# Task Refinement for Autonomous Robots using Complementary Corrective Human Feedback

Çetin Meriçli, Manuela Veloso, and H. Levent Akın

***Abstract*— A robot can perform a given task through a policy that maps its sensed state to appropriate actions. We assume that a hand-coded controller can achieve such a mapping only for the basic cases of the task. Refining the controller becomes harder and gets more tedious and error prone as the complexity of the task increases. In this paper, we present a new learning from demonstration approach to improve the robot's performance through the use of corrective human feedback as a complement to an existing hand-coded algorithm. The human teacher observes the robot as it performs the task using the hand-coded algorithm and takes over the control to correct the behavior when the robot selects a wrong action to be executed. Corrections are captured as new state-action pairs and the default controller output is replaced by the demonstrated corrections during autonomous execution when the current state of the robot is decided to be similar to a previously corrected state in the correction database. The proposed approach is applied to a complex ball dribbling task performed against stationary defender robots in a robot soccer scenario, where physical Aldebaran Nao humanoid robots are used. The results of our experiments show an improvement in the robot's performance when the default hand-coded controller is augmented with corrective human demonstration.**

## I. INTRODUCTION

Transferring the knowledge of how to perform a certain task to a complex robotic platform remains a challenging problem in robotics research with an increasing importance as robots start emerging from research laboratories into everyday life and interacting with ordinary people who are not robotics experts. A widely adopted method for transferring task knowledge to a robot is to develop a controller using a model for performing the task or skill, if such a model is available. Although it is usually relatively easier to develop a controller that can handle trivial cases, handling more complex situations often requires substantial modifications on the controller. Due to interference among the newly added cases and the existing ones, it becomes a tedious and time consuming process to ameliorate the controller and the underlying model as the number of such complex cases increases. That brings out the need for a new approach to robot programming.

The first author is supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610

Ç. Meriçli is with the Department of Computer Engineering Boğaziçi University Bebek, Istanbul 34342, Turkey, and the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA 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

Learning from Demonstration (LfD) paradigm is one such approach that utilizes supervised learning for transferring task or skill knowledge to an autonomous robot without explicitly programming it. Instead of hand-coding a controller for performing a task or skill, LfD methods make use of a teacher who demonstrates the robot how to perform the task or skill while the robot observes the demonstrations and synchronously records the demonstrated actions along with the perceived state of the system. The robot then uses the stored 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 for defining a proper reward function, which is considered to be a difficult problem [1]. Moreover, since the LfD approaches do not require the robot to be programmed explicitly, they are very suitable for cases where the task knowledge is available through a user who is an expert in the task domain but not in robotics.

Providing a way for humans to transfer task and skill knowledge to robots via natural interactions, the LfD approaches are also suitable for problems, where 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 number of examples is a very time consuming process when working with robots with highly complex body configurations; such as humanoids, and for sophisticated tasks with very high dimensional state and action spaces.

LfD based methods have been applied to many learning scenarios involving high level task and low level skill learning on different robotic platforms varying from wheeled and legged robots to autonomous helicopters. Here we present a few representative studies and strongly encourage the reader to resort to [1] for a comprehensive survey on LfD.

While learning to perform high level tasks, it is a common practice to assume that the low level skills required to perform the task are available to the robot. Task learning from demonstration have been studied in many different contexts; such as, (from reinforcement learning point of view) learning how to bake a cake [2], (from an active learning point of view) learning concepts [3], and (from the sliding autonomy point of view) learning of general behavior policies from demonstration for a single robot [4], [5], [6], [7] and multi-robot systems [8], [9] via the "Confidence Based Autonomy (CBA)" approach.

Several approaches to low level skill learning in the literature utilize LfD methods with different foci. Tactile

interaction has been utilized for skill acquisition through kinesthetic teaching [10] and skill refinement through tactile correction [11], [12], [13], [14], [15]. Motion primitives have been used for learning biped walking from human demonstrated joint trajectories [16] and learning to play air hockey [17].

Several regression based approaches have been proposed for learning quadruped walking on a Sony AIBO robot and learning low level skills for playing soccer [18], [19], learning several skills with different characteristics (cyclic, skills with multiple constraints, etc.) using a probabilistic approach that utilizes Hidden Markov Models (HMM) along with regression [20], and learning non-linear multivariate motion dynamics [21].

Interacting with the learner using high level abstract methods has been introduced in forms of natural language [22], [23] and advice operators as functional transformations for low level robot motion, demonstrated on a Segway RMP robot [24], [25]. Reinforcement learning methods have been investigated in conjunction with the LfD paradigm for teaching a flying robot how to perform a complex skill [26], learning to swing up a pole and keep it balanced [27], [28], learning constrained reaching tasks [29], and hierarchical learning of quadrupedal locomotion on rough terrain [30].

In our previous work on complementary skill refinement, we used real-time corrective human demonstration to improve the biped walk stability of a Nao humanoid robot [31], [32]. An existing walk algorithm was used to capture a complete walk cycle, and the captured walk cycle was played back to obtain a computationally cheap open-loop walking behavior. A human demonstrator monitored the robot as it walked using the open-loop controller and modified the joint commands in real-time via a wireless game controller to keep the robot stable. The recorded demonstration values together with the corresponding sensor readings were used to derive a policy for computing proper joint command correction values for a given sensory reading to recover the balance of the robot.

