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

Multi-Resolution Corrective Demonstration for Efficient Task Execution and Refinement

Çetin Meriçli · Manuela Veloso · H. Levent Akın

Received: date / Accepted: date

Abstract Computationally efficient task execution is very important for autonomous mobile robots endowed with limited on-board computational resources. Most robot control approaches assume a fixed state and action representation, and use a single algorithm to map states to actions. However, not all situations in a given task require equally complex algorithms and equally detailed state and action representations. The main motivation for this work is a desire to reduce the computational footprint of performing a task by allowing the robot to run simpler algorithms whenever possible, and resort to a more complex algorithm only when needed. We contribute the Multi-Resolution Task Execution (MRTE) algorithm that utilizes human feedback to learn a mapping from a given state to an appropriate detail resolution consisting of a state and action representation, and an algorithm providing a mapping from states to actions at that resolution. The robot learns a policy from human demonstration to switch between different detail resolutions as needed while favoring lower detail resolutions to reduce computational cost of task execution. We then present the Model Plus Correction (M+C) algorithm to improve the performance of an algorithm using corrective human feedback without modifying the algorithm itself. Finally, we introduce the Multi-Resolution Model Plus Correction (MRM+C) algorithm as a combination of MRTE and M+C. MRM+C learns how to select an appropriate detail resolution to operate at in a given state from human demonstration. Furthermore, it allows the teacher to provide corrective demonstration at different detail resolutions to improve overall task execution performance. We provide formal definitions of MRTE, M+C, and MRM+C algorithms, and show how they relate to general robot control problem and Learning from Demonstration (LfD) approach. We present experimental results demonstrating the effectiveness of proposed methods on a goal-directed humanoid obstacle avoidance task.

Keywords Learning from Human Demonstration · Complementary Corrective Demonstration · Multi-Resolution Task Execution

Ç. Meriçli
Computer Science Department
Carnegie Mellon University
E-mail: cetin@cmu.edu

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

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

1 Introduction

Computational footprint of a robot controller is often overlooked as long as the controller is able to perform as expected on the robot. As robots become more ubiquitous and general-purpose, multiple software modules with different purposes are likely to run on the robot simultaneously. Despite the continuous advancements in the computational technology that enables the mobile robots to be equipped with more and more powerful computers, the on-board computational resources to be shared among these multiple software components are still limited.

Most robot control approaches consider a fixed state representation computed using the sensory input, an action representation to be executed on the robot, and an algorithm for executing the task at hand that maps a given state into an action. Employing a complex algorithm that uses the most detailed state representation and action definitions available might be computationally expensive and infeasible for continuous use. Although some instances of the task might be

handled using simpler algorithms operating on less detailed state representations and action definitions, a system that relies solely on such an algorithm might fail to capture the details in more complex situations, and that might eventually lead to a failure in the task execution.

We contribute the Multi-Resolution Task Execution algorithm (MRTE), a general framework that employs a set of detail resolutions where each resolution has its own state and action representations, and an algorithm using these representations to perform the task. A detail resolution selection policy is learned from human demonstration and used to determine which detail resolution to operate at in a particular state of the system. We then present Model Plus Correction (M+C), an algorithm based on our previous work for improving the performance of an existing algorithm by augmenting it with corrective human feedback [13, 14]. Finally, we combine MRTE with M+C into the Multi-Resolution Model Plus Correction (MRM+C) algorithm. Through multi-resolution corrective demonstration, MRM+C learns not only how to dynamically change the detail resolution at which to operate for a given state, but also how to improve the execution performances of individual algorithms for each detail resolution through multi-resolution corrective demonstration.

Over the course of a training session, a teacher observes the robot executing the task using hand-coded algorithms, and intervenes if the current algorithm needs a correction, or if the detail resolution in use is too coarse to cope with the current situation. The robot learns a detail switching policy for deciding which detail resolution to use in a particular state while also building up individual corrective demonstration databases for the algorithms at each detail resolution. During the autonomous execution of the task, the robot first chooses the most convenient detail resolution to run at, and then computes the action to be performed in the perceived state at the selected detail resolution.

2 Background and Related Work

We define the general robot control problem formally as a tuple $< Z, A, \pi >$. The world consists of states S, and A is the set of actions the robot can take. The state is not fully observable; instead, the robot has access to an observed state Z through the mapping $M : S \rightarrow Z$. The robot uses an execution policy $\pi : Z \rightarrow A$ for selecting the next action $a \in A$ based on the current observed state $z \in Z$.

2.1 Learning from Demonstration

Learning from Demonstration (LfD) is a supervised learning approach for transferring task or skill knowledge to an autonomous robot by means of the demonstrations of the task or skill execution [2]. The LfD methods make use of a teacher who demonstrates the robot how to perform the task or skill at hand while the robot records the demonstrated actions along with the perceived state of the system synchronously. The robot then derives an execution policy to reproduce the demonstrated task or skill using the stored state-action pairs. There are two main methods in which the demonstrations can occur:

- *Learning from observation:* In this category, the teacher performs the task or skill and the robot acquires the demonstration examples passively through observing the teacher. This type of demonstration requires the robot to be able to identify and map the teacher's actions to its own action set. This problem is also known as *the correspondence problem*.
- *Learning from experience:* In this category, the teacher makes the robot execute the task or skill by means of either manipulating the body parts of the robot or through instructing the robot using its own action set.

We define the LfD problem formally as an instance of general robot control problem using a tuple $< Z, A, \pi_{demo} >$. The execution policy $\pi_{demo} : Z \rightarrow A$ is extracted from a demonstration dataset D consisting of teacher demonstrations $d \in D$, where $d =< z, a_{demo} >, z \in Z, a_{demo} \in A$, and a_{demo} is the action demonstrated by the teacher in the observed state z. During execution, the robot uses π_{demo} for selecting the next action a based on the current observed state z. The execution model of generic learning from demonstration system is given in Fig. 1.

