Complementary Humanoid Behavior Shaping using Corrective Demonstration

Çetin Meriçli, Manuela Veloso, and H. Levent Akın

***Abstract*— A humanoid robot can perform a task through a policy mapping from its sensed state to the appropriate task actions. We assume that a hand-coded controller can capture such a mapping only for the basic cases of the given task. As the complexity of the situation increases, the harder it becomes to refine the controller, and such refinements are often tedious and error prone. Based on the fact that a human can detect the failures of a robot executing the hand-coded controller, in this paper we present a corrective learning from demonstration approach to improve the robot performance. Corrections are captured as new state action pairs, and during the autonomous humanoid robot execution, the controller is replaced by the demonstration corrections when the new state is found to be similar to the corrected state. We focus on the Aldebaran Nao humanoid robot and a concrete complex ball dribbling task in an environment with obstacles. We present experimental results showing an improvement in the humanoid task performance when the corrective demonstration is used in addition to the basic hand-coded controller.**

I. INTRODUCTION

Creating a humanoid robot to effectively perform a task or a skill, even trivial ones, remains a difficult problem due to the high dimensional state and action spaces resulting from the large number of degrees of freedom for their motion and sensors. The partial observability of the environment due to limited and noisy sensing, and imperfect actuation makes the problem even more challenging.

Developers typically transfer task knowledge to a robot through a custom controller for performing the task or skill. Although it is relatively easy to develop a controller that can handle straightforward cases, it usually requires substantial changes in the controller to handle the real and complex cases, and as the number of such cases increases, it becomes cumbersome and time consuming to add new cases without interfering with the existing ones.

Learning from demonstration (LfD) is a recently popularized approach to policy learning from *examples*, or *demonstrations*, provided by a teacher [1]. LfD approaches are not only suitable for transferring task and skill knowledge via natural ways for humans, they are also suitable for the problems in which an overall analytical model for the task or skill is not available but a human teacher can tell which action to take in a particular situation. However, providing sufficient examples is a very time consuming process for robots with highly complex body configurations, like humanoids, and for sophisticated tasks with very high dimensional state and action spaces.

Multiple approaches to robot policy learning utilize learning from demonstration, in particular biped walk learning from human demonstrated joint trajectories using dynamical movement primitives [2], and quadruped walk learning for a Sony AIBO robot using a regression-based approach [3]. Learning from demonstration was introduced in the form of advice operators as functional transformations for low level robot motion, and was demonstrated for a Segway RMP robot [4], [5]. Furthermore the "confidence based autonomy" approach enables robot learning from demonstration of general behavior policies for both single robots [6] and multi-robot systems [7].

In our previous work, we utilized real-time corrective human demonstration to improve the biped walk stability of the Nao humanoid robot [8], [9]. We used an existing walk algorithm and we captured a complete walk cycle using the computed joint commands by the algorithm as the robot walks. We played back the obtained walk cycle to have a computationally cheap open-loop walking behavior. Finally we utilized a wireless game controller to allow the human demonstrator to modify the joint commands in real-time to keep the robot stable as the robot walks. We used the recorded demonstration values along with the sensor readings to derive a policy for computing proper modification values to the joint commands for a given sensory reading to recover the balance of the robot.

In this paper, we present a new corrective demonstration approach for task and skill improvement where a hand-coded algorithm exists but is imperfect in terms of being able to handle complex cases. The human demonstrator observes the robot doing the task and provides corrective feedback when necessary by taking over the control of the robot. The received demonstration points are stored along with the state of the robot as a complement to the initial hand-coded algorithm. During autonomous execution, if there is a demonstration point given in a state similar to the current state, the robot executes the demonstration action instead of the action computed by the hand-coded algorithm. The key idea is instead of deriving a policy out of the demonstrations and the output of the hand-coded algorithm, keep the algorithm as the primary source of the action policy, and use the demonstration data only to make exceptions

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 is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA and the Department of Computer Engineering Boğaziçi University Bebek, Istanbul 34342, Turkey cetin@cmu.edu

M. Veloso is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA veloso@cmu.edu

H.L. Akın is with the Department of Computer Engineering Boğaziçi University Bebek, Istanbul 34342, Turkey akin@boun.edu.tr

as needed. Furthermore, those exceptions do not have to be applied to the entire task space. Instead, assuming a decomposition of the task or skill into smaller tasks and skills is available to the demonstrator, a subspace of the task space can be selected and corrective demonstration can be given only for the selected subspace.

