Noname manuscript No.
(will be inserted by the editor)

Improving Biped Walk Stability with Complementary Corrective Demonstration

Çetin Meriçli · Manuela Veloso · H. Levent Akın

Received: date / Accepted: date

Abstract We contribute a method for complementing an existing algorithmic solution to performing a low level skill with corrective human demonstration to improve skill execution performance. We apply the proposed method to biped walking problem, which is a good example of a complex low level skill due to the complicated dynamics of the walk process in a high dimensional state and action space. We introduce an incremental learning approach to improve the Nao humanoid robot's stability during walking. First, we identify, extract, and record a complete walk cycle from the motion of the robot while it is executing a given walk algorithm as a black box. Second, we apply offline advice operators for improving the stability of the learned open-loop walk cycle. Finally, we present an algorithm to directly modify the recorded walk cycle using real time corrective human demonstration. The demonstration is delivered using a commercially available wireless game controller without touching the robot. Through the proposed algorithm, the robot learns a closed-loop correction policy for the open-loop walk by mapping the received corrective demonstration to the sensory readings while walking autonomously using the refined open-loop walk cycle. Experiment results demonstrate a significant improvement in the walk stability.

Keywords Learning from demonstration · Complex motor skill acquisition · Motion and sensor model learning

The first author is supported by The Scientific and Technological Research Council of Turkey Programme 2214 and the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

Ç. Meriçli(✉)
Computer Science Dept., Carnegie Mellon University, USA
Dept. of Computer Engineering, Boğaziçi University, Turkey
E-mail: cetin@cmu.edu

M. Veloso
Computer Science Department,
Carnegie Mellon University, USA
E-mail: veloso@cs.cmu.edu

H. L. Akın
Department of Computer Engineering
Boğaziçi University, Turkey
E-mail: akin@boun.edu.tr

1 Introduction

Learning from Demonstration (LfD) paradigm is getting increasingly popular in robotics research for transferring task or skill knowledge to an autonomous robot without explicitly programming it. LfD methods involve a teacher giving a demonstration of how to perform a task or a skill to the robot. There are two major approaches followed for delivering the demonstration: either the teacher performs the task or skill herself and lets the robot observe her, or the teacher makes the robot perform the task or skill through teleoperation or tactile interaction and lets the robot observe itself. The robot records the demonstrated actions along with the perceived state of the system at the time of the demonstration and then uses the recorded state-action pairs to derive an execution policy for reproducing the demonstrated task or skill. Compared to more traditional exploration based methods, LfD approaches aim to reduce the learning time and eliminate the necessity of defining a proper reward function which is usually considered to be a difficult problem [6].

Corrective demonstration is a form of teacher demonstration focusing on correcting an action selected by the robot to be performed in a particular state by proposing either an alternative action to be executed in that state, or a modification to the selected action. The usual form of employing corrective demonstration is either through adding the corrective demonstration example to the demonstration dataset or replacing an example in the dataset with the corrective ex-

ample, and re-deriving the action policy using the updated demonstration dataset.

If an algorithmic solution exists for performing the skill but is partially successful, the corrective demonstration approach can be used to correct the algorithm if an appropriate method for complementing the algorithm with human demonstration can be found. Seeking an answer to the question of whether it is possible to complement an algorithm with human demonstration by treating the algorithmic solution as a black-box system, in this article we present a complementary corrective demonstration method for improving the biped walk stability on a commercially available humanoid robot platform.

Being actively studied in humanoid robot research, biped walking is a challenging problem due to the high dimensional state and action space and the complex dynamics of the walking process. In our approach, we make use of an existing walk algorithm to obtain an initial open-loop walk cycle, and then we improve the stability of the walk in two corrective demonstration phases.

The phases of learning in our approach are as follows:

(a) An initial modeling of the walk motion by using the output of an existing walk algorithm
(b) Offline improvement of the obtained walk model via high level human advice
(c) Acquisition of a closed-loop gait via real time complementary corrective human demonstration while the robot is walking using the open-loop walk obtained in the previous phases

The teacher uses a commercially available wireless game controller to provide feedback without touching the robot. The feedback signals given by the teacher are transmitted to the robot over a host computer via wireless network. This setup allows the teacher to closely follow the robot and deliver the corrective demonstration without physical contact. The received correction signals are recorded together with the state of the robot in the form of sensory readings. A correction policy is then derived out of the recorded state-action pairs using a learning algorithm of choice. Finally, the learned correction policy is used to modify the open loop walk cycle in such a way to keep the robot balanced as it walks autonomously.

We present different types of correction and different methods for state-action association, policy derivation, and the application of the correction with different complexities. In particular, we present two different correction types (applying correction in the joint space or in the task space), two different state-action association methods (associating a single sensor to a correction value without taking the current position in the walk cycle into account or associating multiple sensors with a correction value while taking the current position in the walk cycle into account), two different policy extraction methods (fitting normal distributions on the received correction values in the discretized sensory reading space or using locally weighted regression with Gaussian kernel), and two methods for deciding when to apply correction to the system (applying the correction at each N^{th} timestep within the walk cycle or applying the correction only if the sensory readings go beyond the normal values according to a certain statistical definition of the *normal*). We present experiment results evaluating the performances of the different combinations of the aforementioned methods compared to each other and compared to the initial open-loop walk. All of the presented methods improved the stability of the walk with an increase in the overall performance as the used method gets more complex.

The organization of the rest of the paper is as follows. In Section 2, we present an overview of the related work and describe the hardware platform on which the proposed methods are implemented. Section 3 presents a formal definition of biped walking, and elaborates on how an open-loop walking behavior can be acquired from an existing walk algorithm and how the acquired walking behavior can be improved using human advice. We explain our real-time complementary corrective demonstration approach thoroughly in Section 4. Section 5 describes how we combine the corrective demonstration with the state of the robot to obtain a closed-loop walk. We present experiment results and evaluate the performances of the proposed methods in Section 6. We conclude the paper in Section 7, pointing out potential issues to be addressed in the future.

2 Background

2.1 Related Work

LfD methods have been applied to many learning scenarios involving high level task and low level skill learning on different robot platforms varying from wheeled and legged robots to flying ones. Here we present a few representative studies with their standing points in comparison to our approach, and we strongly encourage the reader to resort to [6] for a comprehensive survey on LfD.

For learning how to perform high level tasks, it is a common practice to assume that the required low level skills are available to the robot. Task learning from demonstration have been studied in many different contexts. Thomaz and Breazeal proposed a method for utilizing human feedback as the reward signal for the Reinforcement Learning (RL) system for a simulated robot that tries to learn how to bake a cake [29]. The notion of observing the robot executing the task and intervening to provide feedback bears a resemblance with our approach. However, they utilize the feedback as a reward signal to an action selected by the

robot whereas our approach makes use of the received feedback to improve the performance of an existing algorithm. Chernova and Veloso introduced an approach called *"Confidence Based Autonomy"* (CBA) for learning behavior policies from human demonstration and applied it on single robot [13] and multi-robot problems [12], where the robot builds a statistical model of the received demonstration examples and gradually reduces the number of demonstration requests as it becomes more confident. The main difference between the CBA approach and our approach is that instead of starting from scratch, our approach utilizes an existing algorithm as the baseline controller and needs teacher feedback only when the algorithm fails to compute a proper action to execute.

