Int J Soc Robot (2010) 2: 195–215
DOI 10.1007/s12369-010-0060-0

Confidence-Based Multi-Robot Learning from Demonstration

Sonia Chernova · Manuela Veloso

Accepted: 5 May 2010 / Published online: 19 May 2010

Abstract Learning from demonstration algorithms enable a robot to learn a new policy based on demonstrations provided by a teacher. In this article, we explore a novel research direction, *multi-robot learning from demonstration*, which extends demonstration based learning methods to collaborative multi-robot domains. Specifically, we study the problem of enabling a single person to teach individual policies to multiple robots at the same time. We present *flexMLfD*, a task and platform independent multi-robot demonstration learning framework that supports both independent and collaborative multi-robot behaviors. Building upon this framework, we contribute three approaches to teaching collaborative multi-robot behaviors based on different information sharing strategies, and evaluate these approaches by teaching two Sony QRIO humanoid robots to perform three collaborative ball sorting tasks. We then present scalability analysis of *flexMLfD* using up to seven Sony AIBO robots. We conclude the article by proposing a formalization for a broader multi-robot learning from demonstration research area.

Keywords Learning from demonstration · Multi-robot learning · Human–robot interaction · Multi-robot systems

1 Introduction

Research on robot *learning from demonstration (LfD)* explores techniques for learning a policy from examples, or demonstrations, provided by a teacher [2]. LfD algorithms utilize a dataset of state-action pairs recorded during teacher demonstrations to derive a policy that reproduces and generalizes the demonstrated behavior. Proposed techniques span a wide range of policy learning methods, ranging from reinforcement learning [3, 24, 38, 46] to classification [13, 43] and regression [7, 23], and utilizing a variety of interaction methods, such as natural language [6, 42], teleoperation [23, 43], kinesthetic teaching [8, 25] and observation [5, 39].

Despite this rich diversity of approaches, all of the above algorithms are designed for single-robot domains in which the teacher instructs a single robot learner. We are interested in tasks that require the collaboration of multiple robots, in which by combining their unique abilities, or by simply extending their coverage, multiple robots are able to perform more complex tasks than a single robot alone. The development of algorithms for the management and coordination of multiple robots is a challenging problem that has been extensively studied in existing literature [16, 26, 49]. However, no previous work has yet explored methods for teaching multi-robot interaction and control through demonstration.

In this work, we extend LfD to introduce *multi-robot learning from demonstration (MLfD)*, which we define as the problem of teaching multiple independent robots through demonstration by a single teacher. MLfD presents many novel research challenges. Interaction between the robots and the teacher must allow the teacher to perform demonstrations to individuals, while also maintaining awareness of the group as a whole. However, when working with multiple robots, the teacher is not able to pay full attention to all robots at the same time. Each robot must therefore be tolerant of periods of neglect from the teacher during the learning process. Finally, most collaborative tasks require robots to coordinate their actions through the exchange of information. The demonstration learning algorithm must

S. Chernova (✉) · M. Veloso
Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15217, USA
e-mail: soniac@cs.cmu.edu

M. Veloso
e-mail: veloso@cs.cmu.edu

therefore not only support inter-robot communication, but also enable the teacher to teach collaborative elements of the task.

In this article, we explore the problem of enabling a single person to teach individual policies to multiple robots at the same time. We present *flexMLfD*, a task and platform independent multi-robot demonstration learning framework that supports both independent and collaborative multi-robot behaviors. Our approach is based on the Confidence-Based Autonomy (CBA) single robot demonstration learning algorithm, an interactive mixed-initiative approach to demonstration learning that enables the robot and teacher to jointly control the learning process and selection of demonstration training data [10, 11, 13]. The CBA algorithm enables the robot to identify the need for and request demonstrations for specific parts of the state space based on confidence thresholds characterizing the uncertainty of the learned policy. Our multi-robot framework takes advantage of the adjustable robot autonomy provided by CBA to enable each robot to learn a unique task policy. The resulting learning method can be applied to a single or multiple, independent or collaborative, robot learners.

Building upon the *flexMLfD* framework, we formalize three approaches to teaching emergent collaborative behavior based on different information sharing strategies: *implicit coordination*, *coordination through active communication*, and *coordination through shared state*. We evaluate and compare these techniques by teaching two Sony QRIO humanoid robots to perform three collaborative ball sorting tasks utilizing a continuous and noisy state representation.

Additionally, we present a case study analysis of the scalability of *flexMLfD* using up to seven Sony AIBO robots. For three different learning conditions, we examine how the number of robots being taught by the teacher at the same time affects the number of demonstrations required to learn the task, the time and attention demands on the teacher, and the delay each robot experiences in obtaining a demonstration.

We conclude the paper by discussing the broader research questions posed by the challenge of teaching multiple robots at the same time. We propose a formal definition of the MLfD learning problem, as well as key design choices and evaluation metrics, with the goal of providing a foundational structure for future work in this research area.

2 The *FlexMLfD* Framework

We investigate multi-robot learning from demonstration in the context of *loosely-coordinated* tasks, which we define as tasks that contain elements that can be independently performed by individual robots, but that require a degree of coordination to couple their execution. We present *flexMLfD*, a task and platform independent multi-robot demonstration learning framework that enables a single person to teach multiple robots to perform collaborative tasks.

Our multi-robot approach is based on the Confidence-Based Autonomy single-robot learning from demonstration algorithm [10, 11, 13]. We begin with definitions of the notation used throughout this article, followed by an overview of CBA. We then present the *flexMLfD* learning framework and introduce three techniques for teaching multi-robot collaboration through demonstration.

2.1 Definitions

The robot's state is represented by the n-dimensional vector $s \in \mathbb{R}^n$. Elements of the vector, called features, have continuous or discrete numerical values representing information known to the robot. The robot's actions, a, are bound to a finite set A of action primitives, which are the basic actions that can be combined to perform the overall task.

Each demonstration performed by the teacher is recorded as the pair (s, a), representing the correct action a the robot should perform from state s. Given a sequence of demonstrations (s_i, a_i), the goal is for the robot to learn to imitate the teacher's behavior by generalizing from the demonstrations and learning a policy $\pi : S \rightarrow A$ mapping from all possible states S to actions in A. The Confidence-Based Autonomy algorithm represents and learns the policy using a classifier; the algorithm can be combined with any supervised learning approach that provides a measure of confidence in its classification.

Our goal and definitions reflect several assumptions that we make in designing our approach. First, we assume that the teacher is able to demonstrate the task being taught, and that the robot's goal is to imitate the behavior of the teacher. Second, we assume that the robot's state information contains all sensory information necessary to learn the task policy (e.g., if some information, for example the time of day, is required to correctly select among different actions, we assume that this information is available to the robot). Finally, this learning approach is aimed at high level behavioral tasks, which is reflected in our representation by the discrete action space.

2.2 Confidence-Based Autonomy

The Confidence-Based Autonomy algorithm enables the robot to learn a policy imitating the behavior of the teacher. It is a mixed-initiative approach that enables the robot and teacher to jointly control the learning process and selection of demonstration training data. The algorithm consists of two components:

- *Confident Execution*: an algorithm that enables the robot to learn a policy based on demonstrations obtained by regulating its autonomy and requesting help from the teacher.

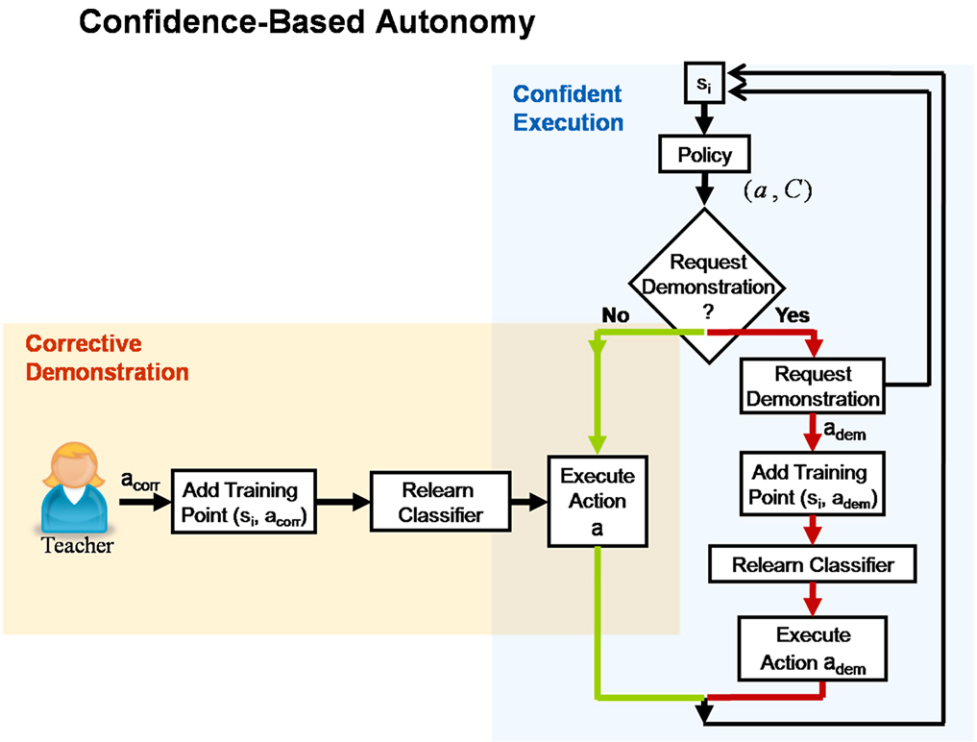


Fig. 1 Confidence-Based Autonomy: overview of the Confident Execution and Corrective Demonstration learning components

Demonstrations are selected based on automatically calculated classification confidence thresholds.

- *Corrective Demonstration*: an algorithm that enables the teacher to provide supplementary demonstrations and improve the learned policy by correcting possible mistakes made by the robot during autonomous execution.

Figure 1 presents an overview of the CBA algorithm, highlighting the interplay between its two components. At the beginning of each execution timestep, the robot's current state, s_i, is used to query the robot's policy. The policy returns the recommended action, a, and a classification summary, C, that provides information from the classifier regarding the query, including the classification confidence. The Confident Execution algorithm uses the classification summary to regulate the autonomy of the robot and decide between executing the policy action or requesting a demonstration from the teacher.

If autonomous execution is selected by the algorithm, the robot performs the policy selected action a. Alternatively, if a demonstration is required, the robot pauses and attracts the attention of the teacher through speech, lights, movement, or some other behavior. Once the teacher provides a demonstration action a_{dem}, the algorithm updates the policy with the new datapoint (s_i, a_{dem}) and executes the demonstrated action.

Note that while waiting for a demonstration, the environment around the robot may change. It is therefore important that the robot continues to update its state information and re-evaluate its decision for a demonstration request. This mechanism serves two purposes. First, there is a possibility that the environment will change in such a way that the robot will find itself in a high confidence state. In this case, the robot will abandon its demonstration request and perform the policy action autonomously. Second, it is possible that the teacher may be distracted and take some time to pay attention to the robot. In this case, the update ensures that once a demonstration action is received, that it is associated with the robot's most current state.

At a high level, the Confident Execution algorithm can be viewed as a supplementary layer added over a classification-based action policy. The algorithm serves two purposes: the selection of training data (our experimental results show that Confident Execution can often select more informative training data than a human) and the regulation of robot autonomy in new or uncertain situations by preventing autonomous execution in such states. However, since the algorithm interleaves learning and execution, the policy used to select actions and regulate autonomy is typically incomplete. As a result, the algorithm sometimes selects an incorrect action for autonomous execution.