The organization of the rest of the paper is as follows: In Section II we give the problem definition and the hardware platform. In Section III we thoroughly describe the hand-coded behavior for a difficult humanoid dribbling task. In Section IV we present the special vision processing system and the corrective demonstration framework. We present experimental results in Section V yielding an improvement in the task completion time using corrective demonstration over the original hand-coded algorithm. The conclusions are given in Section VI.

II. PROBLEM DEFINITION AND THE HARDWARE PLATFORM

We use the Aldebaran Nao robot, (Fig. 1), which is a 4.5 kg, 58 cm tall humanoid robot with 21 degrees of freedom (www.aldebaran-robotics.com). The Nao has an on-board 500 MHz processor, to be shared between the low level control system and the autonomous perception, cognition, and motion algorithms. It is equipped with a variety of sensors including two color cameras, two ultrasound distance sensors, a 3-axis accelerometer, a 2-axis, gyroscope (X-Y), an inertial measurement unit for computing the absolute orientation of the torso, 4 pressure sensors on the sole of each foot, and a bump sensor at the tiptoe of each foot.

Fig. 1. The Aldebaran Nao robot.

Since 2008, the Nao has been used as the robot for the RoboCup Standard Platform League (SPL) (www.robocup.org), in which teams of autonomous humanoid Nao robots play robot soccer autonomously in teams of 3 robots each on a 6 meters by 4 meters green carpeted field (www.tzi.de/spl).

Technical challenges are a part of the RoboCup SPL competitions with the aim of fostering research on complex soccer playing skills. In 2010, there is a dribbling challenge, where a robot needs to score a goal in up to three minutes on a field where three stationary robots are placed in a way that a direct shot is not available from the starting position. The humanoid has to manipulate the ball placed at the middle of its own half in such a way to move the ball into the goal. During dribbling, if the ball or the robot touches one of the stationary robots, the ball is placed back to the starting point. We use the dribbling challenge as our evaluation domain.

III. THE HAND-CODED ALGORITHM

The Nao humanoid robots perceive their environment via their sensors, namely the two color cameras, the ultrasound distance sensors, and the accelerometer. The objects in the game (i.e., the field, the goals, the ball, and the robots) are all color coded to facilitate object recognition. However, perception of the environment remains the most challenging problem due to the extremely limited field of view (FoV) of the camera of the humanoid robot ($\approx 58^o$ diagonal), the sensitivity of the camera to changes in light characteristics like the temperature and luminance levels, and the inevitably limited on-board processing power that prevents the use of intensive and sophisticated vision approaches.