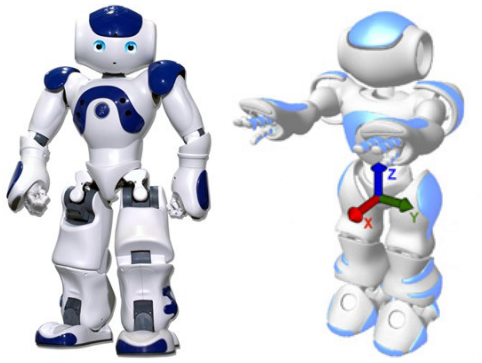
Several approaches to robot motion learning in the literature utilize LfD methods with different foci. Tactile interaction has been utilized for skill acquisition through kinesthetic teaching [18] and skill refinement through tactile correction [5]. Their approach shares a similarity with our approach as both methods utilize human feedback interleaved with the skill execution. A major difference, however, is that while they use the received feedback to refine the action policy, our method keeps the feedback separate and learns a *correction policy* instead.

Interacting with the learner using high level abstract methods has been introduced in forms of natural language [11, 26] and advice operators as functional transformations for low level robot motion, demonstrated on a Segway RMP robot [3, 4]. Reinforcement learning methods have been investigated in conjunction with LfD paradigm for teaching a flying robot how to perform a complex skill [1], learning to swing up a pole and keep it balanced [9, 8], and hierarchical learning of quadrupedal locomotion on rough terrain [19]. Motion primitives have been used for learning biped walking from human demonstrated joint trajectories [22] and learning to play air hockey [10].

Efficient biped walking is also of great importance for RoboCup Standard Platform League (SPL), in which teams of autonomous Aldebaran Nao humanoid robots play robot soccer [24, 27]. There have been various studies on developing efficient biped walking methods that are suitable for the Nao hardware. The proposed approaches include an effective omni-directional walking algorithm using parameterized gaits and sensor feedback [25], an efficient Zero Moment Point (ZMP) search method [20, 21], an evolutionary strategy to tune the parameters of a Central Pattern Generator based walk [15], a ZMP based omni-directional walking algorithm [28], and a preview control based method for keeping the ZMP on the desired path [14]. Additionally, the Nao is delivered with a default open-loop uni-directional walk algorithm[1].

2.2 Hardware Platform

Nao (Fig. 1), is a 58 cm tall humanoid robot with 21 degrees of freedom, weighing 4.5 Kg [2]. It is equipped with an on-board processor running at 500MHz, and a variety of sensors including a 3-axis accelerometer, a 2-axis (Roll-Pitch) gyroscope, and a special circuitry for computing the absolute torso (upper body of the robot) orientation using the accelerometer and gyroscope data. The torso angle estimator, the accelerometer, and the gyroscope sensors use a right-hand frame of reference (Fig.1(b)). As opposed to most other humanoid robot designs, Nao does not have separate hip yaw joints for each leg [16], instead, the two legs have mechanically coupled hip yaw-pitch joints that are perpendicular to each other along the $Y-Z$ plane and driven by a single motor, drastically limiting the reuse of biped walk algorithms developed for other humanoid platforms.



(a) (b)

Fig. 1 a) The Nao robot. b) The frame of reference for sensors.

The internal controller software of the robot runs at 50Hz; therefore, it is possible to read new sensor values and send actuator commands every 20ms[2].

3 Open-loop Biped Walking

Biped walking is a periodic phenomenon consisting of consecutive walk cycles. A walk cycle (wc) is a motion segment that starts and ends with the same configuration of the joints. Each walk cycle consists of four phases:

[1] The Nao V3 model is delivered with a new closed-loop omni-directional walk; however, that model was not available to us during our experimental study.

[2] The mentioned frequency is for the Nao V2 model which was the platform used in this study. The internal control software on the more recent V3 model runs at 100Hz.

- First single support phase (left)
- First double support phase
- Second single support phase (right)
- Second double support phase

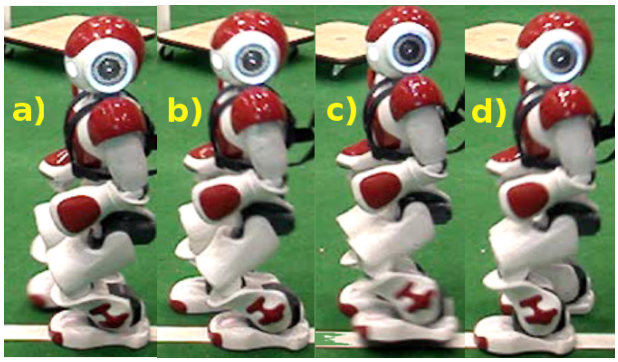


Fig. 2 Walk cycle phases: a) first single support, b) first double support, c) second single support, and d) second double support.

During the first single support phase, the robot stands on its left foot, and swings the right leg forward. During the double support phases, both feet are on the ground, differing in the offsets along the X axis from the first double support phase to the second. During the second single support phase, the robot stands on its right foot to lift and swing the left leg forward as shown in Fig. 2. The walk cycle has a duration of T timesteps, where $wc_t^j, t \in [0, T), j \in Joints$ is the *command* to the joint j provided at timestep t.

In principle, if we could generate the correct joint command sequence for a walk cycle, it would then be possible to make the robot walk indefinitely by executing this cycle repeatedly in an open loop fashion. However, in reality, various sources of uncertainty associated with sensing, planning, and actuation affect biped walking.

- In ***sensing***, the main source of uncertainty is the noise in the sensor readings due to the lack of precision/accuracy (e.g., high noise rate on the gyroscopes and the accelerometers and imprecise position sensing on the joints).
- In ***planning***, the simplifications and assumptions that have been made while building the mathematical model of the system prevent the developed model from capturing all physical aspects of the real world.
- In ***actuation***, several factors such as friction inside the gearboxes, the backlash in the gears, and unmodeled payload effects constitute the main sources of uncertainty.

As a result, the actual movement of the robot differs from the desired one as seen in Fig. 3. Here, the plot with crosses illustrates the joint commands, i.e., the desired trajectory, and the plot with asterisks shows the actual trajectory followed by the joint. The section towards the end where

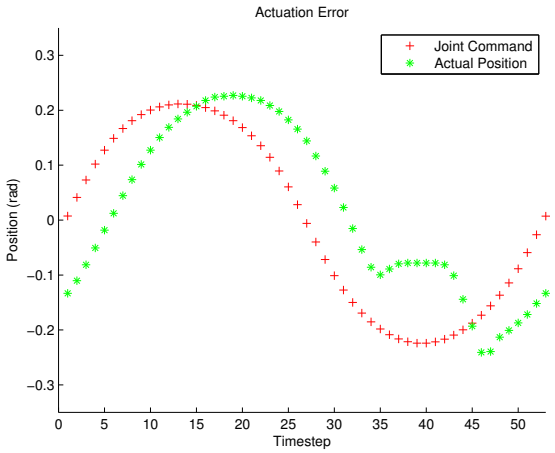


Fig. 3 An example actuation error. The ankle roll joint of the left leg is unable to follow the desired trajectory due to the weight of the body.

the actual joint position significantly digresses from the desired trajectory corresponds to a moment where the robot is standing on its left foot in the first single support phase and the movement of the ankle joint is affected by the weight of the whole body.

Failing to follow the desired trajectory of the joint causes the robot to act differently than expected and this difference affects the balance negatively. This kind of unforeseen or poorly modeled sources of uncertainty are the typical drawbacks of an open-loop controller, which is a type of controller that solely uses its model of the system to compute the next action and does not take any feedback from the environment into account to determine whether the desired state is reached. A closed-loop controller, on the other hand, uses both its model of the system and the feedback received from the system to determine the next action. Similarly, an open-loop walk algorithm generates a set of joint angle commands at each execution timestep to form a walk pattern without taking the actual state of the robot (i.e., sensory readings) into account while a closed-loop walk algorithm incorporates the sensory feedback into the joint command generation process in such a way that the resulting walk motion keeps the robot balanced.

In the remainder of this section, we first present how an open-loop walk cycle can be captured by observing the output of an existing walk algorithm. We then present how the obtained open-loop walk can be further improved offline using the *Advice Operators Policy Improvement* method [4]. In the following sections, we present how a closed-loop walk can be built on top of the obtained open-loop walk.

