A Confidence-Based Approach to Multi-Robot Learning from Demonstration

Sonia Chernova and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA USA

Abstract

This paper presents an overview of a series of projects exploring multi-robot learning from demonstration. We present *flexMLfD*, a robot independent and task independent demonstration learning system that supports a variable number of robot learners. This learning system has been fully implemented and tested, and we present three example domains, utilizing different robotic platforms, to which it has been applied. Additionally, we present scalability analysis, using up to seven real robots, examining 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.

Introduction

One of the central goals of robotics research is to create robots that are able to assist humans in work environments and engage in close interaction with families in homes. As robots increasingly become a part of people's everyday lives, methods for developing new robot behaviors in a natural and intuitive way that is accessible to non-programmers are required. Inspired by the way humans and animals teach each other, *Learning from Demonstration (LfD)* provides an intuitive interface for robot programming based on human-robot interaction. In this learning approach, a teacher, typically a human, performs demonstrations of the desired behavior to the robot. The robot records the demonstrations as sequences of state-action pairs, which it then uses to learn a policy that reproduces the observed behavior.

LfD has been gaining widespread attention in the robotics community, and recent work has led to the development of a wide variety of *single-robot demonstration learning* algorithms. Proposed techniques span a wide range of policy learning methods, such as reinforcement learning (Smart and Kaelbling 2002), classification (Saunders, Nehaniv, and Dautenhahn 2006) and regression (Bentivegna 2004; Grollman and Jenkins 2007), and utilize a variety of interaction methods, including natural language (Lockerd and Breazeal 2004), gestures (Steil et al. 2004), joysticking (Grollman and Jenkins 2007) and observations of human demonstrators (Nicolescu and Mataric 2003; Calinon and Billard 2007).

Despite many variations, a unifying feature of these approaches is that they are designed for a single learner, frequently relying on close one-to-one interaction between the robot and teacher. However, many robotic tasks require the collaboration of multiple robots. In such domains, utilizing an individual teacher for each robot is inefficient and impractical in many real-world settings. A natural question therefore arises about the feasibility of a single person teaching independent, and possibly unique, policies to multiple robots at the same time. We refer to this policy learning method as *multi-robot learning from demonstration (MLfD)*.

Our research has explored multi-robot demonstration learning in a series of projects, and in this paper we present a condensed overview of our work. We present *flexMLfD*, the first task-independent and robot-independent control interface for multi-robot demonstration learning. Our approach is based on the Confidence-Based Autonomy (CBA) single-robot algorithm (Chernova and Veloso 2009), which enables a robot to learn a task policy through interaction with a human teacher. The generalized representation and adjustable robot autonomy provided by the CBA algorithm enable the flexible system design and multi-robot learning capabilities of *flexMLfD*. Our multi-robot system includes a robot-independent and task-independent modular software architecture for robot learning, interaction, and control, which can be applied to a single or multiple, independent or collaborative, robot learners.

In the following section, we present an overview of the CBA algorithm, followed by a description of the *flexMLfD* learning system. *flexMLfD* has been fully implemented and tested using multiple real-world domains and robotic platforms. In this paper we present examples of three real-world tasks that have been successfully learned using this system. Additionally, we present a scalability analysis, using up to seven robots, examining 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.

