In Proceedings of 8th Workshop on Humanoid Soccer Robots
13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA, 2013.

Trunk Controlled Motion Framework

Stefan Glaser, Klaus Dorer
Hochschule Offenburg, Germany
Email: {Stefan.Glaser, Klaus.Dorer}@hs-offenburg.de

***Abstract*—In this paper we propose a motion framework for bipedal robots that decouples motion definitions from stabilizing the robot. This simplifies motion definitions yet allows dynamic motion adaptations. Two applications, walking and stopping on one leg, demonstrate the power of the framework. We show that our framework is able to perform walking and stopping on one leg even under extreme conditions and improves walking benchmarks significantly in the RoboCup 3D soccer simulation domain.**

I. Introduction

A robot motion can be defined in many ways. When using inverse kinematics, one automatically has to decide for a reference frame for calculations. The obvious choice for the robot's trunk/torso as reference frame suits well for the purpose of IK calculations. However, with respect to balancing motion definitions such as walking, the torso as a twisting system lacks the adaptation to the robot's current tilt (x- and y-rotation) and thereby as well to its center of mass (CoM) (Figure 1).

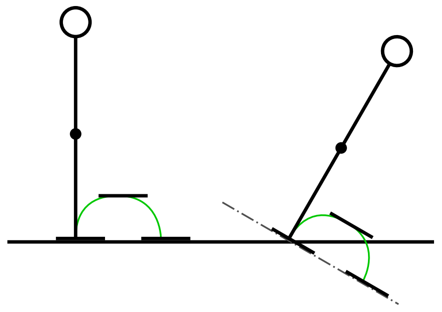
Fig. 1. Locally defined motions do not adapt to the current tilt and CoM.

Extending the trajectory interpolations of the motion models to also cope for the current situation (the robot's tilt and CoM) can be very complex and counter intuitive to the basic calculations. Moreover it has to be done for each motion separately and makes testing and debugging of those calculations even more difficult.

The Trunk controlled Motion Framework described in this paper addresses this issue by the introduction of a virtual reference frame. Motion trajectories defined in this frame are then automatically adapted to the current situation during execution. This way motion definitions keep as simple as in the local case, but implicitly gain the power of dynamic situation adaption. The control strategy is only based on the current CoM and the orientation estimation of the trunk. Two common and well investigated measures of humanoid robots.

The remainder of this paper is organized as follows: Section II puts our approach in relation to existing work. In Section III we describe our approach of the Trunk controlled Motion Framework. Section IV describes two applications of the TcMF: walking and stopping on one leg. Results for those two applications in the RoboCup 3D soccer simulator are presented in Section V. Finally in Section VI we propose some future work and conclude.

In the context of this paper, the coordinate frames follow the right hand rule, where the x-axis points to the right, the y-axis to the front and the z-axis upwards. Furthermore, we use transformation matrices of the form $\mathbf{T}_b^a$, which expresses the pose of b in the coordinates of frame a. In homogeneous coordinates $\mathbf{T}_b^a$ consists of a 3×3 rotation matrix $\mathbf{R}_b^a$ and a 3×1 translation vector $\mathbf{t}_b^a$:

$$\mathbf{T}_b^a = \begin{bmatrix} \mathbf{R}_b^a & \mathbf{t}_b^a \\ \mathbf{0} & 1 \end{bmatrix} \tag{1}$$

The videos referenced in the text are available at [9].

II. Related Work

There are many approaches to create motions of robots and more specifically to create walking gaits for bipedal robots [7]. Simple approaches define or learn motions in joint space, i.e. define a choreography of joint angles that produce the desired behavior. Especially for walking, zero moment point (ZMP) based approaches gained popularity [3]. Many successful teams in the RoboCup competition use ZMP-based linear inverted pendulum models (LIPMs) for walking [1], [2], [4].

ZMP-based methods have in common that they abstract the body to a single point mass. This has the advantage that LIPM can directly be applied reducing mathematical complexity. On the other side they do not consider the current orientation of the robot when mapping from the globally modelled trajectories to the local trunk frame. This means that any deviation from the assumed upright orientation of the torso leads to the problems shown in Figure 1.

In order to respond to such deviations, three recovery strategies are typically applied: ankle push recovery, hip push recovery and step push recovery [8]. Ankle push recovery applies forces to the ankle joints in order to keep the CoM within the base of support. This is what our framework actively applies. Hip push recovery uses angular accelerations in the hip to keep the CoM within the base of support. Step push recovery uses so called capture steps to react to major disturbances by changing the step plan and therefore the base of control [8], [5]. The latter two are 'passively' supported by the proposed framework as explained later.