Our second algorithmic component, Corrective Demonstration, addresses this problem by enabling the teacher to provide corrections for the robot's mistakes. If an incorrect action is executed by the robot, the teacher can provide a corrective demonstration, a_{corr}, during the execution of the incorrect action to indicate which action should have been executed in its place. The resulting demonstration datapoint,

(s_i, a_{corr}), associates the new action with the original decision state s_i.

Together, Confident Execution and Corrective Demonstration form an interactive, mixed-initiative online learning algorithm in which both the robot and human teacher have the ability to identify situations in which additional training data is required. The complete definition of the algorithm can be found in [13]. Learning is complete once the robot is able to perform the task correctly multiple times without requesting demonstrations or requiring corrections. If multiple variations of the task exist (e.g., interaction with different objects, or navigation in different environments), all variations the robot is expected to encounter during operation should be performed.

2.3 Multi-Robot Learning Approach

Within the *flexMLfD* framework, each robot acquires its own set of demonstrations and learns an individual task policy using an independent instance of the CBA algorithm [12]. This approach is in contrast to learning a single joint action for the complete multi-robot system.

The unique feature of the Confidence-Based Autonomy algorithm that enables it to be applied to multi-robot learning is the Confident Execution component, which enables each robot to regulate its own autonomy, and to pause execution when faced with an uncertain situation. The resulting self-regulation achieved by each learner enables the teacher to switch attention between robots on an as-needed basis.

Given a group of robots R, each robot $r \in R$ uses Confidence-Based Autonomy to learn a policy $\pi_r : S_r \rightarrow A_r$ from the robot's states to its actions. Each robot may have a unique state and action set, allowing distinct policies to be learned by possibly heterogeneous robots. The general representation and modularity provided by the CBA learning approach and interface result in a flexible task-independent and robot-independent learning framework.

Algorithm 1 outlines the general procedure followed by the teacher in performing multi-robot demonstrations. The teacher alternates between responding to demonstration requests when they are present, and correcting any mistakes in the autonomous behavior of the robots. The function $f(D)$, represents a demonstration request selection policy, such as first-in-first-out or round-robin ordering. In evaluations presented in this article $f(D)$ represents random selection.

Robots operating in the same space are likely to be interacting in some way, whether simply as obstacles in each other's way or through direct action. However, each robot is controlled by its own action policy, and during each demonstration the teacher interacts independently with only a single robot by instructing it what to do given its current world state. In the following section, we discuss how to teach robots to perform *collaborative* tasks by encoding relevant data about the interaction into the state information of each robot.

Algorithm 1 Multi-robot demonstration

Let D be set of current demonstration requests
while the robots request demonstrations or incorrect behavior is observed **do**
  if $D \neq \emptyset$ **then**
    – Select robot demonstration request r according to function $f(D)$
    – Perform demonstration for robot r
  else
    – Observe autonomous execution of the robots
    if correction is required for robot r **then**
      – Perform correction for robot r

2.4 Teaching Multi-Robot Collaboration

The approach we present for teaching collaborative behavior through demonstration relies on *emergent multi-robot coordination* [36], in which the solution to the shared multi-robot task emerges from the complementary actions performed by robots based on their independent policies. To achieve this coordination, each robot's action abilities may include communication. We define each robot's action set by $A = A_p \cup A_c$, where A_p is the set of physical robot actions and A_c is the set of communication actions. All actions within A are available to the teacher for demonstration.

Multi-robot coordination requires a rich state representation consisting of both local and communicated information. To this end, we categorize the robot's state features based on their source and purpose; for example, information locally observed by the robot's sensors may be private to the robot (e.g., current wheel angle), shared with its teammates at all times (e.g., robot position), or shared only under particular conditions (e.g., robot position only when known with high confidence). We define the robot's state as $s = \{F_o \cup F_s \cup F_c \cup F_e \cup F_t\}$, where

- F_o = locally *observed* state features that are private to the robot
- F_s = *shared* state features that are automatically communicated to teammates each time their value changes
- F_c = state features *communicated* using communication actions A_c as defined by policy π
- F_e = state features *extrapolated* from other features or robot actions
- F_t = state features containing data either directly contained in, or calculated based on, information communicated from *teammates*

In this representation, we differentiate local and communicated data, as in [41], but seamlessly integrate both into the robot's state. Coordination between robots occurs when complementary actions are selected by the policy based on this input. Using this representation, we present three methods for teaching emergent multi-robot coordination using demonstration.

2.4.1 Implicit Coordination

The most basic level of multi-robot coordination is implicit coordination, in which physical actions and observed state allow complementary behaviors to occur without communication or shared intent [27, 28, 37]. Using implicit coordination, robots make decisions based only on locally observed information and are often not aware of the coordination or even of each other's presence. Teaching implicit coordination through demonstration can therefore be reduced to the problem of teaching multiple robots to perform independent tasks at the same time. Within our framework, implicit coordination is represented by the policy:

$$\pi : \{F_o, \emptyset, \emptyset, \emptyset, \emptyset\} \rightarrow \{A_p, \emptyset\}$$

which maps the robot's locally observed state directly to the physical actions. Coordination occurs through the environmental changes resulting from the executed actions.

2.4.2 Explicit Communication

Domains in which a robot cannot acquire all needed information solely through its own sensors require *explicit communication* through the use of actions, such as by sending wireless messages. Explicit communication is widely used in multi-robot research, and a broad range of algorithms have been proposed with different approaches to what data is to be communicated, how often, and to whom [9, 18, 20]. Below we present two approaches for teaching collaboration based on explicit *state communication* [4].

Coordination Through Active Communication

Coordination through active communication enables the teacher to use demonstration to explicitly teach *when* communication is required. Based on demonstrations of communication actions A_c obtained from the teacher, communication is incorporated directly into a robot's policy along with the physical actions A_p. This technique enables the teacher to specify the conditions under which communication should take place. The resulting policy is defined as:

$$\pi : \{F_o, \emptyset, F_c, F_e, F_t\} \rightarrow \{A_p, A_c\}$$

While most physical actions have an observable effect that changes the robot's state (i.e., moving an object changes its location), the immediate effect of communication actions is not observable. To prevent the robot from remaining in the same state following a communication action, we utilize internal state features F_e to represent the last communicated value of each element of F_c ($\forall f \in F_c \rightarrow f \in F_e$). A mismatch between the value of a particular feature in F_e and F_c indicates that the local state no longer matches the teammates' knowledge. All state information received from other robots through communication is stored within the set F_t.

Coordination Through Shared State

Robot coordination frequently relies on shared state information that must be maintained up to date at *all* times, not just under specific conditions. For example, a robot performing a navigation task with its teammates may always need to know their locations. Coordination through shared state automates the communication process for this common case, enabling robot coordination based on automatically updated state features. Specifically, we define F_s as the set of local features that are automatically communicated to teammates each time their value changes. We therefore define coordination through shared by the policy:

$$\pi : \{F_o, F_s, \emptyset, \emptyset, F_t\} \rightarrow \{A_p, \emptyset\}$$

Since communication occurs automatically in this approach, communication actions are not demonstrated or incorporated into the robot policy. Using this technique, the teacher is able to focus on demonstrating only the physical actions to be performed based on state information shared between robots. Note that this approach assumes that shared features do not change very rapidly; attempting to share a sensor value which changes at a high frequency would quickly cause network congestion.

2.4.3 Discussion

Implicit coordination allows collaborative behaviors to be performed without communication, while active communication and shared state rely on communication to coordinate the robots' actions. The difference between the two communication-based approaches is most significant in domains in which communicated state features take on a *range* of values, and in which such features have importance only over a narrow segment of that range. Such features are commonly encountered in robotic problems. Consider, for example, a robot that only needs to know the location of its teammate if the teammate has located an object of interest. Under all other conditions, the teammate's position, if known, would be ignored. In this scenario, the teammate can choose between two communication strategies: (1) to communicate its location only when it finds an interesting object, or (2) to communicate its location at all times and rely on its teammate to ignore this information when it is not relevant. Both of the approaches are valid, and preference between them depends on the relative costs of sending communication messages and learning to ignore irrelevant information. This tradeoff between the amount of communication and information is captured by the active communication and shared state techniques.

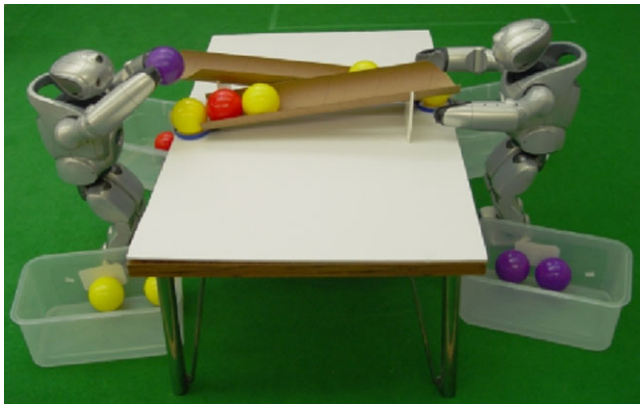


Fig. 2 The Sony QRIO robots performing ball sorting task

3 Evaluation Domains

Evaluation of multi-robot learning and the scalability of the presented approach was performed in two robotic domains using Sony AIBO and QRIO robots.

3.1 Ball Sorting Domain

In the ball sorting domain, which we designed, two Sony QRIO humanoid robots are located at two sorting stations connected by ramps (Fig. 2). Each station has an individual queue of colored balls (red, yellow or blue) that arrive via a sloped ramp for sorting. The robots' task is to sort the balls by color into four bins.

The following set of physical actions is available to each robot: $A_p = \{SortLeft, SortRight, PassRamp, Wait, Leave\}$. Actions *SortLeft* and *SortRight* enable the robot to pick up a ball and place it into a bin on either side. The *PassRamp* action causes the ball to be placed into the teammate's ramp, where it rolls down and takes position at the tail end of the other robot's queue. The *Wait* and *Leave* actions enable the robot to wait for a short duration or walk away from the table, respectively. Passing a ball to the teammate's ramp causes the ball to roll down and take the position at the tail end of the other robot's queue. The color and location of the balls is determined by each robot using its onboard vision system. Using these abilities, the robots are taught to perform the following three tasks:

Task 1: Each robot begins with multiple balls of various colors in its queue. QRIO A sorts red and yellow balls into the left and right bins, respectively, and passes blue balls to QRIO B. QRIO B sorts blue and yellow balls into the left and right bins, respectively, and passes red balls to QRIO A. If a robot's queue is empty, it should wait until additional balls arrive.

Task 2: Extend Task 1 such that each robot communicates to its teammate the status of its queue, empty or full. When its teammate's queue is empty, a robot in possession of a ball should pass the ball to the teammate's queue. However, only balls that can be sorted by the other robot should be passed. For example, QRIO A should pass only the blue and yellow balls, and QRIO B should pass only the red and yellow balls. If both queues are empty, the robots should wait.

Task 3: Each robot begins with multiple balls of various colors in its queue. QRIO A first sorts all of the red balls on the table into its left bin, while QRIO B passes balls of all colors, thereby helping to rotate the queue. Once all the red balls are sorted, QRIO B sorts all the blue balls into its left bin while QRIO A passes. Once all the blue balls are sorted, both robots sort the remaining yellow balls into their right bins. Whenever both queues are empty, the task is complete and the robots leave the table.

Each of the above tasks is designed to test different aspects of multi-robot teaching and coordination. In the first task, coordination between robots emerges naturally based solely on the physical actions of the robots and communication is not required. Task 2 requires coordination through communication to ensure that the sorted balls are distributed more evenly between the robots, while Task 3 adds ordering constraints and additional coordination requirements. State and action representations for each task will be defined in Sect. 4.