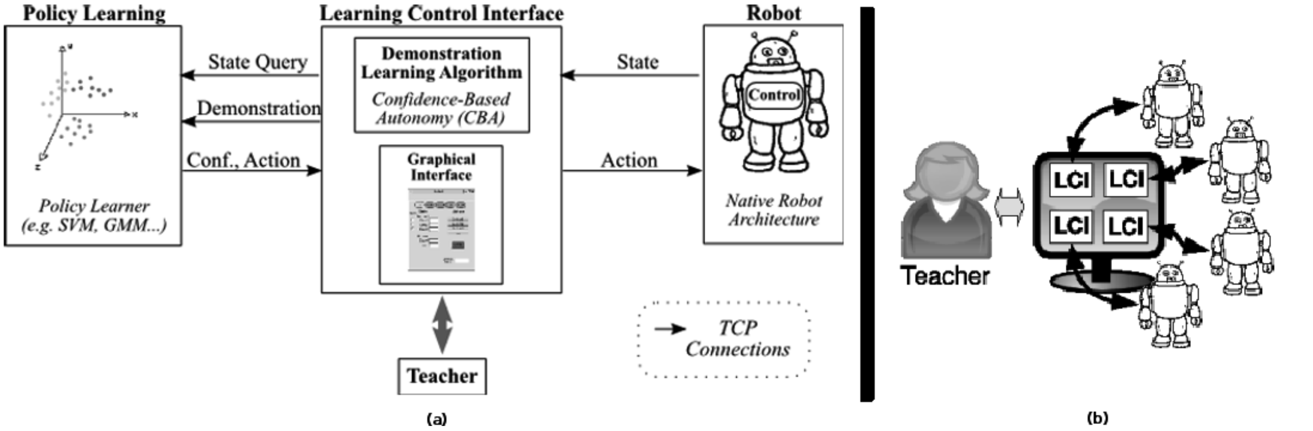

Policy Learning
Learning Control Interface
Robot
State Query
Demonstration
Conf., Action
Demonstration Learning Algorithm
Confidence-Based Autonomy (CBA)
Graphical Interface
State
Action
Control
Policy Learner (e.g. SVM, GMM...)
Native Robot Architecture
Teacher
TCP Connections
(a)
Teacher
LCI
LCI
LCI
LCI
(b)

Figure 1: Overview of the *flexMLfD* demonstration learning system: (a) single robot learning architecture (b) high-level overview of multi-robot learning.

Single-Robot Demonstration Learning

In this section, we present a summary of the CBA demonstration learning algorithm that lies at the heart of the *flexMLfD* learning system. For full details and evaluation of CBA, please see (Chernova and Veloso 2009).

Confidence-Based Autonomy is a single-robot algorithm that enables a robot to learn a policy through interaction with a human teacher. In this learning approach, the robot begins with no initial knowledge and learns a policy incrementally through demonstrations acquired as it practices the task.

Each demonstration is represented by a state-action pair, (s, a), symbolizing the correct action to perform in a particular state. The robot's state s is represented using an n-dimensional feature vector that can be composed of continuous or discrete values. The robot's actions are bound to a finite set $a \in \mathcal{A}$ of action primitives, which are the basic actions that can be combined to perform the overall task. The goal is for the robot to learn to imitate the demonstrated behavior by learning a policy mapping states s_i to actions in $\mathcal{A}$. The policy is learned using supervised learning and is represented by classifier $\mathcal{C} : s \rightarrow (a, c)$, trained using state vectors s_i as inputs, and actions a_i as labels. For each classification query, the model returns the highest confidence action $a \in \mathcal{A}$ and action-selection confidence c. CBA can be combined with any supervised learning algorithm that provides a measure of confidence in its classification.

The most important element of the CBA algorithm is the method for obtaining demonstration examples, which consists of the following two components:

Confident Execution This algorithm *enables the robot to select demonstrations* in real time as it interacts with the environment, targeting states that are unfamiliar or in which the current policy action is uncertain. At each timestep, the algorithm evaluates the robot's current state and actively decides between autonomously executing the action selected by its policy and requesting an additional demonstration from the human teacher. Demonstrations are selected based on the action selection confidence of classifier $\mathcal{C}$.

Corrective Demonstration This algorithm *enables the teacher to correct the robot's mistakes by performing additional demonstrations*. If an incorrect action is selected for autonomous execution by the Confident Execution algorithm above, Corrective Demonstration allows the teacher to retroactively demonstrate what action should have been selected in its place. In addition to indicating that the wrong action was selected, this method also provides the algorithm with an additional training point, leading the robot to learn quickly from its mistakes.

Together, Confident Execution and Corrective Demonstration form an interactive learning algorithm that takes advantage of the robot's and teacher's complementary abilities – the robot's knowledge of its underlying policy and the teacher's knowledge of the task. Note that we assume that the domain allows the robot to pause and request demonstrations during the learning process.

Software Architecture Design

To utilize the CBA algorithm, we present the single-robot learning architecture shown in Figure 1. The single robot architecture consists of three software modules: the policy learner, the Learning Control Interface (LCI), and the robot's onboard software controller. The LCI is a task-independent and robot-independent software component that provides a central control point for learning. It contains the CBA algorithm and manages the interaction between all system components. Additionally, the LCI provides a standard graphical user interface (GUI) for interaction between the learning software and the teacher.