In this paper, we present a corrective demonstration approach for task execution refinement where a hand-coded algorithm for performing the task exists but is inadequate in handling complex cases. The human demonstrator observes the robot carry out the task by executing the hand-coded algorithm and provides corrective feedback when the hand-coded controller computes a wrong action. The received demonstration actions are stored along with the state of the robot at the time of correction as complements (or "patches") to the base hand-coded algorithm. During autonomous execution, the robot substitutes the action computed by the hand-coded algorithm with the demonstrated action if the corrective demonstration history database contains a demonstration provided in a similar state. The key idea is to keep the base controller algorithm as the primary source of the action policy, and use the demonstration data as exceptions only when needed instead of deriving the entire policy out of the demonstrations and the output of the controller algorithm. We applied this approach to a complex ball dribbling task in humanoid robot soccer domain. Experiment results show considerable performance improvement when the hand-coded algorithm is complemented by corrective human feedback. Since the human teacher provides correction only when the robot performs an erroneous action, the number of corrective demonstration examples needed to improve the task performance is smaller compared to other LfD approaches in the literature.

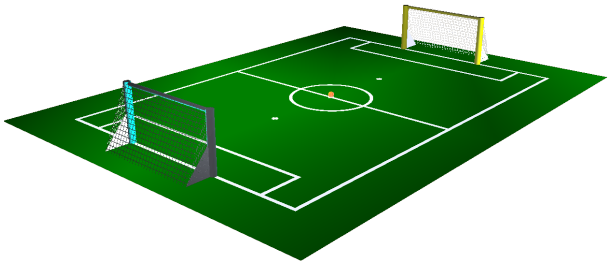
The rest of the paper is organized as follows: In Section II, we describe the robot soccer domain and the hardware platform used in this study, followed by a brief overview of the software infrastructure used in our humanoid robot soccer system. In Section III, we first give the problem definition for the ball dribbling task, which is our application domain, and then we present a thorough explanation of the special image processing system for free space detection, the ball dribbling behavior developed using the available low level skills and the software infrastructure described in Section II, and a hand-coded algorithm for action selection parts of the ball dribbling behavior. Section IV contains the explanation of the corrective demonstration setup for delivering the demonstration to the robot and a domain-specific correction reuse system used for deciding when to apply a correction based on the similarity of the current state of the system to the states in the correction database. We present our experimental study in Section V with results showing a considerable improvement in the task completion time using corrective demonstration as a complement to the original hand-coded algorithm over using the original hand-coded algorithm alone. Pointing out some future directions to be further explored, we conclude the paper in Section VI.

## II. BACKGROUND

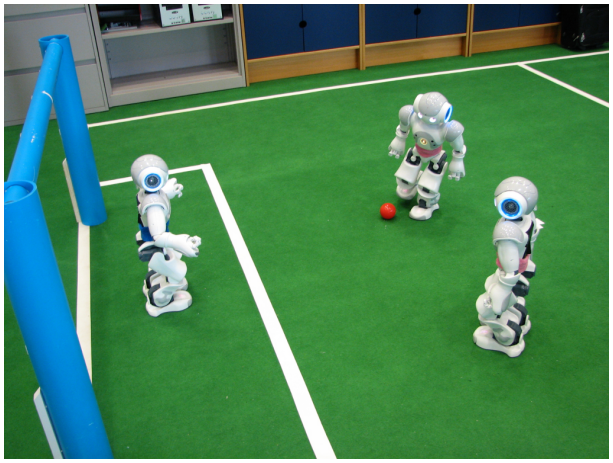
### A. Robot Soccer Domain

RoboCup is an international research initiative that aims to foster research in the fields of artificial intelligence and robotics by providing standard problems to be tackled from different points of view; such as, software development, hardware design, and systems integration [33]. Soccer was selected by the RoboCup Federation as the primary standard problem due to its inherently complex and dynamic nature, allowing scientists to conduct research on many different sub-problems ranging from multi-robot task allocation to image processing, and from biped walking to self-localization. With its various categories focusing on different challenges in the soccer domain; such as, playing soccer in simulated environments (the 2D and 3D Simulation Leagues) and physical environments using wheeled platforms (the Small Size League and the Middle Size League), humanoid robots of different sizes and capabilities (the Humanoid League), and a standard hardware platform (the Standard Platform League), the ultimate goal of RoboCup is to develop, by 2050, a team of 11 fully autonomous humanoid robots that can beat the human world champion soccer team in a game that will be played on a regular soccer field complying with the official FIFA rules.

In the Standard Platform League (SPL) of RoboCup [34], teams of 3 autonomous humanoid robots play soccer on a 6 meters by 4 meters green carpeted field (Figure 1(a)). The league started in 1998 as an embodied software competition with a common and standard hardware platform, hence the name. Sony AIBO robot dogs had been used as the standard robot platform of the league until 2008, and the Aldebaran Nao humanoid robot was decided to be the new standard platform thereafter. A snapshot showing the Nao robots playing soccer is given in Figure 1(b).



(a)



(b)

Fig. 1. a) The field setup for the RoboCup Standard Platform League (SPL), and b) a snapshot from an SPL game showing the Nao robots playing soccer.

### B. Hardware Platform

The Aldebaran Nao robot (Figure 2) which is the standard hardware platform for the RoboCup SPL competitions, is a 4.5 kg, 58 cm tall humanoid robot with 21 degrees of freedom[1]. 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-axisaccelerometer, 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.

[1]More information can be found on http://www.aldebaran-robotics.com.

