening, i. e. $\beta_z = \alpha_z/4$. We chose $\beta_z = 25$ and $\alpha_z = 6.25$, but the exact values do not matter as long as the system is critically dampened.

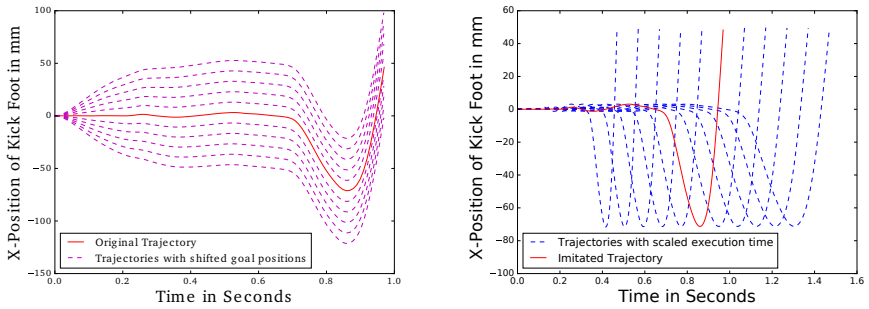


Fig. 4. (a) Behavior of the kick motion when the goal position is adapted. (b) Temporal scaling capability of the DMP.

The forcing term defines the movement's shape. With $f(s) = 0$, the system is just a PD controller converging to g and reaching it at time τ. One could say that f superimposes its shape onto the PD controller. To ensure that f does not keep the system from reaching the desired goal position, f is scaled by the phase, thus diminishing the influence of f towards the end of the movement. To be able to express arbitrary movements, f is typically chosen to be a radial basis function approximator:

$$f(s) = \frac{\sum_{i=1}^{N} \psi_i(s) w_i}{\sum_{i=1}^{N} \psi_i(s)}, w_i \in \mathbb{R} \tag{9}$$

ψ_i is the i-th Gaussian radial basis function with mean c_i and variance σ_i^2:

$$\psi_i(s) = exp(-\frac{1}{2\sigma_i^2}(s - c_i)^2) \tag{10}$$

The weights w_i can be chosen to create any desired function and thus define the shape of the whole movement. Different learning and optimization approaches can be used to find the weights for a certain movement, e. g. Schaal et al. [14] describe how to imitate a given trajectory.

Since the goal position g is part of eq. 7, the shape also depends on g. This means that the weights can only force the system into a certain shape for one specific value of g. When g changes, the shape "bends" to reach the new goal position. In general, this is undesired behavior and is solved by scaling f if g changes. However, we found that kick motions "bend" in a natural way (Fig. 4), i. e. if the goal position is moved closer to the start position, the robot will swing further back and vice versa. This is exactly the behavior that one would expect from a kicking motion because it ensures that the distance between the inflection point and the goal position remains the same. This is important because the kicking velocity depends on this distance.

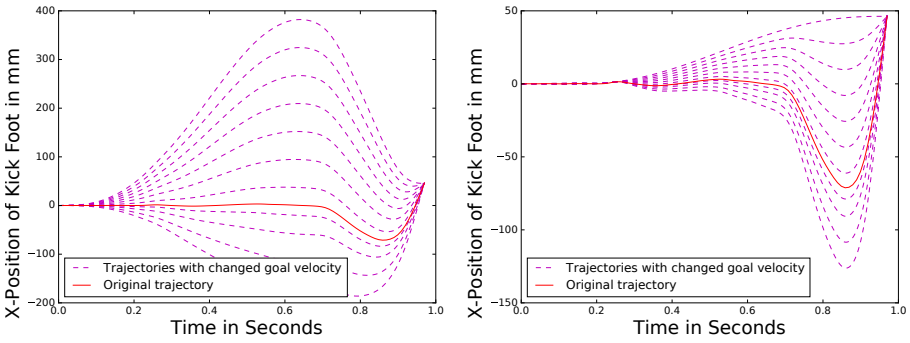


Fig. 5. Reaction of the DMP to changes in the final velocity. (a) Shows the reaction of the original DMP while (b) shows how the DMP reacts with our new scaling term A.

Thus, the DMP formulation allows us to define an arbitrary movement in task space and later scale its execution time as well as to move the goal position while retaining a sane shape.