One disadvantage of typical implementations of these strategies is that the adjustments are embedded into the motion model. That means they are tightly coupled to the specific motion. The framework proposed here provides a generic adjustment to the current tilt decoupled from specific motion definitions. To demonstrate this, we show two applications of the framework, walking and stopping on one leg. Also, the proposed framework explicitly models the orientation of the torso eliminating the implicit assumption of an upright robot. Both is new to our knowledge.

III. TRUNK CONTROLLED MOTION FRAMEWORK

A. Approach

The idea behind the Trunk controlled Motion Framework (TcMF) follows a simple thought: A balanced motion should always support the center of mass. In order to achieve this, motions should be defined with respect to a reference frame with the center of mass as its origin. Furthermore supporting the center of mass usually means acting against the gravity, regardless of the actual tilt of the robot. The orientation of the reference frame should therefore always point upright, but still facing the horizontal view direction of the robot. We call this reference frame *Motion Frame* $\mathcal{M}$.

So far we simply switched the reference frame in which we define the motions from the local trunk frame $\mathcal{T}$ to the virtual Motion Frame. In order to use an inverse kinematics solver, we now need to transform the motion trajectories back into the trunk frame, using the following equation:

$$\mathbf{T}_l^{\mathcal{T}} = \mathbf{T}_{\mathcal{M}}^{\mathcal{T}} \ \mathbf{T}_l^{\mathcal{M}} \tag{2}$$

where $\mathbf{T}_l^{\mathcal{M}}$ and $\mathbf{T}_l^{\mathcal{T}}$ expresses the pose of limb l with respect to frame $\mathcal{M}$ respectively $\mathcal{T}$. The transformation $\mathbf{T}_{\mathcal{M}}^{\mathcal{T}}$ from the Motion Frame to the trunk follows directly from the definition:

$$\mathbf{T}_{\mathcal{M}}^{\mathcal{T}} = \begin{bmatrix} \mathbf{R}_{\mathcal{M}}^{\mathcal{T}} & \mathbf{t}_{CoM}^{\mathcal{T}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

where $\mathbf{R}_{\mathcal{M}}^{\mathcal{T}}$ describes the current tilt of the trunk and $\mathbf{t}_{CoM}^{\mathcal{T}}$ the position of the CoM with respect to the trunk frame. By using a static CoM position together with the identity rotation to construct $\mathbf{T}_{\mathcal{M}}^{\mathcal{T}}$, we end up with the same local motion as we had before. However, by using the current estimation for tilt and CoM to transform the motion trajectories, we get a motion which is automatically adjusted to the current CoM and tilt of the robot.

The current tilt is obtained from the orientation $\mathbf{R}_{\mathcal{T}}^{\mathcal{W}}$ of the trunk with respect to the world frame $\mathcal{W}$ by removing the horizontal rotation part of the orientation:

$$\mathbf{R}_{\mathcal{T}}^{\mathcal{M}} = \mathbf{R}_{\mathcal{W}}^{\mathcal{M}} \ \mathbf{R}_{\mathcal{T}}^{\mathcal{W}} \tag{4}$$

where $\mathbf{R}_{\mathcal{W}}^{\mathcal{M}}$ by definition is the rotation around the z-axis about the horizontal view angle of the robot. In our case, the current tilt of the robot corresponds to the orientation of the trunk with respect to the Motion Frame. In order to plug formula 4 into 3, we need to invert formula 4:

$$\mathbf{R}_{\mathcal{M}}^{\mathcal{T}} = (\mathbf{R}_{\mathcal{T}}^{\mathcal{M}})^{-1} = (\mathbf{R}_{\mathcal{W}}^{\mathcal{M}} \ \mathbf{R}_{\mathcal{T}}^{\mathcal{W}})^{-1} \tag{5}$$

So far motions are closed-loop in the sense that trajectories of legs are adjusted to any tilt of the trunk and movement of the CoM. However, they are not counteracting any tilt of the robot. If the orientation estimation of the robot would be perfect, this tilt adjustment would basically try to preserve the current tilt of the trunk. In order to continuously push the current z-axis of the trunk upright (towards the global z-axis), we can manipulate $\mathbf{R}_{\mathcal{M}}^{\mathcal{T}}$ by applying an additional rotation, interpolating from the actual z-axis of the trunk to the global z-axis. The resulting adapted trunk frame is called $\mathcal{T}^{'}$.