Our evaluation of the learning algorithm and teaching approaches utilizes continuous and noisy features. Before presenting this evaluation, we provide a walk-though of a simplified example of Task 1 with a noiseless, boolean state representation. Only a single demonstration of each state is required in this representation, helping to illustrate elements of the learning process. To achieve this representation, we calibrate the robot's onboard vision system to classify each ball into one of the discrete color classes, such that $F_o = \{red, yellow, blue\}$.

Figure 3 presents an interactive learning sequence between the teacher (T) and the robots (R). The robots begin in the top configuration, with no initial knowledge about the task. Both robots request a demonstration upon encountering the first state, and are instructed by the teacher to *SortLeft*, i.e., place their respective balls, red and blue, into the left bin. Upon receiving their individual instructions, each robot executes the specified action.

Step two in the figure shows the task state after both robots have completed their first action. One ball has been sorted on each side, and the next ball in the queue is now available to each robot. QRIO B receives a second blue ball, and therefore executes the previously demonstrated action autonomously. QRIO A requests a demonstration request for the new ball color, yellow.

At each following timestep, the robots observe their state, invoke the CBA classifier, and select between autonomous execution and demonstration. At steps 3 and 5 the robots are

taught to pass balls to their teammate, causing these balls to appear at the end of the other robot's queue. The learned policy also allows the robots to wait while no balls are present, and respond once a ball has been received (QRIO A, steps 4–6). Although the figure presents the learning process as a sequence of synchronized steps, no synchronization occurs in real-life learning. In this representation, learning the entire task requires 8 demonstrations, 4 per robot.

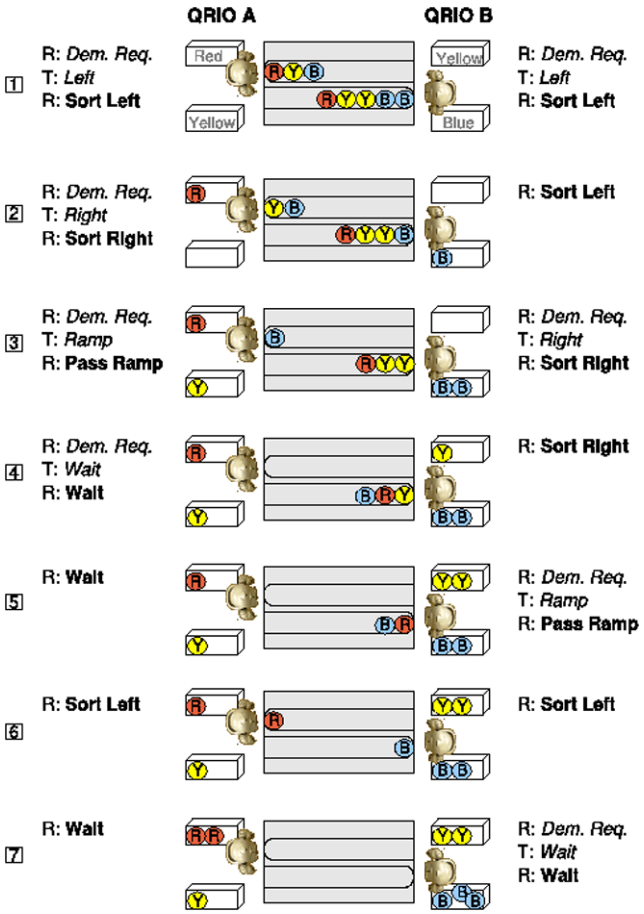


Fig. 3 Collaborative ball sorting example

3.2 Beacon Homing Domain

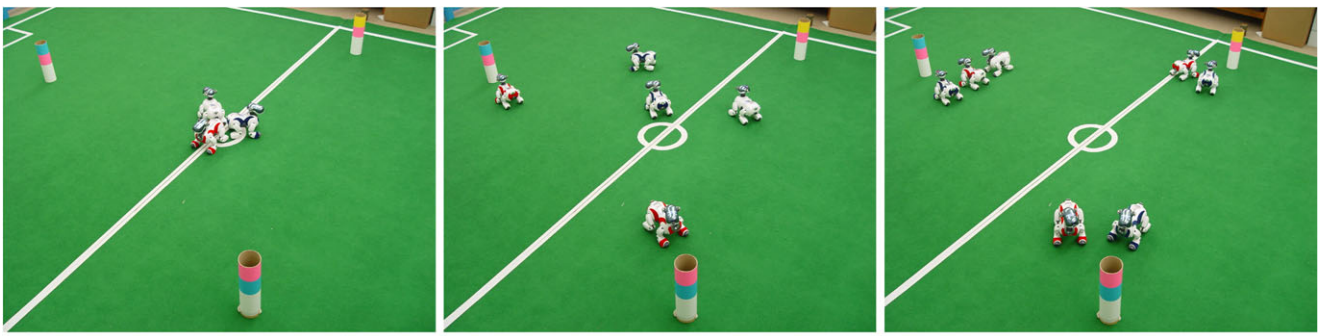
Figure 4 shows three examples of AIBO robots operating in the beacon homing domain, which we designed, consisting of an open area with three uniquely-colored beacons ($B = \{B_1, B_2, B_3\}$) located around the perimeter. Each robot is able to observe the relative position of a beacon using its onboard camera, and to communicate information via the wireless network. The set of action available to each robot is limited to basic movement commands, $A_p = \{Forward, Left, Right, Search, Stop\}$, used by each robot to navigate in the environment.

Using the representation defined in Sect. 6.1, we define the robot's state as:

- $F_o = \{d_1, d_2, d_3, a_1, a_2, a_3\}$
- $F_s = \{myBeacon\}$
- $F_t = \{n_1, n_2, n_3\}$
- $F_c = F_e = \emptyset$

The set of observed features, F_o, contains information about the robot's relative distance d_i and angle a_i to each beacon $i \in B$. For any beacon not currently in view, the distance and angle are set to the default values 4000 mm and 1.8 rad, respectively, to indicate that this beacon is far away. The set of shared state features, F_s, contains a single value, *myBeacon*, which is set to a beacon's ID number if the robot is within a set distance x of a beacon, and -1 if the robot is not located near a beacon. Each robot communicates the value of this feature to its teammates. In turn, all robots use this shared information to determine the values of state features $F_t = \{n_1, n_2, n_3\}$, which maintain the count of the current number of robots occupying each beacon.

In summary, using the above representation, each robot knows its position relative to beacons that it observes, and the number of other robots already located at each of the beacons. Using this information, the teacher can teach each robot to navigate from a random initial location in the center



(a) (b) (c)

Fig. 4 Beacon homing domain: (**a**) Example starting configuration, 3 robots. (**b**) Example intermediate stage, 5 robots. (**c**) Example final configuration, 7 robots

of the open region to one of the colored beacons. Specifically, the teacher instructs the selection of a beacon according to the following rules: *Given a maximum limit m for the number of robots that can occupy a marker, search until beacon i is found for which the number of robots, n_i, is less than m. Navigate to that beacon and occupy it by stopping within a set distance d. If at any point the number of robots at the selected beacon exceeds m, search for another beacon.*

These explicit rules of the task are known only to the teacher. During the learning process, each robot in the experiment learns an independent policy representing this behavior from demonstrations. All robots were taught the same task to ensure a fair comparison between robots for the scalability evaluation. We set the maximum number of robots allowed per beacon for each experiment to $m = \lceil \frac{\#Robots}{\#Beacons} \rceil$, such that at least one beacon must contain the maximum number of robots. Each experiment began with all robots located in the center of the open region (Fig. 4(a)) and ended once all robots had reached a beacon (Fig. 4(c)). Training continued until all robots executed the desired behavior efficiently and correctly without requesting demonstrations.

4 Evaluation of Coordination

We now present an evaluation of our three approaches to teaching multi-robot coordination. We begin by evaluating each coordination approach independently by applying it to one of the ball sorting tasks. We then evaluate the performance of the communication-based coordination approaches in greater detail using Task 3.

For all evaluations, the robots observe ball color as the average RGB values of the pixels in the detected ball region. Results are reported as the total number of demonstrations required to learn the task, averaged over ten trials. Note that in this discussion we are not distinguishing between demonstrations obtained from Confident Execution and Corrective Demonstration algorithms. While most demonstrations are initiated by the robot through Confident Execution, the Corrective Demonstration algorithm does play a significant role in aiding policy learning of each individual robot. Consider, for example, a robot that first observes demonstrations of passing both a blue and a yellow ball to a teammate. Upon encountering a red ball, the robot may assume, due to generalization of the classifier, that the same action should also be performed for this new color. Corrective Demonstration enables the teacher to correct this assumption if necessary.

4.1 Implicit Coordination Without Communication

We evaluate implicit coordination learning by training Task 1, in which the teacher's demonstrations are limited to the robot's physical actions A_p (the *Leave* action is not used for tasks 1 and 2). The following state representation, consisting only of locally observed information, is used:

- $F_o = \{R, G, B\}$
- $F_s = F_c = F_e = F_t = \emptyset$

Using this representation, both robots successfully learned their individual policies, which enabled them to collaboratively sort the balls by coordinating through their actions. Training the entire task required an average of 20 demonstrations (10 per robot), with a standard deviation of 1.1.

During the execution of the task, the robots encounter both noisy and noiseless states. The number of demonstrations required to learn each state-action mapping is proportional to the level of noise in the sensor readings. For example, an empty queue contains no ball color information and consistently results in the state vector $S = \{0, 0, 0\}$. Since this value is not affected by noise, only a single demonstration is required to teach each robot to *Wait* when the queue is empty. Ball color readings, on the other hand, vary due to both sensor noise and slight variations in the robot's view angle and ball distance between actions. An average of 3.1 demonstrations is therefore required for the model to learn to generalize over each ball color class.

4.2 Coordination Through Active Communication

Coordination through active communication enables communication to be represented directly within each robot's policy. We evaluate this approach using Task 2 and the following state representation:

- $F_o = \{R, G, B\}$
- $F_s = \emptyset$
- $F_c = F_e = F_t = \{Empty\}$

We define a new boolean feature, $F_c = \{Empty\}$, to represent the status of the robot's ball queue, empty or full. This feature is locally observed and communicated to each robot's teammate using communication action $A_c = \{SendEmpty\}$. For each robot, feature sets F_e and F_t contain the robot's own last communicated value of *Empty* and the most recent value of *Empty* received from its teammate, respectively.

The complete state vector consists of six features. Note that all of the information available to a robot, whether continuous or discrete, locally observed or communicated, is combined to form the robot's state vector, allowing the algorithm to generalize over all of these variations. Each feature value is typically normalized by the classifier prior to analysis to give each feature equal weight.