A major downside of the formulation is that the final velocity is always zero making it unsuitable for kick motions because kick motions need to reach the target with a specific velocity. Solutions to this problem were proposed by Kober et al. [6] and Mülling et al. [9]. They replaced the goal g in eq. 7 with the position, velocity and acceleration of a moving target g_p.

$$\tau \dot{z} = \alpha_z(\beta_z(g_p - y) + \tau \dot{g_p} - z) + \tau^2 \ddot{g_p} + sf(s) \tag{11}$$

While Kober et al. used a target that is moving on a straight line, Mülling et al. used a fifth order polynomial.

$$g_p(t) = \sum_{i=0}^{5} b_i t, \qquad \dot{g_p}(t) = \sum_{i=i}^{5} i b_i t^{i-1}, \qquad \ddot{g_p}(t) = \sum_{i=2}^{5} (i^2 - i) b_i t^{i-2} \tag{12}$$

The coefficients b_i are calculated by applying the bounding conditions:

$$g_p(t_0) = y_0, \quad \dot{g_p}(t_0) = \dot{y_0}, \quad \ddot{g_p}(t_0) = \ddot{y_0} \tag{13}$$

$$g_p(\tau) = g, \quad \dot{g_p}(\tau) = \dot{g}, \quad \ddot{g_p}(\tau) = 0 \tag{14}$$

Due to the time dependency, the coefficients need to be recalculated if τ changes.

In this way, a new parameter $\dot{g}$ is introduced. It represents the velocity at the end of the movement. However, the weights now depend on $\dot{g}$ as well. This means that if the goal velocity is changed, the shape of the movement will change. As shown in Fig. 5(a), the trajectory reacts to changes in the goal velocity with huge changes and becomes inexecutable. We propose to fix this by scaling the forcing term with the novel factor $A = (\dot{g}_{new} - \dot{y}_0)/(\dot{g} - \dot{y}_0)$, where $\dot{y}_0$ is the starting velocity of the trajectory and $\dot{g}_{new}$ is the new goal velocity. Fig. 5(b) shows that

	B-Human left	Imitated left	B-Human right	Imitated right
Avg. distance	5.3 m	4.3 m	4.61 m	3.6 m
Avg. angular deviation	5.62°	8.06°	9.69°	7.48°
Avg. ball location (cm)	(-2.53, 496.93)	(-1.63, 430.03)	(7.52, 453.53)	(4.41, 361.56)
Royston H-value	4.48	0.407	1.62	3.08
Royston p-value	0.106	0.812	0.43	0.000062
Is normal distributed	Yes	Yes	Yes	No

Table 3. Kick distance comparision between the B-Human kick motion of 2015 and an imitation of that motion. The data for the imitated kick with the right leg contained two outliers. If those outliers are removed the result is normally distributed as well.

this produces much better results. If the velocity is increased, the wind up phase gets longer, if it is reduced, the wind up phase gets shorter until it completely disappears if the requested goal velocity is zero. This is exactly the behavior that one would expect from a kick motion.

Thus, the final form of the DMP used in our experiments is:

$$\tau \dot{z} = \alpha_z(\beta_z(g_p - y) + \tau \dot{g_p} - z) + \tau^2 \ddot{g_p} + sf(s)A \tag{15}$$

$$\tau \dot{y} = z \tag{16}$$

$$\tau \dot{s} = -\alpha_s s \tag{17}$$

This DMP responds well to changes in goal position and goal velocity. It is noteworthy that both parameters can be changed mid-execution without causing discontinuities. The implementation used in our experiments has been released as part of the B-Human Code Release 2015 [11] and is available online[1].