In addition to the LCI, the software architecture consists of a robot control module and policy learning component. All communication between software components occurs over Transmission Control Protocol (TCP) sockets. The modular structure of the presented learning system has many benefits. Most importantly, it allows individual components to be switched in and out freely, which not only enables the user to apply the base system to a variety of robotic platforms and tasks, but also allows for independent development and testing of each system element.

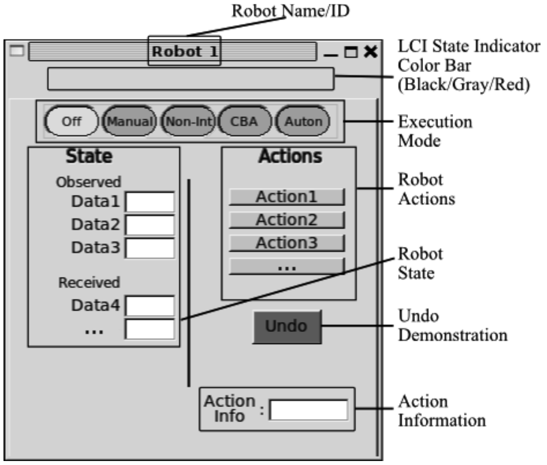


Figure 2: Screenshot of the LCI graphical user interface.

Our system makes no assumptions about the physical embodiment and onboard software architecture of the robot beyond the following requirements: 1) the robot's onboard controller is able to establish a TCP connection with the LCI for data transfer; 2) the robot's onboard controller is able to perform a set of predetermined actions the execution of which can be triggered by the LCI. Any robotic, or even software, system that meets these general requirements can be used for demonstration learning. Communication between the LCI and the robot consists of the exchange of state and action information.

The policy learning component can be viewed as a black box containing the classifier of the teacher's choosing, for which the LCI provides a generic interface. Communication between the LCI and policy learner consists of three types of information: the LCI provides the state information acquired from the robot, and the policy returns its highest confidence action and the action selection confidence.

For each learning task, the teacher configures each robot's instance of the LCI by specifying the following learning parameters using an XML configuration file: robot information (name, IP, and port), choice of policy learning algorithm (e.g. SVM), list of features composing the robot state, list of actions available for demonstration, and the name of the log file in which a record of demonstrations is maintained.

Graphical Interface

The graphical user interface of the LCI serves as a two-way communication device between the robot and the teacher. A screenshot of the GUI is shown in Figure 2 with labels highlighting different regions of the display. The interface displays system information, such as the associated robot's name and current state, and enables the teacher to perform demonstrations by selecting among a set of possible actions for the robot to execute. Additionally, the GUI also provides the teacher with the capability to select among multiple execution modes, as discussed below, and to undo incorrect demonstrations. The undo functionality is useful in the case that the wrong action was accidentally selected by the teacher. The undo operation erases the last demonstration performed from the policy database. However, it does *not* undo the effects of the incorrect action that was executed by the robot as a result of the mistaken demonstration.

Additionally, feedback is provided by a color bar, which enables the teacher to identify at a glance the current status of the robot: not connected, connected and executing an action, or waiting for a demonstration. An action information display is used to show the current action being performed by the robot. When the robot is idle and a demonstration request is pending, the display shows the highest confidence policy action as a recommendation to the teacher.

LCI Execution Modes

The LCI operates in one of five execution modes, each of which provides the teacher with a different level of control and interaction with the robot. The list below presents each execution mode in detail. A summary of the interaction between system components for each execution mode is presented in Figure 3.

Off – This is the initial mode of *flexMLfD*, in which the LCI is inactive. No communication occurs between components.

Manual – This mode provides the teacher with manual control of the robot's actions, similar to a joystick. The LCI displays the robot's current state and allows the teacher to select actions for execution. The robot executes the specified actions, but no learning takes place. This mode is useful for basic interaction with the robot, such as for testing action performance or for teleoperation.

Non-Interactive – This mode enables the teacher to perform demonstrations without receiving feedback from the LCI regarding when and what demonstrations should be performed. Instead, the LCI requests a demonstration at every learning timestep, regardless of action selection confidence. This mode enables the teacher to perform long batch demonstration sequences, which can be useful for bootstrapping the learning process, as discussed in (Chernova and Veloso 2009). This learning mode is not used in the experiments presented in this paper.