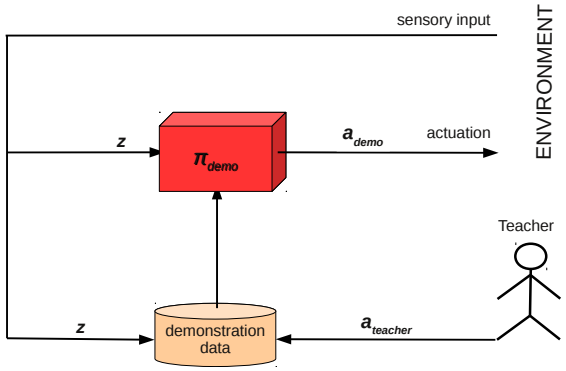


Fig. 1 The schematic representation of the generic LfD system.

Corrective demonstration is a form of teacher demonstration focusing on correcting an action of the robot by proposing one of the following types of feedback:

- An alternative action to be executed in that state
- 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. The action policy is then re-derived using the updated demonstration dataset.

However, re-deriving the execution policy each time a correction is received can be cumbersome if the total number demonstrations 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.

2.2 Related Work

LfD is a rapidly growing field in robotics research as it enables people who know how to perform a task but not how to program robots to transfer the task knowledge to the robot through demonstration. Being a very active field of research, LfD approach has been utilized in many different learning scenarios, focusing on different aspects of the learning. Here, we present a few representative studies and state how our approach relates to them.

Thomaz and Breazeal proposed a method for utilizing human feedback as the reward signal for the Reinforcement Learning (RL) system [16]. They used a simulated kitchen environment modeled as a Markov Decision Process (MDP) where a robot tries to learn how to bake a cake. The human teacher observes the robot operating, and provides a reward signal at any time without interrupting the operation. The notion of observing the robot executing the task and intervening to provide feedback bears a resemblance with our proposed approach. However, they utilize the feedback as a reward signal to an action selected by the robot whereas in our proposed approach, the teacher provides corrective feedback on robot's actions. The second main difference is that our proposed approach updates its existing task definitions to improve task execution performance while their approach utilizes the received feedback for training a RL system. They evaluated three different ways of making queries where each of the methods differ in the conditions of when to ask teacher for a demonstration using an upper-torso humanoid robot in a concept learning task. They presented a user study where they evaluate the performance of the different ways of asking for feedback against each other and against a baseline supervised learning method.

Chernova and Veloso introduced an approach for learning behavior policies from human demonstration called Confidence Based Autonomy (CBA) [4]. The CBA approach utilizes a confidence calculation mechanism for assessing how confident the robot is about the action selected by its execution policy. If the confidence value is above a certain threshold, the robot proceeds with the execution of the selected action. Otherwise, it asks for teacher demonstration. The system builds a statistical model of the received demonstration examples, and becomes more confident in situations where it has received a higher number of demonstrations. The CBA approach reduces the need for teacher attendance, hence it makes the teaching process less tedious and time consuming for the teacher. The main difference between the CBA approach and our proposed research is that instead of starting from scratch, our approach needs teacher feedback only when the existing algorithms fail to behave properly.

Simultaneous execution and feedback has been utilized for skill refinement through tactile correction. Argall *et al.* proposed a method for refining a demonstrated skill execution policy using kinesthetic feedback from the teacher during the execution of the skill using the execution policy extracted from the demonstration examples [3]. In the Tactile Policy Correction approach, if tactile feedback is detected, the policy is modified according to the received corrective tactile feedback. This approach shares a similarity with our proposed approach as both systems utilize provided human feedback interleaved with the task or skill execution.

Another method for learning low level skills from human demonstration through high level communication methods is the Advice Operator Policy Improvement (A-OPI) approach proposed by Argall *et al.* [1]. A set of defined verbal operators are associated with functional transformations for low level robot motion. The teacher provides feedback in the form of defined verbal operators and the corresponding transformations are applied on the specified portion of the demonstration database. A new execution policy is then re-derived out of the modified demonstration database. The A-OPI approach is evaluated on a trajectory learning task using a Segway RMP robot platform. Kolter *et al.* proposed a hierarchical apprenticeship learning approach for learning complex skills which are non-trivial even for the domain experts [9]. They propose a method that allows the teacher to provide advice at different hierarchical levels as providing isolated advice for a smaller part of the skill is often easier for the teacher. This approach shares similarities with our multi-resolution task and skill refinement approach since one of the two key advantages of our multi-resolution approach is the ability to cover a larger portion of the state-action space with demonstration provided at a low detail resolution. However, our approach differs from the hierarchical apprenticeship learning approach as it utilizes multiple algorithms with different computational complexities to handle different situations of the same task. [15] proposed a hierarchical LfD approach on humanoid robots. The learned behaviors are represented with a hierarchy of finite state machines, where different sub-parts of the task hierarchy can be addressed during the demonstration.

Grollman and Jenkins propose a learning from demonstration framework called "Dogged Learning" [7], and applied it to learning quadruped walking and a set of skills related to playing soccer on a Sony AIBO. The Dogged Learning algorithm share similarities with our approach on having multiple "boxes" generating output and an arbitrator for computing the final output but they use this approach for learning input-output associations whereas our approach utilizes human feedback to learn how to select the appropriate detail resolution, and to correct the underlying default algorithms.

Cobo *et al.* proposed a method for learning state abstractions from demonstration data [5]. Using two different algorithms, they select a subset of the originally available features that would yield least performance loss. Reducing the dimensionality of the state space improves the learning performance of the RL system they use. Automatically selecting a subset of available features for constructing a reward function for RL has also been investigated. Levine *et al.* proposed an algorithm that selects relevant features through building logical conjunctions of the features to the example policy [11]. Meriçli *et al.* introduced an automated method for learning reward function as a linear combination of available features [12]. They use Genetic Algorithms to learn appropriate feature combination and use the learning performance with the candidate reward function as the fitness value. Our approach differs from all these approaches as in all these approaches the underlying assumptions are i) a properly learned single state representation, and ii) a single algorithm using the learned state representation would suffice whereas we advocate that different instances of the same task can be handled using different algorithms and different state representations.

3 Approach

Our approach has two components: i) a multi-resolution task execution framework, and ii) a complementary corrective demonstration approach. In the remainder of the section, we first introduce our multi-resolution task execution framework. We then explain complementary corrective demonstration for augmenting the algorithms employed at each detail resolution with corrective human demonstration. Finally, we present the extended multi-resolution task execution and refinement approach that combines the multi-resolution algorithm execution and complementary corrective demonstration approaches.