Fig. 2. The Aldebaran Nao humanoid robot.

The Nao runs a Linux-based operating system and has a software framework named *NaoQi*, which allows users to develop their own controller software and access the sensors and actuators of the robot. The internal controller software of the robot runs at 100Hz, making it is possible to read new sensor values and send actuator commands every 10ms.

### C. Software Overview

Being able to play soccer requires several complex software modules (i.e., image processing, self localization, motion generation, planning, communication, etc.) to be designed, implemented, and seamlessly integrated with each other. In this section of the paper, we present a brief overview of the software infrastructure developed for the RoboCup SPL competitions and also used in this study.

*1) Image Processing:* The Nao humanoid robots perceive their environment via their sensors, namely the two color cameras, the ultrasound distance sensors, the gyroscope, and the accelerometer. All the important objects in the game environment (i.e., the field, the goals, the ball, and the robots) are color coded to facilitate object recognition. However, perception of the environment remains the most challenging problem primarily due to the extremely limited on-board processing power that prevents the use of intensive and sophisticated computer vision algorithms. The very narrow fields of view (FoV) of the robot's cameras ($\approx 58^o$ diagonal) and their sensitivity to changes in light characteristics like the temperature and luminance levels are among the other contributing factors to the perception problem.

The job of the image processing module is to extract the relative distances and bearings of the objects detected in the camera image. In addition to the position information, the image processing module also reports confidence scores indicating the likelihood of those objects being actually present in the camera image.

*2) Self Localization and World Modeling:* These modules are responsible for determining the location of the robot as well as the locations of the other important objects (e.g. the ball) on the field. Our system uses a variation of Monte Carlo Localization (MCL) called Sensor Resetting Localization [35] for estimating the position of the robot on the field.

For calculating and tracking the global positions of the other objects, we employ a modeling approach which treats objects based on their individual motion models defined in terms of their dynamics [36], [37].

*3) Planning and Behavior Control:* Our planning and behavior generation module is built using a hierarchical Finite State Machine (FSM) based multi-robot control formalism called Skills, Tactics, and Plays (STP) [38]. Plays are multi-robot formations where each robot is executing a tactic consisting of several skills. Skills can be stand-alone or formed via a hierarchical combination of other skills.

*4) Motion Generation:* The motion generation module is responsible for all types of movement on the field including biped walking, ball manipulation (e.g., kicking), and some other motions such as getting back upright after a fall. For the biped walking, we use the omni-directional walk algorithm provided by Aldebaran. For kicking the ball and the other motions, we use predefined actions in the form of sequences of keyframes, each of which define a vector of joint angles and a duration value for the interpolation between the previous pose and the current one. Two variations (strong and weak) of three types of kick (side kick to the left, side kick to the right, and forward kick) are implemented to be used in the games.

## III. PROPOSED APPROACH

### A. Problem Definition

Technical challenges are held as a complementary part of the RoboCup SPL competitions with the aim of creating a research incentive on complex soccer playing skills that will help leverage the quality of the games and enable the league to gradually approach the level of real soccer games both in terms of the field setup and the game rules. Each year, the technical challenges are determined accordingly by the Technical Committee of RoboCup SPL.

Our application and evaluation domain, the "Dribbling Challenge", was one of the three technical challenges of the 2010 SPL competitions. In that challenge, an attacker robot is expected to score a goal in three minutes without having itself or the ball touching any of the three stationary defender robots that are placed on the field in such a way to block the direct shot paths. The positions of the obstacle robots are not known beforehand; therefore, the robot has to detect the opponent robots, model the free space on the field, and plan its actions accordingly. An example scenario is illustrated in Figure 3.

### B. Free Space Detection using Vision

Instead of trying to detect the defender robots and avoid them, our attacker robot detects the free space in front of it and builds a free space model of its surroundings to decide which direction is the best to dribble the ball towards. The soccer field is a green carpet with white field lines on it. The robots are also white and gray, and they wear pink or blue waist bands as uniforms (Figure 1(b), Figure 3). Therefore, anything that is non-green and lying on the field can be considered as an obstacle, except for the detected

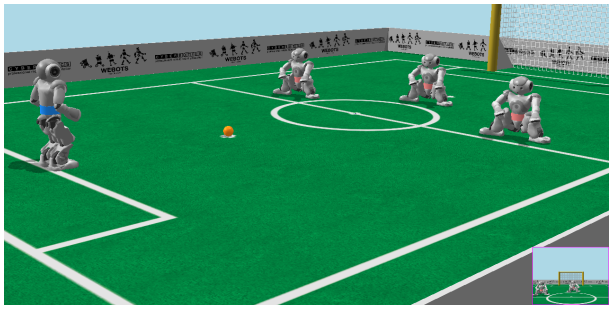


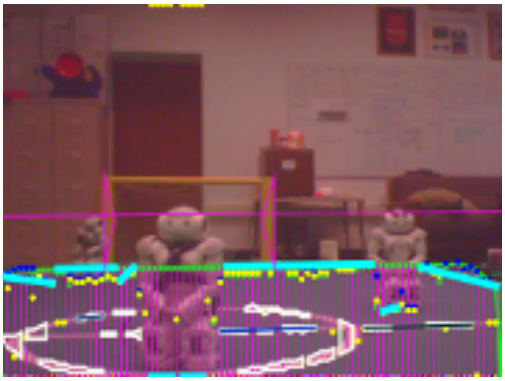
Fig. 3. An example scenario for the dribbling challenge.