CBA – In this mode, the LCI uses the CBA algorithm to control robot autonomy and select demonstration. Additionally, it enables the teacher to initiate corrective demonstrations by selecting the correct action in the GUI when an incorrect action is observed. When this occurs, the LCI records the new demonstration, communicating it to the policy learner. Note that once initiated, the incorrect action is allowed to complete without interruption; interrupting an action may cause the robot to enter an unstable or unsafe state.

Autonomous – In this mode, robot's actions are fully controlled by the current learned policy. The robot is autonomous, not relying on the teacher for any demonstrations.

flexMLfD Learning Framework

At the core of the *flexMLfD* system lies the CBA algorithm and the presented learning architecture, which establishes a general state and action representation and provides a means

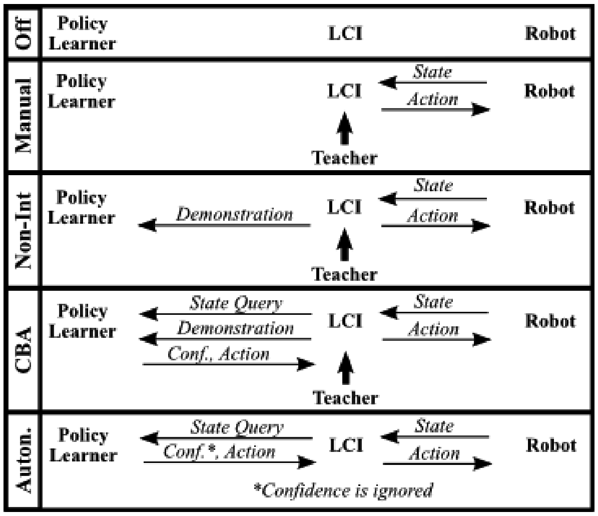


Figure 3: Interaction between the LCI and system components in each execution mode.

for single-robot policy learning through adjustable autonomy. We believe *flexMLfD* to be the first approach that enables a single teacher to teach multiple robots at the same time.

One of the greatest challenges that prevents most single-robot algorithms from generalizing to multi-robot domains is 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. The *flexMLfD* system addresses this problem by replicating instances of the single-robot architecture, as shown in Figure 1(b). This approach enables the system to take advantage of the underlying CBA algorithm, specifically of the fact that the Confident Execution component of CBA prevents the autonomous execution of actions in low-confidence states. This effect makes each learner robust to periods of teacher neglect, enabling multiple robots, each utilizing its individual instance of CBA, to be taught at the same time.

Using this approach, each robot acquires its own set of demonstrations and learns an individual task policy. Specifically, given a group of robots R, our goal is for each robot $r_i \in R$ to learn policy $\Pi_i : S_i \rightarrow A_i$ mapping from the robot's states to its actions. Note that each robot may have a unique state and action set, allowing distinct policies to be learned by possibly heterogeneous robots. The flexibility of the underlying single-robot architecture enables this approach to be applied to a wide variety of tasks with minimal configuration. During learning, the teacher is able to monitor the activities of all robots either visually or through the LCI graphical interface. Demonstrations are provided to one robot at a time.

Example MLfD Domains

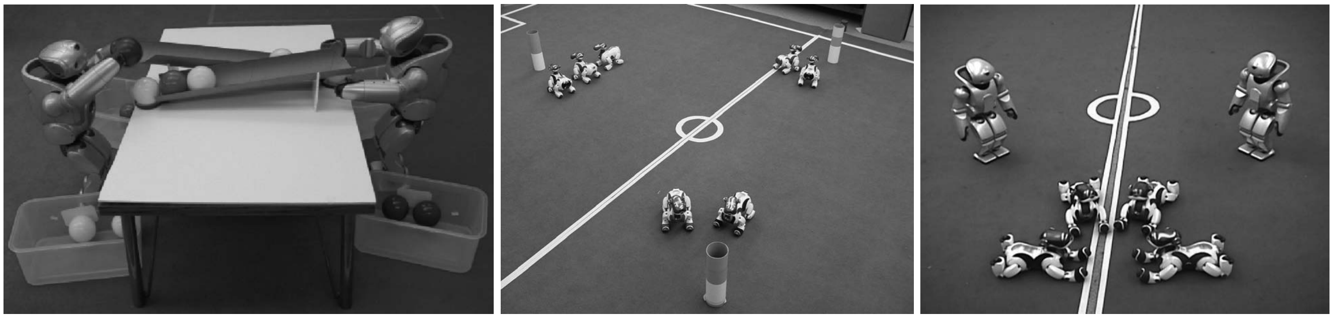
In this section, we present three example multi-robot tasks that showcase the flexible design of the *flexMLfD* system. For each robotic platform, we assume the existence of a set of sensing and acting abilities that are known to the user. At the beginning of the learning process, the teacher selects among these basic abilities and specifies the state features and actions relevant to the current task using the XML configuration file. This selection process speeds up learning by reducing the state representation only to relevant features, and simplifies the user interface for demonstration.