3.1 Multi-Resolution Task Execution (MRTE)

We define Multi-Resolution Task Execution (MRTE) framework as a tuple:

$$< \pi_{arbitrator}, \{c_1, c_2, ..., c_N\} >$$

where c_r is the controller defined for the detail resolution $r \in R$, and $\pi_{arbitrator} : Z \rightarrow R$ is the detail resolution selection policy. A controller for the detail resolution r is defined as a tuple $c_r =< Z_r, A_r, f_r^{state}, f_r^{action}, \pi_{model}(r) >$, where $f_r^{state} : S \rightarrow S_r$ is the function for mapping the global state to the state definition at the detail resolution r, and $f_r^{action} : A_r \rightarrow A$ is the function for mapping the action computed at the detail resolution r into an action representation that can be executed by the robot. $\pi_{model}(r)$ is the task execution policy for each detail resolution r.

The detail resolution selection component acts as an arbitrator among the different detail resolutions, and decides which detail resolution to operate at for computing the next action given the current state. A human teacher provides demonstrations to teach the robot when to switch into a finer detail resolution. Unless stated otherwise, the system always runs at the most coarse detail resolution. In other words, the system assumes that the current situation can be handled by the simplest algorithm and state-action representations unless a feedback for increasing the detail resolution is received. The schematic representation of the MRTE approach is given in Fig. 2.

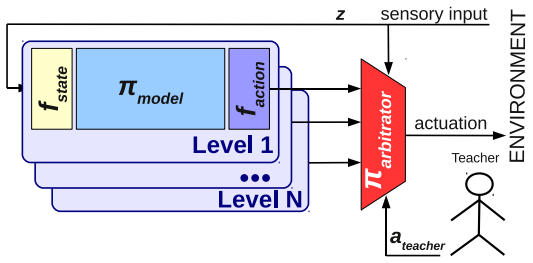


Fig. 2 The schematic representation of the MRTE approach.

3.2 Model Plus Correction (M+C)

We present *Model Plus Correction (M+C)* complementary corrective demonstration approach by extending the learning from demonstration model. Our approach makes use of an algorithm as the default controller, and utilizes corrective human demonstration to further improve the performance of the algorithm without changing the algorithm itself. We define the M+C system as a tuple:

$$< Z, A, \pi_{demo}, \pi_{model}, f_{reuse} >$$

The LfD definition is extended with a model-based algorithm $\pi_{model} : Z \rightarrow A$, and a correction reuse function $f_{reuse}(z, a_{demo}, a_{model}) : Z \times A \times A \rightarrow A$, where a_{demo} is the action computed by π_{demo}, and a_{model} is the action computed by π_{model}. The correction reuse function computes the final action to be executed by the robot as a function of the current observed state, and the actions computed by the model-based and the demonstration policies. During training, a human teacher observes the robot performing the task using the model-based policy, and corrects the action of the robot if the computed action is erroneous. The robot stores the received corrections as exceptions to the underlying algorithm. During execution, if an exception for a similar situation is found in the correction database, the robot substitutes the action computed by the algorithm with a demonstrated correction action. The schematic representation of the M+C system is given in Fig. 3.

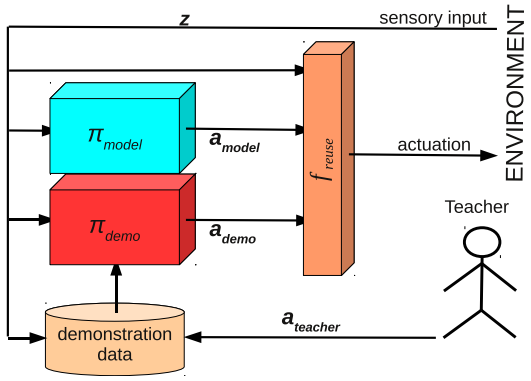


Fig. 3 The schematic representation of the M+C approach.

3.3 Multi-Resolution Model Plus Correction (MRM+C)

We combine M+C with MRTE and contribute the Multi-Resolution Model Plus Correction (MRM+C) algorithm to learn how to reduce the computational cost of task execution, and how to improve the overall task execution performance from human feedback. We define MRM+C as a tuple

$$< \pi_{arbitrator}, \{c_1, c_2, ..., c_N\} >$$

where $c_r =< Z, A, f_{state}, f_{action}, \pi_{demo}, \pi_{model}, f_{reuse} >$ is an instance of a modified version of the M+C model defined as a tuple at the detail resolution $r \in R$. A schematic diagram of the Multi-Resolution M+C framework is given in Figure 4.

3.4 Demonstration Delivery: Training the System

During the demonstration sessions, the teacher uses a custom user interface running on a host computer to access the

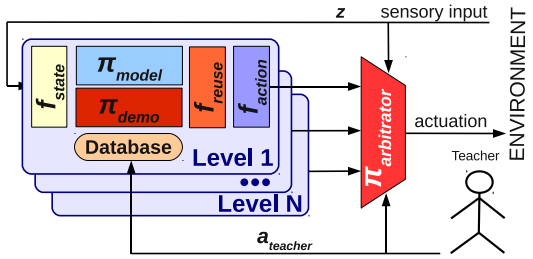


Fig. 4 The schematic representation of the MRM+C approach.

current detail resolution as well as the representation of the observed state at that resolution. The same user interface is also used for delivering the action corrections and changing the detail resolution. The host computer communicates with the robot over wireless Ethernet connection. The robot broadcasts its computed state, the current detail resolution, and other task-specific information back to the host computer at each step. The robot also uses a text-to-speech software system to announce the inferred state of the system and the action selected to be executed. The received state information of the robot is then visualized on the display. The state visualization part of the interface is task-specific.

Along the course of a demonstration, the teacher observes the robot as it executes the task, and intervenes by means of the following feedback types:

- The **elaborate command**: This type of feedback switches to the next finer detail resolution.
- The **correct command**: This type of feedback replaces the computed action to be executed with another action defined in the same detail resolution.