3.1 Obtaining an Open-loop Walk

If a walk algorithm is readily available at hand, one way of obtaining a walking behavior without directly employing the algorithm is to observe the output of the algorithm and

generate a single walk cycle out of those observations to be played back in a loop. To accomplish this, we use the existing walk algorithm as a black-box and record a number of walk sequences where the robot walks forwards for a fixed distance at a constant speed using the selected algorithm. We record the sequences in which the robot was able to travel the predetermined distance while maintaining its balance.

A set of examples of the robot walking without falling provide data D for each $t, t \in [0, T)$, in the form of the commands received by each joint $\overrightarrow{D_j}(t)$ and the corresponding sensory readings $\mathbf{S}(t)$ provided by the set of sensors $Sensors$. We obtain a single walk cycle wc using D as $wc_t^j = \mu(\overrightarrow{D_j}), j \in Joints, t \in [0, T)$. In addition, we fit a normal distribution $N(\overrightarrow{\mu(t)}, \overrightarrow{\sigma(t)})$ to the readings of each sensor at each t, where $\mu_s(t)$ is the mean, and $\sigma_s(t)$ is the standard deviation for the readings of the sensor $s \in Sensors$ at time t in the walk cycle (Fig. 4).

Sending the joint commands in the obtained walk cycle to the robot repetitively, hence playing back the captured walk cycle in a loop yields an open-loop walk behavior that performs similar to the original walking algorithm without employing the algorithm itself. Although the Nao robot has a total of 21 joints, for our experiments we utilize only 12 of them, which are all the leg joints except the shared hip yaw-pitch joint and the shoulder roll joints for the arms, constituting the set $Joints$.

3.2 Corrective Demonstration Using Advice Operators for Offline Improvement

Advice Operators Policy Improvement (A-OPI) is a method for improving the execution performance of the robot in a human-robot learning from demonstration (LfD) setup [4]. Advice operators provide a *language* between the human teacher and the robot student, allowing the teacher to give *advice* as a mathematical function to be applied on the actions in the demonstration database and/or the observations corresponding to those actions. The resulting data is then used to re-derive the execution policy. More formally, for the defined advice operators $O = \{o_1, o_2, \ldots, o_N\}$, there is a set of corresponding mathematical functions

$$F = \{f_1(X_1), f_2(X_2), \ldots, f_N(X_N)\}$$

where $X_o =< x_1, x_2, \ldots, x_K >$ is the parameter vector for the advice operator o. For each received advice o along with its parameter vector X_o, the corresponding mathematical function $f_o(X_o)$ is applied on the observations Z and/or actions A such that $Z' \leftarrow f_o(X_o, Z)$ and/or $A' \leftarrow f_o(X_o, A)$. Advice operators are especially useful in domains with continuous state/action spaces where the correction must be provided in continuous values.

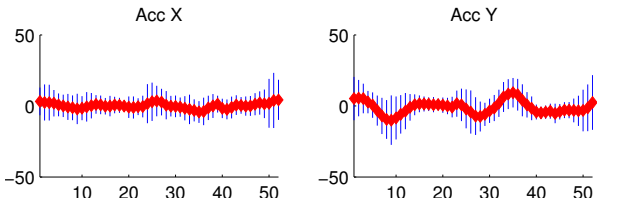


Fig. 4 Distribution of the sensor values over the complete walk cycle for a stable walk sequence. The middle line denotes the mean and the vertical lines denote +/- 3σ variance. The x-axis is timesteps, and the y-axis is the sensor value.

We use A-OPI for correcting the obtained walk cycle in its open-loop form based on human observations of the executed walk behavior. We defined three advice operators $O = \{ScaleSwing, ChangeFeetDistance, ChangeArms\}$ that are applied on the walk cycle:

- **ScaleSwing(k):** Scales the joint commands of the hip roll joints (along the X axis) in the walk cycle by a factor of k where $k \in [0, 1]$. The hip roll joints generate the lateral swinging motion while walking. The implemented function for the ScaleSwing operator is a coefficient to be applied on the joint commands for both hip-roll joints:

  $$f_{ScaleSwing}(k) = wc_t^j \leftarrow k \cdot wc_t^j$$

  where $j \in \{HipRollLeft, HipRollRight\}$, and $\forall t \in [0, T]$.
- **ChangeFeetDistance(d):** Applies an offset of d millimeters to the distance between the feet along the Y axis. The operator function is implemented using the forward and inverse kinematics methods explained in Section 4.3. For each timestep t in wc, first the position of the feet is computed using forward kinematics, then the offset d is applied on the inter-feet distance, and finally the modified joint angles wc'_t are computed using the inverse kinematics.
- **ChangeArms($angle$):** Raises or lowers the arms by $angle$ radians along the $Y - Z$ plane. The implemented function for the ScaleSwing operator is an offset to be applied to the shoulder roll joint commands:

  $$f_{ChangeArms}(angle) = wc_t^j \leftarrow wc_t^j + angle$$

  where $j \in \{ShoulderRollLeft, ShoulderRollRight\}$, and $\forall t \in [0, T]$.

During the A-OPI improvement, the robot executes the walk with the current walk cycle as the teacher observes the robot. Then, the teacher applies the operators on the walk cycle using a custom user interface running on the host computer. Finally, the walk cycle modified by the defined functions for the operators is transmitted to the robot over wire-

less network. It replaces the old walk cycle on the robot and the robot goes back to executing the walk motion using the newly received modified walk cycle. Repeating this process for a set of iterations, an improvement on the walk stability is achieved over the initial walk cycle. The initial and improved versions of hip roll joint values to generate lateral swinging motion are shown in Fig. 5 as an example. Here, decreasing the amplitude of the hip roll joint signal causes the robot to swing less, which contributes to preservation of balance positively.

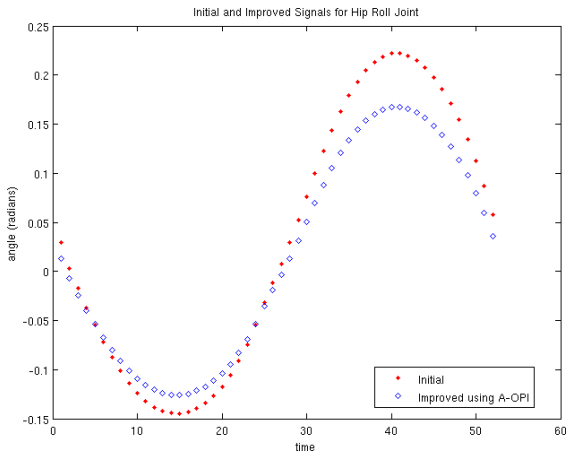


Fig. 5 Initial and improved joint commands for hip roll joints generating swinging motion while walking.

4 Real-Time Complementary Corrective Demonstration

The complementary corrective demonstration approach complements the output of an algorithm with human feedback, whenever the default output of the algorithm fails to perform as expected. A correction policy is learned using the received corrective demonstrations, and the perceived state of the system in which those demonstrations were received. The underlying algorithm remains untouched; the learned correction policy encapsulates the algorithm output and modifies the output if a correction is needed. A sketch of the complementary corrective demonstration system applied on biped walk is depicted in Fig. 6.

Considering the open-loop playback walk with the advice operator improvement as the underlying algorithm, the next step is to close the loop by adding a mechanism to modify the open-loop walk cycle during the autonomous execution according to the feedback received from the system. In the remainder of this section, we first present our corrective demonstration setup, elaborating on the implementation details. We then present two different correction methods for forward walking along with a simplified inverse kinematics model for the Nao.