field lines. We utilize a simplified version of the *Visual Sonar* algorithm by Lenser and Veloso [39] and the algorithm by Hoffmann *et al.* [40]. We scan the pixels on the image along evenly spaced vertical lines called *scanlines*, starting from the bottom end and continue until we see a certain number of non-green pixels. Although the exact distance function is determined by the position of the camera, in general the projected relative distance of a pixel increases as we ascend from the bottom of the image to the top, assuming all the pixels lie on the ground plane. 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 starts is marked as the end of the free space towards that direction. To further save some computation time, we do not process every vertical line on the image. Instead, we process the lines along every fifth pixel and every other pixel along those lines. As a result, we effectively process only $1/10^{th}$ of the image(Figure 4(b)). The pixels denoting the end of the free space are then projected onto the ground to have a rough estimate of the distance of the corresponding obstacle in the direction of the scanned line. In order to cover the entire $180^o$ space in front of it, the robot pans its head from side to side. As the head moves, the computed free space end points are combined and divided into 15 slots, each covering $12^o$ in front of the robot. In the mean time, each free space slot is tagged with a flag indicating whether that slot points towards the opponent goal or not based on the location of the opponent goal in the world model, or the estimated location and orientation of the robot on the field (Figure 4(c)).

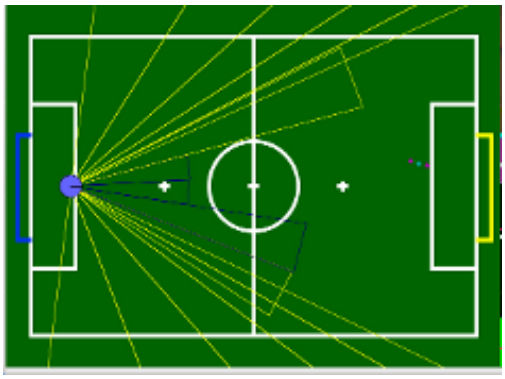
### C. Ball Dribbling Behavior

We used the Finite State Machine (FSM) based behavior system explained in Section II-C.3 for developing the ball dribbling behavior. The FSM structure of the ball dribbling behavior is depicted in Figure 5. The robot starts with "searching for the ball" by panning its head from side to side several times using both cameras. If it cannot find the ball at the end of this initial scan, it starts turning in place while tilting its head up and down, and this cycle continues until the ball is detected. Once the ball is located on the field, "approach the ball" behavior gets activated and the robot starts walking towards the ball. Utilizing the omni-directional walk, it is guaranteed that the robot faces the

(a)



(b)



(c)

Fig. 4. The environment as perceived by the robot: a) the color segmented image showing the potential obstacle points marked as red dots, b) the computed perceived free space segments illustrated with the cyan lines overlaid on the original image, and c) the free space model built out of the perceived free space segments extracted from a set of images taken while the robot pans its head. Here, the dark triangles indicate the free space slots pointing towards the opponent goal.

ball when the "approach the ball" behavior is executed and completed successfully. After reaching the ball, the robot pans its head one more time to gather information about the free space around it, calculates its current state, selects an action that matches its state, and finally kicks the ball towards a target point computed according to the selected action. If the robot loses the ball at any instant of this process, it goes back to the "search for the ball" state.

Except for the lightly colored *select action* and *select dribble direction* states shown on the state diagram, each state in the given FSM corresponds to a low level skill. We use the existing low level skills in our robot soccer system without any modifications; namely, looking for the ball, approaching the ball, lining up for a kick, and kicking the ball to a specified point relative to the robot by selecting an appropriate kick from the portfolio of available kicks.

### D. 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 using corrective demonstrations. The hand-coded algorithms for both the action and the dribble direction selection parts utilize the free space model in front of the robot. After lining up with the ball properly, the robot selects one of the following two actions:

- Shoot
- Dribble

The *shoot* action corresponds to kicking the ball directly towards the opponent goal using a powerful and long range kick. The *dribble* action, on the other hand, corresponds to dribbling the ball towards a more convenient location on the field using a weaker and shorter range kick. When the robot reaches the decision point; that is, after it aligns itself with the ball and scans the environment for free space modeling, the action selection algorithm checks if any of the free space 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 the *dribble* action is selected in that situation. Otherwise, the path is considered *"clear"* and the *shoot* action targeting the center of the opponent goal is selected. The pseudo-code of the action selection algorithm is given in Algorithm 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 the distance of the goal. In our implementation, we use $\Gamma = 0.5$.

```
goalDist ← getGoalDist()
goalAngle ← getGoalAngle()
if goalAngle < −π/2 or goalAngle > π/2 then
  return dribble
else
  for all i ∈ getGoalSlots() do
    distDiff ← |goalDist − dist_i|
    if distDiff > Γ 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 steps in to determine the

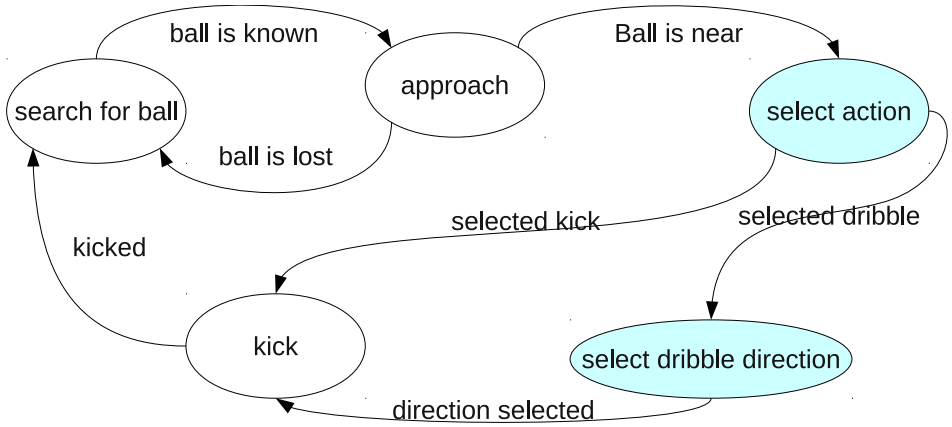



Fig. 5. The state diagram of the base system.

best way to dribble the ball. All slots in the free space model are examined and assigned a score computed as the weighted sum of the distance values of the slot itself and its left and right neighbors. The free space slot with the maximum score is selected as the dribble direction. The algorithm for dribble direction selection is given in Algorithm 2.

**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** $dribbleAngle$

Using the two algorithms explained above for the two action selection states in the behavior FSM, the robot is able to perform the ball dribbling task and score a goal with limited success. We define the success metric for this task to be the time it takes for the robot to score a goal. The performance evaluation results for the hand-coded action selection algorithms are provided in Section V. In the following section, we present the corrective demonstration system developed as a complement to the hand-coded action selection algorithms for refining the task performance.

## IV. CORRECTIVE DEMONSTRATION

Argall *et al.* [1] formally define the learning from demonstration problem 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 focusing on correcting an action selected by the robot to be performed by proposing

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

However, re-deriving the execution policy each time a correction is received can be cumbersome if the total number of state-action pairs in the demonstration database is large. On the other hand, accumulating a number of corrective demonstration points and then re-deriving the execution policy may be misleading or inefficient since the demonstrator will not be able to see the effect of the provided corrective feedback immediately.

In our approach, we store the collected corrective demonstration points separate from the hand-coded controller, and utilize a reuse algorithm to decide when to use correction. In the following subsections, we first describe how the corrective demonstration is delivered to the robot, and then we explain how the stored corrections are used during autonomous execution.

### A. Delivering Corrective Demonstration

During the demonstration session, the robot is set to a semi-autonomous mode in which it executes the dribbling behavior with the hand-coded action selection algorithms and performs the dribbling task. Each time an action selection state is reached, the robot computes an action, announces the selected action, and asks the teacher for the feedback. The teacher can give one of the following commands as the feedback:

1) Perform the selected action
2) Revise the action

If (1) is selected, the robot continues with the action computed by the hand-coded action selection system. If (2) is selected, the robot provides the available actions and asks the teacher to select one. If the robot is in the dribble direction selection state, it first selects a free space slot according to the hand-coded dribble direction selection algorithm. If the teacher wants to change the selected dribble slot, the robot provides the directions that can be chosen and waits for the teacher to select one. In both revision cases, the action provided by the teacher is paired with the current state of the system and stored in a database. No records are stored in the database when the teacher approves the robot's selection as the next action to be executed.

### B. Reusing the Corrections

By the end of the demonstration session, the robot has built a demonstration database of state-action pairs denoting what action is provided by the teacher as a replacement of the action computed by the hand-coded algorithms and what was the robot's state when that correction is received. During autonomous execution, the decision of when to execute the action selected by the hand-coded algorithms and when to use corrective demonstration samples is made by a correction reuse system based on the similarity of the current state of the robot to the states in which the demonstration samples were collected.

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 inside slot $i$, and $goal_i \in \{true, false\}$ is a Boolean flag which is set to $true$ if the slot $i$ intersects with the goal, and set to $false$ otherwise.