If an **elaborate** command is received, the system checks if there is a finer detail resolution available. If such a resolution is found, the received elaborate command is stored with the current state of the system represented with the state definition for the finest detail resolution available. The system then switches to that detail resolution and goes back to the action computation step. If a **correct** command is received, the action to be executed by the robot is replaced with the corrected action. The received action is also stored in a correction database along with the observed system state at the current detail resolution. The algorithm for MRM+C training is given in Alg. 1.

3.5 Demonstration Reuse: Autonomous Execution

Each time the robot reaches an intermediate destination point during the autonomous task execution, the MRM+C algorithm switches to the most coarse detail resolution available and computes an action using the algorithm associated with that detail resolution. Then, the system starts searching in this particular order:

Algorithm 1 The algorithm for training the MRM+C system

```
1: resolution ← COARSEST
2: state ← computeState(resolution)
3: action ← computeAction(state)
4: executeAction(action)
5: if feedbackReceived() then
6:     feedback ← readFeedback()
7:     if feedback == ELABORATE then
8:         if resolution < FINEST then
9:             saveDetailDemonstration()
10:            increaseResolution()
11:            goto 2
12:        end if
13:    else if feedback == CORRECT then
14:        action ← readCorrection()
15:        saveCorrectionDemonstration()
16:        executeAction(action)
17:    end if
18: end if
```

Algorithm 2 The algorithm for MRM+C execution

```
1: resolution ← COARSEST
2: state ← computeState(resolution)
3: mostSimilar ← ∅
4: maxSim ← 0
5: for each demo ∈ correctionDB_resolution do
6:     sim ← getSimilarity(state, demo(state))
7:     if sim > maxSim then
8:         maxSim ← sim
9:         mostSimilar ← demo
10:    end if
11: end for
12: threshold ← getCorrectionThreshold(resolution)
13: if maxSim > threshold then
14:    action ← demo(action)
15: else
16:    mostSimilar ← ∅
17:    maxSim ← 0
18:    fState ← computeState(FINEST)
19:    for each demo ∈ elaborationDB do
20:        sim ← getSimilarity(fState, demo(fState))
21:        if sim > maxSim then
22:            maxSim ← sim
23:            mostSimilar ← demo
24:        end if
25:    end for
26:    threshold ← getElaborationThreshold()
27:    if maxSim > threshold then
28:        if resolution < FINEST then
29:            increaseResolution()
30:            goto 2
31:        else
32:            action ← computeAction(state)
33:        end if
34:    else
35:        action ← computeAction(state)
36:    end if
37: end if
38: executeAction(action)
```

– A correction sample in the demonstration database for the current detail resolution
– An elaborate command in the elaboration demonstration database for switching to the next detail level with finer resolution.

If a correction sample is found in the corrective demonstration database that is received when the robot was in a state *similar enough* to the current state of the system, the provided correction action is selected as the next action. If an elaborate command is received in a state *similar enough* to the current state, the system changes its detail resolution to the next finer detail resolution and recomputes an action using the hand-coded algorithm specified for that resolution. We use a domain and task specific similarity metric with empirically determined parameters. The algorithm for the autonomous MRM+C execution is given in Alg. 2.

4 Humanoid Obstacle Avoidance Task

We apply the proposed multi-resolution task execution and refinement approach on an obstacle avoidance task using a humanoid robot in a robot soccer environment. We define the obstacle avoidance task for a humanoid soccer robot as the problem of walking to a destination position without colliding with various obstacles placed on the field. In our implementation, the robot starts in its own goal area and the goal of the task is to reach within 1 meter distance of the opponent goal. The number, shapes, and locations of the obstacles on the field are unknown to the robot beforehands. We use the regular field of the RoboCup Standard Platform League (http://www.robocup.org, http://www.tzi.de/spl) as the experimentation area, and Aldebaran Nao humanoid robot (http://www.aldebaran-robotics.com) as the robot platform. Fig. 5 presents an example instance of the humanoid obstacle avoidance task with three obstacles. The dashed lines represent an example traversal of the course by the robot with the yellow circles denoting the intermediate destination points selected by the robot.

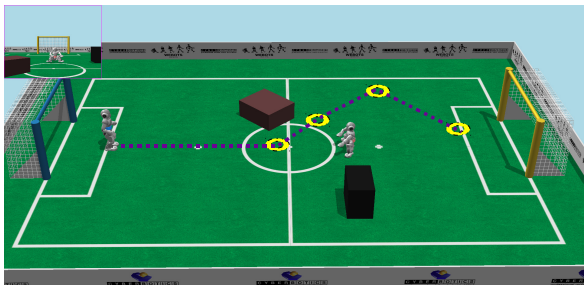


Fig. 5 An example instance of the humanoid obstacle avoidance task with an example solution in a configuration where two box-shaped obstacles and another humanoid robot are placed on the field.

4.1 Free Space Modeling using Vision

Instead of trying to detect the obstacles on the field, we build a free space model of the surroundings to decide which direction is the best to move to. The soccer field is a green carpet with white field lines on it. Therefore, anything that is non-green and lying on the field can be considered as an obstacle, except for the detected field lines (Fig. 5). We utilize a simplified version of the *Visual Sonar* algorithm by Lenser and Veloso [10] and the algorithm by Hoffmann *et al.* [8]. We scan the pixels on the image along evenly spaced vertical lines called *scanlines*, starting from the bottom end and continue until we encounter a certain number of non-green pixels. Although the exact distance of a certain pixel from the robot is a function of the position of the camera, the 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 (Fig. 6(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. The final computed free space is then divided into 15 slots, each covering an arc of 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 (Fig. 6(c)). Here, the dark triangles indicate the free space slots pointing towards the opponent goal.

For the humanoid obstacle avoidance task, we define three detail resolutions: $R = \{coarse, medium, fine\}$. In the remainder of this section, we explain the state and action definitions, and the destination point selection algorithms for each detail resolution.