$$\mathbf{R}_{\mathcal{M}}^{\mathcal{T}^{'}} = \mathbf{R}_{\mathcal{T}}^{\mathcal{T}^{'}} \ \mathbf{R}_{\mathcal{M}}^{\mathcal{T}} \tag{6}$$

If we now use this manipulated tilt estimation $\mathbf{R}_{\mathcal{M}}^{\mathcal{T}^{'}}$ in formula 3 to construct $\mathbf{T}_{\mathcal{M}}^{\mathcal{T}^{'}}$ and plug $\mathbf{T}_{\mathcal{M}}^{\mathcal{T}^{'}}$ into formula 2 to transform the motion trajectories into $\mathcal{T}^{'}$, we get a motion that is not only adjusted to the current CoM and tilt of the robot, but also continuously trying to push the trunk upright. The amount by how much we interpolate between the current and upright situation specifies the weighting between the two adjustments. Zero percent interpolation ($\mathcal{T}^{'} = \mathcal{T} \rightarrow \mathbf{R}_{\mathcal{T}}^{\mathcal{T}^{'}} = \mathbf{R}_{\mathcal{T}}^{\mathcal{T}} = \mathbf{I}$) results solely in the first tilt adjustment. Whereas 100 percent interpolation ($\mathcal{T}^{'} = \mathcal{M} \rightarrow \mathbf{R}_{\mathcal{T}}^{\mathcal{T}^{'}} = \mathbf{R}_{\mathcal{T}}^{\mathcal{M}}$) would simply cancel out the first tilt adjustment, with the consequence that the motion is still related to the current CoM, but without any further adjustment to the current tilt of the trunk. Any interpolation in between describes a trade off between adapting to the current tilt and pushing the trunk upright.

The interpolation rotation $\mathbf{R}_{\mathcal{T}}^{\mathcal{T}^{'}}$ can be constructed in different ways. A simple method to construct $\mathbf{R}_{\mathcal{T}}^{\mathcal{T}^{'}}$ is using a quaternion slerp between $\mathcal{T}$ and $\mathcal{M}$. However, a quaternion slerp entails an implicit weighting between the Sagittal and Coronal adjustment, which restricts the flexibility of the controller. We therefore use the Euler/Cardan angles convention to describe the difference from $\mathcal{T}$ to $\mathcal{M}$, which implicitly requires the separation of $\mathbf{R}_{\mathcal{T}}^{\mathcal{M}}$ into three elemental rotations. Since we only consider the tilt of the trunk without the horizontal direction, $\mathbf{R}_{\mathcal{T}}^{\mathcal{M}}$ can be composed by only two elemental rotations, one around the x-axis (Sagittal axis) and one around the y-axis (Coronal axis), which can be interpolated separately. The interpolation itself is realized by a simple two-dimensional proportional controller.

Until now, the above described framework tries to maintain an upright direction of the trunk. While this may be sufficient to support for example a walking motion, it lacks flexibility with respect to general motions, which may intend to maintain a specific tilt, different from upright. However, extending the above framework to maintain an arbitrary tilt is straight forward. The second adjustment, responsible for maintaining an upright state, is constructed by an interpolation between the current state $\mathcal{T}$ and the upright situation $\mathcal{M}$. More precisely by an interpolated rotation between the current z-axis of the trunk and the global z-axis. By switching the interpolation target from the global z-axis to an arbitrary unit vector, we can guide the second adjustment towards maintaining an arbitrary tilt. We call this unit vector describing the intended tilt of the trunk *(intended) leaning vector*. With the requirement to specify an intended leaning vector to the TcMF, each motion automatically describes its own adjustment target. An example motion definition together with the two adjustments is illustrated in Figure 2.

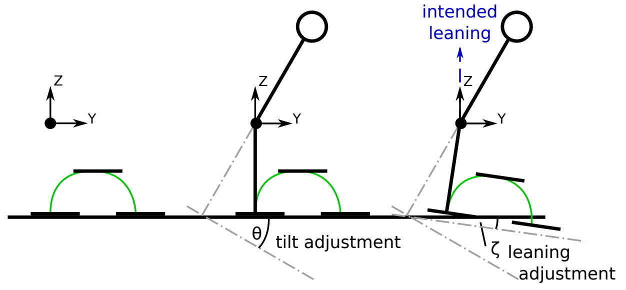

Fig. 2. An automatically adapted motion defined in the virtual motion frame. The general motion definition (left) is first adapted to the current tilt of the trunk (center) and then manipulated to push the trunk towards the intended leaning vector (right).