4.1 Corrective Demonstration Setup for Biped Walking

A major challenge in providing corrective demonstration for the biped walking process is to find a proper way of delivering the demonstration as fast as possible without physically contacting the robot. Fast delivery is needed because biped walk is such a delicate dynamic process that it might be too late to recover from a balance loss if the robot receives the provided correction signal with a significant delay. Another problem with late delivery is that in such a case the received demonstration is associated with the wrong set of sensory data; hence, it results in an erroneous association of the demonstration points with the state information in the policy generation process. The necessity of delivering the demonstration without touching the robot also stems from the delicate dynamics of the biped walking process since interfering with those dynamics of the robot affects the learned policy negatively.

We utilize a wireless control interface using the commercially available *Nintendo Wiimote* game controller [23] to deliver corrective demonstration to the robot. Both the Wiimote controller and its *Nunchuk* extension are equipped with accelerometers measuring the acceleration of the controllers as well as allowing their absolute roll and pitch orientations to be computed. Therefore, the Wiimote with its extension has four measurable axes allowing four different correction signals to be delivered simultaneously. The computed roll and pitch angles are in radians and they use the right-hand frame of reference.

We use a custom developed software framework for delivering the correction signal received from the Wiimote by the host computer to the robot over wireless Ethernet connection as fast as possible (Fig. 6). The custom software also provides the demonstrator an interface to define scaling and shifting operators on the received signals from the Wiimote before transmission to the robot, allowing the demonstration signal to be scaled up or down. By scaling down the demonstration signals, it is possible to reduce the undesirable noise factors like trembling hands of the demonstrator. The position of the Wiimote which is connected to the host computer over a Bluetooth connection is sampled and the processed demonstration signals are transmitted to the robot over a UDP connection via wireless network at a frequency of 1KHz; therefore, even if some of the UDP packets are dropped due to the network conditions, we can still deliver the demonstration signal packets at around 50Hz, which is the update frequency for the sensors and the actuators of the robot.

The demonstrator delivers the corrective demonstration signals to the robot by changing the orientations of the Wiimote and the Nunchuk controllers in real time while the robot is walking using the open-loop walk cycle. We record the received correction signals during the demonstrations

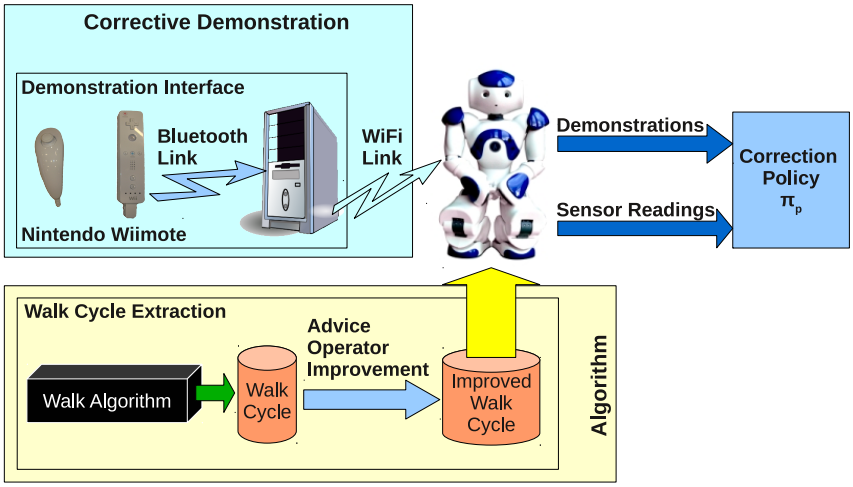


Fig. 6 The diagram for the complementary corrective demonstration framework along with the biped-walk specific parts shown within the main components.

synchronously with the rest of the sensor readings at 50Hz. The Nunchuk extension and the Wiimote control the left and the right side corrections on the robot, respectively (Fig. 7). We define two different methods for converting the Wiimote signals into the correction signals to be applied on the robot:

- Applying correction signals in the joint space by means of direct modifications to the joint commands.
- Applying correction signals in the task space by means of feet position displacements.

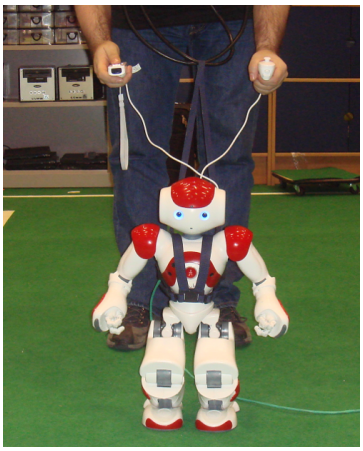


Fig. 7 A snapshot from a demonstration session. A loose baby harness is used to prevent possible hardware damage in case of a fall. The harness neither affects the motions of the robot nor lifts it as long as the robot is in an upright position.

4.2 Applying Correction in the Joint Space

In this correction method, we associate the four correction signals received from the demonstrator to the four individual joints on the hip. Namely, we use the hip roll and the hip pitch joints to apply the correction signals. To provide a finer control ability to the demonstrator, a scaling factor γ is applied on the Wiimote readings using the interface described above for scaling the demonstration signals before they are transmitted to the robot. We used $\gamma = 0.1$ in our implementation. The received roll corrections are applied on the hip roll joints and the received pitch corrections are applied on the hip pitch joints. The following correction values are applied on the ankle roll and the ankle pitch joints to keep the feet parallel to the ground:

$$C_{AnkleRoll} = -C_{HipRoll}$$
$$C_{AnklePitch} = -C_{HipPitch}$$

At each timestep, we compute the correction values for all joints $j \in Joints$ using the defined correction functions. We then add the calculated values to the joint command values in the walk cycle for that timestep before sending the joint commands to the robot. The correction is applied to the system at each m^{th} timestep where $1 \leq m \leq T$ where T is the length of the walk cycle in timesteps.

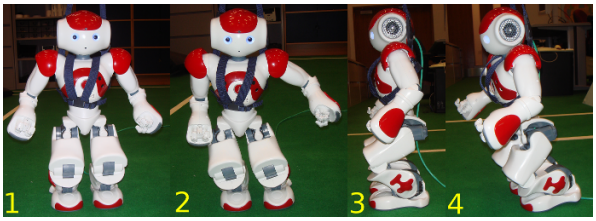


Fig. 8 Applying correction in the joint space. Rolling the Wiimote to the right transitions the robot from its neutral posture (1) to a posture bent along the Y axis (2). Similarly, tilting the Wiimote forward transitions the robot from its neutral posture (3) to a posture bent along the X axis (4).

4.3 Applying Correction in the Task Space

In this correction method, we modify the feet positions in the 3D space with respect to the torso center by mapping the received correction values to the offsets along the X-Y plane instead of applying the correction signals directly to the joints. At each timestep of playback, the vector of joint command angles for that timestep is used to calculate relative positions of the feet in 3D task space with respect to the torso using forward kinematics. The calculated corrections (in the autonomous mode), or the received corrections (during the demonstration) are applied on the feet positions in 3D space and the resulting feet positions are converted back into a vector of joint command angles using inverse kinematics and sent to the robot (Fig. 9).

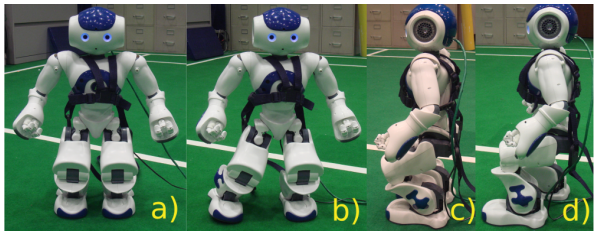


Fig. 9 Applying correction in the task space as feet position displacement. Rolling the Wiimote to the right takes the right leg of the robot from its neutral posture (a) to a modified posture along the Y axis (b). Similarly, tilting the Wiimote forward brings the right leg of the robot from its neutral posture (c) to a modified posture along the X axis (d).