Since the robot is expected to kick/dribble the ball into the opponent goal, rather than the mere position of the robot on the field , the distribution of the free space with respect to the direction towards the goal needs to be taken into account. Therefore, we calculate the sum of the absolute differences of the free space slots using the slot pointing towards the center of the goal as the origin if the goal is in sight. If the goal is not somewhere within the $180^o$ 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 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 Algorithm 3.

**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** $similarity$

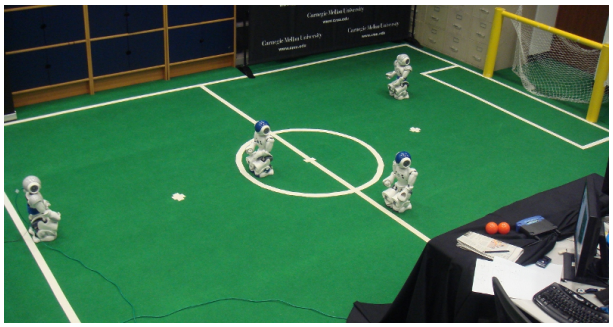
During execution, when the robot reaches the action selection or dribble direction selection states, it first checks its demonstration database and fetches the demonstration sample with the highest similarity to the current state. If the similarity value is higher than a threshold value $\tau$, the robot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

## V. RESULTS

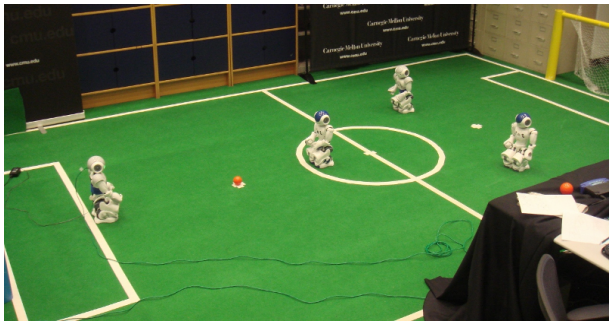
We evaluated the efficiency of the complementary corrective demonstration using three instances of the ball dribbling task with different opponent robot placements in each of them. The test cases were designed in such a way that the robot using the hand-coded action selection algorithm would be able to complete the task, but not through following an optimal sequence of actions (Figure 6). The following criteria were kept in mind while designing the test cases:

- **Case 1:** In this scenario, we place two robots on the periphery of the centercircle, leaving a narrow, but passable corridor. The third robot is placed on the virtual

(a)



(b)



(c)

Fig. 6. Three different configurations used in the experiments. a) Case 1, b) Case 2, and c) Case 3.

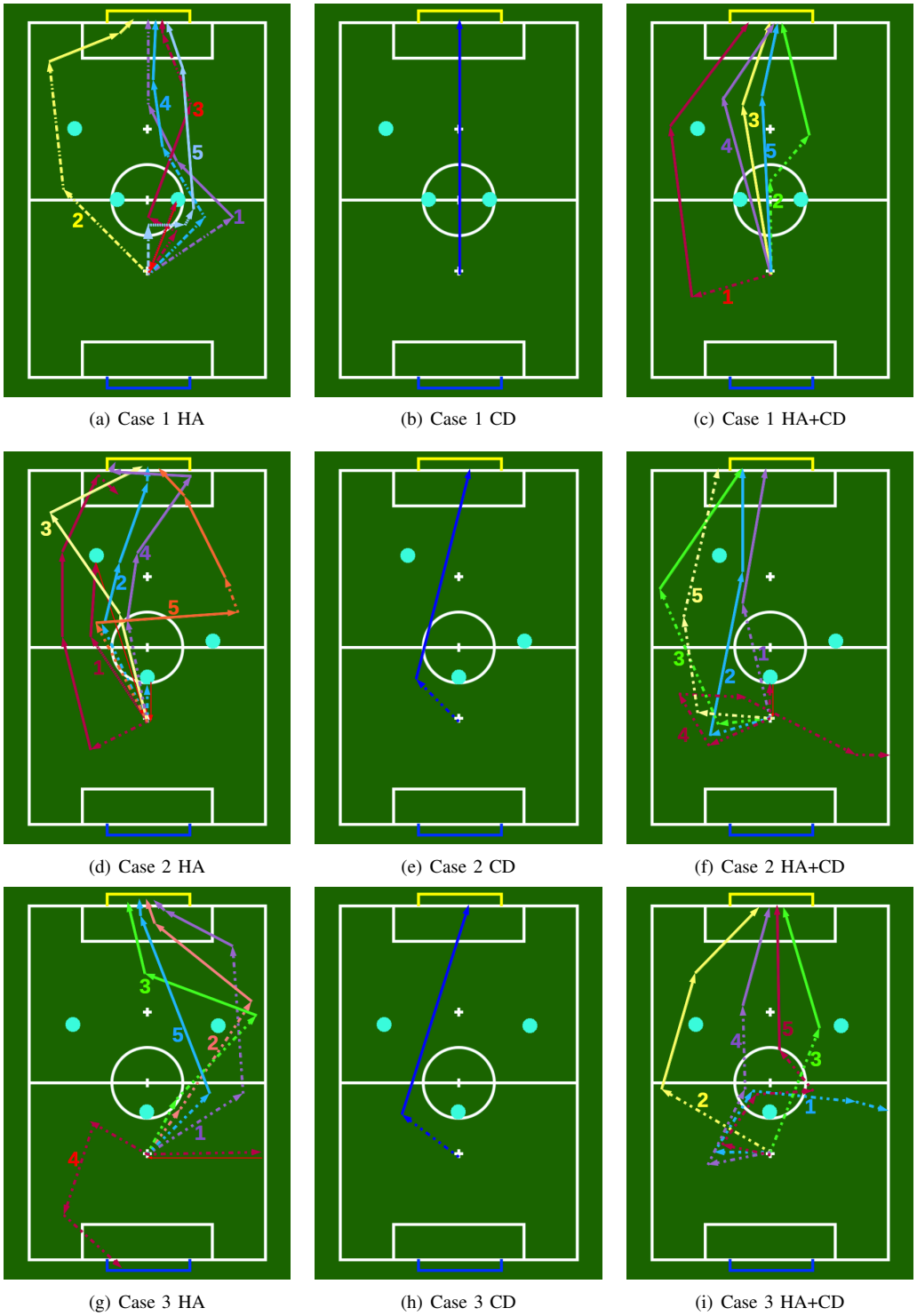
intersection of the opponent penalty mark and the left corner of the opponent penalty box. The hand-coded behavior computes the corridor between the two center robots to be too narrow to pass. Therefore, the robot tries to avoid the two robots at the center and mostly chooses a right dribbling direction to avoid the third robot as well. During the demonstration, we advised the robot to take a direct shot between the two robots at the center (Figure 7(b)). This scenario was a good showcase for illustrating how to refine the otherwise imprecise output of a very simple algorithm; no additional complexity were introduced to the algorithm and a limited number of demonstrations were provided only when the robot tried dribbling the ball whereas it could take a direct shot.

- **Case 2:** In this case, a direct shot is not possible from the initial position, and the robots are placed asymmetrically on the field in such a way that dribbling the ball towards the robot placed further away is advantageous. During the demonstration, the given advice was to first dribble the ball to the left, and then take a direct shot towards the goal (Figure 7(e)). The hand-coded algorithms tend to choose the right action by dribbling the ball to the left, but then the robot decides to advance the ball through a series of dribbles before kicking it into the goal instead of taking a direct shot.
- **Case 3:** This case was also designed to emphasize the ability of the proposed algorithm to reshape the behavioral response in addition to correcting mistakes. Similar to Case 2, a direct shot is not possible from the initial position, and the robots are placed symmetrically so no clear advantage of choosing one initial dribbling direction over another exists. During the demonstration, we gave a very similar advice to the one we gave in Case 2 to investigate whether we can create a bias towards a specific action in certain cases (Figure 7(h)).

We gathered corrective demonstration data from all three cases and formed a common database. A total of 42 action selection and 21 dribble direction selection demonstration points were collected in a roughly 30 minutes long demonstration session. Time required to score a goal being the success measurement metric, we then evaluated the performance of the system with and without the use of the corrective demonstration database.

We ran 10 trials for each case, 5 with the hand-coded action and dribble direction selection algorithms (HA), and another 5 trials with the corrective demonstration data (CD) in addition to the HA (HA+CD). The sequence of actions taken by the robot at each trial are depicted in Figure 7, and the timing information is presented in Table I. In the figures, a dashed line indicates 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 bounds from the sides, “missed” means that the robot chose the right actions but the ball did not roll into the goal due to imperfect actuation, and “own goal” means that the robot accidentally kicked the ball into its own goal. The failed attempts are excluded from 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.

The decrease in the timings in all three test cases when using (HA+CD) compared to the system using (HA) alone shows an improvement in the overall performance since according to the problem definition, the shorter completion times are considered more successful. In **Case 1**, where the average completion time is reduced by around one minute, the improvement in the task performance was mostly due to the bias created by the corrective demonstration which favors taking direct shots as opposed to the dribbling action computed by the hand-coded algorithm as given in Figure 7(c). In **Case 2**, the complementary corrective demonstration was able to correct the wrong decision made by the hand-coded algorithm on taking a second dribble action instead of a