4.2 Coarse Detail Resolution

At the coarse detail resolution, the 180^o space in front of the robot is divided into five equal slices of 36^o each. The existence of an obstacle along a free space slot is represented with a boolean value in the state vector where **true** indicates the slot is occupied. The slot is marked as occupied if the mean distance of the most detailed free space representation

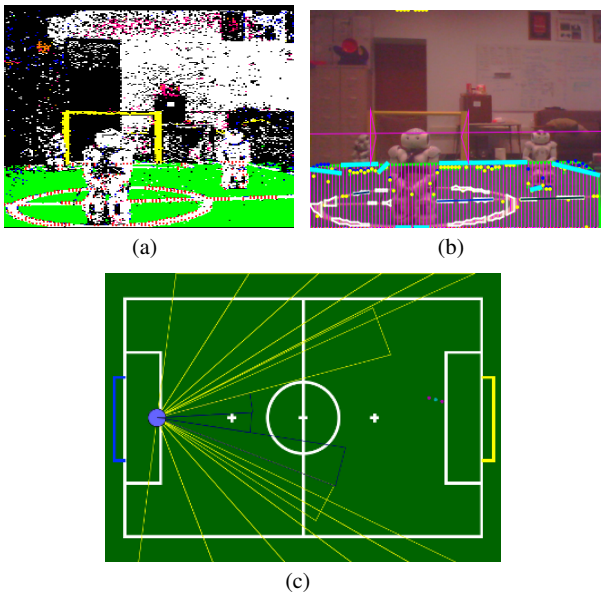


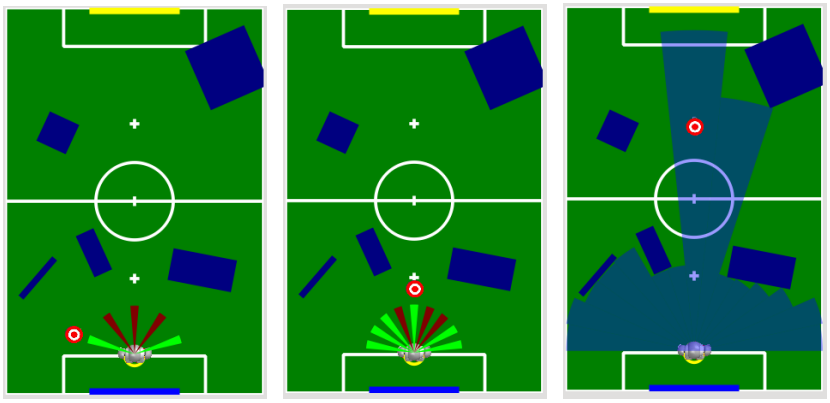
Fig. 6 The environment as perceived by the robot: a) the color segmented image, b) the computed perceived free space segments, and c) the resulting free space model.

slots that fall within it is less than a certain threshold. In our implementation, we use a threshold of 120 centimeters for considering a free-space slot as occluded. The visualization of the state in the low detail resolution is given in Fig. 7(a).

At this detail resolution, the destination point can be selected from among the five free space slot directions with a distance of 120 centimeters. However, the hand-coded algorithm for this resolution only selects from the three walking directions: *forward*, *left*, or *right*. If the middle slot (the slot number 2) is free, the algorithm selects the forward direction, otherwise it checks the right and left slots to decide. The algorithm also favors the left direction over the right direction, if the leftmost free-space slot (the slot number 4) is free. The destination point selection algorithm for the first detail level is given in Alg. 3.

Algorithm 3 Destination point selection algorithm for the coarse detail resolution.

```
1: slot ← −1
2: state ← getBooleanState(COARSE)
3: if ¬state(2) then
4:     slot ← 2
5: else
6:     if ¬state(0) then
7:         slot ← 0
8:     else
9:         slot ← 4
10:    end if
11: end if
12: angle ← calculateDirection(slot)
13: distance ← 120
14: return calculateGlobalPoint(angle, distance)
```



(a) (b) (c)

Fig. 7 Example visualizations for the state representations at different detail resolutions for the same situation. a) coarse detail resolution, b) medium detail resolution, and c) fine detail resolution. For the coarse and medium level resolutions, a green slot means no obstacle towards that direction, and a red slot means this direction is occluded by an obstacle. The target sign represents the selected destination point on the field according to the algorithm for that detail resolution.

4.3 Medium Detail Resolution

The state representation for the medium detail resolution uses the same principles as the coarse state representation with the exception of using nine slots instead of five. An example visualization of a medium detail resolution state representation is given in Fig. 7(b).

The hand-coded algorithm for this resolution goes over each free-space slot and selects the direction of the closest available slot to the opponent goal as the destination direction, using a fixed walking distance of 120 centimeters. The destination point selection algorithm for the medium detail resolution is given in Alg. 4.

Algorithm 4 Destination point selection algorithm for the medium detail resolution.

$state \leftarrow getBooleanState(MEDIUM)$
$goal \leftarrow getGoalSlot()$
$closestSlot \leftarrow 0$
$minDist \leftarrow 9$
for $slot \leftarrow 0; slot < 9; i \leftarrow slot + 1$ **do**
  if $|goal - slot| \leq minDist$ **and** $\neg state(slot)$ **then**
    $minDist \leftarrow |goal - slot|$
    $closestSlot \leftarrow slot$
  end if
end for
$angle \leftarrow calculateDirection(closestSlot)$
$distance \leftarrow 120$
return $calculateGlobalPoint(angle, distance)$

4.4 Fine Detail Resolution

At the finest detail resolution, the free space is divided into 15 slots, and the occupancy status for each slot is represented with a distance value in centimeters. This value denotes the distance to the closest detected obstacle lying within the coverage of that particular free-space slot. Fig.7(c) shows an example visualization of the state representation for the fine detail resolution.

In addition to the destination direction, the algorithm for the fine detail resolution also determines the walk distance towards that direction. We go over each free-space slot and for each slot we compute a weighted distance value using a sliding window of size three with the weights 0.25 at both ends and 0.5 for the center. The direction of the free-space slot with highest weighted distance is then selected as the walking direction and the computed weighted distance is used as the walking distance.

To be able to calculate the similarity of two given states in any of the detail resolutions, in this implementation we use the following function:

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

where K is a coefficient for shaping the similarity function, and $diff$ is the calculated sum of absolute differences of the slot distances. For the boolean slots, $diff$ is 0, if both slot values are the same, and 1 otherwise. In our implementation, we use $K = 5$.