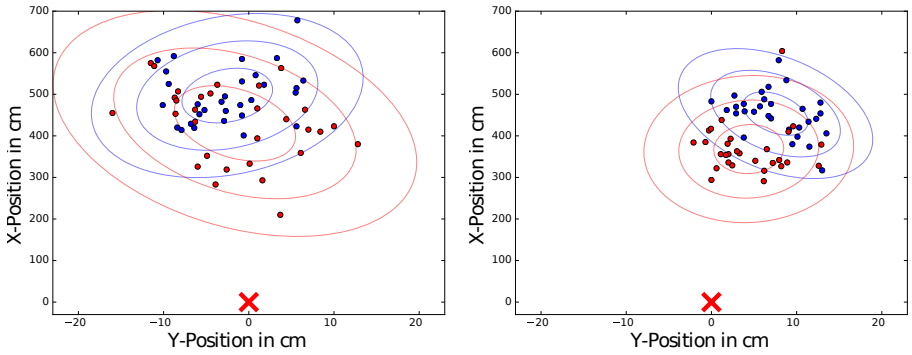
5 Evaluation

Several experiments have been done to evaluate the kick motion. The setup is identical for all experiments: The robot is standing at the side line of the field and kicks the ball into the field, as depicted in Fig. 7. All experiments have been done with the official RoboCup SPL ball of 2015, a Mylec street hockey ball that is 65 mm in diameter and weighs 55 g. Each experiment consists of 30 kicks.

We used the Royston H-Test [10] with a significance level of 0.05 to determine that the measured kick distances are normally distributed. Normally distributed kick results indicate that the results have only been influenced by natural noise, i. e. there is probably no systematic error in the test setup or the implementation.

To compare the performance of the kick motion to an existing one, we used imitation learning [14] to learn weights that imitate the kick motion of team B-Human of 2015 [8,11] and executed 30 kicks with each leg and each kick motion. The results can be seen in Tab. 3 and Fig. 6.

[1] `https://github.com/bhuman/BHumanCodeRelease/tree/master/Src/Tools/Motion`



(a) (b)

Fig. 6. Kick positions of the B-Human and imitated kick motions: The red cross is the kick origin. The dots are the positions where the balls came to a halt. The blue dots originate from the B-Human kick, the red dots from the imitated kick. (a) shows the results of kicks with the left foot while (b) shows the right foot.

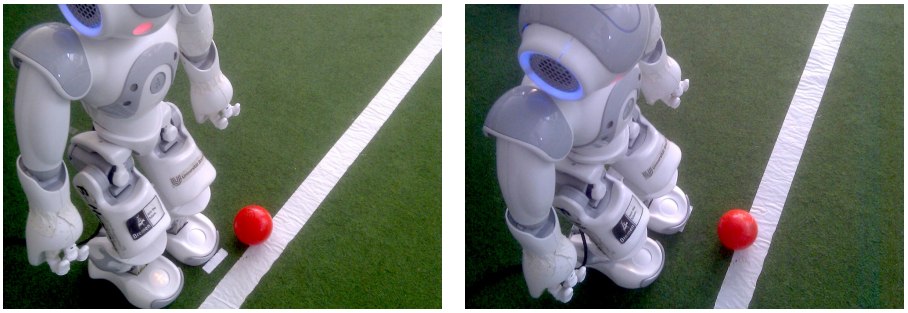
	Left ball	Forward ball	3/4 speed	1/2 speed
Avg. distance	1.31 m	3.29 m	2.4 m	1.83 m
Avg. angular deviation	5.91°	4.98°	5.83°	4.73°
Avg. ball location (cm)	(12.63, 130.93)	(22.87, 328.0)	(5.44, 239.16)	(2.96, 182.90)
Royston H-value	6.17	1.15	3.65	3.43
Royston p-value	0.04	0.56	0.144	0.142
Is normal distributed	No	Yes	Yes	Yes

Table 4. Kick distance results for generalized kicks. The data of the left generalization contained one outlier. If it is removed, the result is normally distributed.

To test the generalization qualities of the kick motion, we conducted four experiments with different ball positions and velocities. In the first experiment, the ball is positioned 65 mm to the left. In the second experiment it is moved 80 mm forward, the third and fourth experiment reduced the kick velocity by 1/4 and 1/2 of the original kick velocity respectively. The results can be seen in Tab. 4. To reach the position of the left ball, the robot had to fully stretch the leg. Therefore, the knee motor could not be used to generate a forward force, thereby significantly reducing the reached kick distance. The other experiments show a reasonable scaling towards the desired kick distance. Videos showing the kick generalization can be found at `https://youtu.be/g73pPCWcQvw` and `https://youtu.be/eANtiAiMmTg`.