Due to the physically coupled hip-yaw joints of the Nao, inverse kinematics for feet positions cannot be calculated independently for each foot. Graf *et al.* propose an analytical solution to inverse kinematics of the Nao, presenting a practical workaround for the coupled hip-yaw pitch joints constraint [17]. We used a simplified version of this approach by assuming the hip-yaw joints to be fixed at 0 degrees for the straight walk. The desired position Pos of the foot with respect to the hip joints is given in the form of a homogeneous transformation matrix.

We assume a stick figure model for the feet as shown in Fig. 10. The thigh and the tibia (upper and lower leg parts) form a triangle with the imaginary edge d_{foot} which represents the distance of the foot from the hip. This distance equals the magnitude of translation vector $\mathbf{t}$ and can easily be calculated as $d_{foot} = |\mathbf{t}|$. The angle β between the upper and lower leg parts can be calculated using the law of cosines.

$$d_{foot}^2 = l_{thigh}^2 + l_{tibia}^2 + 2l_{thigh}l_{tibia}\cos\beta$$

$$\beta = \arccos\frac{l_{thigh}^2 + l_{tibia}^2 - d_{foot}^2}{2l_{thigh}l_{tibia}}$$

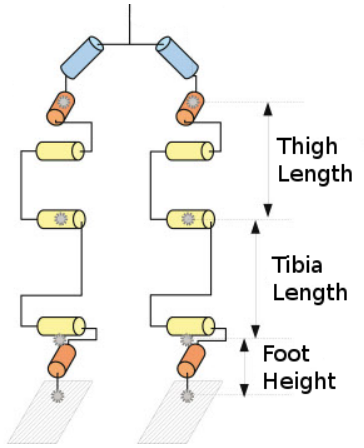


Fig. 10 Kinematic configurations for the legs of the Nao robot.

where l_{thigh} and l_{tibia} are the length of the thigh and the tibia, respectively. When the leg is fully extended, the knee pitch joint angle $\alpha_{KneePitch} = 0$; therefore, the resulting angle for knee pitch is calculated as $\alpha_{KneePitch} = \pi - \beta$. The angle between lower leg and foot plane constitutes the first part of the final ankle pitch angle and can be computed by the law of cosines.

$$\gamma = \arccos\frac{l_{tibia}^2 + d_{foot}^2 - l_{thigh}^2}{2l_{tibia}d_{foot}}$$

The second part of the ankle pitch angle is calculated using the components of the translation vector

$$\theta = atan2(t_x, \sqrt{t_y^2 + t_z^2})$$

where, $atan2(y, x)$ calculates the angle between the X axis, and the point (x, y).

The final ankle pitch angle value is the sum of its two components; that is, $\alpha_{AnklePitch} = \gamma + \theta$. The hip roll angle value is also calculated using the translation vector. Similar to the hip pitch joint, its final angle value is equal to the exterior angle value; that is, $\alpha_{HipRoll} = \pi - atan2(t_y, t_z)$. The value of the ankle roll angle is the difference between the desired absolute orientation of the foot along the X axis (calculated using the rotation matrix part of Pos), and the calculated hip roll joint angle value

$$\alpha_{AnkleRoll} = \arcsin(p_{32}) - \alpha_{HipRoll}$$

where p_{32} is the third row and the second column of Pos. Finally, the hip pitch angle value is calculated as

$$\alpha_{HipPitch} = -(\alpha_{KneePitch} + \alpha_{AnklePitch})$$

Any given valid joint command vector satisfying the assumptions stated at the beginning of this subsection can be converted into the relative positions of the feet in the 3D task space using the method described above.

5 Closed-Loop Walking Using Playback And Corrective Demonstration

With the correction methods described in the previous section, we can collect demonstration data consisting of the sensory readings representing the state of the system as it is perceived by the robot, and the correction values provided by the demonstrator based on her observation of the state of the robot. To obtain a closed-loop gait, we need a function that associates the current state of the robot as it is perceived by the robot to the corresponding demonstration values so that we can use that association to infer the appropriate correction values to be applied for a given system state. We use the sensors on the robot to estimate the current state of the robot. The changes in sensor readings when the robot is about to lose its balance (Fig. 11) are used to derive a correction policy by mapping these changes to corrective feedback signals.

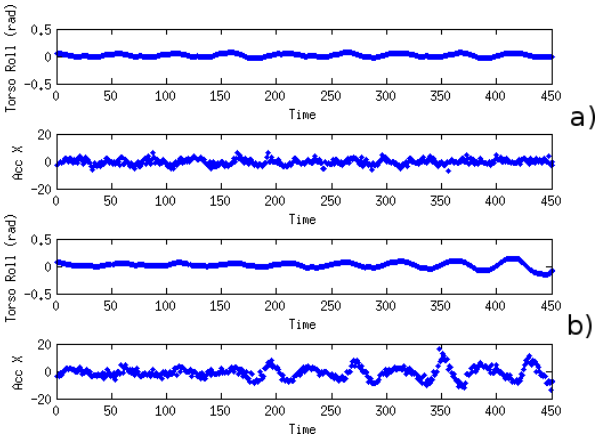


Fig. 11 Sample torso orientation and accelerometer readings: a) a stable walk sequence, and b) a walk sequence where the robot starts losing its balance after around 200^{th} timestep.

Due to the noisy nature of the sensors, fluctuations may occur in the sensor readings and that may result in jerky motions that lead to loss of balance when the correction values calculated as a function of the sensor readings are applied to the joints directly. Therefore, the readings need to be filtered. Running mean and median smoothers are widely used methods for filtering noisy data. In running smoothers, the data point in the middle of a running window of size N is replaced with the mean or the median of the data points lying within that window. The filtered signal gets smoother as the window size increases. The delicate trade-off in filtering lies in the selection of an appropriate window size for smoothing the data just enough to filter out the noise without rendering the patterns in the data hard to detect.

We evaluated the running mean and median smoothers with window sizes 5 and 10 (Fig. 12), and decided to use a running mean filter with window size 5 since it filters out the noise reasonably well and is computationally cheaper than the running median filter. Also, considering our sensor sampling rate is 50 Hz, we can still detect a significant change in the sensor readings in at most $1/10^{th}$ of a second.

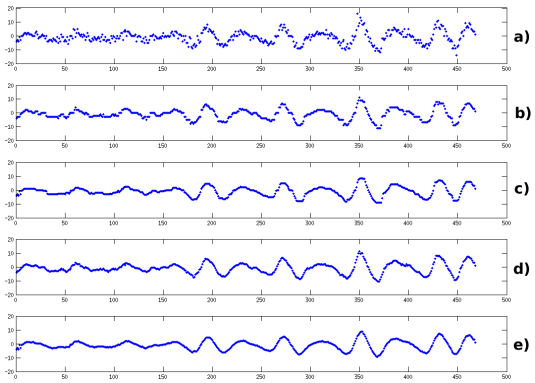


Fig. 12 Results of applying various smoothers on an example accelerometer data: a) the raw data, b) median smoother with window size 5, c) median smoother with window size 10, d) mean smoother with window size 5, e) mean smoother with window size 10.