Below, we present a brief summary of each task, which were performed using the legged Sony AIBO and the humanoid Sony QRIO robots. Note that within each of the presented domains, all robots of the same type utilize the same state and action representation. However, this is not a requirement of the algorithm or learning system.

Ball Sorting Domain The ball sorting domain (Chernova and Veloso 2008), shown in Figure 4(a), consists of two sorting stations connected by ramps. Each station has an individual queue of colored balls (red, yellow or blue) that arrive via a sloped ramp for sorting. The task of the two Sony QRIO humanoid robots is to sort the balls by color into four bins. This is achieved by picking up and sorting each ball into the left or right bin, or by passing the ball to the robot's teammate by placing it into the teammate's ramp. Additionally, 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 share the ball with the teammate by passing it. However, only balls that can be sorted by the other robot should be passed. If both queues are empty, the robots should wait.

Beacon Homing Domain The beacon homing domain, shown in Figure 4(b), consists of an open area with three uniquely-colored beacons ($B = \{B1, B2, B3\}$) located around the perimeter. Seven Sony AIBO robots operate in the domain, able to identify the relative position of each beacon and to navigate in the environment by selecting the direction of motion. All robots begin at the center of the open region and must navigate to and occupy one of the beacons. Specifically, each robot must search for a beacon until one is found that is occupied by fewer than 3 robots. Upon locating such a beacon, the AIBO should navigate to its location and occupy it by stopping within a set radius r. If at any point the number of robots at the selected beacon exceeds 3, the AIBO must search for another beacon.

Playground Domain The playground domain, shown in Figure 4(c), consists of an open space simulating a school playground. Two humanoid QRIO robots represent "teachers", and four AIBO robots represent "students"; each QRIO is assigned two AIBO students as its "class". The task simulates a playground scenario in in which the teachers collect their students at the end of recess and take them back to lessons. The task begins with recess, during which the QRIO robots talk to each other while the AIBOs play. Once the bell sounds, the QRIOs stop their conversation and walk to opposite sides of the playground. Each QRIO then calls its respective class and waits for it to arrive. The AIBOs play in the open space until they are called. Once called, each robot should navigate to its QRIO teacher. Once a QRIO has all of its students around it, it leaves the playground with the AIBOs following.



(a) Ball sorting domain (b) Beacon homing domain (c) Playground domain

Figure 4: Multi-robot demonstration learning domains with Sony QRIO and AIBO robots.

	Ball Sorting	**Beacon Homing**	**Playground**
Robots	2 QRIOs	7 AIBOs	2 QRIOs and 4 AIBOs
Actions	*Wait, SortBallLeft, SortBallRight, PassBallRamp, SendHaveBall*	*Forward, TurnLeft, TurnRight, Stop, Search*	**Q:** *Talk, WalkToOpenSpace, CallAIBOs, Wait, LeadAIBOs* **A:** *Forward, Left, Right, Stop, Search, Play*
Action Description	High-level manipulation, communication	Low-level navigation	Low-level navigation, high-level behavior, communication
State	*HaveBall, TeammateHaveBall,* $BallColor_R$, $BallColor_G$, $BallColor_B$, *SentHaveBall*	$B1_a$, $B1_d$, $B1_{nr}$, $B2_a$, $B2_d$, $B2_{nr}$, $B3_a$, $B3_d$, $B3_{nr}$, $OccupiedBeaconID$	**Q:** *Bell, InOpenSpace, CalledAIBOs, NumStudents* **A:** $RecvdQRIOCall$, $Q1_a$, $Q1_d$, $Q2_a$, $Q2_d$
State Description	Noisy real-valued and boolean features	Noisy real-valued and discrete features	Noisy real-valued and boolean features
Communication	Explicit communication actions	Passive communication	Explicit and passive communication
Interaction	Loosely collaborative task, no physical interaction	Non-collaborative task, full physical interaction	Loosely collaborative task, full physical interaction

Table 1: Overview of demonstration learning tasks.