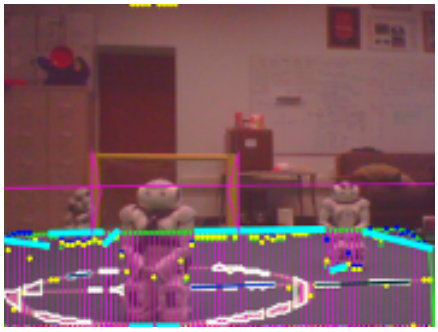
A. Free Space Detection

We model the free space in front of the robot to decide where to dribble the ball over. The soccer field is a green carpet, with white field lines. The robots are also white and gray, and they wear pink or blue waist bands as uniforms so any non-green thing lying on the field is an obstacle, except for the field lines which are white. We utilize a simplified version of the *Visual Sonar* algorithm by Lenser and Veloso [10] and the algorithm by Hoffmann *et al.* [11]. We scan the image along evenly spaced vertical lines starting from the bottom end and continue until we see a certain number of non-green pixels. If we do not encounter any green pixels along a scanline, we consider that scanline as fully occupied. Otherwise, the point where the non-green block started is considered the end of the free space towards that direction. To save computation time, we do not process every vertical line in the image. Instead, we skip every 4 pixels and process the line along the fifth pixel and we process every other pixel along a line. As a result, we effectively process only $1/10^{th}$ of the image (Fig. 2(a)). We project the pixel on the scanline denoting the end of the green onto the ground to have a rough estimate of the distance of that obstacle towards the direction of the scanned line. The robot does a horizontal scan, covering the entire 180^o space in front of the it and we combine the computed free space end points along the lines computed during the scan into 15 slots, each covering 12^o. In addition, we tag each slot with a flag indicating whether that slot points toward the opponent goal or not (Fig. 2(c)).

B. Dribbling Behavior

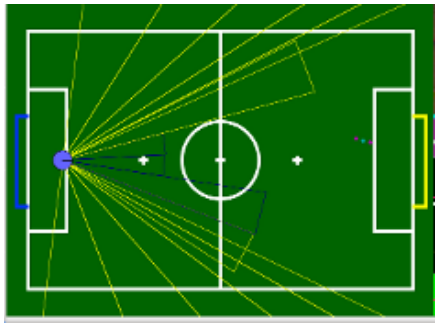
We developed a base controller implemented as a finite state machine (FSM). The robot looks for the ball, approaches the ball once sees it, selects an action, and finally kicks the ball towards a target point computed according to the selected action. If it loses the ball at any stage, it goes back to the search state to look for the ball again (Fig. 3).

We use existing low level skills without any change, namely, looking for the ball, approaching to the ball, lining up for a kick, and kicking the ball to a specified point with respect to the robot by selecting an appropriate kick from the portfolio of available kicks.



(a) Original image with perceived free space information

(b) Color segmented image



(c) Resulting free space model. Dark triangles indicate the slots pointing towards the goal

Fig. 2. The environment as perceived by the robot.

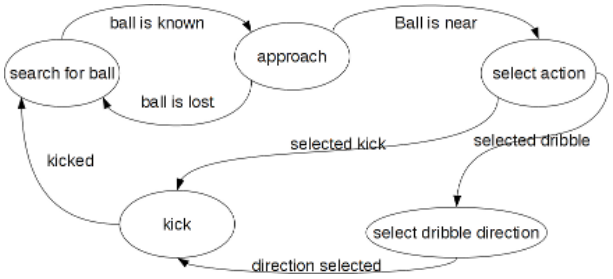


Fig. 3. The state diagram of the base system.

C. Action and Dribble Direction Selection

The "Select action" and the "Select dribble direction" states constitute the main decision points of the system we aim to improve. Both of the algorithms utilize the model of the free space in front of the robot detected using the color camera. Action selection tries to decide either to take a shoot or to dribble to the ball. Action selection algorithm checks if any of the slots pointing towards the opponent goal has a distance less than a certain fraction of the distance to the goal. If so, the path to the opponent goal is considered *"occupied"* and *dribble* is selected as the action. Otherwise, the path is considered *"clear"* and *shoot* is selected as the action with the center of the opponent goal being set as the kick target. The pseudo-code of the action selection algorithm is given in Alg. 1.

Algorithm 1 Action selection algorithm. $\Gamma \in [0, 1]$ is a coefficient for specifying the maximum distance to be considered as free space in terms of goal distance. In our implementation, we use $\Gamma = 0.5$.