Using the above setup, the teacher required an average of 35 demonstrations to teach Task 2. While each ball color still required multiple demonstrations, the algorithm was able to

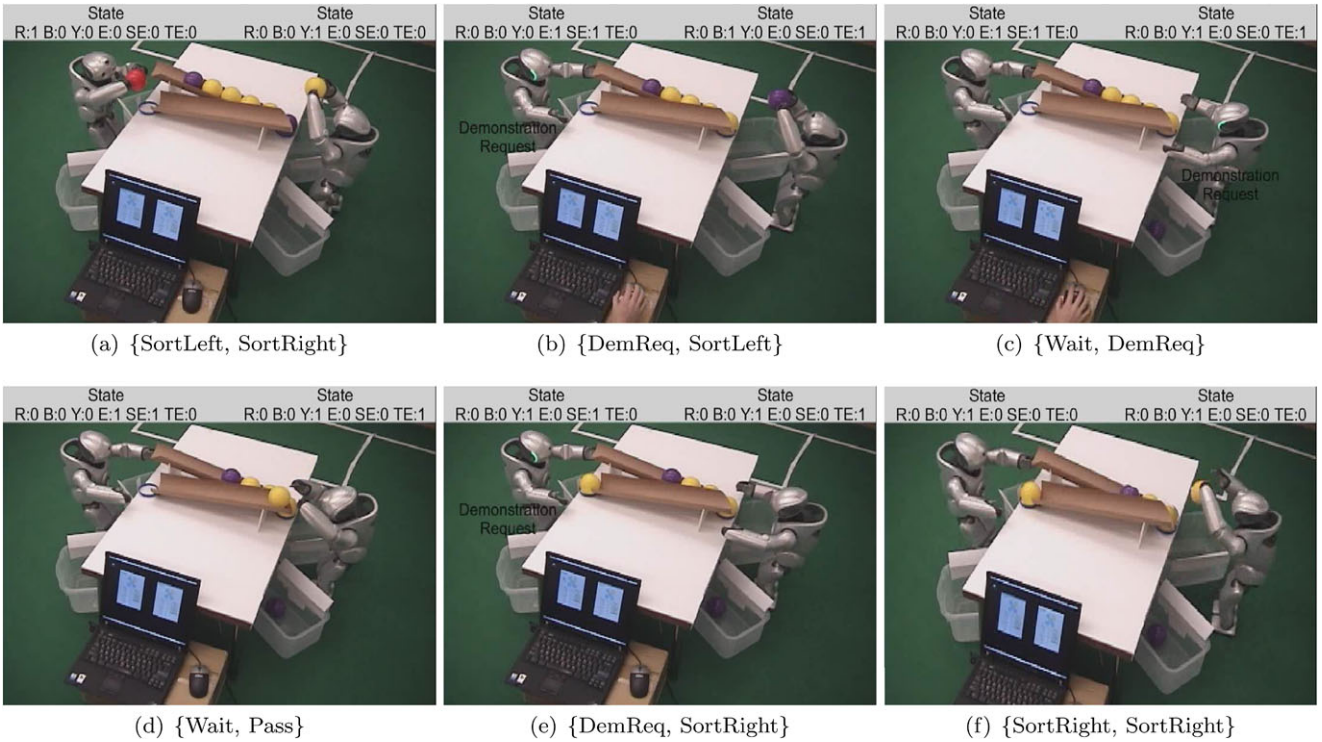


Fig. 5 Example Task 2 learning sequence using coordination through active communication

rapidly generalize over the boolean *Empty* features. For example, learning that communication updates must be performed each time the value of the feature *Empty* in F_c and F_e does not match, regardless of the ball color being observed, required only two demonstrations.

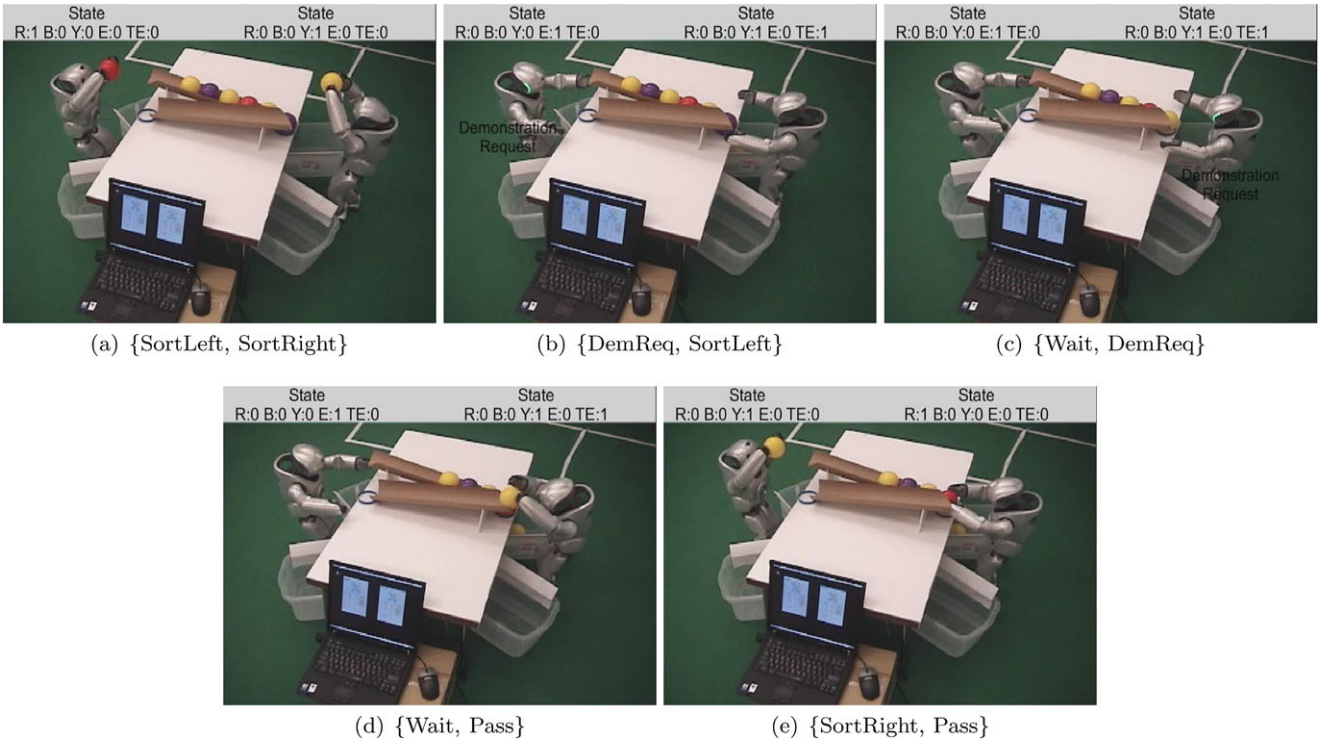
Figure 5 presents a sequence of images showing how the teacher uses demonstration to teach the ball sorting task using active communication. The teacher uses the laptop computer shown at the bottom of the image to perform demonstrations independently for each robot using its own instance of the CBA GUI interface. Each figure is annotated with the robot's current state, shown at the top of the image.

In Fig. 5(a), the QRIO on the left sorts a red ball into its left bin, while the QRIO on the right sorts its yellow ball into its right bin. After completing its action, the left QRIO observes that its ramp is empty and stops to request a demonstration from the teacher. During this time, the right QRIO continues to perform its task, sorting the next ball in its queue, which is blue, into its left bin. Since communication between robots has not yet occurred, the state of the right QRIO does not accurately reflect its teammate's status because the teammate's empty (TE) state feature value is 0.

In response to the demonstration request, the teacher instructs the left QRIO to communicate its queue status using the *SendEmpty* action. Figure 5(b) shows the robot immediately following this step—the right robot is now aware that its teammate's queue is empty. The left QRIO requests a demonstration for its new state, and the teacher instructs it to *Wait*. In Fig. 5(c) the right QRIO requests a demonstration for what to do given that it has a yellow ball and its teammate's queue is empty. The teacher instructs the robot to pass this ball to its teammate; the robot is shown performing this action in Fig. 5(d). Once the left QRIO observes that a yellow ball has arrived, it requests a demonstration (5(e)) and the teacher instructs the robot to communicate its updated ramp status to its teammate to indicate that the queue is no longer empty. Once this information is updated, the left QRIO continues by sorting the yellow ball into its right bin (5(f)). During this time the right QRIO autonomously continues with the ball sorting task, sorting its next yellow ball into the right bin. Once the left robot's queue becomes empty again, it will autonomously communicate the status of its queue to the other robot, and the ball sharing process will repeat autonomously.

4.3 Coordination Through Shared State

Coordination through shared state automates the communication process, leaving the teacher to demonstrate the robot's physical behavior based on shared information. In the place of explicit communication actions, this technique utilizes a set of *shared state features*. Any of the robot's locally observed state features may be selected by the teacher to be shared with a teammate. The status of each shared feature is then tracked by the system, and updates are communicated to the teammate each time the value changes.



(a) {SortLeft, SortRight} (b) {DemReq, SortLeft} (c) {Wait, DemReq}

(d) {Wait, Pass} (e) {SortRight, Pass}

Fig. 6 Example Task 2 learning sequence using coordination through shared state

Coordination through shared state was similarly evaluated using Task 2. State information shared between robots consists of the state feature $Empty$. The complete state is represented by:

- $F_o = \{R, G, B\}$
- $F_s = F_t = \{Empty\}$
- $F_c = F_e = \emptyset$

where $F_s = \{Empty\}$ represents a robot's local shared ramp status, and $F_t = \{Empty\}$ represents the ramp status of its teammate. The complete state contains five features, and the entire task required an average of 27 demonstrations to learn.

Figure 6 presents a sequence of images that shows the robots learning to perform Task 2 using collaboration through shared state. Figure 6(a) begins with both robots performing autonomous sorting, with the left robot sorting its last ball into its left bin. Figure 6(b) shows what happens once the left QRIO observes that its queue is now empty. Unlike the active communication approach described in the previous section, the value of the robot's *Empty* feature is automatically communicated to its teammate, such that the teammate's empty (TE) state feature for the right QRIO is set to 1. The left QRIO requests a demonstration, asking for instructions about what to do when its queue is empty. The teacher instructs the robot to *Wait*. In Fig. 6(c), in response to a demonstration request the teacher instructs the right QRIO to pass its yellow ball to its teammate, as shown in Fig. 6(d). Once the ball arrives at the bottom of the ramp, the left QRIO automatically communicates its updated ramp status to its teammate and autonomously resumes the sorting process, Fig. 6(e).

4.4 Combined Approach and Comparison

In the above evaluation, we showed that each of the presented coordination approaches can be successfully used to learn a variation of the ball sorting task. In this section, we further compare and evaluate our two communication-based methods—coordination through shared state and coordination through active communication—as well as a combined approach utilizing a combination of these techniques.

While our evaluation of Task 2 shows the feasibility of both active communication and shared state, it provides little information about the tradeoffs between the two approaches. The communication requirements of Task 2 are very limited, with only a single communicated boolean feature that had to be updated each time its value changed. A more informative analysis can be obtained by examining a domain with communicated state features that take on a *range* of values, and in which such features have importance only over a narrow segment of that range.

In this section, we use Task 3 to compare the performance of the three communication-based coordination approaches with respect to the number of demonstrations and number of communication messages. Task 3 is a variation of the ball

sorting task, in which the robots must sort balls in the order of their color class, such that all red balls are sorted into their bin first, then blue, and finally yellow. Once sorting is complete, the robots turn around and walk away from their sorting stations. To represent this domain, we add two new types of state features. The feature *SortColor* is used to represent the current color being sorted (red, blue or yellow), and the feature *PassCount* is used to represent the number of consecutive *PassRamp* actions that have been executed by a particular robot. Since the robots do not have a global view of the world, the *PassCount* feature is required to determine when all balls of a particular color have been removed. For example, when sorting red balls, both robots pass any blue or yellow balls they encounter into their teammate's ramp. The resulting effect is that the entire queue of balls rotates, enabling the robots to examine all balls one at a time. Once the value of the *PassCount* feature for both robots passes some threshold, in our case 10, this indicates that the entire queue has been examined and no additional balls of the current *SortColor* remain on the table.

In total, three pieces of information are communicated between robots: 1) the current color being sorted (*SortColor*), 2) the number of consecutive passes each robot has performed (*PassCount*), and 3) the status of each robot's ball queue (*Empty*). Actions available to the robots for performing this task include the previously defined physical actions A_p, and the communication actions *SendEmpty*, *IncrementSortColor*, and *SendPassCount*.