All three of the above tasks were successfully learned using the *flexMLfD* system. Table 1 presents an overview of the various aspects of each learning task, including the state and action representations and styles of interaction and communication. Different elements of each task showcase the flexibility of the proposed system, including the following features:

- Multiple robotic platforms, the Sony QRIO and AIBO.
- Variable number of robots, from 2 to 7.
- Heterogeneous and homogeneous groups of robots.
- Each robot platform was utilized for multiple tasks, with distinct states, actions and policies. The QRIO robots in particular use very different abilities in the ball sorting and playground domains.
- The ability to train an individual policy for each robot provides the teacher with the choice of train all robots the same policy, as in the beacon homing domain, or distinct policies, as in the ball sorting and playground domains.
- The flexible action representation supports wide range of actions, ranging from low-level navigation commands, such as $TurnLeft$, to high-level behavioral commands, such as $Talk$, an action that causes the QRIO to use conversational pose and gestures to simulate speaking.
- The flexible state representation includes boolean, discrete and real-valued features and both locally observed and communicated information.
- Communication actions are incorporated seamlessly into a robot's policy along with physical actions. Teaching communication actions explicitly, such as $SendHaveBall$ in the ball sorting domain, enables the teacher to specify the exact conditions under which communication should occur, a useful technique for domains with high communication costs. Alternately, passive communication can be used to automatically communicate data at a fixed time interval, as is done in the beacon homing domain for the $OccupiedBeaconID$ feature.
- Policy learning supports variable degrees of collaboration and interaction between robots, ranging from physically separate but collaborating QRIO robots in the ball sorting task, to competitive and physically interacting AIBO robots in the beacon homing domain.

All of the above variations are fully supported by the *flexMLfD* learning system and no special adaptations to the underlying architecture are required for each task.

Scalability Analysis

In addition to exploring the flexibility of the learning system, we are interested in analyzing the scalability of *flexMLfD*. In this section, we discuss how the number of robots taught by the teacher at the same time affects 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. Evaluation was performed in the beacon homing domain using 1, 3, 5, and 7 robots.

All evaluation results presented in this paper 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 was then discarded from the evaluation. 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 effect multiple robots have on teacher performance.

Robot Autonomy

Figure 5 shows how the level of autonomy, measured as the percentage of autonomous actions versus demonstrations, changes for an individual robot over the course of training. Data in the figure presents the average autonomy over time of robots in the 5-robot beacon homing experiment. The shape of the curve seen in this figure 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. The domain knowledge acquired from these initial demonstrations provides 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. Learning is complete once the correct action is selected for all states with high confidence.

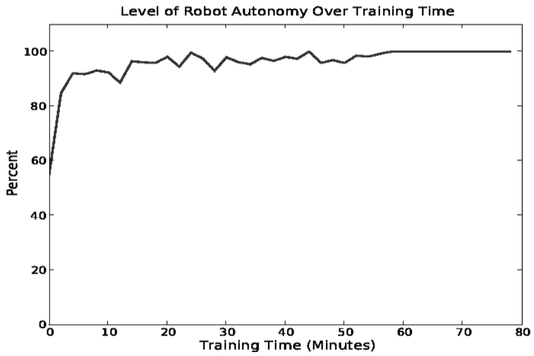


Figure 5: Average level of autonomy of a single robot over the course of training (5-robot learning example).

Training Time

Figure 6 presents the change in the overall experiment training time with respect to the number of robots. 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 to even larger tasks.

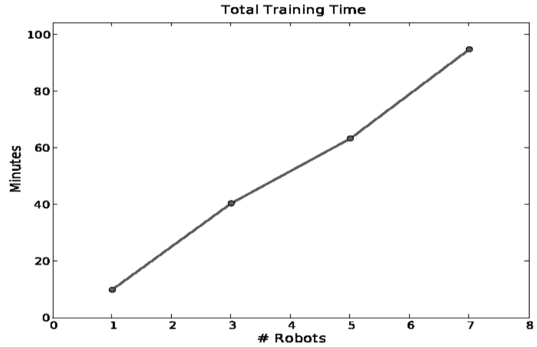


Figure 6: Total training time with respect to number of robots.