4.5 Demonstration Interface

For the humanoid obstacle avoidance task, we use a task-specific user interface for both monitoring the task execution, and delivering feedback. The user interface visualizes the perceived free-space information, the position of the robot on the field, and the current selected destination point that

Algorithm 5 Destination point selection algorithm for the fine detail resolution.

$goalAngle \leftarrow getGoalAngle()$
if $goalAngle < -\frac{\pi}{2}$ **or** $goalAngle > \frac{\pi}{2}$ **then**
  if $|angle_0 - goalAngle| < |angle_{N-1} - goalAngle|$ **then**
    $destAngle \leftarrow angle_0$
  else
    $destAngle \leftarrow angle_{N-1}$
  end if
  $destDistance \leftarrow 120$
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
  $angle \leftarrow angle_{maxSlot}$
  $distance \leftarrow maxDist$
end if
return $calculateGlobalPoint(angle, distance)$

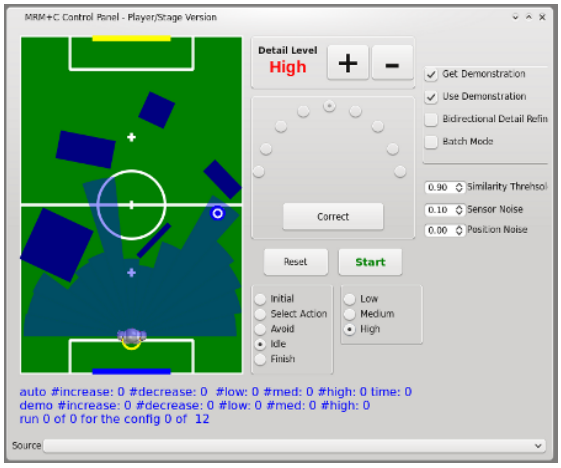


Fig. 8 The user interface for delivering corrective demonstration to the robot.

the robot walks to. A snapshot from the developed software is given in Fig. 8.

The teacher uses the elaborate button to issue a detail resolution refinement command. The current detail resolution is also displayed on the screen. If the current detail resolution is either *Coarse* or *Medium*, the teacher uses the radio buttons located on the bottom-right part of the interface. At the *Fine* detail resolution, the user specifies the destination point by clicking on the field visualization on the interface. There are 9 radio buttons placed on an arc, each representing a free space slot. For the *Medium* detail resolution, all radio buttons are enabled. For the *Coarse* detail resolution, every other button is enabled, reducing the number of enabled buttons to 5. At the *Fine* detail resolution, all radio buttons are disabled as the system expects a correction in the form of a global point on the field. Similarly, at the *Medium* and *Coarse* detail resolutions, it is not possible to specify a destination point by clicking on the visualized field.

5 Results

Our experimental evaluation of the proposed approach is two fold. In the first part, we present an experimental evaluation performed on a real RoboCup field with Nao humanoid robot. In the second part, we present an extensive experimental analysis of the proposed approach on a simulated version of the humanoid obstacle avoidance task.

5.1 Real World Experiments

We evaluated the performance of MRM+C approach against the hand-coded controllers at the lowest and the highest detail resolutions (Coarse Model and Fine Model algorithms) on the obstacle avoidance task using two different obstacle configurations, and an empty field as the base case (Fig. 9).

We used the task completion time as the performance measure for the cases the robot was able to complete the task. The results are given in Table 1. We ran 5 trials per method for each configuration. The **Rate** column presents the success rate. The **Time** shows the average time it took the robot to complete the task for the successful trials. The units for the rate and the average time columns are percentages and seconds, respectively.

An examination of the results yields that the success rate drops and the average task completion time gets longer as the number of obstacles increase, as expected. For the empty field configuration, all algorithms performed well in terms of success rate, while the hand coded algorithm for the fine detail resolution outperformed the others. The main reason behind this result is since the fine resolution algorithm uses free space slot distances to compute the destination point, it selects a destination point very close to the opponent goal and the task ends once the robot reach the destination so the robot does not lose any time in localizing itself and scanning the field for free space modeling. The performance of MRM+C was better than the coarse detail resolution algorithm but was worse than fine detail resolution algorithm mainly due to the number of field scans it has executed.

For the single obstacle case, the performance of the coarse detail resolution algorithm degraded considerably but the fine detail resolution algorithm and MRM+C were able to achieve high success rates. The fine detail resolution algorithm outperformed the MRM+C since it uses the most detailed state representation and computes long distance destination points, yielding a smaller number of field scans.

For the three obstacles case, the coarse detail resolution algorithm was too simple to handle the case, and the

(a) (b) (c)

Fig. 9 The obstacle configurations used in the experimental evaluation. a) empty field, b) a single obstacle placed on the center of the field, and c) three obstacles placed around the center circle.

fine detail resolution algorithm was not able to compute the propoer actions in most of the times. Combining the use of simpler algorithms when the current obstacle model does not yield the need for very detailed actions, and the corrective demonstration actions provided by the teacher, the MRM+C algorithm outperformed both hand-coded algorithms despite a considerable performance degradation compared to the previous configurations.

In 8 out of 15 failed trials, the failure was mostly due to the poor self localization data. The destination points computed by the algorithms are in global world coordinates; therefore, the performance gets heavily affected by the error in the estimated position.

Table 1 Performance evaluation results for the MRM+C approach in Humanoid Obstacle Avoidance domain. The times are in seconds.

	Empty Field		1 Obstacle		3 Obstacles	
Method	**Rate**	**Time**	**Rate**	**Time**	**Rate**	**Time**
Coarse Res.	80%	115	60%	195	0%	N/A
Fine Res.	100%	59	80%	94	40%	133
MRM+C	80%	96	100%	103	60%	182

5.2 Simulation Experiments

To evaluate the proposed approach extensively without suffering from the indirect factors like the occasional self localization errors in the real world experiments, we modeled a simulated version of the humanoid obstacle avoidance domain as the experimental testbed. We use the Player/Stage framework [6] to model the environment in 2D. We model the Nao humanoid robot with an omnidirectional wheeled robot base, and we use a laser range finder to emulate the vision-based free space perception used on the real Nao robots. The laser range finder readings are processed and converted into the same format as the free-space detection module on the real Nao provides. The omnidirectional walk of Nao is modeled as a holonomic motion on 2D ground plane and the speed of the wheeled robot is limited to 10 cm/s, which is roughly the speed of a real Nao robot. We imitate the self-localization information in the simulation with a global positioning system distorted with a certain amount of white noise.