Before discussing each coordination approach, we define the *SortColor* state feature in detail. Unlike previously encountered communicated features, the sorting color is a value that is common to both robots; to achieve accurate performance, the robots must maintain the same value for this feature at all times. As a result, instead of representing this value both as a local and teammate copy of the information, we use a single value for each robot. The value of this feature can be updated both locally and through communication from the teammate using the *IncrementSortColor* action. Specifically, the execution of *IncrementSortColor* by a robot first increments its local copy of the variable, and then communicates the new value to its teammate where it immediately updates the other robot's state. Additionally, this action resets the value of the *PassCount* feature to 0. In summary, this approach achieves state synchrony between robots through the use of a single feature and a multi-function communication action.

The above representation for the *SortColor* feature is used for each coordination approach applied to Task 3. Table 1 presents a summary of the complete state representations used by each learning method for this task. Note that for each approach, the information locally observed by each robot (F_o) and received from its teammate (F_t) remains the same. The differences consist in the local representation of the state features to be communicated. Below we summarize the distinguishing features of each approach.

Table 1 Representation of Task 3 using active communication, shared state and a combined approach

	Active Communication	Shared State	Combined
F_o	$\{R, G, B\}$	$\{R, G, B\}$	$\{R, G, B\}$
F_s	$\emptyset$	{*PassCount*, *Empty*}	{*Empty*}
F_c	{*PassCount*, *Empty*}	$\emptyset$	{*PassCount*}
F_e	{*PassCount*, *Empty*}	$\emptyset$	{*PassCount*}
F_t	{*PassCount*, *Empty*, *SortColor*}	{*PassCount*, *Empty*, *SortColor*}	{*PassCount*, *Empty*, *SortColor*}
A_c	{*IncSortColor* *SendPassCount*, *SendEmpty*}	{*IncSortColor*}	{*IncSortColor*, *SendPassCount*}

Active Communication—In the active communication approach, both the *PassCount* and *Empty* features are communicated explicitly using communication actions. The teacher selects the *PassCount* feature to be communicated each time its value reaches 10. This representation requires a total of 10 state features and 8 action classes.

Shared State—In the shared state approach, the values of all communicated features are shared each time their value changes. The main difference between this approach and the active communication method is that each robot always knows the exact number of passes that its teammate has performed. This approach utilizes a more compact representation, requiring 8 state features and 6 action classes.

Combined Approach—In the combined approach, active communication is used for some state features, and shared state for others. Specifically, shared state is used for the *Empty* feature, the value of which must be communicated each time it changes. Active communication is used for the *PassCount* feature, the value of which is communicated only once it passes a set threshold.[1] This representation uses a total of 9 state features and 7 action classes.

Table 2 presents the results of this experiment in terms of the number of demonstrations required to learn the task policy, and the average number of communication messages sent by each robot during the execution of this policy. We find that the shared state approach requires significantly fewer demonstrations than both active communication and

[1] Alternatively, *PassCount* could also be defined as a boolean feature representing whether enough passes have occurred or not. We do not use this representation here for evaluation purposes.

Table 2 Task 3 evaluation result summary. Comparison of the average number of demonstrations required per robot to learn the task policy, and the average number of communication messages sent by each robot while applying the final policy to sorting 10 randomly colored balls

	Number of Demonstrations	Number of Communication Messages
Active Communication	135.0 ± 5.9	5.8 ± 1.1
Shared State	52.6 ± 6.6	37.7 ± 2.6
Combined	92.2 ± 4.8	5.8 ± 1.1

the combined approach. Even in the presence of variables spanning a range of values that contain no useful information, the shared state approach requires less training data. This suggests that learning a threshold for a particular input feature (in our case *PassCount*) is easier for the underlying classifier than dealing with an additional state feature ($F_e = PassCount$) and class label (*SendPassCount*).

The number of communication messages required to perform the task is dependent upon the number, order and color of balls to be sorted. Table 2 presents the average number of communication messages required to sort 10 balls of random color and sequence. Analysis of the average number of communication messages reveals the tradeoff between the coordination approaches. The active communication approach utilizes significantly fewer communication messages than shared state since it does not communicate the value of *PassCount* each time it changes.

Additionally, this evaluation highlights the benefit of using a combined approach that takes advantage of both training methods. The combined approach provides the communication benefits of active communication by communicating *PassCount* only when necessary, while also reducing the number of demonstrations by using state sharing for features that must be communicated each time they change. The result is a low communication rate and demonstration requirements in between those of either method alone.

In summary, we find that complex domains are likely to benefit from a combination of active communication and shared state techniques if a cost is associated with communication. If communication is free, the shared state approach should be used since it results in the fewest number of demonstrations.

5 Evaluation of Scalability

We present a case study analysis the scalability of the *flexMLfD* framework. Scalability was evaluated in the beacon homing domain using 1, 3, 5, and 7 robots and communication through shared state. Through experimental evaluation, we examine how the number of robots being taught by the teacher affects the following metrics: number of demonstrations, learning time, teacher workload, teacher response time, and number of simultaneous interaction requests.

We evaluate three approaches to teaching multiple robots at the same time:

- **Synchronous Learning Start Times**—Each robot learns an individual task policy. All robots begin learning at the same time.
- **Offset Learning Start Times**—Each robot learns an individual task policy. Learning begins with a single robot. All remaining robots are introduced incrementally, one at a time, once all previous robots have gained partial autonomy at the task.
- **Common Policy Learning**—All robots begin learning at the same time and learn a single common policy by consolidating their knowledge and sharing demonstration data.

In each approach, robots begin with no knowledge about the task, and learning is complete once all robots are able to perform the task correctly. The synchronous and offset learning approaches examine the most general learning scenario, one in which each robot learns an individual policy, allowing different tasks or roles to be taught at the same time. In the synchronous learning approach, all robots begin at the same time. As we show in our evaluation, this approach places great demand on the teacher for demonstrations early in the training process. The offset learning approach presents an alternate method in which robots are introduced one at a time, after all active learners have already gained partial autonomy at the task, thereby helping to disperse the demand for demonstrations over a longer time period. Common policy learning examines a special case of demonstration learning in which robots consolidate all their demonstration knowledge to learn a single common policy. We evaluate the scalability of our multi-robot learning framework in a beacon homing domain using Sony AIBO robots.

All evaluations were performed with a single teacher. As with all human user trials, we must account for the fact that the human teacher also learns and adapts over the course of the evaluation. To counter this effect, the teacher performed a practice run of each experiment, which we then discarded from the evaluation. Results averaged over three additional trials are presented. An alternate evaluation method would be to eliminate the human factor by using a standard controller to respond to all demonstration requests in a consistent manner. This approach, however, would prevent us from evaluating the demands multiple robots place on a human teacher.

5.1 Synchronous Learning Start Times

First, we present a detailed evaluation of the scalability of the synchronous learning start times approach in which all

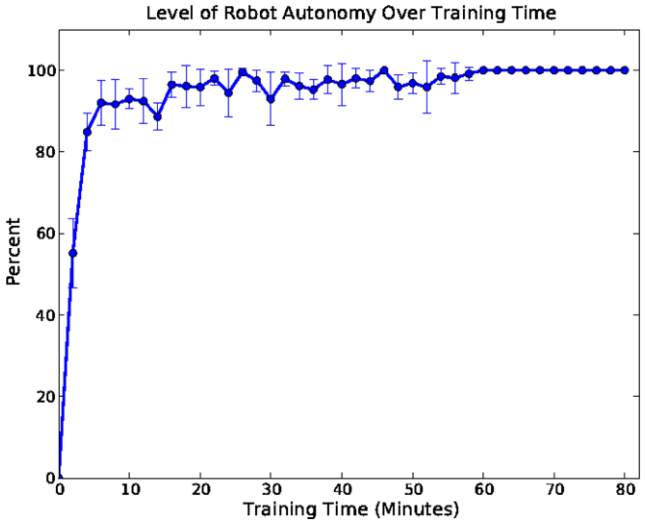


Fig. 7 Average level of autonomy of a single robot over the course of training

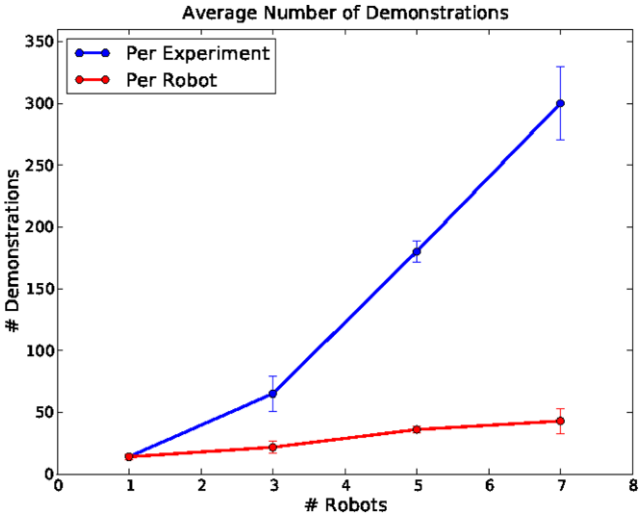


Fig. 8 Average number of demonstrations performed by the teacher for each robot and in each experiment

robots begin learning at the same time. In Sects. 5.2 and 5.3, we compare the performance of the other two techniques—offset learning start times and common policy learning—in the 7-robot beacon homing domain. We begin by presenting the typical autonomy curve for CBA-based learning methods.

5.1.1 Robot Autonomy

Figure 7 shows how the level of autonomy, which is measured as the percentage of autonomous actions versus demonstrations, changes for an individual robot over the course of training. The data presents the average autonomy of robots in the 5-robot beacon homing experiment. The shape of the curve is typical of CBA learning, in which robots begin with no initial knowledge about the task and request many demonstrations early in the training process. These initial demonstrations provide the robot with the experience for handling most commonly encountered domain states. As a result, following the initial burst of demonstration requests, the robot quickly achieves 80–95% autonomous execution. The remainder of the training process then focuses on refining the policy and addressing previously unencountered states. The duration of this learning time is dependent upon the frequency with which novel and unusual states are encountered.

5.1.2 Number of Demonstrations

Figure 8 shows how the number of demonstrations performed by the teacher on average for each robot, and in total for each experiment, changes with respect to the number of robots. As the number of robots grows, we observe a slight increase in the number of demonstrations required per robot. This increase is due to the fact that, although the number of state features in the representation of our domain does not change, the *range* of possible feature values does.

Specifically, in an N-robot experiment, the value of features representing the number of robots located at a beacon, n_i, has the range $[0, N]$. As a result, extra demonstrations are required in the presence of a greater number of robots to provide guidance in the additional states. While similar effects are present in many domain representations, state features can often be designed or modified in such a way that their range is independent of factors such as the number of robots. For example, in the beacon homing domain this could be achieved by converting n_i to a boolean feature that indicates whether the beacon's capacity has been reached or not.

The total number of demonstrations required for each experiment grows at a nearly linear rate with respect to the number of robots. The overall number of demonstrations that the teacher must perform has a significant effect on the overall training time, as discussed in the next section. Seven robots require a total of approximately 300 demonstrations to learn the task.

5.1.3 Training Time

Figure 9 presents the change in the overall experiment training time with respect to the number of robots. Importantly for the scalability of *flexMLfD*, the data shows a strongly linear trend, with seven robots requiring just over 1.5 hours to train. This result is significant as it suggests that this approach will continue to scale linearly with the number of robots if the size of the state space remains the same. In the following sections, we examine the factors that contribute to the training time, such as the number of demonstrations and demonstration delay.