B. Framework Integration

In the context of the TcMF, we consider motion definitions based on inverse kinematics, that model trajectories in three-dimensional space for all relevant limbs. A motion using the TcMF typically consists of three components (see Figure 3):

- Motion Model:
  The motion model specifies the basic motion trajectories for all relevant limbs of the robot. It provides the virtual targets (left/right foot/arm target poses in the motion frame) together with an intended leaning vector for each cycle.
- Trunk Controller:
  The trunk controller implements the above presented approach. It is responsible for the mapping from the motion frame to the local trunk frame, dependent on the current estimations for trunk orientation and CoM position.
- Inverse Kinematics Solver:
  An arbitrary IK solver is required to map the adjusted target poses into joint space.

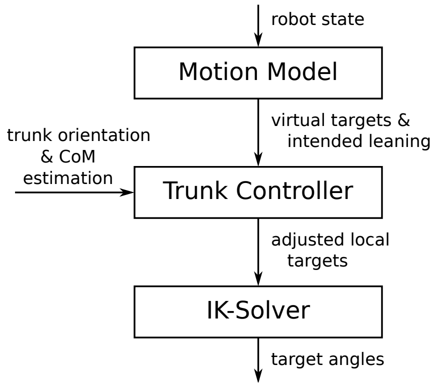

Fig. 3. Component diagram of a motion using the TcMF.

In the beginning of each execution cycle, the motion model is triggered to update its target state (e.g. to progress in the modelled motion trajectories). After that, the virtual targets and the intended leaning vector are forwarded to the trunk controller, which maps the virtual targets to the local trunk frame. The resulting adjusted set of target poses is then used by an inverse kinematics solver to determine the target angles of the involved joints.

A key aspect of the TcMF is its generality. The three components mentioned above can be implemented in a layered structure. In our case, the IK solver is integrated in the robot model abstraction, a layer below the actual motion definitions. The TcMF can be integrated as an intermediate layer between the motion model and the IK solver. By extending the interface to the TcMF with a Boolean parameter, a motion model can command the TcMF to replace the dynamic trunk controller with a static adjustment to the intended leaning vector. To get a completely static adjustment of the motion, we even use a static CoM position in the pelvis of the robot in this case. This way, the TcMF can be reduced to a shift of the motion about this static CoM position, which allows for an easy integration of existing motion definitions.

IV. Applications

As explained before, the TcMF is a framework to decouple motion definitions from the problem of maintaining a specified trunk orientation. In this section we show two applications of the TcMF, (omnidirectional) walking and stopping balanced on one leg (for kicking).

A. Walking

The walk motion consists of a simple trajectory of a linear movement for the support leg and a sinoidal movement for the free leg. The trajectory of the right leg while moving forward with a step size of 9 cm is shown in Figure 4. The walk is specified as an 18 cycle (50 Hz) motion so a complete step cycle takes 0.36 s. The trajectory has been optimized to the maximum motor speed of the simulated Nao of 7.03 degrees per 20 ms. Only in 3 of the 18 cycles the inverse kinematics is not limited by the maximum speed as shown in Figure 5.

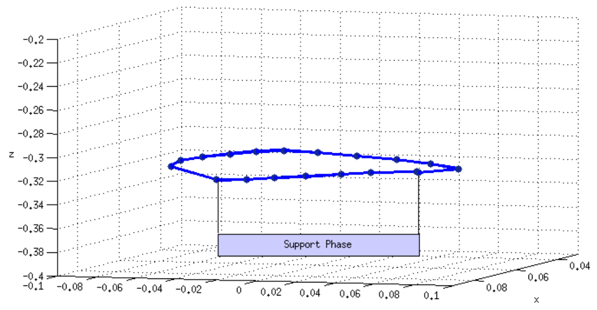

Fig. 4. Right foot trajectory when walking forward.

Independent from this trajectory, a torso leaning can be specified. Figure 6 shows two examples of static leaning definitions. Images three and four show a 30 degree forward leaning while walking forward at 0.8 m/s. Images five and six show a 20 degree sideward leaning while walking forward at 0.5 m/s. The CoM is shown as an olive circle in the wire frames. Dynamic leaning definitions are possible. A more natural example would be to lean forward dependent on the forward acceleration of the robot.