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

(d) Case 2 HA (e) Case 2 CD (f) Case 2 HA+CD

(g) Case 3 HA (h) Case 3 CD (i) Case 3 HA+CD

Fig. 7. The illustrations of the performance evaluation runs. Different colors denote different runs. For each run, a dashed line represents a dribble and a solid line represents a kick. HA stands for hand-coded algorithm and CD stands for corrective demonstration. HA+CD shows the cases where the robot is in autonomous mode using both hand-coded algorithm and the corrective demonstration database.

direct shot after dribbling the ball to the left. As a result, the average task completion time was reduced almost to the half of the time it took on average when only (HA) was used. The effectiveness of corrective demonstration in Case 2 is presented in Figure 7(f). In **Case 3**, the corrective demonstration was again proven to be effective in creating a bias in situations where it is not analytically possible to prefer an action over another. Presenting a preference for dribbling to the left (Figure 7(h)), the corrective demonstration was able to change the initial response of the hand-coded algorithm

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** |

from dribbling the ball to the right (Figure 7(g)) to dribbling the ball to the left (Figure 7(i)).

## VI. CONCLUSIONS AND FUTURE WORK

In this paper, we contributed a task and skill improvement method which utilizes corrective human demonstration as a complement to an existing hand-coded algorithm for performing the task. We applied the method on one of the technical challenge tasks of the RoboCup 2010 Standard Platform League competitions called the "Dribbling Challenge". Corrective demonstrations were supplied at two levels: action selection and dribble direction selection. The hand-coded action selection algorithms for kick type and dribble direction selections can handle most of the basic cases; however, they are unable to perform well on all the defined test cases, as the test cases are designed in such a way to measure different aspects of the developed ball dribbling behavior. A human teacher monitors the execution of the task by the robot and intervenes as needed to correct the output of the hand-coded algorithm. During the advice session, the robot saves the received demonstration examples in a database, and during autonomous execution, it fetches the demonstration example received for the most similar state in the database using a domain specific similarity measure. If the similarity of those two states is above a certain threshold, the robot executes the fetched demonstration 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 improve the task performance by applying "patches" to the hand-coded algorithm through only a small number of demonstrations as well as "shaping" a hand-coded behavior by complementing it with corrective demonstrations instead of modifying the underlying algorithm itself.

Investigating the possibility of developing a domain-free state similarity measure, applying the corrective demonstration to the other sub-spaces of the task space, considering other demonstration retrieval mechanisms allowing better generalization over the state-action landscape, and applying the proposed method to more sophisticated tasks are among the future work that we aim to address.

## ACKNOWLEDGMENTS

The authors would like to thank the other members of the CMurfs RoboCup SPL team for their invaluable help, and to Tekin Meriçli for proofreading the manuscript.

## 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] Andrea L. Thomaz and Cynthia Breazeal. 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*, pages 1000–1005. AAAI Press, 2006.

[3] M. Cakmak, C. Chao, and A.L. Thomaz. Designing interactions for robot active learners. *Autonomous Mental Development, IEEE Transactions on*, 2(2):108 –118, jun. 2010.

[4] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *In Proceedings of AAMAS'07, the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, Honolulu, Hawaii, May 2007*, 2007.

[5] Sonia Chernova and Manuela Veloso. Learning equivalent action choices from demonstration. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08), Nice, France, September 2008*, 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. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34, 2009.

[8] Sonia Chernova and Manuela Veloso. Multiagent collaborative task learning through imitation. In *In Proceedings of the 4th International Symposium on Imitation in Animals and Artifacts, AIBS'07, Artificial and Ambient Intelligence, Newcastle, UK, April 2007*, 2007.

[9] 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.

[10] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467, 2008.

[11] B. Argall, E. Sauser, and A. Billard. 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*, 2010.

[12] B. Argall, E. Sauser, and A. Billard. Policy adaptation through tactile correction. In *In Proceedings of the 36th Annual Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB '10), Leicester, UK, March 2010*, 2010.

[13] B. Argall, E. Sauser, and A. Billard. Tactile correction and multiple training data sources for robot motion control. In *In NIPS 2009 Workshop on Learning from Multiple Sources with Application to Robotics, Whistler, British Columbia, December 2009*, 2010.