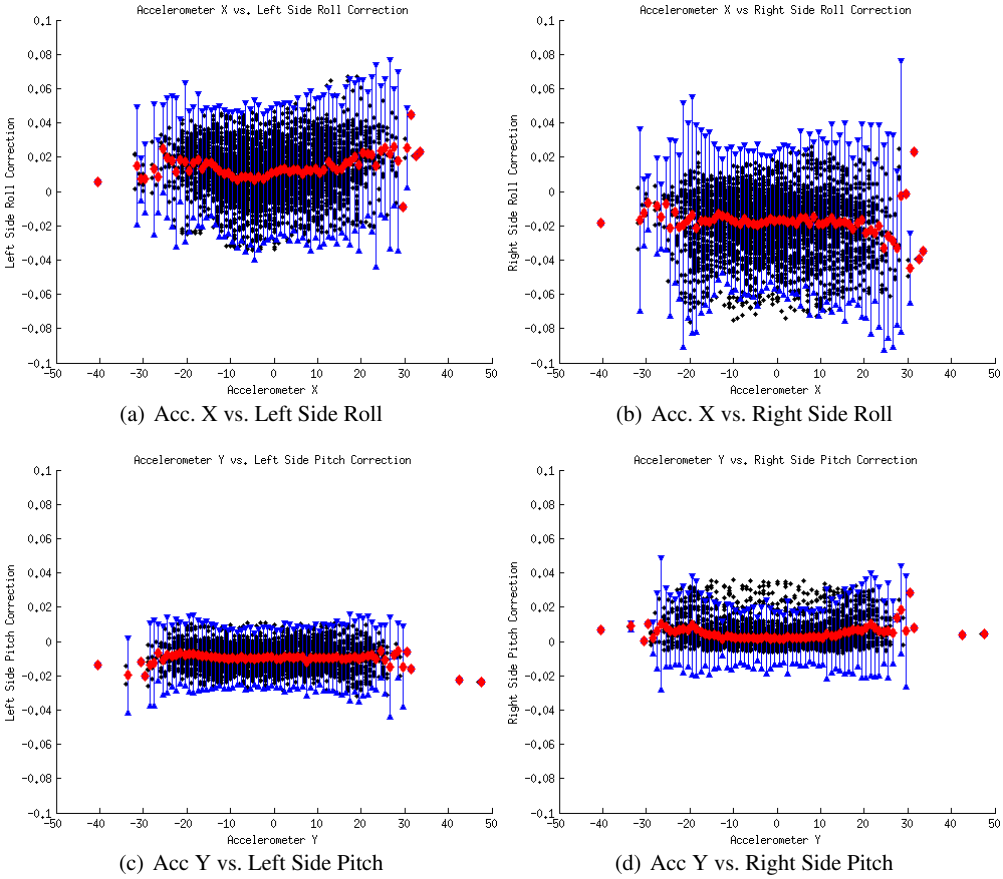
We present two different sensor-correction association methods:

- Associating a single sensor with joint space correction
- Associating multiple sensors with task space correction

5.1 Associating a Single Sensor with Joint Space Correction

In this method, we apply the correction on individual joints, and we define the correction value for a joint as a function of a single sensor reading. We use the accelerometer readings along the X and Y axes as the sensory input. Each point in the resulting demonstration dataset is a tuple $< \overrightarrow{S}, \overrightarrow{C} >$ where $\overrightarrow{S} = \{Acc_X, Acc_Y\}$ is the vector of accelerometer readings, and $\overrightarrow{C} = \{C^{left}_{roll}, C^{left}_{pitch}, C^{right}_{roll}, C^{right}_{pitch}\}$ is the vector of received correction values for the left hip roll, the left hip pitch, the right hip roll, and the right hip pitch joints, respectively. The accelerometers on the Nao can measure accelerations in the range $[-2g, 2g]$ where g is the standard gravity and their readings are integer values in the interval $[-128, 127]$. To model the noise associated with the demonstration data, we fit a normal distribution on the correction data points received for all 256 values of the accelerometer. The resulting distributions versus the accelerometer readings populated using approximately 6000 correction points out of about 30000 points recorded in a single demonstration session of roughly 10 minutes are given in Fig. 13.

Any discontinuity or a sudden change in the correction signal causes a jerky movement of the robot and further con-



(a) Acc. X vs. Left Side Roll (b) Acc. X vs. Right Side Roll

(c) Acc Y vs. Left Side Pitch (d) Acc Y vs. Right Side Pitch

Fig. 13 The normal distributions fit on the received correction data versus the accelerometer readings for the single sensor - joint space correction association. The bold points in the middle denote the mean, and vertical lines denote the variance of the normal distribution fit on that sensor value interval.

tributes to the loss of balance. To deal with it, the correction is modified to be a mapping from the sensory reading to the mean of each joint command to be corrected, namely, the left hip roll, the left hip pitch, the right pitch, and the right hip roll. During autonomous execution, given the perceived sensory data, the corresponding mean value is added to the walk cycle commands. The computed correction values are applied to the walk cycle commands at each N^{th} timestep, where N is a predefined value that does not change during the execution. The pseudo-code of that process is given in Algorithm 1.

In addition, we defined a hand-tuned simple linear function to be used as a benchmark closed-loop gait in our experiments. We use the roll and the pitch angles of the torso, calculated by the inertial measurement unit as the sensor readings and associate them with the hip roll and the hip pitch joints. The inertial measurement unit returns the roll and pitch orientations of the torso in radians with respect to the ground. The used linear coupling functions are of the form $C = AX + B$ where A is a gain value, B is an offset value, X is the sensor reading, and C is the calculated correction value. For the four hip joints to be corrected, we have four functions with individually set A and B values. We hand-tuned the parameters of these four functions using expert knowledge and visual observation of the robot walking. The resulting hand-tuned policy provided an improve-

Algorithm 1 Closed-loop walking using single sensor-joint space correction association.

$t \leftarrow 0$
loop
  $\mathbf{S} \leftarrow readSensors()$
  $\mathbf{S} \leftarrow smoothen(\mathbf{S})$
  for all $j \in Joints$ **do**
    if $timestep$ MOD $correctioninterval = 0$ **then**
      $C_j = Correction(\mathbf{S}, j)$
    else
      $C_j = 0$
    end if
    $NextAction_j \leftarrow wc_t^j + C_j$
  end for
  $t \leftarrow t + 1 \pmod T$
end loop

ment over the initial open-loop walk. Details of the results are given in the Results section.

5.2 Associating Multiple Sensors with Task Space Correction

In this method, the correction values received during the demonstration are recorded synchronously with the sensory readings, tagged with the current position in the walk cycle. Each point in the resulting demonstration dataset is a tuple $< t, \overrightarrow{S}, \overrightarrow{C} >$, where t is the position in the walk cycle at the time when this correction is received, $\overrightarrow{S} = \{Acc_X, Acc_Y\}$ is the vector of accelerometer readings, and $\overrightarrow{C} = \{C_X^{left}, C_Y^{left}, C_X^{right}, C_Y^{right}\}$ is the vector of received correction values for the left foot along the X axis, the left foot along the Y axis, the right foot along the X axis, and the right foot along the Y axis, respectively.

We utilize locally weighted regression with a Gaussian kernel [7] for generalizing a policy using the recorded correction and sensor values. For each received sensor reading vector $\overrightarrow{S}$, we calculate the correction vector $\overrightarrow{C}$ as follows:

$$d_i(t) = e^{-\sqrt{(\overrightarrow{S}-\overrightarrow{S_i}(t))^T \Sigma^{-1} (\overrightarrow{S}-\overrightarrow{S_i}(t))}}$$

$$\overrightarrow{C}(t) = \frac{\sum_i d_i(t)\overrightarrow{C_i}(t)}{\sum_i d_i(t)}$$

where Σ is the covariance matrix of the sensory readings in the demonstration set, $\overrightarrow{C_i}(t)$ is the i^{th} received correction signal for the walk cycle position t, $\overrightarrow{S_i}(t)$ is the i^{th} sensory reading for the walk cycle position t, $\overrightarrow{S}(t)$ is the current sensory reading, $\overrightarrow{C}(t)$ is the calculated correction value to be applied, and t is the current position in the walk cycle.