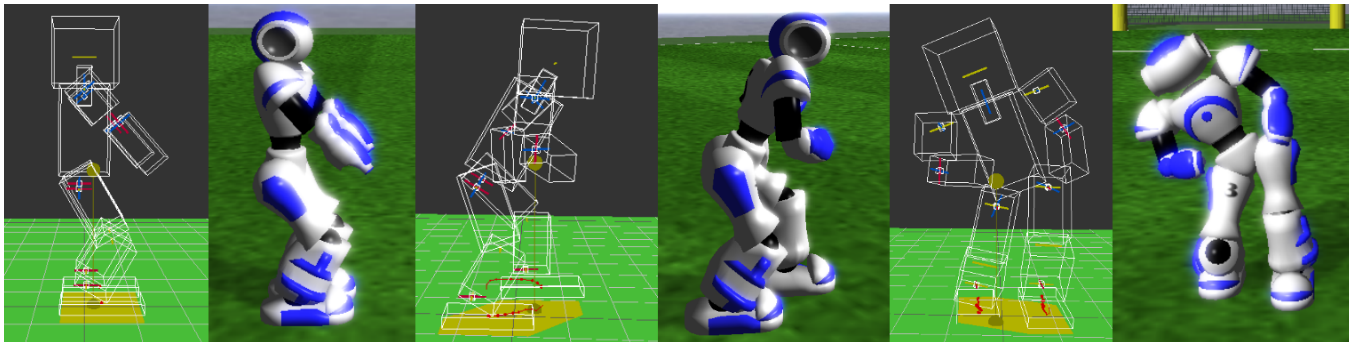

Fig. 6. Walk as an application of the TcMF. Image one and two: Nao upright. Image three and four: walking with 30 degrees forward leaning. Image five and six: walking with 20 degrees sideward leaning.

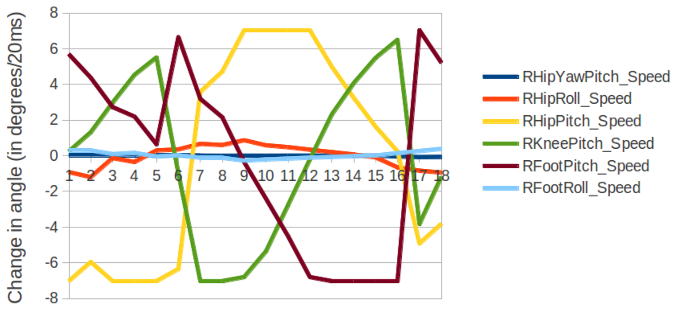

Fig. 5. Motor speeds of the right leg when walking forward.

B. *One Leg Stopping*

A considerable amount of time can be saved for kicking, if the robot does not have to stop on both legs before kicking. It also makes it possible to place the support foot at the side of the ball. This section describes how TcMF is used to stop on one leg and keeping balance for kicking.

One leg stopping is divided into three motions: a final step motion, a get on leg motion and a balance on leg motion. The final step motion is defined by the ball relative position of the support foot, the desired global kick direction and which of the two legs should be support foot. The motion itself places the second last step before getting on one leg. Unlike walking this motion performs 2/3 of the step distance by the support leg to shift the CoM to the final balancing leg. Also it makes sure to turn appropriately. Since the Nao robot is not able to do inward turns, any remaining turn angle to the right, if the final support foot is the right foot, have to be done with the second last step. The get on leg motion performs the final step and specifies an appropriate trunk leaning. It also makes sure that the free foot is started to be moved towards a kicking position. Finally, it also performs any remaining turn to the desired kick direction. Information like which is the support foot is gained from the previous motion by a mechanism called motion morphing. The balance on leg motion finally stabilizes the robot standing already on one leg, adjusts the height of the hip to proper kick height and defines the final trunk leaning required for the kick. The kick itself is currently performed in joint space outside the TcMF. The end of each motion is shown in Figure 7 for a straight kick.

V. Results

Experiments in this section have been conducted with the RoboCup 3D soccer simulation server version 0.6.6 [6] on a simulated Nao robot.

A. *Walking*

To demonstrate the effectiveness of TcMF a first experiment let the robot walk forward for four seconds with a speed of 0.5 m/s. The first two seconds the desired torso leaning was set to 30 degrees forward leaning. The last two seconds the desired torso leaning was upright again to see how effective the trunk controller can adjust to the new leaning.