$goalDist \leftarrow getGoalDist()$
$goalAngle \leftarrow getGoalAngle()$
if $goalAngle < -\frac{\pi}{2}$ **or** $goalAngle > \frac{\pi}{2}$ **then**
  return *dribble*
else
  for all $i \in getGoalSlots()$ **do**
    $distDiff \leftarrow |goalDist - dist_i|$
    if $distDiff > \Gamma goalDist$ **then**
      return *dribble*
    end if
  end for
end if
return *shoot*

If the action selection algorithm deduces that the path to the opponent goal is blocked, and subsequently selects the dribbling action, a second algorithm comes in to determine the best way to dribble the ball over. All slots are examined and the slot with the maximum weighted distance is selected as the dribble slot. The weighted distance is calculated as a weighted sum of the slot distance and the distances of its left and right neighbor slots. The algorithm for dribble direction selection is given in Alg. 2.

IV. CORRECTIVE DEMONSTRATION

Argall *et al.* defines the learning from demonstration problem formally as follows. The world consists of states S, and A is the set of actions the robot can take. Transitions between states are defined with a probabilistic transition function $T(s'|s, a) : S \times A \times S \rightarrow [0, 1]$. The state is not fully observable, instead, the robot has access to an observed state Z with the mapping $M : S \rightarrow Z$. A policy $\pi : Z \rightarrow A$ is employed for selecting the next action based on the current observed state.

Corrective demonstration is a form of teacher demonstration focused on correcting an action the robot performed by proposing an alternative action to be executed in that state. The usual form of employing corrective demonstration is to either add the corrective demonstration example to the demonstration dataset, or replacing an example in the dataset with the corrective example, and then re-deriving the action

Algorithm 2 Dribble direction selection algorithm. N is the number of free space slots.

$goalAngle \leftarrow getGoalAngle()$
if $goalAngle < -\frac{\pi}{2}$ **or** $goalAngle > \frac{\pi}{2}$ **then**
  if $|angle_0 - goalAngle| < |angle_{N-1} - goalAngle|$ **then**
    $dribbleAngle \leftarrow angle_0$
  else
    $dribbleAngle \leftarrow angle_{N-1}$
  end if
else
  $maxDist \leftarrow 0$
  for $i \leftarrow 1; i < N-1; i \leftarrow i+1$ **do**
    $distance \leftarrow 0.25dist_{i-1} + 0.5dist_i + 0.25dist_{i+1}$
    if $distance > maxDist$ **then**
      $maxDist \leftarrow distance$
      $maxSlot \leftarrow i$
    end if
  end for
  $dribbleAngle \leftarrow angle_{maxSlot}$
end if
return $dribbleAngle$

policy using the updated demonstration dataset. This way of applying corrective demonstration requires either to have a demonstration dataset which is large enough to derive a generalized policy, or, in case of an existing algorithm for the task, to have an understanding and access to the underlying analytical model of the algorithm.

We define the observed state of the robot as

$$Z =< slotDist_0, ..., slotDist_{N-1}, goal_0, ..., goal_{N-1} >$$

where $slotDist_i$ is the distance to the nearest obstacle towards the slot i, and $goal_i \in \{true, false\}$ is a Boolean flag which has the value $true$ if the slot i intersects with the goal, and $false$ otherwise.

Since the robot is expected to move the ball into the opponent goal, the distribution of the free space with respect to the direction to the goal needs to be taken into account rather than the position of the robot on the field. Therefore, we calculate the sum of absolute differences of the free space slots using the slot pointing towards the center of the goal as origin, if the goal is in sight. If the goal is not within the 180^o area in front of the robot, we calculate the sum of absolute differences of the free space slots using the rightmost slot as the origin. The similarity value which is in the range $[0, 1]$ is then calculated as

$$similarity = e^{-K\,diff^2}$$

where K is a coefficient for shaping the similarity function, and $diff$ is the calculated sum of absolute differences of the slot distances. In our implementation, we selected $K = 5$. The algorithm for similarity calculation is given in Alg 3.