We used randomly distorted variations of two different obstacle configurations with three obstacles each. For each trial, small random translational and rotational offsets are applied on the obstacle positions in the configuration.

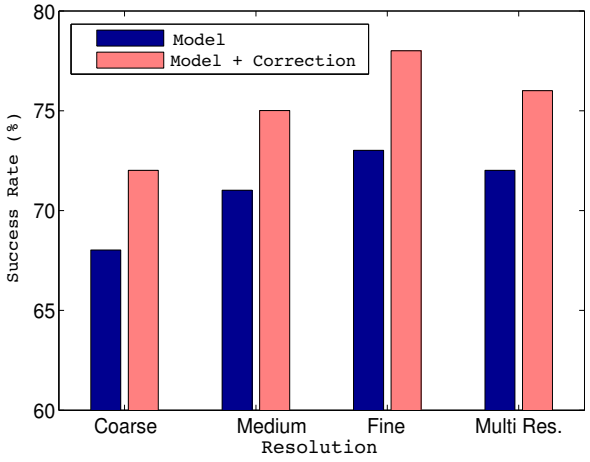


Fig. 10 The overall performance results for the individual algorithms and M+C instances for each detail resolution, along with the multi resolution performances without (MRTE) and with (MRM+C) corrective demonstration. The performance is measured with the per cent of succeeded runs.

We evaluated the hand-coded algorithms for each detail resolution (M), the improved algorithms using corrective demonstration (M+C), multi-resolution task execution using the hand-coded algorithms (MRTE), and the multi-resolution corrective demonstration (MRM+C) algorithm. During the training session, 23 low level, 8 medium level, and 14 high level demonstrations were collected for the corrective demonstration part, and 22 demonstrations for chang-

ing the detail resolution were recorded for the detail resolution selection part.

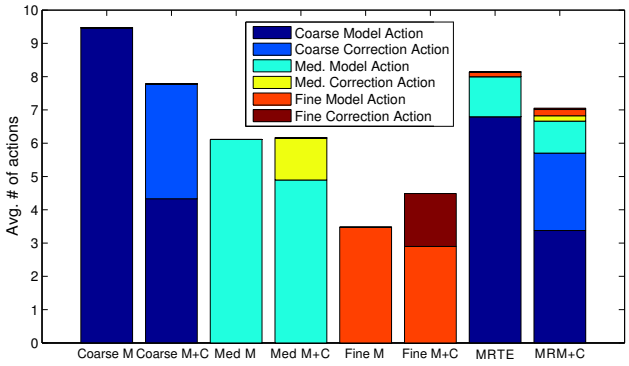


Fig. 11 The average number of actions executed per individual algorithms, M+C instances, MRTE, and MRM+C systems.

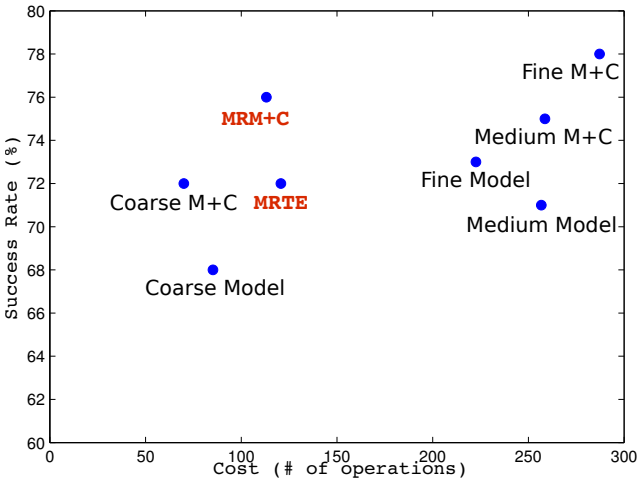


Fig. 12 The cost vs. performance plot for individual algorithms, M+C instances, MRTE, and MRM+C systems.

We ran 100 experiments for each algorithm. Fig. 10 shows the success rates of the algorithms. The blue bar in the Multi Resolution group is the success rate for the MRTE algorithm, and the red bar in the same group is the success rate for the MRM+C algorithm. As expected, the success rate of the algorithms increase as the algorithm gets more complex and runs at a higher detail resolution. In all four configurations (three detail resolutions, and the multi-resolution execution), the M+C instances outbested the algorithms alone, and the MRM+C algorithm outperformed the MRTE algorithm. The composition of executed actions per evaluated algorithm is given in Fig. 11. In both the MRTE and MRM+C evaluations, the majority of the executed actions were computed by the low detail resolution algorithm with and low detail resolution demonstration database. Yet, the success rates for the MRTE and MRM+C algorithms are better than the low and medium level algorithms, and close to the high level algorithm. To be able to assess the effectiveness in terms of the success rate versus the computational cost, we approximated the computational complexity of an algorithm as the total number of comparison and assignment operations in the pseudo-code algorithms for the worst case execution scenario. The computational cost versus success rate results are given in Fig. 12. In the figure, the horizontal axis is the average number of operations per algorithm, computed as the sum of estimated costs per detail resolution multiplied by the average number of actions at that resolution. For the execution of correction actions, we assume a flat cost regardless of the detail resolution. As it is seen in the figure, the success rate increases as the complexity of the underlying algorithm increase. Also it is evident that the augmentation of complementary corrective demonstration actions through M+C approach improves the success rate at each detail resolution. Similarly, the multi resolution task execution with corrective demonstration (MRM+C) performs better than MRTE. The success rate of the finest resolution algorithm with corrective demonstration (Fine M+C) is higher than the MRM+C system, but the success rate / computational cost ratio of MRM+C is better.