Figure 8 shows the x and y components of the torso's z-vector over time. Starting from upright position, it takes the robot less than one second to get into 30 degree forward leaning. When the desired trunk leaning is changed to upright at 2s after the start, the robot is upright again within almost half a second. During the whole time the robot remains walking.

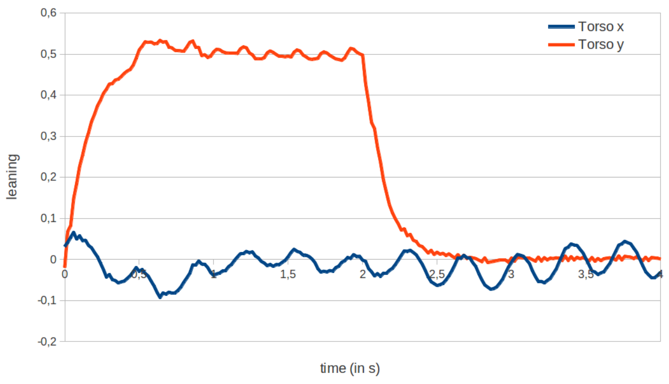

Fig. 8. Adjustment of torso orientation over time.

Figure 9 shows the range of torso rotations in which the robot is still able to walk. The setup is like above, 2 seconds walking with a leaning followed by two seconds walking upright. The leaning was varied from 30 degrees backward leaning to 30 degrees forward leaning and for each of these values also from 30 degrees left to 30 degrees right leaning. Each point in Figure 9 is the average of 40 runs counting the average probability not to fall down.

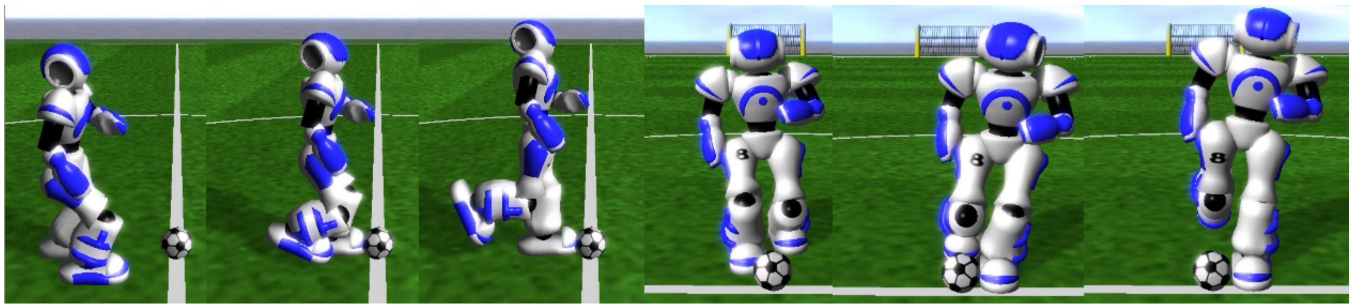

Fig. 7. Stop on one leg as an application of the TcMF. First image: end of final step motion. Second image: end of get on leg motion. Third image: end of balance on leg motion. Fourth to Sixth image: same from front.

The robot is able to walk with 30 degrees forward and 25 degrees backward leaning, if there is no sideward leaning. It is also able to walk with 20 degrees side leaning in both sides if there is no forward-backward leaning. Even a 15 degrees forward and sideward leaning can be sustained while walking and corrected upright when desired after two seconds. The asymmetry of forward versus backward leaning is due to limitations of the robot's hip and knee pitch joint's working areas. The walked distance in the green area is almost unaffected by the leaning of the torso as can be seen in Figure 10 [Videos 1-4].

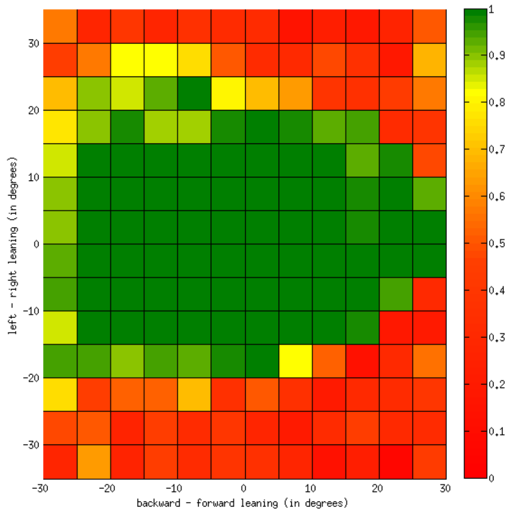

Fig. 9. Possibility to not fall down while walking forward with the specified leaning.