During the execution, when the robot reaches to the action selection or dribble direction selection states, it first searches its demonstration database and fetches the demonstration point with the highest similarity to the current state. If the similarity value is higher than a threshold value τ, the robot executes the demonstration action instead of the action computed by the hand-coded algorithm. In our implementation, we use $\tau = 0.9$

Algorithm 3 The algorithm for computing the similarity of two given states

$dist_{curr} \leftarrow getSlotDist(Z_{curr})$
$dist_{demo} \leftarrow getSlotDist(Z_{demo})$
$diff \leftarrow 0$
if $goalAngle < -\frac{\pi}{2}$ **or** $goalAngle > \frac{\pi}{2}$ **then**
  for $i \leftarrow 0; i < N; i \leftarrow i+1$ **do**
    $diff \leftarrow diff + |dist_{curr}(i) - dist_{demo}(i)|$
  end for
  $diff \leftarrow diff/N$
else
  $goalSlot_{curr} \leftarrow getGoalSlot(Z_{curr})$
  $goalSlot_{demo} \leftarrow getGoalSlot(Z_{demo})$
  $num \leftarrow 0$
  $s_1 \leftarrow goalSlot_{curr}, s_2 \leftarrow goalSlot_{demo}$
  while $s_1 < N$ **and** $s_2 < N$ **do**
    $diff \leftarrow diff + |dist_{curr}(s_1) - dist_{demo}(s_2)|$
    $num \leftarrow num + 1, s_1 \leftarrow s_1 + 1, s_2 \leftarrow s_2 + 1$
  end while
  $s_1 \leftarrow goalSlot_{curr}, s_2 \leftarrow goalSlot_{demo}$
  while $s_1 >= 0$ **and** $s_2 >= 0$ **do**
    $diff \leftarrow diff + |dist_{curr}(s_1) - dist_{demo}(s_2)|$
    $num \leftarrow num + 1, s_1 \leftarrow s_1 - 1, s_2 \leftarrow s_2 - 1$
  end while
  $diff \leftarrow diff/num$
end if
$similarity \leftarrow e^{-K\,diff^2}$
return $similarity$

V. EXPERIMENTAL RESULTS

We evaluated the efficiency of the corrective demonstration using three different robot placement cases. We defined the cases in a way that the robot using the hand-coded action selection algorithm can score a goal, but not following an optimal sequence of actions (Fig. 4). The cases are as follows:

- **Case 1:** We put two robots on the periphery of the center circle, leaving a narrow, but passable corridor. We place the last robot at the penalty point distance from the center point, and 1 meter to the left from the penalty point. The hand-coded behavior tries to avoid the two robots at the center, and mostly chooses a right dribbling direction to also avoid the third robot. During the demonstration, we advised the robot to take a direct shot between the two robots at the center (Fig. 5(a)).
- **Case 2:** This is a position where a direct shot is not possible, and the robots are placed asymmetrically on the field so dribbling the ball towards the direction where the robot placed further away gives an advantage.

(a)

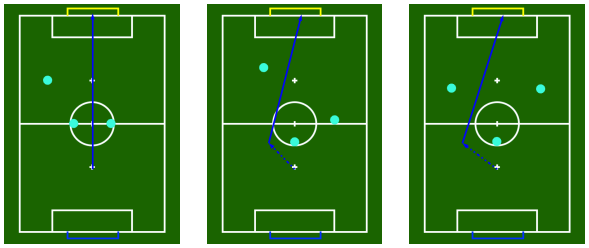
(b)

(c)

Fig. 4. Three different configurations used in the experiments. a) Case 1, b) Case 2, and c) Case 3.

During the demonstration, the given advice was first to dribble the ball to the left, and then to take a direct shot on the goal (Fig. 5(b)).

- **Case 3:** This is also a situation where a direct shot is not possible, and the robots are placed symmetrically so no clear advantage in choosing an initial dribbling direction over another exists. During the demonstration, we gave a very similar advice to the Case 2 advice to investigate whether we can create a bias towards a specific action in certain cases or not (Fig. 5(c)).