Importantly for the scalability of the CBA algorithm, training time grows linearly with the number of robots. In the following sections, we examine the factors that contribute to the training time, such as the number of demonstrations and demonstration delay.

Number of Demonstrations

In this section, we examine 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.

Figure 7 shows that as the number of robots grows, we observe a slight increase in the number of demonstrations

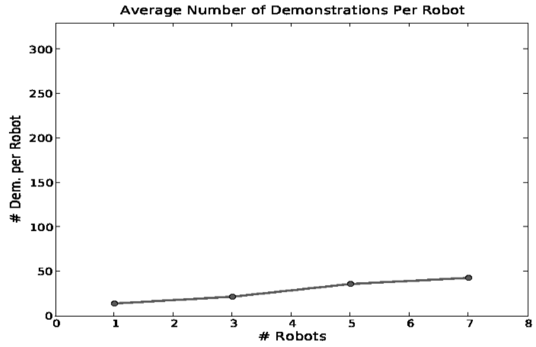


Figure 7: Average number of demonstrations performed by the teacher for each robot.

required per robot. This possibly surprising 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, b_{nr}, have 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 b_{nr} to a boolean feature that indicates whether the beacon's capacity has been reached or not.

Figure 8 shows how the total number of demonstrations required for each experiment changes with respect to the number of robots. The rate of growth is nearly linear, with seven robots requiring nearly 300 total demonstrations to learn the task. The overall number of demonstrations that must be performed has a significant effect on the overall training time, as discussed in the next section.

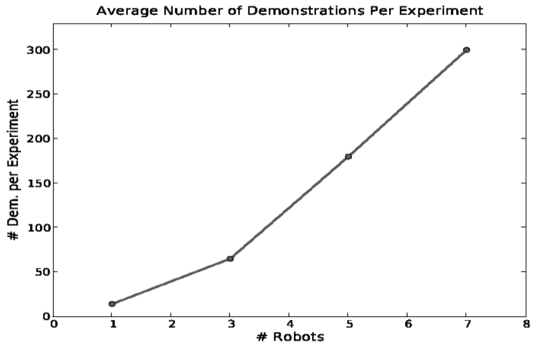


Figure 8: Total number of demonstrations performed in each experiment.

Attention Demand on the Teacher

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. 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 mental effort that is required from the teacher.

Figure 9 plots the duration of the longest continuous demonstration request period for each experiment. The data shows that the duration 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.

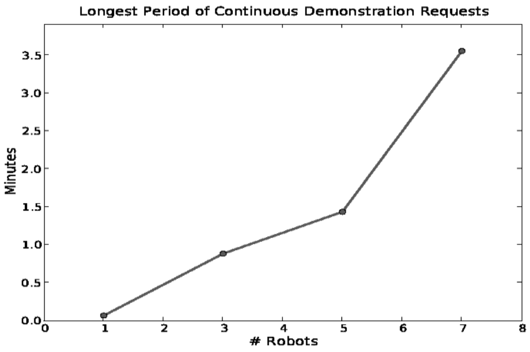


Figure 9: Attention demand on the teacher.

Demonstration Delay

As discussed in the previous section, simultaneous demonstration requests from multiple robots become common as the number of robots increases. As a result, robots are often required to wait while the teacher responds to other robots. Figure 10 shows that the average time a robot spends waiting for a demonstration grows with respect to the number of learners from only 2 seconds for a single robot to 12 seconds for seven robots.

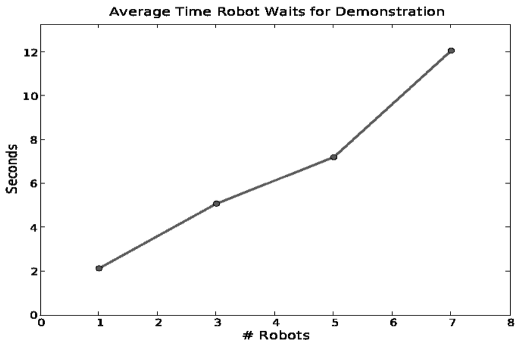


Figure 10: Average amount of time a robot spends waiting for a demonstration response from the teacher.

Figure 11 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. A promising direction for future work is to examine the possibility of staggering the times at which novice robots are introduced to the task in order to reduce the demand of the initial training phase on the teacher.