In a next and less artificial experiment, the robot was benchmarked on more typical motions that occur during a soccer game. Each of the results is an average of 100 runs of 6 seconds each. Table I shows the results. The trunk controlled motions show significantly better results in backward, sideward, diagonal and turning tasks. For the latter two the difference is directly visible when looking at the motion. Especially during fast turning the robot drifts from its upright orientation from time to time. Trunk control establishes upright orientation quickly, while without trunk control the robot even falls in six of the 100 turn motions. Since the forward walk motion is relatively stable by itself in the absence of other robots or accelerations, trunk control does not have a significant impact on the forward benchmark [Videos 5-8].

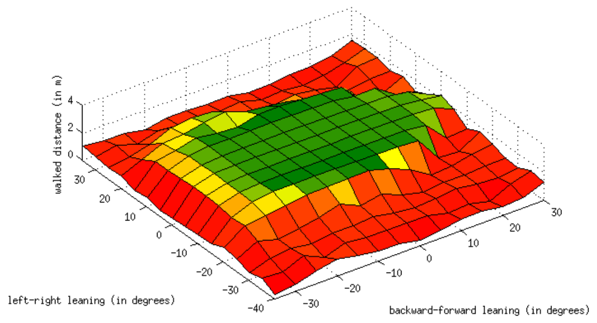

Fig. 10. Distance walked with leaning.

TABLE I. COMPARISON OF BENCHMARK RESULTS (SIGNIFICANTLY BETTER RESULTS IN BOLD FONT).

Benchmark	Classic (uncontrolled)	Trunk controlled
Forward (m/s)	0.85	0.84
Backward (m/s)	0.75	**0.78**
Sideward (m/s)	0.48	**0.56**
Diagonal (m/s)	0.62	**0.73**
Turning (deg/s)	158.1	**179.4**

B. *One Leg Stopping*

To evaluate stopping on one leg, the robot walked straight for 1.4 seconds and then stopped on the left leg. We varied the speed at which the robot was walking from 0 to 0.85 m/s and the desired horizontal turn angle from 0 to 100 degrees. Each result is an average of 40 trials.

Figure 11 shows the reliability of the one leg stopping motions. Even with 0.8 m/s the robot is able to stop on one leg reliably for a range of turn angles. For a considerable speed of 0.6 m/s an effective turn of 50 degrees is reliably achievable.

Figure 12 shows the effective turning angle for the various settings. The values are somewhat lower than the desired angles especially for huge turns. The main reason is that the fix amount of time we defined for a step does not suffice for huge turns. The yaw-pitch joint in the hip is at its limit [Videos 9,10].

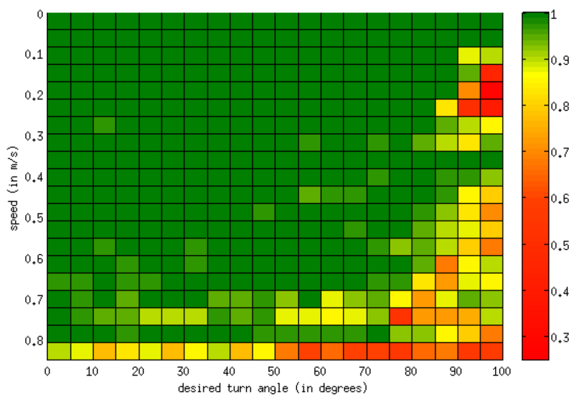

Fig. 11. Probability of not falling down when stopping at different speeds and with different turn angles.

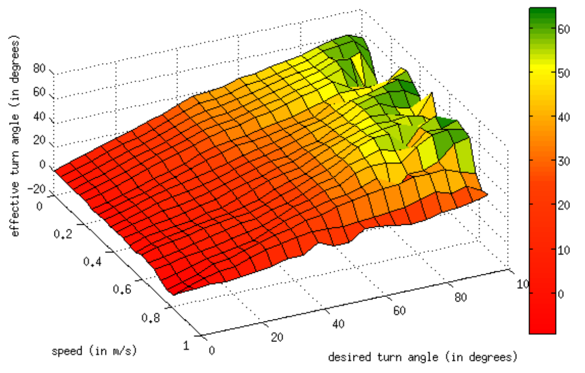

Fig. 12. Effective turn amount when stopping.