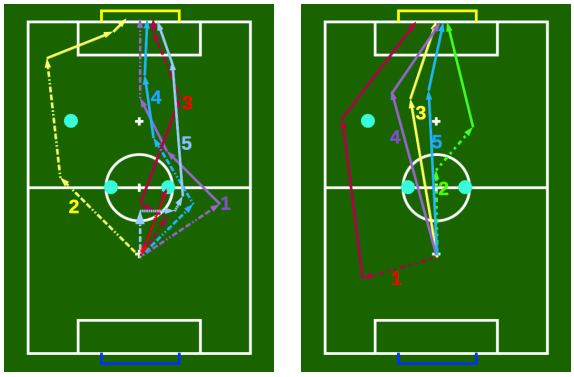


(a) Advised path for Case 1 (b) Advised path for Case 2 (c) Advised path for Case 3

Fig. 5. The advised paths for different cases.

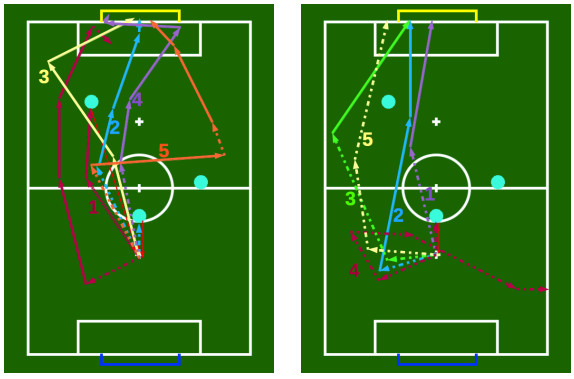
We gathered corrective demonstration data from all three cases, forming a common database. We collected a total of 42 action demonstration and 21 dribble direction demonstration points in a demonstration session of roughly 30 minutes. We then evaluated the performance of the system with and without the use of corrective demonstration database by means of the time required to score a goal.

We ran 10 trials for each case, 5 with the hand-coded action and dribble direction selection algorithm (HA), and another 5 trials with the corrective demonstration data (CD) on top of the HA. The sequence of actions taken by the robot at each trial are given in Fig. 6- 8, and the timing information given in Table I. In the figures, a dashed line indicates a dribble action, a solid line indicates a shoot action, and a thin line indicates the replacement of the ball to the initial position after committing a foul. In the table, 'out' means that the robot kicked the ball out of the field from sides, 'missed' means that the robot did the right actions but due to imperfect actuation, the kick missed the goal and went outside, and 'own goal' means that the robot accidentally kicked the ball into its own goal. The failed attempts are excluded in the given mean and standard deviation values. The failures were mostly due to the imperfection of the lower level skills like aligning with the ball, and the high variance in both the kick distance, and the kick direction.



(a) Case 1 HA (b) Case 1 HA+CD

Fig. 6. The followed paths in Case 1.



(a) Case 2 HA (b) Case 2 HA+CD

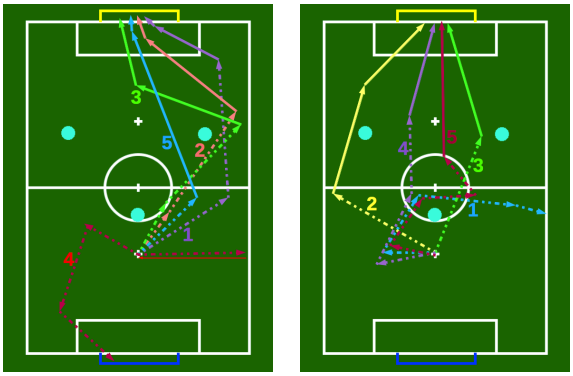
Fig. 7. The followed paths in Case 2.

The decrease in the timings in HA+CB case compared to the HA base case shows an improvement in the overall performance since according to the problem definition, the shorter completion times are considered more successful.

TABLE I

ELAPSED TIMES DURING TRIALS.

