Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots

Judith Müller[1], Tim Laue[2], and Thomas Röfer[2]

[1] Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
judy@informatik.uni-bremen.de
[2] Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. Complex motions like kicking a ball into the goal are becoming more important in RoboCup leagues such as the Standard Platform League. Thus, there is a need for motion sequences that can be parameterized and changed dynamically. This paper presents a motion engine that translates motions into joint angles by using trajectories. These motions are defined as a set of Bezier curves that can be changed online to allow adjusting, for example, a kicking motion precisely to the actual position of the ball. During the execution, motions are stabilized by the combination of center of mass balancing and a gyro feedback-based closed-loop PID controller.

1 Introduction

For successfully playing soccer with humanoid robots, much attention is paid to the development of not only fast but also robust gaits, i. e., to be flexible regarding sudden changes of the walking direction and to be able to compensate for disturbances such as collisions with other robots or the asperity of the ground floor. These two kinds of robustness are also a requirement for the second most important kind of motion in soccer: kicking. During the execution of a kicking motion, the position of the ball might change, either in reality after having been touched by a robot, or virtually in the kicking robot's world model after having perceived the ball at a slightly different place. In these cases, an online adaption of the kicking foot's trajectory becomes necessary to precisely hit the ball. In addition, even slightest disturbances during the kick phase might cause a loss of precision or even prevent a successful kick at all. Thus, balancing mechanisms for a robot standing on one foot only are a crucial factor.

In [5], a key-frame based approach that is one of the most common methods for motion designing is described. This method defines motions as a series of static joint angle sets. Each joint angle set is a key-frame. All joint positions between two key-frames are interpolated. The major disadvantage of this approach is its inflexibility. Once a motion execution has started, the robot cannot react on any new information. Similar approaches are currently used for kicking motions by

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 109–120, 2011.

many RoboCup teams such as *Cerberus* [1], *Nao Team HTWK* [12], *Kouretes* [15], or *B-Human* [17].

In [7], the static key-frame based approach is extended by the possibility of balancing to compensate external disturbances. In this concept, key-frames are defined as Cartesian limb positions in between which it is interpolated; the actual joint angles are calculated by inverse kinematics. A similar approach is used by the *Nao Team Humboldt* [4], where kicking motions are also defined by Cartesian positions, and in addition, they are stabilized.

The contribution of this paper is the concept of a motion engine that extends the idea of trajectory-based motions of current walking approaches such as [17] and [14] to execute more complex motions such as kicking a ball. In order to stabilize the resulting motions, the engine contains a balancing module that is based on center of mass stabilization of [6] and a sensor feedback closed-loop PID controller based on [9] and [2]. The approach has been applied to a *Nao* humanoid robot as it is used in the RoboCup Standard Platform League.

This paper is organized in four main sections: section 2 introduces the main concept and the procedure of the motion engine. After that, Sect. 3 describes the definition of related motions and the concept of dynamic changes. The components providing stability are described in Sect. 4. Finally, in Sect. 5 the experiments conducted are presented and the results are discussed.

2 Motion Engine Design

The motion engine is developed as a module of the framework introduced in [17] and can be used in parallel to other modules such as a walking engine. The main idea behind the engine is to combine a set of simple motion curves to more complex curves. That way a whole motion is divided into *motion phases* such as lifting the foot or kicking the ball. Simple curves define the motion of a limb of the robot for a certain period of time. They are also referred to as trajectories. A set of simple motion curves or *motion phases* contains six different curves. Each foot and each arm has its own trajectory to control its position. Each foot has an additional trajectory to control the rotational movement.

Since simple trajectories are combined to more complex trajectories, the engine has to make sure that the combination of trajectories of two different phases is smooth to prevent unwanted twitching. The system guarantees smoothness by checking whether the connection point of two curves is continuously differentiable (cf. Sect. 3).

The motion engine receives a list of dynamic points as input from other modules to accomplish dynamic changes. Dynamic points contain information about the desired target position of certain curves and their motion direction (cf. Sect. 3.2). Since these points provide the system with information as, for example, the ball position, they are applied at times during motion execution.

As each leg of the robot contains six different joints, the unknown joint angles are calculated by inverse kinematics. The actual calculation is based on a geometric approach that was described by [11].