Fig. 9 Total training time with respect to number of robots

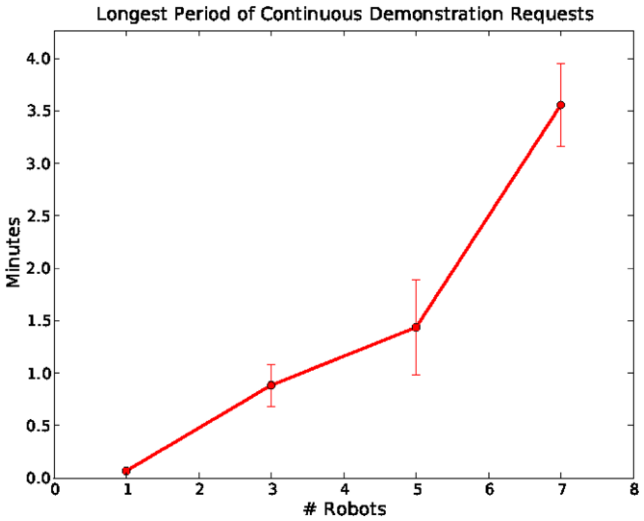


Fig. 10 Attention demand on the teacher

5.1.4 Teacher Workload

In addition to the overall training time and number of demonstrations, it is important to understand the demands that multiple robots place on the teacher. The teacher experiences the greatest number of demonstration requests during the earliest stages of learning, possibly from multiple robots at the same time. As the level of autonomy of the robots increases over time, the teacher receives fewer requests, until finally all robots become fully autonomous. To evaluate the demand on the teacher's attention during this most laborious training segment, we calculate the longest continuous period of time during which the teacher has at least one demonstration request pending. This value provides insight into the degree of effort that is required from the teacher.

Figure 10 plots the duration of the longest continuous period of demonstration requests for each experiment. The data shows that the length of this time period grows quickly, possibly exponentially, with the number of robots. In experiments with only a single robot, demonstration requests last only a few seconds at a time; as soon as the teacher responds to the request, the robot switches to performing the demonstrated action. As the number of robots increases, however, so does the number of simultaneous requests from multiple robots. In the 7-robot experiment, this results in a 3.5 minute uninterrupted segment of demonstration requests for the teacher. In Sect. 5.2, we examine an alternate approach to teaching multiple robots in which novice learners are added incrementally, reducing the demand for demonstrations.

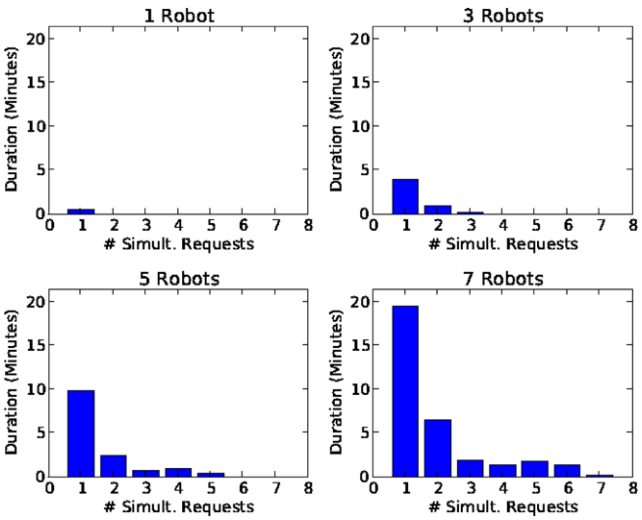


Fig. 11 Histograms showing the distribution of the number of simultaneous requests for each experiment

Additionally, we examine the total time per experiment that multiple demonstration requests are pending. Figure 11 presents a set of bar graphs showing the distribution of the number of simultaneous requests for each experiment. This data indicates that for all experiments, the teacher spends the greatest percentage of time with only a single demonstration request. However, the teacher spends over 3 minutes in the 5-robot experiment, and over 13 minutes in the 7-robot experiment, faced with multiple queries. This growing number of simultaneous queries has a significant impact on demonstration delay, the amount of time that passes between the robot's initial request and the teacher's response.

5.1.5 Teacher Response Time

As discussed in the previous section, multiple simultaneous demonstration requests become common as the number of robots increases. As a result, robots often must wait while the teacher responds to other robots. Figure 12(a) shows that the average time a robot spends waiting for a demonstration grows with the number of learners from only 2 seconds for a single robot to 12 seconds for seven robots.

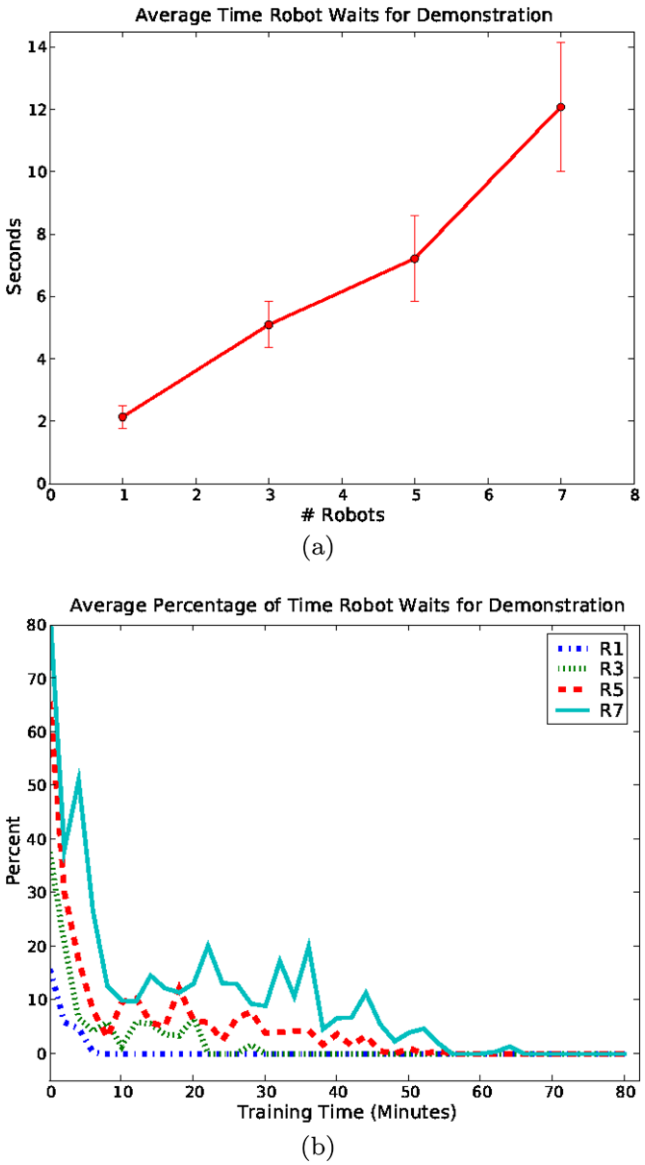


Fig. 12 (**a**) Average amount of time a robot spends waiting for a demonstration response from the teacher. (**b**) Average percentage of time a robot spends waiting for a demonstration over the course of training

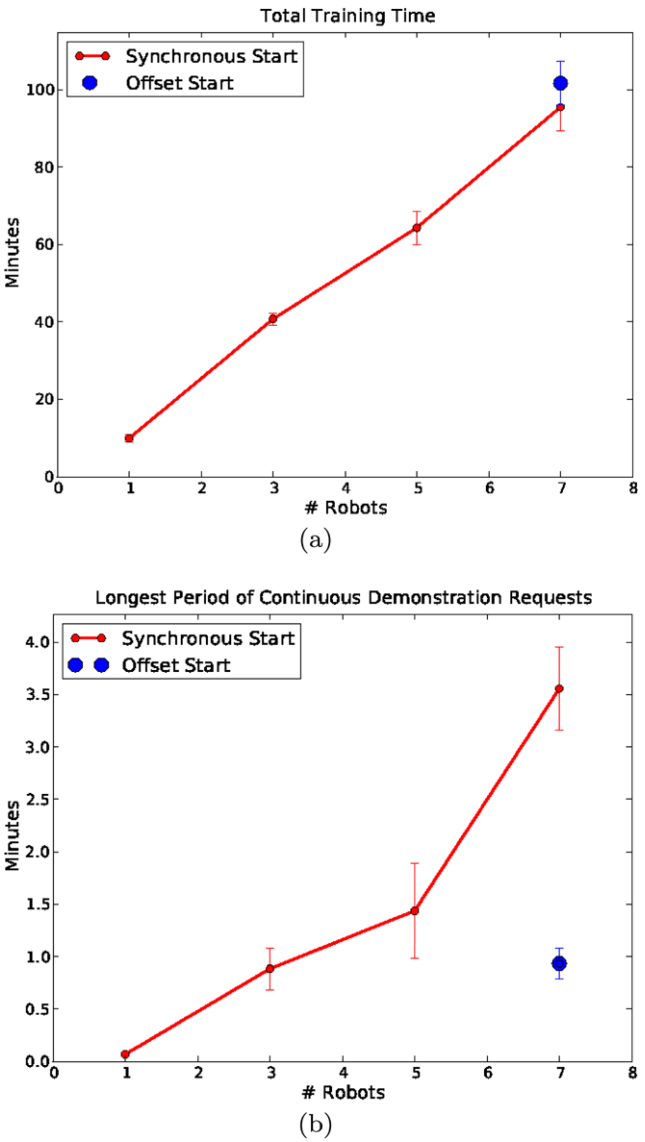


Fig. 13 Comparison of the synchronous and offset learning start time approaches. (**a**) Total training time. (**b**) Attention demand on the teacher

Figure 12(b) plots the percentage of time a robot spends waiting on average for a demonstration over the course of training. Not surprisingly, we observe that the demonstration delay is greatest early in the training process when the teacher is most busy with initial demonstration requests. Staggering the times at which novice robots are introduced to the task may help to alleviate this problem by reducing the demand of the initial training phase on the teacher.

5.2 Offset Learning Start Times

Analysis of the synchronous start times approach presented in the previous section shows that when multiple robots begin learning a new task at the same time, they place great demand on the teacher for demonstrations early in the training process. This, in turn, leads to longer demonstration delays for each robot. In this section, we examine an alternate approach in which novice robots are introduced incrementally. Each new robot is added once all previous learners have gained proficiency at the task and are able to act autonomously approximately 80% of the time. We refer to this learning process as *offset start time* learning.

We compare the performance of offset start time learning to the synchronous start time approach in the 7-robot beacon homing domain. Note that as we add each additional robot learner, the range of domain states experienced by previously active robots expands. For example, a single robot alone can be taught to navigate to various beacons, but with-

out the presence of other robots it will never learn the collaborative aspects of the task. As each additional robot is introduced to the domain, the robot will learn how to act in the presence of others.

Figure 13 shows the effect of offset start time learning on total training time and attention demand on the teacher. Figure 13(a) shows that by offsetting the learning start time of the robots, the total learning time of the experiment increases by approximately 12%. This increase is due to the fact that robots experience novel states over a longer period of time. Since the task itself remains unchanged, the total number of demonstrations required to teach all seven robots their individual policies remains approximately the same (284 demonstrations using offset start times compared to 298 using the synchronous start times approach). However, since the demonstrations are disbursed over a longer period of time, the attention demand on the teacher is significantly reduced, as shown in Fig. 13(b). The longest period of continuous demonstration requests falls to approximately 1 minute, compared to 3.5 minutes in the synchronous learning start case, and the average time a robot waits to receive a demonstration falls from 12 seconds to 4.5 seconds.

In summary, by offsetting the learning start times of the robots we are able to distribute demonstration requests over a longer time period. This approach leads to a slight increase in the overall learning time, but reduces the number of simultaneous demonstration requests the teacher receives, which in turn leads to faster response times, reducing the demonstration delay.

5.3 Common Policy Learning

In the standard formulation of the *flexMLfD* learning approach, the teacher provides each robot with an individual set of demonstrations from which a unique policy is derived. This generalized approach is highly suitable for domains in which robots perform different roles and functions. However, in some domains, as in our experiments, the same behavior may be desirable for each robot. In such cases, teaching the same policy to multiple robots results in a large number of redundant demonstrations. To address this case, we propose that all robots learning the same task learn a single, *common*, policy by consolidating all demonstration data. The sharing of information can occur by collecting all data within a single dataset, or by freely exchanging each demonstration among all robots so as to maintain a distributed set of identical policies. In this section, we evaluate the performance of the common policy approach using the 7-robot beacon homing domain.