	Case 1		Case 2		Case 3	
Trial	**HA**	**HA+CD**	**HA**	**HA+CD**	**HA**	**HA+CD**
1	2:38	1:35	6:34 - out	1:42	2:10	1:57 - out
2	2:27	1:49	3:31	1:47	2:31	1:57
3	1:48	1:32	2:02	2:08	2:24	1:03
4	2:36	1:27	3:52	3:47 - out	3:45 - own goal	1:53
5	3:57	1:31	2:56	1:29 - missed	1:54	2:52
mean	**2:41**	**1:34**	**3:05**	**1:52**	**2:14**	**1:56**
std	**0.0326**	**0.006**	**0.033**	**0:009**	**0.011**	**0.031**



(a) Case 3 HA (b) Case 3 HA+CD

Fig. 8. The followed paths in Case 3.

In **Case 1**, the corrective demonstration created a bias towards taking direct shots as opposed to the dribbling action computed by the hand-coded algorithm and reduced the completion time. In **Case 3**, the corrective demonstration was again proven to be effective in creating bias in situations where it is not analytically possible to prefer an action over another.

VI. CONCLUSIONS

In this paper, we contributed a task and skill improvement method which utilizes corrective human demonstration on top of an existing algorithm. We applied the method on one of the technical challenge tasks of RoboCup 2010 Standard Platform League competitions called "Dribbling Challenge". We applied corrective demonstration at two levels: action selection and dribble direction selection. We developed a hand-coded base system for performing the task which can handle most basic cases but is unable to perform well on all the cases. A human demonstrator monitors the execution of the task by the robot and intervenes as needed to correct the output of the hand-coded algorithm. The robot saves the received demonstration examples in a database, and fetches the demonstration example received from the most similar state in the database using a domain specific similarity measure. If the similarity of the state of the received demonstration and the current state of the robot is above a certain threshold, the robot executes the fetched demonstrated action instead of the output of its own algorithm. We presented empirical results in which the proposed method is evaluated in three different field setups. The results show that it is possible to apply "patches" to the hand-coded algorithm which improves the task performance with only a small number of demonstrations.

Investigating the possibility to develop a domain-free similarity measure, applying the corrective demonstration to the other subspaces of the task space, and applying the proposed method to more sophisticated tasks can be listed among the possible future work we aim to address.

ACKNOWLEDGMENTS

The authors would like to thank the other members of the CMurfs RoboCup SPL team for their invaluable help.

REFERENCES

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Automation Systems*, 57(5):469–483, 2009.

[2] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79 – 91, 2004. Robot Learning from Demonstration.

[3] D.H. Grollman and O.C. Jenkins. Dogged learning for robots. In *International Conference on Robotics and Automation (ICRA 2007)*, pages 2483–2488, Rome, Italy, Apr 2007.

[4] B. Argall, B. Browning, and M. Veloso. Learning from demonstration with the critique of a human teacher. In *Second Annual Conference on Human-Robot Interactions (HRI'07)*, 2007.

[5] B. Argall, B. Browning, and M. Veloso. Learning robot motion control with demonstration and advice-operators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, 2008.

[6] Sonia Chernova and Manuela Veloso. Confidence-based demonstration selection for interactive robot learning. In *In Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI'08), Amsterdam, The Netherlands, March 2008.*, 2008.

[7] Sonia Chernova and Manuela Veloso. Teaching collaborative multi-robot 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*, 2008.

[8] Çetin Meriçli and Manuela Veloso. Biped walk learning through playback and corrective demonstration. In *AAAI 2010: Twenty-Fourth Conference on Artificial Intelligence*, 2010.

[9] Çetin Meriçli and Manuela Veloso. Improving biped walk stability using real-time corrective human feedback. In *RoboCup 2010: Robot Soccer World Cup XIV*, 2010.

[10] Scott Lenser and Manuela Veloso. Visual sonar: Fast obstacle avoidance using monocular vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2003*, pages 886–891, 2003.

[11] Jan Hoffmann, Matthias Jüngel, and Martin Lötzsch. A vision based system for goal-directed obstacle avoidance used in the rc'03 obstacle avoidance challenge. In *In 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence*, pages 418–425. Springer, 2004.