The calculated correction values are applied only if any of the sensor values are not in the range $\mu_t \pm K\sigma_t$ (i.e., if an abnormal value is read from that sensor, meaning that the robot is losing its balance) where K is a coefficient, and t is the current position in the walk cycle. In our implementation, we chose $K = 3$ so the correction values are applied only if the current sensory readings are outside the range $\mu_s(t) \mp 3\sigma_s(t)$, corresponding to the $\%99$ of the variance of the initial sensory model given in Section 3.1. The pseudo-code for multiple sensors - feet position displacement association is given in Algorithm 2.

Algorithm 2 Closed-loop walking using multiple sensors-task space correction association. Pos_{left} and Pos_{right} are the positions of the feet in 3D space.

```
t ← 0
loop
  S(t)⃗ ← readSensors()
  S(t)⃗ ← smoothen(S(t)⃗)
  Pos_left, Pos_right ← forwardKine(wc_t)
  if (μ_s(t) − Kσ_s(t) ≤ S_s(t) ≤ μ_s(t) + Kσ_s(t)) then
    C_left, C_right ← 0
  else
    C_left, C_right ← correction(S(t)⃗)
  end if
  Pos_left ← Pos_left + C_left
  Pos_right ← Pos_right + C_right
  NextAction ← inverseKine(Pos_left, Pos_right)
  t ← t + 1 (mod T)
end loop
```

6 Results

To evaluate the performance of the proposed methods, we conducted a set of walking experiments on a flat surface covered with carpet. We used the walking algorithm proposed by Liu and Veloso as the black-box algorithm [20]. The duration of the extracted walk cycle is 52 individual timesteps, approximately corresponding to one second.

We evaluated different combinations of the proposed correction, sensory association, and policy derivation methods as follows:

(a) ***OL :***
Initial open-loop playback walk.

(b) ***OL+JS+SS+HT+FC :***
Closed-loop playback walk with the joint space correction policy (JS) using hand-tuned (HT) single sensor-correction association (SS) on top of the original open loop walk cycle (OL), and the fixed frequency application of the correction (FC) twice a walk cycle ($N = 26$).

(c) ***OL+JS+SS+NF+FC :***
Closed-loop playback walk with the joint space correction policy (JS) using single sensor-correction association (SS), normal distribution fit (NF) on top of the original open loop walk cycle (OL), and the fixed frequency application of the correction (FC) twice a walk cycle ($N = 26$).

(d) ***OL+AO :***
Open-loop playback walk cycle (OL) after offline improvement using advice operators (AO).

(e) ***OL+AO+TS+MS+LWR+AC :***
Closed-loop playback walk with the task space correction policy (TS) using multiple sensors - correction association (MS), locally weighted regression (LWR) as the policy extraction method on top of the advice improved walk cycle (OL+AO), and the application of the correction under state anomaly (AC), in other words, when the sensory readings go beyond the $\pm 3\sigma$ of the normal sensory readings (Fig. 4).

We used two benchmark combinations (case (a) and case (b)), the former being the base case and the latter being

a simple closed-loop method with hand tuned parameters as described in Section 5.1. For each combination, we performed 10 runs and measured the distance traveled before falling. The results are given in Fig. 14 as boxplots, where the lines within the boxes mark the mean, the marks at both ends of boxes indicate minimum and maximum distances, and the left and right edges of boxes mark 25^{th} and 75^{th} percentiles, respectively. The black bar at the bottom of the figure marks 7.2 meters, which is the longest possible straight walk distance on a regular RoboCup SPL field which is sized 6 meters by 4 meters.

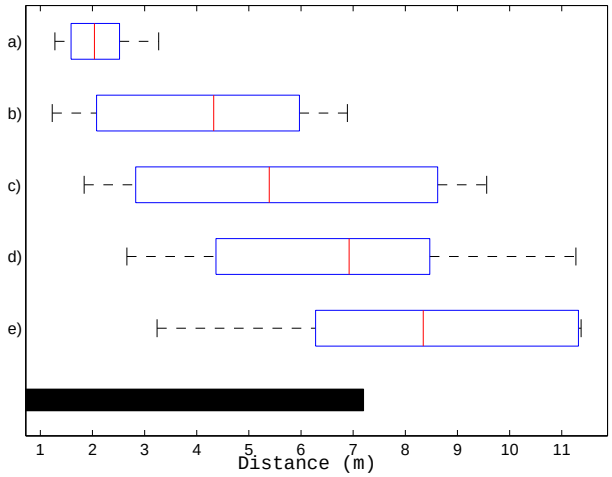


Fig. 14 Performance evaluation results: a) OL, b) OL+JS+SS+HT+FC, c) OL+JS+SS+NF+FC, d) OL+AO, e) OL+AO+TS+MS+LWR+AC. The black bar at the bottom marks 7.2 meters, the longest straight walk distance on a regular RoboCup SPL field.

During three demonstration sessions of 28 minutes, a total of about 83000 demonstration points are recorded for both joint space and task space corrections, and 25428 of them corresponding to about 489 walk cycles are selected as good examples of corrective demonstration by visually inspecting the demonstration data based on the changes in the sensory readings towards the recovery of balance.

The mean and maximum distances that the robot could travel using the initial open loop benchmark walk (case (a)) were 2.03 and 3.27 meters, respectively, while the mean and the maximum distances the robot was able to travel using the closed loop benchmark walk (case (b)) were 4.32 and 6.89 meters, respectively. The performance difference between the two benchmark cases stems from the fundamental difference between the open-loop and the closed-loop control paradigms under the presence of uncertainty and noise in the environment.

All of the combinations involving proposed methods outperformed the benchmark cases (a) and (b). The combination (c) which is directly comparable to the combination (b) demonstrated considerable improvement over (b), reaching a maximum traveled distance of 9.56 meters with a mean traveled distance of 5.39 meters[3]. The improvement in the performance could be accounted for the non-linear relationship between the computed means for the received correction and the accelerometer readings (as seen in Fig. 13), which could not be captured appropriately by the assumed linear relation function in the hand tuned case.

The open-loop walk improved with advice operators (d) outperformed the closed-loop case (c), reaching a maximum traveled distance of 11.27 meters with a mean traveled distance of 6.92 centimeters. During the advice operator improvement, the teacher continuously observes the robot and gives high level advice which corresponds to a systematic correction to the walk cycle. Taking a closer look at the mean correction values in Fig. 13, we see that the mean values are off from the zero position by a fixed offset in addition to the nonlinear relation of the sensory readings to the received correction value. This offsets are results of the implicit high level correction of the same systematic error by the demonstrator. An explanation for why the improved open loop walk did better compared to case (c) could be that it is easier to focus on the *"big picture"* and hence to spot the systematic error when the teacher is solely observing the robot rather than being actively involved in delivering real-time correction to the robot.

Despite the fact that the application of the advice operators on the walk cycle resulted in a considerably improved walk performance with the maximum and mean traveled distances of 11.37 (the maximum length of the available experimentation area) and 8.34 meters, respectively, the last case (case (e)) shows that there is still room for improvement with the real-time corrective demonstration over the improved open-loop walk.

Although the results suggest that the more complex options for the correction type (task space correction instead of joint space correction), sensor-correction association (multiple sensors - task space correction association instead of single sensor - joint space correction association), policy derivation method (locally weighted regression for the individual timesteps within the walk cycle instead of fitting normal distributions for the whole walk cycle), and the application of the correction (applying correction only if the current perceived state differs from the normal values instead of applying correction at each N^{th} timestep regardless of the sensory readings) yielded better performance, we do not possess enough experimental evidence to claim such superiority. The main motivation of this study was to answer the question of whether it would be possible to improve the performance of a system using human demonstration if a

[3] The open-loop walk performance was comparable to the performance of the original ZMP-based walk, which was not available to be accounted for in this empirical comparison.

solution to the problem with partial success exists and available as a black box. Therefore, we left the comparison of all the individual components against each other in all possible settings (e.g., evaluating two policy extraction methods against each other by keeping all other components the same for all possible combinations of the other components) as future work.