Algorithm 1 shows the general structure of the motion engine: first the motion definitions of the requested motion *id* are retrieved. Then, at the start of each phase, the dynamic points are applied to the current phase and the trajectories are initialized. Besides dynamically changing, in this step the smoothness of the curves of the current phase is adapted. Afterwards, the limb positions are retrieved from the current point in time and the trajectories. Since the position of the end effector is known, the joint angles are calculated using inverse kinematics. The final step is to apply the balance corrections.

Algorithm 1. The main procedure of the motion engine

```
1: if wasActive ≠ true then
2:    phase ← 0
3:    currentParameters ← getParameters(id)
4:    addDynValues(phase, currentParameters, dynValues)
5:    initCurrentTrajectories(phase, currentParameters)
6: end if
7: if phase ≤ maxPhase then
8:    time ← getTime(getCurrentTime(), getDeltaT(currentParameters))
9:    if time == 1 then
10:      phase ← phase + 1
11:      addDynValues(phase, currentParameters, dynValues)
12:      initCurrentTrajectories(phase, currentParameters)
13:      time ← 0
14:   end if
15:   positions ← getLimbPos(currentParameters, phase, time)
16:   joints ← calcJoints(positions)
17:   addBalance(joints, currentParameters, gyroData)
18:   wasActive ← true
19:   return joints
20: else
21:   wasActive = false
22:   return stand
23: end if
```

3 Motion Design

In many applications, trajectories are mostly combined equations that are selected by a case differentiation over time. These equations are continuous but not continuously differentiable as the derivative is not defined at the connection points. This means that a real servo motor is supposed to change its speed in no measurable time, which causes a twitch.

A better suiting equation for motion controlling is the Bezier curve. Such a curve is defined through $n+1$ control points. Each of these control points affects the whole curve. Bezier curves can be combined easily and even the connection points between two curves are continuously differentiable under certain conditions. The motion engine presented in this paper uses combined cubic Bezier curves ($n = 3$) as shown in equation 1.

$$b(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} P_i \tag{1}$$

3.1 Combined Trajectories

A motion is a set of phases and a phase is a set of six trajectories. Each trajectory controls a limb, e. g. the left foot. That means that if a motion has two phases, the motion of the left foot is described by two trajectories. Such a combination of two trajectories has to satisfy some conditions for continuous differentiability in the connection point.

The connection point of two Bezier curves is the fourth control point of one curve and the first control point of the next curve. The forth control point of a curve $b_1(t_1)$ is reached if $t_1 = 1$ for $0 \leq t_1 \leq 1$ and the first control point of a curve $b_2(t_2)$ if $t_2 = 0$ for $0 \leq t_2 \leq 1$. For that reason, two Bezier curves $b_1(t_1)$ and $b_2(t_2)$ with the control points P_0, P_1, P_2, P_3 and Q_0, Q_1, Q_2, Q_3 have a connection point if $b_1(1) = b_2(0)$.

If Δt_1 and Δt_2 are equal, i. e., the timings of both phases are equal, continuously differentiability is given if P_2, P_3, and Q_1 of $b_1(t_1)$ and $b_2(t_2)$ are collinear and equidistant [10]. This means that those three points are on a straight line and the distance between P_2 and P_3 equals the distance of P_3 and Q_1 with $P_3 = Q_0$.

If $P_3 = Q_0$, equation 1 is valid:

$$b_1(1) = b_2(0) \tag{2}$$

The tangents in the points P_3 and Q_0 are calculated using the first derivative of the Bezier curve:

$$\dot{b_1}(1) = 3 \cdot (P_3 - P_2) \tag{3}$$

$$\dot{b_2}(0) = 3 \cdot (Q_1 - Q_0) \tag{4}$$

Equation 3 and 4 applied to 2 provides the condition to guarantee continuous differentiability in the connection point between two Bezier curves with equal timings [10]:

$$P_3 - P_2 = Q_1 - P_3 \tag{5}$$

If the timings of two Bezier curves are not equal, i. e., when Δt_1 and Δt_2 are not equal, then Eq. 5 is insufficient. According to [10], the connection point of two Bezier curves with different timings is continuously differentiable if the points P_2, P_3 and Q_1 are collinear and the ratio of the distance between P_2 and P_3 and the distance of P_3 and Q_1 equals the ratio between Δt_1 and Δt_2. Equation 6 integrates that condition into equation 5.

$$\frac{P_3 - P_2}{\Delta t_1} = \frac{Q_1 - P_3}{\Delta t_2} \tag{6}$$

Since the phases have an order, the system only has to calculate point Q_1 using Eq. 6 for each phase except for the first one. In the first phase, all control points can be set without conditions as long as the control points in the following phases are calculated correctly.