6 Conclusion

We presented a kick motion for the NAO robot that can imitate arbitrary kick trajectories and adapt them to different ball positions as well as different kick



(a) (b)

Fig. 7. Ball position generalization experiments: In (a) the ball was moved 65 mm to the left, in (b) it was moved 80 mm to the front.

velocities. The kick motion is modeled using a slightly modified variant of DMPs. While executing the kick, the robot is kept dynamically stable using a ZMP preview controller. Additionally, we proposed a model of the NAO's motors and used it to improve the calculation of the ZMP.

We have shown that this method of generating kick motions works but cannot kick the ball as far as a manually tuned motions. However, they are more versatile. The ball does not need to be placed perfectly to be kicked and the kick speed can be adjusted. Additionally, the underlying DMPs are easy to extend and lend themselves well to a multitude of optimization algorithms [7].

The kick motion presented in this paper has been successfully used in the corner kick challenge competition at RoboCup 2015.

Acknowledgement

We would like to thank the members of the team B-Human for providing the software framework for this work.

References

1. Alcaraz-Jiménez, J.J., Herrero-Pérez, D., Martínez-Barberá, H.: Robust feedback control of ZMP-based gait for the humanoid robot Nao. The International Journal of Robotics Research 32(9-10), 1074–1088 (2013)
2. Czarnetzki, S., Kerner, S., Urbann, O.: Applying dynamic walking control for biped robots. In: RoboCup 2009: Robot Soccer World Cup XIII, Lecture Notes in Computer Science, vol. 5949, pp. 69–80. Springer (2010)
3. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009). pp. 2124–2129. Kobe, Japan (2009)

4. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural computation 25(2), 328–373 (2013)
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: Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003). vol. 2, pp. 1620–1626. Taipei, Taiwan (2003)
6. Kober, J., Mülling, K., Krömer, O., Lampert, C.H., Scholkopf, B., Peters, J.: Movement templates for learning of hitting and batting. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA 2010). pp. 853–858. Anchorage, Alaska, USA (2010)
7. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32(11), 1238–1274 (2013)
8. Müller, J., Laue, T., Röfer, T.: Kicking a ball – modeling complex dynamic motions for humanoid robots. In: RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes in Artificial Intelligence, vol. 6556, pp. 109–120. Springer (2011)
9. Mülling, K., Kober, J., Krömer, O., Peters, J.: Learning to select and generalize striking movements in robot table tennis. The International Journal of Robotics Research 32(3), 263–279 (2013)
10. Royston, J.: Some techniques for assessing multivarate normality based on the Shapiro-Wilk W. Journal of the Royal Statistical Society. Series C (Applied Statistics) 32(2), 121–133 (1983)
11. Röfer, T., Laue, T., Richter-Klug, J., Schünemann, M., Stiensmeier, J., Stolpmann, A., Stöwing, A., Thielke, F.: B-Human Team Report and Code Release 2015 (2015), only available online: `http://www.b-human.de/downloads/publications/2015/CodeRelease2015.pdf`
12. Sardain, P., Bessonnet, G.: Forces acting on a biped robot. center of pressure – zero moment point. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 34(5), 630–637 (2004)
13. Schaal, S.: Dynamic movement primitives – a framework for motor control in humans and humanoid robotics. In: Adaptive Motion of Animals and Machines, pp. 261–280. Springer (2006)
14. Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.: Control, planning, learning, and imitation with dynamic movement primitives. In: Proceedings of the workshop on bilateral paradigms on humans and humanoids, IEEE International Conference on Intelligent Robots and Systems (IROS 2003). pp. 1–21. Las Vegas, Nevada, USA (2003)
15. Ude, A., Nemec, B., Petric, T., Morimoto, J.: Orientation in cartesian space dynamic movement primitives. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014). pp. 2997–3004. Hong Kong, China (2014)
16. Urbann, O., Tasse, S.: Observer based biped walking control, a sensor fusion approach. Autonomous Robots 35(1), 37–49 (2013)
17. Vukobratović, M., Borovac, B.: Zero-moment point -— thirty five years of its life. International Journal of Humanoid Robotics 01(01), 157–173 (2004)
18. Wenk, F., Röfer, T.: Online generated kick motions for the NAO balanced using inverse dynamics. In: RoboCup 2013: Robot Soccer World Cup XVII. Lecture Notes in Artificial Intelligence, vol. 8371, pp. 25–36. Springer (2014)