VI. Conclusion

With the introduction of a virtual reference frame for motion definitions in combination with the mapping to the trunk frame of the robot, we constructed a framework which is able to dynamically adjust motions to the current tilt and CoM of the robot. The intended leaning vector allows motion definitions to describe the intended tilt of the trunk during the motion in an intuitive way. By using a control strategy to interpolate between the current and intended tilt of the trunk, the framework is able to support motions in reaching and holding an intended tilt during their execution. The presented framework can be easily integrated as an intermediate layer between the actual motion model and the IK solver. An additional parameter, allowing to switch between the dynamic controller and a static transformation, provides more flexibility and helps integrating existing motion definitions.

In general, the approach described in this paper is not able to dynamically adjust an arbitrary motion in such a way that the robot keeps stable and balanced. The suggested framework merely provides a generalized version of a motion adjustment to the current tilt and CoM of the robot, which is abstracted by the intended leaning. This is just one low level approach of maintaining the balance. If the actually executed motion is generally unstable, the TcMF will not prevent the robot from falling down. However, as results show, the TcMF is able to support generally stable motions during execution by maintaining a certain trunk stability. External disturbances and deviations from the expected tilt are automatically incorporated in the motion during execution and thus less harmful.

The TcMF is based on the estimations for the position of the CoM and the orientation of the trunk. While the CoM can be calculated quite exactly and is less important in the adjustment, the estimation for the orientation of the trunk is more complex and error prone and have crucial influence on the adjustment. Any drift in orientation is directly reflected by the TcMF and if the drift in orientation estimation becomes bigger than the interpolation rate of the trunk controller, the TcMF will rapidly cause the robot to collapse. Another problem is a flickering orientation estimation, which has direct impact on the smoothness and amplitudes of the resulting motion trajectories. Particularly in the case of a kick trajectory, where smoothness, speed and accuracy are very important, this framework might not be the best choice. However, the TcMF is only dependent on the x- and y-rotation part of the orientation estimation and well known probabilistic methods, like the Kalman Filter, provide more stable orientation estimations by incorporating multiple sensor information.

The current framework uses a proportional control strategy for adapting to the intended leaning vector. While first results show an acceptable general control behavior, different control strategies might perform better across different motions. By taking the velocities and accelerations of the CoM into account, more sophisticated control strategies can be applied to react more systematically to external disturbances. Apart from classical controllers, machine learning techniques can be used to learn beneficial control strategies.

References

[1] M. Friedmann, J. Kiener, S. Petters, H. Sakamoto, D. Thomas, and O. von Stryk. *Versatile, high-quality motions and behavior control of humanoid soccer robots*. In Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), pages 916, 2006

[2] C. Graf and T. Röfer, *A Closed-loop 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid* Nashville, TN, USA: Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots, 2010

[3] S. Kajita, F. Kanehiro, K. Kaneko, and K. Yok, *The 3D Linear Inverted Pendulum Mode : A simple modeling for a biped walking pattern generation Pendulum Mode* International Conference on Intelligent Robots and Systems, no. 4, pp. 239-246, 2001

[4] D. Lee, et.al. *RoboCup 2011 humanoid league winners* RoboCup 2011: Robot Soccer World Cup XV, pp. 37–50, Springer, 2012

[5] M. Missura and S. Behnke, *Lateral Capture Steps for Bipedal Walking* 11th Bled, Slovenia: IEEE-RAS International Conference on Humanoid Robots (Humanoids), October 2011

[6] O. Obst and M. Rollmann, *SPARK A Generic Simulator for Physical Multiagent Simulations* Computer Systems Science and Engineering, 20(5), September 2005

[7] N. Onn, M. Hussein, T. H. Hing, L. W. Ying, M. Z. Zain, M. S. Che Kob, *Human Gait Modelling Considerations of Biped Locomotion for Lower Limb Exoskeleton Designs* Malaysia: 13th International Conference on Robotics, Control and Manifacturing Technology, April 2013

[8] S. Yi, B. Zhang, D. Hong, and D. Lee, *Active stabilization of a humanoid robot for impact motions with unknown reaction forces* Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on intelligent robots and systems, pp. 4034–4039, October 2012

[9] http://www.et-it.hs-offenburg.de/prof/kdorer/robocup/go/videos/HSR13/