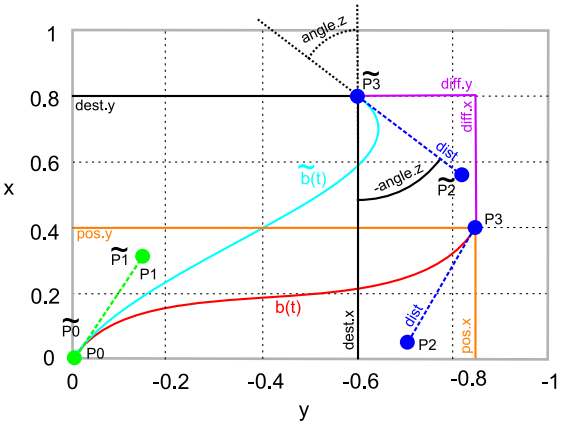

Fig. 1. Applying a dynamic point to curve $b(t)$

3.2 Dynamic Motions

The engine allows changing motions online. Thus, it receives parameters from other components such as the robot's behavior control that can affect a motion specified offline. These parameters are a target position, a motion direction, the phase number, and the limb name. The phase number and the limb name address the curve that should be changed and the target position modifies the fourth control point of that curve. The motion direction rotates the third control point of a curve around the fourth control point to change a forward kick for example into a sideways kick.

The dynamic points are added to the curves at the start of the execution of the phase, not at the start of the execution of the whole motion. This implies that a motion can be changed during its execution, e. g., to follow a ball with the foot.

The actual addition of a dynamic point to a curve $b(t)$ is moving the forth control point P_3 to the desired position and the rotation of the third control point P_2 to the desired angle or motion direction. In Fig. 1, two curves $b(t)$ and $\tilde{b}(t)$ are shown. Curve $b(t)$ is the curve created offline and $\tilde{b}(t)$ is the corresponding curve changed dynamically. When P_3 is moved to the target position, P_2 has to be translated by the same amount to keep the continuous differentiability of the connection point. After the translation of P_3, P_2 is rotated around P_3.

4 Stability

Center of mass (COM) based approaches are already in use for the creation of walking gaits, e. g. by [14] who use a trajectory to set the body motion and thus stabilize the gait. The approach presented in [13] introduces a concept that – as *foresighted car driving* – not only considers the actual location of the COM but also the future movements. However, it is not possible to absorb external disturbances using the COM only.

For that reason, a second balancing module was added to the system to compensate for external disturbances. The approach for the secondary module is inspired by [9] and [2]. Disturbances become compensated by the use of a closed loop PID controller that uses gyroscope measurements as input.

4.1 Center of Mass Balancing

Since the robot used is a biped, there are three different stand cases: standing on two feet, standing on the left foot, and standing on the right foot. The desired COM depends on these stand cases. The COM position projected to the ground should always be inside the support polygon to maintain static stability [6]. This support polygon equals the size of the standing foot. In this approach, a sufficient stability is assumed if the COM projected to the ground is at the center of the standing foot.

The calculation of the desired COM projected to the ground is realized by computing the harmonic mean of the n future positions of the stand foot. This lets the desired COM approach the next stand case early, and the COM adaption can take place before the feet actually move.

To realize stabilization, the robot's tilt angle is controlled by a PID controller. The input for this controller is the error between the measured COM projected to the ground and the desired COM. The calculation of the measured COM is made by equation 7 that is obtained from [16].

$$\overrightarrow{com} = \frac{\sum_{i=0}^{n} \overrightarrow{part_i} m_i}{m_{total}} \tag{7}$$

The manipulated variables obtained by the PID controller using equations 8a and 8b cannot directly be used as tilt angles for the body, because the height of the COM is not included.

$$balance_{y_t} = k_{py} * error_{y_t} + k_i \cdot \int_0^t error_\tau d\tau + k_{dx} \cdot \frac{derror_{y_t}}{dt} \tag{8a}$$

$$balance_{x_t} = k_{px} * error_{x_t} + k_i \cdot \int_0^t error_\tau d\tau + k_{dy} \cdot \frac{derror_{y_t}}{dt} \tag{8b}$$