6 Discussion and Conclusion

In this paper, we introduced Multi-Resolution Task Execution (MRTE) approach, a novel method for handling different situations in a task through running different algorithms with varying complexities and using state and action representations at different detail resolutions. The system favors the most coarse detail resolution by default, and learns how to dynamically change the detail resolution to operate at from corrective human feedback. To the best of our knowledge, this is the first study that uses human feedback to learn a policy for changing its internal representation according to the observed state of the system.

We further elaborated on the proposed multi-resolution task execution to allow the human teacher to provide corrective feedback at different detail resolutions to improve execution performances of individual algorithms associated with each detail resolution. We introduced Multi-Resolution Model Plus Correction (MRM+C) algorithm as a combination of the Multi-Resolution Task Execution (MRTE) and the Model Plus Correction (M+C) algorithms. M+C enables a human teacher to improve an algorithm through corrective feedback without changing the algorithm itself. We presented formal models for the proposed approaches that relate the presented models with each other, and with the traditional LfD methods.

Our experimental results demonstrate the computational efficiency of multi-resolution approach compared to con-

trollers with fixed state and action definitions. In the real world experiments, the MRM+C system outperformed the algorithm of the finest detail level, demonstrating the task execution improvement using corrective demonstration. The experiment results from the simulated version of the task also confirms the improvement on the task execution performance in presence of corrective demonstration. Furthermore, the results demonstrate that the computational footprint of the overall task execution can be reduced drastically without critically suffering from the task execution performance.

Since our main motivation was to evaluate the effectiveness of the underlying formal model, in this paper we assumed that the teachers have a technical understanding of the task at hand, and are able to use the custom user interface for both monitoring the current state of the robot as well as for delivering feedback. We assume that the teacher will have no difficulties in interpreting the observed state of the system at different resolutions. Furthermore, we assume that the teacher can accurately foresee if a higher detail resolution would render the task doable in a given situation if the current computed action needs correction. As a result, we assumed that the demonstration data collected through the training session has good quality, and contains little or no wrong feedback actions. In our experimental evaluation, the teacher was the first author for fulfilling the assumptions listed above. The teacher used the user interface to infer the state of the system, and chose the best action according to his observations without having access to an optimal policy.

In our current evaluation, the finest state representation is the set of all available state features, and we compute coarser detail resolutions using the finest state representation. Therefore, in the current implementation, using finest detail resolution for the elaboration commands does not create a computational burden. However, in other tasks, the finest detail resolution might require intensive computations. To address this issue, we will investigate methods for using the coarser detail resolutions for handling elaboration commands in the future.

Our approach provides a very convenient framework for mass deployment of robots into real world as it allows the robots to be shipped with a basic set of abilities, and allows the users to tail the robot behavior according to the needs of their particular environmental or task conditions. Therefore, our future plans include an experimental evaluation of our approach with people having varying levels of technical understanding. Although the formal framework is suitable for any task or skill, we believe there is much room for improvement in the social interaction between the teacher and the robot. In particular, we expect non-technical everyday users to have difficulties in interpreting the state of the world as observed by the robot. As a consequence, we expect untrained users to have difficulties in developing an intuition on when to provide correction and when to switch into a finer detail resolution, solely by interpreting the current observed state displayed on the user interface. We hypothesize that a possible way of improving the teacher's understanding of the perception of the world as seen by the robot is to make the robot more interactive and give the teacher the ability to inquiry robot state through dialog.

We plan to evaluate the efficiency of our approach on a variety of more complex real world tasks to understand and tackle the challenges in the teacher-robot interaction. Furthermore, we plan to evaluate the robustness of our approach against uncertainty in sensing and action, and noisy demonstration data.

References

1. Argall, B., Browning, B., Veloso, M.: Learning robot motion control with demonstration and advice-operators. In: Proceedings of IROS'08 (2008)
2. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Automation Systems **57**(5), 469–483 (2009). DOI http://dx.doi.org/10.1016/j.robot.2008.10.024
3. Argall, B.D., Sauser, E., Billard, A.: Tactile Guidance for Policy Adaptation. Foundations and Trends in Robotics **1(2)**, 79–133 (2010)
4. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research **34** (2009)
5. Cobo, L.C., Zang, P., Jr., C.L.I., Thomaz, A.L.: Automatic state abstraction from demonstration. In: Proceedings of IJCAI 2011
6. Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of ICAR 2003
7. Grollman, D., Jenkins, O.: Dogged learning for robots. In: Proceedings of ICRA 2007
8. Hoffmann, J., Jüngel, M., Lötzsch, M.: A vision based system for goal-directed obstacle avoidance used in the rc'03 obstacle avoidance challenge. In: Proceedings of RoboCup 2004 Symposium
9. Kolter, J.Z., Abbeel, P., Ng, A.Y.: Hierarchical apprenticeship learning with application to quadruped locomotion. In: Proceedings of NIPS'07 (2007)
10. Lenser, S., Veloso, M.: Visual sonar: Fast obstacle avoidance using monocular vision. In: Proceedigns of IROS 2003
11. Levine, S., Popovic, Z., Koltun, V.: Feature construction for inverse reinforcement learning. In: Procs. of NIPS 2010, pp. 1342–1350
12. Mericli, C., Mericli, T., Akin, H.L.: A reward function generation method using genetic algorithms: A robot soccer case study. In: Proc. of AAMAS 2010 (2010)
13. Mericli, C., Veloso, M., Akin, H.L.: Task refinement for autonomous robots using complementary corrective human feedback. International Journal of Advanced Robotic Systems (2011)
14. Mericli, C., Veloso, M., Akin, H.L.: Improving biped walk stability with complementary corrective demonstration. Autonomous Robots **32**, 419–432 (2012)
15. Sullivan, K., Luke, S., Ziparo, V.A.: Hierarchical learning from demonstration on humanoid robots. In: Humanoids 2010 Workshop on Humanoid Robots Learning from Human Interaction (2010)
16. Thomaz, A.L., Breazeal, C.: Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In: Proceedings of AAAI 2006

Çetin Meriçli is a post-doctoral fellow in the Computer Science Department at Carnegie Mellon University. He received his Ph.D. from the Department of Computer Engineering at Boğaziçi University, Turkey in 2011. 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 the president of the AAAI (Association for the Advancement of Artificial Intelligence), and a trustee of 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 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 Bogazici University. He is a trustee 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.