Note that common policy learning in domains independent robots, results in the same final policy as if each of those robots was taught alone. Using multiple robots in this case may speed up learning since uncommon states are more likely to be encountered with many learners. Common policy learning in domains with non-independent robots, as in our example of the beacon homing domain, differs from training and replicating a single robot policy as it additionally allows the robots to experience the collaborative aspects of the task.

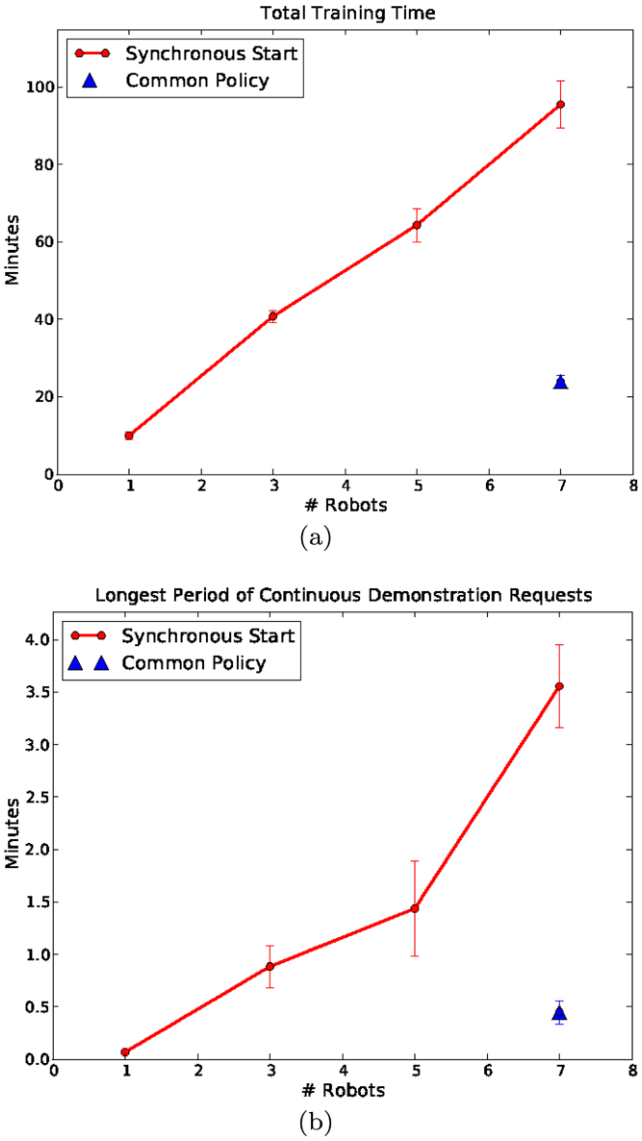


Fig. 14 Comparison of the single common policy and multiple individual policy *flexMLfD* learning approaches. (**a**) Total training time. (**b**) Attention demand on the teacher

Using this technique, the teacher was required to perform a total of only 44 demonstrations, compared to nearly 300 total demonstrations previously required for this task. Figure 14(a) shows that the overall training time of the experiment is similarly reduced to only 23 minutes, compared to the 95 minutes for the standard approach. In fact, while seven robots learning a common policy require more of the

teacher's time than a single learner, they require less time than three robots learning individual policies.

Figure 14(b) shows that a common policy similarly reduces the attention demand that seven robots require of the teacher. This effect can be attributed to the fact that frequently a demonstration performed for one robot addresses the queries of other currently waiting robots. An additional effect of this occurrence is that the average waiting time of the common policy approach is reduced from 12 seconds to 1.2 seconds.

5.4 Discussion

In summary, our scalability case study demonstrates the first example of a single person training up to seven independent robots. Our analysis has shown promising trends with respect to scalability, but further work is necessary to continue to reduce the overall learning time. The impact of multiple robots on the teacher is a particularly important factor to consider in future research, both in terms of effect on the maximum number of robots that can be taught by a single person, as well as overall task performance. Our experiments found that increasing the number of robots significantly increased the workload of the teacher, as measured by the number of pending demonstration requests. This, in turn, impacted demonstration delay and the time robots spent waiting for the teacher's response. While this has no negative impact on learning in the presented domain, delay may impact performance in other tasks. Additional research is also needed to determine what impact state representation, action duration, and degree of collaboration between robots have on learning performance and scalability.

6 Multi-Robot Learning from Demonstration

In this article, we have examined the problem of teaching collaborative policies to multiple robots by a single teacher. In this section, we consider a broader definition for multi-robot learning from demonstration as a new research area in which we expand the problem definition to include multiple teachers, as well as the ability for robots to teach one another. A number of projects on software agents have already explored research in this direction, such as communities of social agents that are able to observe each other's actions or exchange advice [14, 35, 40].

In this section, we propose a formal definition for the broad multi-robot learning from demonstration problem, with the goal of providing a foundational structure for future work in this area. We then discuss the open research questions of MLfD in the context of already established fields and propose a range of evaluation metrics.

6.1 Problem Definition

Formally, we define the MLfD problem as follows. The world consists of the set R of robots, and the set T of teachers. Each robot $r \in R$ observes the world as a set of states S_r. The robot's actions, a, are bound to a finite set A_r of action primitives, which are the basic actions that can be combined to perform the overall task. The set of states S_r and actions A_r may be unique to each robot.

During the course of training, each robot r incrementally accumulates an independent set of demonstrations, D_r, consisting of j_r demonstration sequences:

$$d_m^t = \{(s^i, a^i), s^i \in S_r, a^i \in A_r, i = 0 \ldots k_m, t \in \{T \cup R\}\}$$

for $m = 0, \ldots, j_r$. Each demonstration sequence contains a temporally sequential series of k_m state-action pairs representing the demonstration performed by a teacher t, where t could be either a human or robot teacher, or another robot. Given the set of demonstrations, the robot's goal is to learn to imitate the teacher's behavior by generalizing from the demonstrations and learning the policy $\pi_r : S_r \rightarrow A_r$ mapping from all possible states to actions in A_r.

6.2 Context and Design Choices

Multi-robot learning from demonstration combines elements of three research fields: robot learning from demonstration, multi-robot systems, and human–robot interaction (HRI). We have chosen, therefore, to analyse MLfD in the context of these fields as a way of categorizing the factors and open challenges that must be considered in designing an MLfD system.

6.2.1 Robot Learning from Demonstration

Single–robot learning from demonstration algorithms may form a natural basis for MLfD approaches. However, with the exception of work presented in this article, all LfD approaches developed to date have been designed for single-robot applications, and have relied on close one-to-one interaction between the robot and the teacher. As a result, these techniques do not scale to multi-robot domains due to the problem of *limited human attention*—the fact that the teacher is not able to pay attention to, and interact with, all robots at the same time.

However, most of the common design choices and challenges of single-robot LfD extend to the multi-robot case. LfD algorithms can be analyzed along two dimensions: the demonstration technique used to gather data, and the learning method used to learn a policy [2]. Many demonstration techniques have been shown to be successful for different applications, including teleoperation [23], kinesthetic teaching [8], and external observation of the teacher [31].

Once training data is collected, algorithms further differ by what algorithm is applied to learn an action policy. Learning methods vary from exploration-based methods, such as reinforcement learning [3, 46], to classification [13] and regression [7].

Additionally, a broad range of other factors must be considered in designing an MLfD system. Is learning performed online or as batch learning? Is learning interactive, allowing the robot to provide feedback or ask for help? Does the robot have the ability to perform self evaluation in order to learn independently, or in order to provide feedback about its confidence in certain tasks? Can the robot expand its state representation and gather data about the world through techniques such as symbol grounding [33]? Does the robot have varying degrees of autonomy? Can the robot learn independently and surpass the performance of its teacher?

Each of the above questions represents an open research problem, and we expect that no single right answer exists to satisfy the specific needs of all applications [2, 32, 44].

6.2.2 *Multi-Robot Systems*

The problems of management, coordination, and control of multiple robots have been extensively studied within the multi-robot systems community [16, 26, 49], and potential transfer exists from these to MLfD. One of the most important factors to consider in designing a multi-robot system is the issue of communication. How does the teacher communicate with each robot? Can the teacher communicate with the group as a whole or with individuals? Do robots have the ability to communicate with each other? Can robots observe and imitate each other's actions [1]? Can robots request demonstrations from and provide aid to their peers [35]?

6.2.3 *Human–Robot Interaction*

Human–robot interaction [21] is a critical component of any demonstration learning system, and a number of important questions need to be considered in the development process. How is the robot's state information presented to the teacher? For example, does the teacher observe the complete internal state of the robot, or is the teacher limited to external visual observation? What means does the teacher use to communicate information back to the robot? Can the robot provide feedback beyond state information to the teacher, such as demonstration requests or indicators of confidence? Is the robot able to recognize and interpret human social cues?

6.3 Evaluation Metrics

Common evaluation metrics are invaluable to any research field to facilitate knowledge transfer through comparison and benchmarking. A major challenge of defining metrics for MLfD, and the reason for the relative absence of standard evaluation techniques in single-robot LfD, is the broad diversity of applications and demonstration styles. Differences in data representation, demonstration and interaction methods, as well as the interactive nature of the learning process, make it difficult to perform broad comparisons between algorithms. Researchers within LfD thus traditionally rely on task-specific metrics, such as the *effectiveness* and *efficiency* of the performance of a particular task, as well as the *number of demonstrations* and overall *learning time*.

While it may not be possible to define evaluation metrics that will work across all algorithms, here we attempt to identify metrics targeted at specific aspects of the MLfD problem based on established metrics from the multi-robot systems and HRI communities. This list is far from exhaustive, but is meant to be a starting point to promote further discussion within the community.

6.3.1 *Multi-Robot Systems*

Much research on multi-robot systems has focused on the problems of multi-robot teaming and the management of multiple robots by human operators [15, 34, 50]. This has led to the introduction of metrics such as *fan-out*, defined as the upper bound on the number of independent, homogeneous robots that can be managed by a single person [22]. Within MLfD, we generalize fan-out to be the number of independent robots that can be managed and taught by a single teacher. Factors such as *neglect tolerance*, the robot's robustness to operating without close operator supervision, demonstration technique and levels of autonomy all affect fan-out.

Measures with respect to the teacher's cognitive workload of managing multiple robots must also be taken into account. In the context of interactive MLfD algorithms, the *number of simultaneous interaction requests* to the teacher is an important metric to consider in determining teacher workload and calculating factors such as the fan-out.

Another useful metric to consider is *teacher response time*, or *intervention response time* [19], which we defined as the delay between the time that a robot encounters a problem and when the human intervenes. This metric can be directly applicable to MLfD, with intervention ranging from the physical takeover of robot control by the teacher to teacher guidance or advice.

6.3.2 *Human–Robot Interaction*

Human–robot interaction research promotes standard evaluation metrics [48], many of which may be inherently important in MLfD. One of the most commonly used metrics in HRI are *subjective ratings*, such as the Lickert scale [30],

that can be used to provide common scoring strategies for vastly different approaches.

Another important evaluation is with regard to the appropriate level of autonomy. An increasing number of robotic systems are being designed to allow varying degrees of robot autonomy, both as a means to regulate learning [13] and to divide a task into parts better performed by a robot or a human operator [47]. Evaluations measuring *level of autonomy discrepancies* provide information as to whether the appropriate level of autonomy is used for different elements of the task. Closely related is the metric of *appropriate utilization of mixed initiative*, which is defined as the robot's ability to effectively regulate who has greater control over the robot's actions [19].