Equations 9a and 9b include the COM height using the Pythagorean theorem. The angles obtained state the rotation of the foot around the origin of the leg, which is considered by the inverse kinematics.

$$\theta_{x_t} = \text{atan2}(balance_{x_t}, com_{z_t}) \tag{9a}$$

$$\theta_{y_t} = \text{atan2}(balance_{y_t}, com_{z_t}) \tag{9b}$$

4.2 Gyroscope Feedback-Based Balancing

Our approach is based on [9] who use a P-controller with the measured data of a gyroscope as input to compensate for external disturbances. The manipulated variables of the approach are directly used to control the servo motors used for balancing. Since gyroscopes are measuring the angular velocity independently from body tilt and roll, they can only be used to keep an angular velocity.

As extension to the method introduced in [9], our approach uses a PID controller instead of a P controller. The PID controller is distinct from the P controller as it is able to eliminate the entire control deviation and it is consequently more reliable in this context. Furthermore, the approach of [9] suggests that stabilization is given when the rotational velocity of the body is kept by $\frac{0^\circ}{s}$. This is not applicable to our approach as the body has to move during the COM balance stabilizing. It has to be distinguished between desired angular velocity and undesired angular velocity to allow the two balancing components to work together.

The desired angular velocities are taken from the commanded angles of the balancing joints of the standing leg with Eq. 10a and Eq. 10b before the gyro feedback balance is added. The joints responsible to compensate for disturbances are the hip pitch joints and the hip roll joints. It is important to use the target angles without any gyro feedback balancing since to obtain the desired angular velocity, there should not be any velocity included, which is introduced by the disturbance compensation. Otherwise, the desired velocity becomes the velocity that is used to compensate for disturbances in the next frame and the system tries to keep the new velocity that is added to the desired velocity in the next frame and so on.

$$ref_{x_t} = \frac{\theta_{joint_{hipRoll_t}} - \theta_{joint_{hipRoll_{t-1}}}}{\Delta time_t} \tag{10a}$$

$$ref_{y_t} = \frac{\theta_{joint_{hipPitch_t}} - \theta_{joint_{hipPitch_{t-1}}}}{\Delta time_t} \tag{10b}$$

Since the gyroscopes of the robot used are inert, our approach uses exponential smoothing to forecast the next possible angular velocity from previous measurements. Eq. 11 calculates a forecast that is factorized by α. It uses the factorized forecast of the last frame and adds it to the current factorized measurement. This equation is defined recursively. Newly made measurements are weighed more than measurements taken a while ago.

$$\tilde{g}_{x_t} = g_{x_t} \cdot \alpha + \tilde{g}_{x_{t-1}} \cdot (1 - \alpha) \tag{11}$$

The desired angular velocity is also known as the reference velocity that should be kept. Thus, the velocity error is obtained by the subtraction of the desired velocities from the measurements of the gyroscopes. This is done by Eq. 12.

$$error_{x_t} = \tilde{g}_{x_t} - ref_{x_t} \tag{12}$$

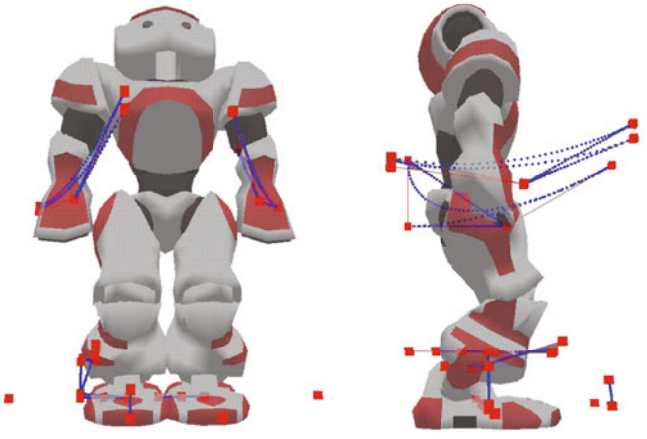

Fig. 2. Visualization of a basic kick motion

The velocity error serves as input for the PID controller in Eq. 13a and Eq. 13b and the manipulated variables serve as offset angles – that become added directly – for the balancing joints (hip pitch and hip roll).

$$\theta_{offset_{hipRoll_t}} = K_p \cdot error_{x_t} + K_i \cdot \int_0^t error_{x_\tau} d\tau + K_d \cdot \frac{derror_{x_t}}{dt} \tag{13a}$$

$$\theta_{offset_{hipPitch_t}} = K_p \cdot error_{y_t} + K_i \cdot \int_0^t error_{y_\tau} d\tau + K_d \cdot \frac{derror_{y_t}}{dt} \tag{13b}$$

5 Experiments

Figure 2 visualizes a kick motion created for the motion engine introduced in this paper. The motion is divided into seven phases: shift the masses, lift the foot, strike out, kick the ball, take the foot back, lower the foot, and shift the masses back.