[14] Brenna Dee Argall, Eric Sauser, and Aude Billard. Tactile Guidance for Policy Adaptation. *Under submission*, 2010.

[15] S. Calinon and A. Billard. What is the Teacher's Role in Robot Programming by Demonstration? - toward Benchmarks for Improved Learning. *Interaction Studies. Special Issue on Psychological Benchmarks in Human-Robot Interaction*, 8(3), 2007.

[16] 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.

[17] Darrin C. Bentivegna, Christopher G. Atkeson, and Gordon Cheng. Learning similar tasks from observation and practice. In *in Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2677–2683, 2006.

[18] 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.

[19] D.H. Grollman and O.C. Jenkins. Learning elements of robot soccer from demonstration. In *International Conference on Development and Learning (ICDL 2007)*, pages 276–281, London, England, Jul 2007.

[20] Sylvain Calinon, Florent D'halluin, Eric Sauser, Darwin Caldwell, and Aude Billard. Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression. *IEEE Robotics and Automation Magazine*, 17(2):44–54, 2010.

[21] Elena Gribovskaya, Khansari Zadeh, Seyed Mohammad, and Aude Billard. Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators [accepted]. *International Journal of Robotics Research*, 2010.

[22] Cynthia Breazeal, Guy Hoffman, and Andrea Lockerd. Teaching and working with robots as a collaboration. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1030–1037, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Paul E. Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M. Veloso. 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*, pages 255–262, 2007.

[24] 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.

[25] 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.

[26] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *In Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[27] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning: (icml '97)*, pages 12–20. Morgan Kaufmann, 1997.

[28] C. G. Atkeson and S. Schaal. Learning tasks from a single demonstration. In *IEEE International Conference on Robotics and Automation (icra97)*, pages 1706–1712. Piscataway, NJ: IEEE, 1997.

[29] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement Learning for Imitating Constrained Reaching Movements. *RSJ Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544, 2007.

[30] J. Zico Kolter, Pieter Abbeel, and Andrew Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *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, 2007.

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

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

[33] RoboCup. RoboCup International Robot Soccer Competition, 2009. http://www.robocup.org.

[34] RoboCup SPL. The RoboCup Standard Platform League, 2009. http://www.tzi.de/spl.

[35] Scott Lenser and Manuela Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of ICRA-2000*, San Francisco, April 2000.

[36] Brian Coltin, Somchaya Liemhetcharat, etin Merili, and Manuela Veloso. Challenges of multi-robot world modelling in dynamic and adversarial domains. In *Workshop on Practical Cognitive Agents and Robots, 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 2010.

[37] Brian Coltin, Somchaya Liemhetcharat, Çetin Meriçli, Junyun Tay, and Manuela Veloso. Multi-humanoid world modeling in standard platform robot soccer. In *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots, December 6-8, 2010, Nashville, TN, USA*, 2010.

[38] Brett Browning, James Bruce, Michael Bowling, and Manuela Veloso. Stp: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering*, 219:33–52, 2005.

[39] 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.

[40] 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.

Çetin Meriçli is a Ph.D. candidate in the Department of Computer Engineering at Boğaziçi University, Turkey. His research interests include robot learning from demonstration, human-robot interaction, developmental robotics, multi-robot systems, robot soccer, and robot vision.

Manuela Veloso is the Herbert A. Simon Professor of Computer Science in the Computer Science Department at Carnegie Mellon University. She researches in artificial intelligence and robotics towards a vision of robots coexisting with humans in a seamless integration of intelligence. She directs the CORAL research laboratory, for the study of agents that Collaborate, Observe, Reason, Act, and Learn. Professor Veloso is a Fellow of the AAAI (Association for the Advancement of Artificial Intelligence), and the President of the RoboCup Federation. She received the 2009 ACM/Sigart Autonomous Agents Research Award. Professor Veloso is the author of one book on "Planning by Analogical Reasoning".

H. Levent Akın received his Ph.D. degree in Nuclear Engineering from Boğaziçi University, Turkey in 1984 and he has been a professor in the Department of Computer Engineering, Boğaziçi University since 1989. He is the director of Artificial Intelligence Lab and the founder of the Robotics Group. He is currently Dean of Faculty of Engineering of Boğaziçi University. He is an Exec of RoboCup Federation. He is the chair of IEEE Computational Intelligence Society Turkey Chapter. His research interests include artificial intelligence, autonomous robots, and computational intelligence and he has published more than 100 papers on these topics.