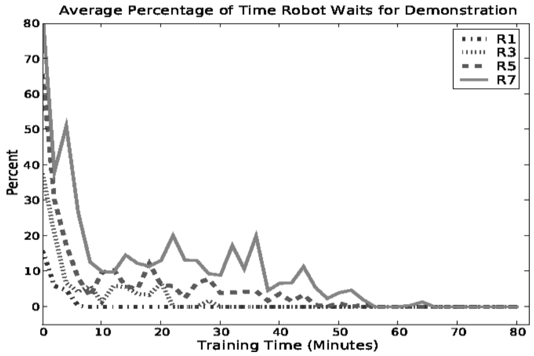


Figure 11: Average percentage of time a robot spends waiting for a demonstration over the course of training.

Evaluation Summary

In summary, our findings show promising trends for the scalability of the presented multi-robot demonstration learning approach. Particularly significant is that the total training time grows linearly with the number of robots, allowing learning to scale to larger tasks. Somewhat unsurprisingly, we also found that increasing the number of robots also significantly increases the workload of the teacher, as measured by the number of pending demonstration requests. In our evaluation, we show that this in turn impacts demonstration delay and robots must spend more time 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.

Further studies are required before making broad conclusions about the scalability of *flexMLfD*. In particular, more research is needed to determine what impact state representation, action duration, and degree of collaboration between robots have on learning performance and scalability.

However, based on the presented case study we find that no absolute upper bound exists on the number of robots that can be taught at the same time. The maximum number of robots used in the experiments, seven, represents our own limitation in terms of time and the number of available robots, not a limitation of the algorithm. Insights gained in this evaluation can be used as a guide for the development of future applications for *flexMLfD*. For example, our knowledge of the trend in overall training time requirements can be used to limit the number of robots in other applications for which the availability of an expert teacher is limited to some fixed time. Similarly, the number of robots in other domains may be affected by the amount of time a robot may remain idle while waiting for a demonstration.

Conclusion

Multi-robot applications are an exciting and promising new direction for demonstration and imitation based learning methods. In this paper, we presented an overview of the *flexMLfD* multi-robot demonstration learning system. Our approach is based on the Confidence-Based Autonomy demonstration learning algorithm, which provides the means for a single robot to learn a task policy through interaction with a human teacher. We utilized the generalized representation and adjustable autonomy provided by the CBA algorithm to develop a flexible multi-robot demonstration learning system. To highlight the generality of the presented approach, we presented three multi-robot domains which showcase learning using multiple robotic platforms in uniform and heterogeneous groups, utilizing different state features, actions, and styles of communication and collaboration. Additionally, we presented an evaluation of the scalability of this approach 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. The results of our case study indicate that no strict upper bound exists on the number of robots due to limitations of the algorithm.

We hope that the presented work serves as a stepping stone for further research into multi-robot demonstration learning. Promising future research direction include, and are not limited to, alternate representations and learning techniques, additional methods of human-robot interaction, interface design, multi-robot control and scalability.

References

Bentivegna, D. C. 2004. *Learning from Observation Using Primitives*. Ph.D. Dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA.

Calinon, S., and Billard, A. 2007. Incremental learning of gestures by imitation in a humanoid robot. In *ACM/IEEE international conference on Human-robot interaction*, 255–262. New York, NY, USA: ACM.

Chernova, S., and Veloso, M. 2008. Teaching collaborative multi-robot tasks through demonstration. In *IEEE Int. Conf. on Humanoid Robots*.

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34:1–25.

Grollman, D., and Jenkins, O. 2007. Dogged learning for robots. In *IEEE International Conference on Robotics and Automation*, 2483–2488.

Lockerd, A., and Breazeal, C. 2004. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Nicolescu, M. N., and Mataric, M. J. 2003. Natural methods for robot task learning: instructive demonstrations, generalization and practice. In *Autonomous Agents and Multiagent Systems*, 241–248. ACM Press.

Saunders, J.; Nehaniv, C. L.; and Dautenhahn, K. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *1st Conference on Human-Robot Interaction*, 118–125. New York, NY, USA: ACM Press.

Smart, W. D., and Kaelbling, L. P. 2002. Effective reinforcement learning for mobile robots. In *ICRA 2002*.

Steil, J.; Rothling, F.; Haschke, R.; and Ritter, H. 2004. Situated robot learning for multi-modal instruction and imitation of grasping. *Robotics and Autonomous Systems* 47(2-3):129–141.