The phases *strike out* and *kick the ball* of this motion can be changed online. In Fig. 3 the motion is changed dynamically to hit the ball at the center to get a straight kick. The dynamic parameters for that motion depend on the ball position and the target angle. In Fig. 3 top the desired target angle was 0°. In Fig. 3 bottom, the same motion that was used for the straight kick was changed online at the phases *strike out* and *kick the ball* to kick the ball from the side. The target angle was 90°.

5.1 Angular Deviation Experiment

An experiment with 25 trials of kicking a ball to a target angle of 0° was made to evaluate how effective the straight kick can be. In each trial, the robot was

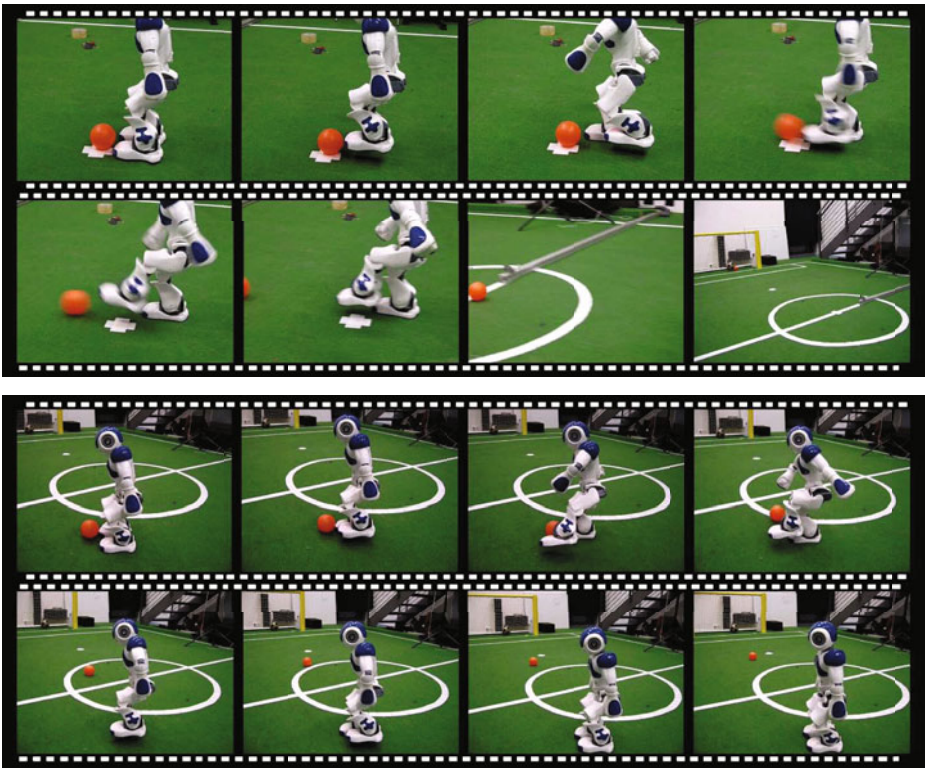

Fig. 3. The dynamic forward kick (top) and the dynamic side kick (bottom) are sharing the same basic motion