7 Conclusions

In this article, we presented an approach that incorporates corrective human demonstration for improving the performance of an existing partially successful system for performing a low-level task and its application on improving the walk stability of the Nao humanoid robot using corrective human demonstration. We analyzed the Nao robot in terms of the variations of joint commands and sensor readings. The key question we tried to answer was whether it would be possible to improve the performance of an existing controller for performing a complex skill on a complex robotic platform without knowing the underlying technical details of the existing controller. We tackled the problem by making use of a human teacher who is able to externally observe the robot performing the skill using the existing controller. We utilized corrective human demonstration given in two phases (first offline and then in real-time) to learn a policy for modifying the joint commands in the open-loop walk cycle during the autonomous execution in such a way to keep the robot balanced.

Our method plays back a single walk cycle extracted from an existing walk algorithm to obtain an open-loop walk behavior. We introduced an offline method using advice operators to improve the stability of the open-loop walk cycle. We utilized the data collected from the real-time corrective human demonstration delivered using a Wiimote wireless game controller and smoothened sensory readings of the robot to learn the appropriate correction values to the joint commands of the open-loop walk. We proposed two correction methods, one in joint space and one in task space, two methods for associating the received corrections with the state of the robot at the time of reception, two different policy extraction methods, and two methods for deciding when to apply the correction signal to the system during the autonomous execution. We presented experiment results for different scenarios using different combinations of the proposed methods, demonstrating a performance improvement over the initial open-loop walk after we applied the correction calculated using the proposed methods. We discussed the experimental results in detail, proposing possible explanations for the comparison of the performances of different combinations.

Addressing the delay between the perception and the actuation of the demonstrator, generalizing the proposed approach to a multi-phase learning framework applicable to other skill learning problems, investigating better policy derivation methods, assessing the demonstration quality and incorporating this information into the policy learning process, relaxing the flat surface assumption to cope with uneven terrain, and extending the balance maintenance capability to endure against moderate disturbances in adversarial domains (e.g., getting pushed in a robot soccer game) are among the issues we aim to address in the future.

Acknowledgements The authors would like to thank Tekin Meriçli for the valuable feedback on the manuscript. We further thank the Cerberus team for their debugging system, and the CMWrEagle team for their ZMP-based robot walk.

References

1. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: In Proceedings of the Twenty-first International Conference on Machine Learning, URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.6759
2. Aldebaran (2008) Aldebaran Robotics - Nao Humanoid Robot. Http://www.aldebaran-robotics.com/pageProjetsNao.php
3. Argall B, Browning B, Veloso M (2007) Learning from demonstration with the critique of a human teacher. In: Second Annual Conference on Human-Robot Interactions (HRI'07)
4. Argall B, Browning B, Veloso M (2008) Learning robot motion control with demonstration and advice-operators. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)
5. Argall B, Sauser E, Billard A (2010) Tactile feedback for policy refinement and reuse. In: In Proceedings of the 9th IEEE International Conference on Development and Learning (ICDL '10), Ann Arbor, Michigan, August 2010
6. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robotics and Automation Systems 57(5):469–483, DOI http://dx.doi.org/10.1016/j.robot.2008.10.024
7. Atkeson C, Moore A, Schaal S (1997) Locally weighted learning. AI Review 11:11–73
8. Atkeson CG, Schaal S (1997) learning tasks from a single demonstration. In: ieee international conference on robotics and automation (icra97), piscataway, nj: ieee, pp 1706–1712, URL http://www-clmc.usc.edu/publications/A/atkeson-ICRA1997.pdf

9. Atkeson CG, Schaal S (1997) robot learning from demonstration. In: machine learning: proceedings of the fourteenth international conference (icml '97), morgan kaufmann, pp 12–20, URL http://www-clmc.usc.edu/publications/A/atkeson-ICML1997.pdf
10. Bentivegna DC, Atkeson CG, Cheng G (2006) Learning similar tasks from observation and practice. In: in Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2677–2683
11. Breazeal C, Hoffman G, Lockerd A (2004) Teaching and working with robots as a collaboration. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, IEEE Computer Society, Washington, DC, USA, pp 1030–1037, DOI http://dx.doi.org/10.1109/AAMAS.2004.258
12. Chernova S, Veloso M (2008) Teaching collaborative multirobot tasks through demonstration. In: In Proceedings of AAMAS'08, the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems, Estoril, Portugal, May 2008
13. Chernova S, Veloso M (2009) Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research 34
14. Czarnetzki S, Kerner S, Urbann O (2009) Observer-based dynamic walking control for biped robots. Robotics and Autonomous Systems 57:839–845
15. Gokce B, Akin HL (2009) Parameter optimization of a signal-based biped locomotion approach using evolutionary strategies. In: Proceedings of the Twelfth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2009), Istanbul, Turkey, 9-11 September 2009, O. Tosun, H. L. Akin, M. O. Tokhi and G S Virk (Ed.), World Scientific, 2009
16. Gouaillier D, Hugel V, Blazevic P, Kilner C, Monceaux J, 0002 PL, Marnier B, Serre J, Maisonnier B (2009) Mechatronic design of nao humanoid. In: ICRA, pp 769–774
17. Graf C, Härtl A, Röfer T, Laue T (2009) 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
18. Hersch M, Guenter F, Calinon S, Billard A (2008) Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. IEEE Transactions on Robotics 24(6):1463–1467
19. Kolter JZ, Abbeel P, Ng AY (2007) Hierarchical apprenticeship learning with application to quadruped locomotion. In: Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, MIT Press, DOI http://books.nips.cc/papers/files/nips20/NIPS2007_0985.pdf
20. Liu J, Veloso M (2008) Online ZMP Sampling Search for Biped Walking Planning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08), Nice, France, September 2008
21. Liu J, Chen X, Veloso M (2009) Simplified Walking: A New Way to Generate Flexible Biped Patterns. In: Proceedings of the Twelfth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2009), Istanbul, Turkey, 9-11 September 2009, O. Tosun, H. L. Akin, M. O. Tokhi and G S Virk (Ed.), World Scientific, 2009
22. Nakanishi J (2004) Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47(2-3):79 – 91
23. Nintendo (2007) Nintendo - Wii Game Controllers. Http://www.nintendo.com/wii/what/controllers
24. RoboCup (2009) RoboCup International Robot Soccer Competition. Http://www.robocup.org
25. Röfer T, Laue T, Müller J, Bösche O, Burchardt A, Damrose E, Gillmann K, Graf C, de Haas TJ, Härtl A, Rieskamp A, Schreck A, Sieverdingbeck I, Worch JH (2009) B-Human 2009 Team Report. Tech. rep., DFKI Lab and University of Bremen, Bremen, Germany, http://www.b-human.de/download.php?file=coderelease09_doc
26. Rybski PE, Yoon K, Stolarz J, Veloso MM (2007) Interactive robot task training through dialog and demonstration. In: In Proceedings of the 2007 ACM/IEEE International Conference on Human-Robot Interaction, Washington D.C, pp 255–262
27. RoboCup SPL (2009) The RoboCup Standard Platform League. Http://www.tzi.de/spl
28. Strom J, Slavov G, Chown E (2009) Omnidirectional walking using zmp and preview control for the nao humanoid robot. In: RoboCup 2009: Robot Soccer World Cup XIII
29. Thomaz AL, Breazeal C (2006) Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In: AAAI'06: Proceedings of the 21st national conference on Artificial intelligence, AAAI Press, pp 1000–1005