Finally, HRI metrics can be used to evaluate the demands on the human teacher in order to facilitate the development of improved interaction interfaces and demonstration methods. Methods for evaluating *situation awareness*, such as the Situation Awareness Global Assessment Technique (SAGAT) [17], can aid in assessing teacher performance and workload levels and their impact on decision-making [45]. Social metrics, such as the degree of the operator's *trust* in the robot, can further impact both training performance, as well as the potential for adaptability of a technology by future users [29].

7 Conclusion

In this article, we presented *flexMLfD*, the first multi-robot demonstration learning framework. Our approach is based on the Confidence-Based Autonomy algorithm, which establishes a general state and action representation and provides a means for single-robot policy learning through adjustable autonomy. Using an independent instance of the CBA algorithm, each robot acquires its own set of demonstrations and learns an individual task policy. The unique feature of the CBA algorithm that enables it to be applied to multi-robot learning is the Confident Execution component, which enables each robot to regulate its own autonomy, and to pause execution when faced with uncertain situations.

Using the *flexMLfD* framework, we examined techniques for teaching emergent multi-robot coordination, in which the solution to the shared multi-robot task emerges from the complementary actions performed by robots based on their independent policies. We contributed three approaches to teaching collaborative behavior based on different information sharing strategies: implicit coordination, coordination through active communication, and coordination through shared state.

Additionally, we contributed a case study analysis of the scalability of the *flexMLfD* learning framework in a beacon homing domain using up to seven AIBO robots. We believe this is the first experiment examining demonstration learning at this scale. We compared three approaches to teaching multiple robots at the same time: synchronous learning start times, offset learning start times, and common policy learning. Learning performance was evaluated with regard to the number of demonstrations required to learn the task, the demands for time and attention placed on the teacher, and the delay that each robot experiences in obtaining a demonstration.

Finally, we contributed a formalization of a broader multi-robot learning from demonstration problem, which we hope will serve as a starting point for further research in this area.

Acknowledgements We thank Sony for their generous four-year loan of the QRIO robots. This research was sponsored by DARPA - BBNT Solutions under grant number 9500007812 and 9500008572, Department of the Interior under grant number NBCH1040007, L3 Communication Integrated Systems, L.D., under grant number 4500244745, SRI International under grant number 03-000211, and Lockheed Martin Corporation under grant number 8100001629. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the social policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

1. Alissandrakis A, Nehaniv CL, Dautenhahn K (2002) Do as i do: Correspondences across different robotic embodiments. In: Kim J, Polani D, Martinetz T (eds) Fifth German workshop on artificial life (GWAL5), pp 143–152
2. Argall B, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483
3. Atkeson CG, Schaal S (1997) Robot learning from demonstration. In: Fisher DH Jr (ed) Machine learning: proceedings of the fourteenth international conference (ICML'97). San Francisco, California, pp 12–20
4. Balch T, Arkin RC (1994) Communication in reactive multiagent robotic systems. Auton Robots 1(1):27–52
5. Bentivegna DC, Ude A, Atkeson CG, Cheng G (2004) Learning to act from observation and practice. Int J Humanoid Robot 1(4)
6. Breazeal C, Hoffman G, Lockerd A (2004) Teaching and working with robots as a collaboration. In: AAMAS '04: Proceedings of the third international joint conference on autonomous agents and multiagent systems. IEEE Computer Society, Washington, DC, pp 1030–1037
7. Browning B, Xu L, Veloso M (2004) Skill acquisition and use for a dynamically-balancing soccer robot. In: Proceedings of nineteenth national conference on artificial intelligence (AAAI'04)
8. Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: Second annual conference on human-robot interactions (HRI'07). Arlington, Virginia, March 2007
9. Chaimowicz L, Campos MFM, Kumar V (2002) Dynamic role assignment for cooperative robots. In: Proc. of the IEEE intl. conf. on robotics and automation (ICRA), pp 293–298
10. Chernova S (2009) Confidence-based robot policy learning from demonstration. PhD thesis, Computer Science Dept., Carnegie Mellon University, Advisor-Manuela Veloso

11. Chernova S, Veloso M (2008) Multi-thresholded approach to demonstration selection for interactive robot learning. In: Proceedings of 3rd ACM/IEEE international conference on human-robot interaction (HRI'08), March 2008
12. Chernova S, Veloso M (2008) Teaching multi-robot coordination using demonstration of communication and state sharing (short paper). In: Proceedings of the international conference on autonomous agents and multiagent systems (AMMAS '08), May 2008
13. Chernova S, Veloso M (2009) Interactive policy learning through confidence-based autonomy. J Artif Intell Res 34(1):1–25
14. Clouse JA (1996) On integrating apprentice learning and reinforcement learning. PhD thesis, University of Massachusetts, Department of Computer Science. Director-Paul E Utgoff
15. Crandall JW, Goodrich MA, Olsen DR Jr, Nielsen, CW (2005) Validating human-robot interaction schemes in multitasking environments. IEEE Trans Syst Man Cybern A 35(4):438–449
16. Dias MB, Zlot R, Kalra N, Stentz A (2006) Market-based multirobot coordination: A survey and analysis. Proc IEEE 94(7):1257–1270
17. Endsley MR, Garland DJ (2000) Situation awareness: analysis and measurement. Lawrence Erlbaum Associates
18. Farinelli A, Farinelli R, Iocchi L, Nardi D (2004) Multi-robot systems: A classification focused on coordination. IEEE Trans Syst Man Cybern B 34:2015–2028
19. Fong TW, Thorpe C, Baur C (2003) Robot, asker of questions. In: Robotics and autonomous systems
20. Gerkey BP, Mataric MJ (2000) Principled communication for dynamic multi-robot task allocation. In: Experimental robotics VII. LNCIS, vol 271. Springer, Berlin, pp 353–362
21. Goodrich MA, Schultz AC (2007) Human-robot interaction: a survey. Found Trends Hum Comput Interact 1(3):203–275
22. Goodrich MA, Olsen DR Jr (2003) Seven principles of efficient human robot interaction. In: Proc IEEE Int Conf Syst, Man and Cybernetics, vol 4, pp 3942–3948
23. Grollman SH, Jenkins OC (2007) Dogged learning for robots. In: Proceedings of the IEEE international conference on robotics and automation (ICRA'07), Roma, Italy
24. Guenter F, Hersch M, Calinon S, Billard A (2007) Reinforcement learning for imitating constrained reaching movements. RSJ Adv Robot 21(13):1521–1544 (Special issue on imitative robots)
25. Hersch M, Guenter F, Calinon S, Billard A (2008) Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Trans Robot 24(6):1463–1467
26. Jan't Hoen P, Tuyls K, Panait L, Luke S, La Poutré JA (2005) An overview of cooperative and competitive multiagent learning. In: LAMAS, pp 1–46
27. Jones C, Shell D, Matarić M, Gerkey B (2004) Principled approaches to the design of multi-robot systems. In: IEEE/RSJ intl conf on intelligent robots and systems, workshop on networked robotics
28. Kube RC, Zhang H (1997) Task modelling in collective robotics. Auton Robots 4(1):53–72
29. Lee JD, See KA (2004) Trust in automation: designing for appropriate reliance. Hum Factors 46:50–80
30. Likert R (1932) A technique for the measurement of attitudes. In: Archives of psychology, pp 1–55
31. Lockerd A, Breazeal C (2004) Tutelage and socially guided robot learning. In: IEEE/RSJ international conference on intelligent robots and systems
32. Mataric MJ (2002) Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. In: Dautenhahn K, Nehaniv C (eds) Imitation in animals and artifacts. MIT Press, Cambridge, pp 392–422
33. Mayo M (2003) Symbol grounding and its implications for artificial intelligence. In: Oudshoorn MJ (ed) Twenty-sixth australasian computer science conference (ACSC2003), CRPIT, vol 16. Adelaide, Australia, ACS, pp 55–60
34. Nielsen CW, Few DA, Athey DS (2008) Using mixed-initiative human-robot interaction to bound performance in a search task. In: international conference on intelligent sensors, sensor networks and information processing. ISSNIP 2008, pp 195–200
35. Oliveira E, Nunes L (2004) Learning by exchanging Advice. Springer, Berlin
36. Ossowski S, Menezes R (2006) On coordination and its significance to distributed and multi-agent systems: Research articles. Concurr Comput Pract Exper 18(4):359–370
37. Pagello E, D'Angelo A, Montesello F, Garelli F, Ferrari C (1999) Cooperative behaviors in multi-robot systems through implicit communication. Robot Auton Syst 29(1):65–77
38. Peters J, Vijayakumar S, Schaal S (2003) Reinforcement learning for humanoid robotics. In: IEEE-RAS international conference on humanoid robots, pp 1–20
39. Pollard N, Hodgins JK (2002) Generalizing demonstrated manipulation tasks. In Workshop on the algorithmic foundations of robotics, December 2002
40. Price B, Boutilier C (2003) Accelerating reinforcement learning through implicit imitation. J Artif Intell Res 19:569–629
41. Roth M, Vail D, Veloso M (2003) A real-time world model for multi-robot teams with high-latency communication. In: IEEE/RSJ international conference on intelligent robots and systems, vol 3. pp 2494–2499
42. Rybski PE, Yoon K, Stolarz J, Veloso MM (2007) Interactive robot task training through dialog and demonstration. In: HRI'07: Proceedings of the ACM/IEEE international conference on human-robot interaction. ACM Press, New York, pp 49–56
43. Saunders J, Nehaniv CL, Dautenhahn K (2006) Teaching robots by moulding behavior and scaffolding the environment. In: HRI '06: proceeding of the 1st ACM SIGCHI/SIGART conference on human-robot interaction. ACM Press, New York, pp 118–125
44. Schaal S, Ijspeert A, Billard A (2003) Computational approaches to motor learning by imitation. Philos Trans R Soc Lond, B, Biol Sci 358:537–547
45. Scholtz J, Antonishek B, Young J (2004) Evaluation of a human-robot interface: Development of a situational awareness methodology. In: HICSS '04: Proceedings of the 37th annual Hawaii international conference on system sciences (HICSS'04)—Track 5, IEEE Computer Society, Washington, DC p 50130.3
46. Smart WD, Kaelbling LP (2002) Effective reinforcement learning for mobile robots. In: IEEE international conference on robotics and automation
47. Steinfeld A (2004) Interface lessons for fully and semi-autonomous mobile robots. In: IEEE international conference on robotics and automation
48. Steinfeld A, Fong T, Kaber D, Lewis M, Scholtz J, Schultz A, Goodrich M (2006) Common metrics for human-robot interaction. In: 1st annual conference on human-robot interaction, Salt Lake City, Utah
49. Stone P, Veloso M (2000) Multiagent systems: A survey from a machine learning perspective. Auton Robots 8(3):345–383
50. Wang J, Lewis M (2007) Human control for cooperating robot teams. In HRI '07: Proceedings of the ACM/IEEE international conference on human-robot interaction, New York, NY, USA, pp 9–16

Sonia Chernova is a postdoctoral associate in the Personal Robots Group at the MIT Media Lab and an adjunct assistant professor at Worcester Polytechnic Institute. She received her Ph.D. from the Computer Science Department at Carnegie Mellon University in 2009, where this work was performed. Her research interests lie in social and interactive robot systems, with a focus on policy learning, human–robot interaction and adjustable autonomy.

Manuela Veloso is the Herbert A. Simon Professor of Computer, 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 (www.cs.cmu.edu/~coral). 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". With her students, Professor Veloso has created many successful teams of RoboCup robot soccer players, and made contributions to efficient distributed perception, automated planning, and robot teamwork.