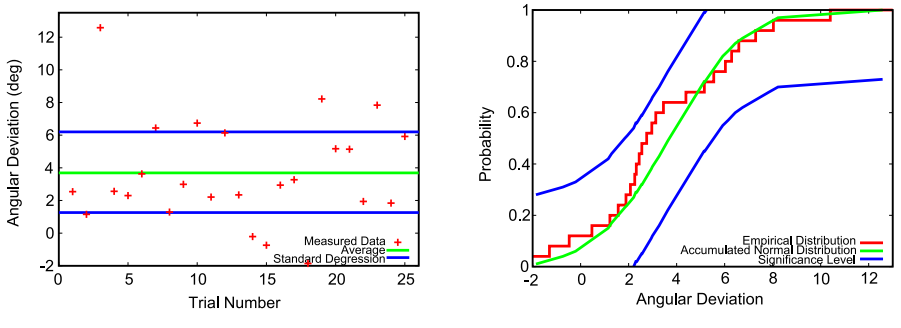

Fig. 4. The measured angular deviations with average (3.32°) and standard deviation (2.11°) (left); Check whether current distribution is normally distributed (right)

set about 30 cm away from the ball but with the face pointing to it. The robot had to go to the ball on its own for reaching a random kick position without human interaction. The ball used was a standard orange tennis ball.

The results of the angular deviations of each of the 25 trials are presented in Fig. 4 left. The overall average (in green) is 3.32° and the standard deviation (in blue) is 2.11°. Furthermore, 16 of 25 samples are within the standard deviation. Since a normal distribution is very reasonable if 68% of the samples are within the standard deviation, a Kolmogorow-Smirnow test for normal distribution was made.The test was made with a significance level of 0.05 and the result is presented in Fig. 4 right. The test is positive, when none of the samples are outside of the tolerance. Since the test proves that the experiment result is normally distributed, it is very likely that there were no unusual disturbances. Furthermore, the test implies that 68% of all kicked balls have an angular deviation within the range of 1.21° to 5.43°. Since normally distributed samples have the characteristic to differ to 95% at most 2σ from the average, the kicked balls have an angular deviation within the range of -0.9° to 7.54° by a chance of 95%.

Besides the angular deviation experiment the maximum distance the kicked balls reached was also measured. The overall average of 25 samples is 5435.65 mm and the standard deviation is 251.03 mm. Since the samples of this test are normal distributed, the kicked balls reach a maximum distance within the range of 4933.59 mm and 5937.65 mm by a chance of 95%.

5.2 Pendulum Experiment

Besides tests about the effectiveness of motions, a pendulum experiment was conducted to evaluate the stability. In this experiment, a pendulum with the weight of 500g was attached to a rope. The rope with the length of 45 cm was attached to a rack with a height of 80 cm. The pendulum is held in a 90° angle relative to the robot. After releasing the pendulum, it hits the robot at a height of 35 cm from the ground. This experiment includes ten trials with balancing

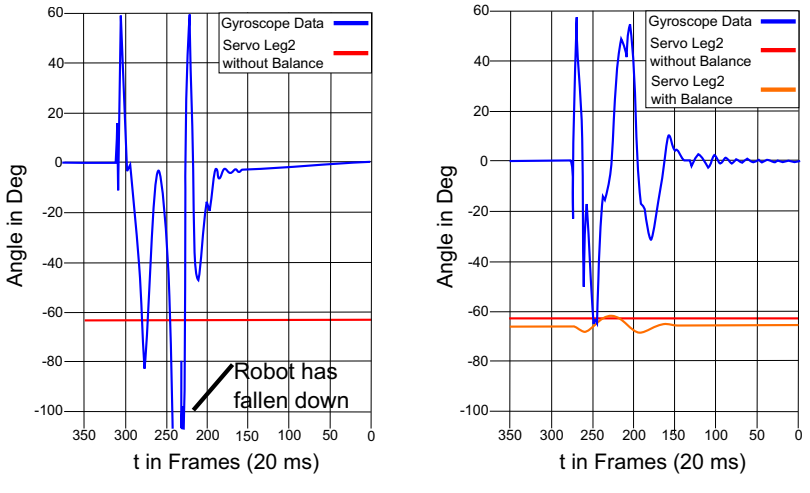

Fig. 5. The plotted data without balance (left) and with balance (right)

and ten trials without. The robot is standing on one leg and the pendulum hits it from the back.

Figure 5 left shows a plot of the test without balancing. The blue curve is the measured error retrieved from the gyroscope and the orange line is the movement of the balancing joint. In this case, the joint is not moving because the balance module is turned off. The reference angular velocity from this test is $\frac{0^\circ}{s}$. Furthermore, the figure shows that the robot could not compensate the disturbance and fell down after 75 frames. In each of the ten trials, the robot fell down.

Figure 5 right shows a plot of the trials with balancing. The robot was able to compensate for the disturbance after 125 frames. Since one frame lasts 20 ms, the whole compensation was completed after only 2.5 seconds. The robot did not fall down in all ten trials, and it was always able to compensate for the disturbance in a time similar to 2.5 seconds.

6 Conclusions and Future Work

The experiments imply that it is possible to create very effective motions with the motion engine introduced. The motions are stable despite the dynamical changing of the kick direction. Since the stabilization is divided into two approaches, it would be an enhancement to replace these by a single method such as the preview control introduced in [8]. The preview control takes the system dynamics into account and uses the principle of the zero moment point to stabilize the motions.

Since Bezier curves are well known, motions can be created very easily even by less experienced users. Nevertheless, an alternative approach would be to record motions by kinesthetic manipulation as in [3].

References

1. Akin, H.L., Mericli, T., Özkucur, K.C., Gökce, B.: Cerberus 2010 team description paper (2009), `http://www.tzi.de/spl/pub/Website/Teams2009/Cerberus10TDP.pdf` (as of April 26, 2010)
2. Bartsch, S.: Steuerung der Fortbewegung eines humanoiden Roboters, robust gegen externe Störungen und geeignet für unebenes Terrain, basierend auf biologisch inspirierter Architektur. Diplomarbeit, Universtität Bremen (January 2007)
3. Berger, E., Amor, H.B., Vogt, D., Jung, B.: Towards a simulator for imitation learning with kinesthetic bootstrapping. In: Menegatti, E. (ed.) Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR), pp. 167–173 (November 2008)
4. Borisov, A., Ferdowsizadeh, A., Mohr, C., Mellmann, H., Martius, M., Krause, T., Hermann, T., Welter, O., Xu, Y.: NAO-Team Humboldt 2009 (2009), `http://www.naoteamhumboldt.de/papers/NaoTH09Report.pdf` (as of April 26, 2010)
5. Brunn, R., Düffert, U., Jüngel, M., Laue, T., Lötzsch, M., Petters, S., Risler, M., Röfer, T., Spiess, K., Sztybryc, A.: GermanTeam 2001. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 705–708. Springer, Heidelberg (2002)

6. Bräunl, T.: Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems. Springer, Heidelberg (2003)
7. Czarnetzki, S., Kerner, S., Klagges, D.: Combining key frame based motion design with controlled movement execution (2010)
8. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. Robotics and Autonomous Systems 57(8), 839–845 (2009)
9. Faber, F., Behnke, S.: Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In: Proceedings of the International Conference on Humanoid Robots (Humanoids) (2007)
10. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics - Principles and Practice, 2nd edn. Addison-Wesley Publishing Company, Reading (1990), xXIII, 1174 S : Ill
11. Graf, C., Härtl, A., Röfer, T., Laue, T.: A robust closed-loop gait for the standard platform league humanoid. In: Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T. (eds.) Proceedings of the Fourth Workshop on Humanoid Soccer Robots in Conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots, Paris, France, pp. 30–37 (2009)
12. Jahn, P.D.K.U., Borkmann, D., Reinhardt, T., Tilgner, R., Rexin, N., Seering, S.: Nao Team HTWK Leipzig team research report 2009 (2009), `http://naoteam.imn.htwk-leipzig.de/documents/techReportHTWK.pdf` (as of April 26, 2010)
13. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Jirukawa, 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 (2003)
14. Niehaus, C., Röfer, T., Laue, T.: Gait optimization on a humanoid robot using particle swarm optimization. In: Pagello, E., Zhou, C., Menegatti, E., Behnke, S. (eds.) Proceedings of the Second Workshop on Humanoid Soccer Robots in Conjunction with the 2007 IEEE-RAS International Conference on Humanoid Robots, Pittsburgh, PA, USA (2007)
15. Panakos, A., Paraschos, A., Pierris, G., Chroni, D., Vafeias, E., Chatzilaris, E., Vazaios, E., Lagoudakis, M.G., Vlassis, N.: Kouretes 2008 - Nao team report (2009), `http://www.intelligence.tuc.gr/kouretes/docs/2008-kouretes-nao-report.pdf` (as of April 26, 2010)
16. Pratab, R., Ruina, A.: Introduction to Statics and Dynamics. Oxford University Press, Oxford (2009), `http://ruina.tam.cornell.edu/Book/` (Preprint)
17. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-Human team report and code release 2009 (2009), `http://www.b-human.de/download.php?file=coderelease09_doc`