Flexible Demonstration Learning System for Variable Number of Robots

Sonia Chernova and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
{soniac, veloso}@cs.cmu.edu

ABSTRACT

In this paper, we present *flexMLfD*, a robot independent and task independent demonstration learning system that supports a variable number of robot learners. Our approach is based on the Confidence-Based Autonomy (CBA) demonstration learning algorithm, which provides the means for a single 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*. Building upon the CBA single-robot algorithm, we contribute a robot-independent modular software architecture for multi-robot learning, interaction and control. To highlight the generality of the presented learning system, we present three example domains, each utilizing from two to seven real robots.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Design, Human Factors

Keywords

human-robot interaction, learning from demonstration, multi-robot learning

1. INTRODUCTION

The interaction of robots with humans is inherently complex and varied and many methods of human-robot interaction have been developed for different robotics applications. In this paper, we examine robot interaction and control in the context of learning from demonstration (LfD). Learning from demonstration, also known as teaching by demonstration, is a learning approach based on human-robot interaction that provides an intuitive technique for robot programming. In this approach, a teacher, usually 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*, a mapping from all world states to actions, that reproduces the observed behavior.

Demonstration learning inherently requires interaction between the robot and the teacher. Many different demonstration learning systems have been proposed utilizing a variety of interaction methods, including natural language [2], gestures [3], joysticking devices [15] and execution of the task by other robots [12]. Regardless of the method of interaction, existing approaches share certain common features, such as application-specific design and dependence on one-to-one interaction with the teacher. In this paper, we highlight two research directions that we believe to be fundamental for the future development of demonstration learning, and present a complete learning system addressing these challenges.

Platform and task independent system design: When developing a learning system with a particular application in mind, it is often tempting to take advantage of task-specific optimizations and simplifications to improve performance. However, task-specific elements severely limit the applicability of the developed technique to new domains. Inspired by general purpose robot development software such as MS Robotics Studio [1] and the Player Project [9], which provide a flexible programming interface for a wide range of robotic platforms, our goal is to develop a task-independent and robot-independent system for learning from demonstration. A general learning system of this kind will serve as a stepping stone for future development in this research area, allowing for faster development times and greater comparison between algorithms.

Multi-robot demonstration learning: For many applications, it is desirable for a single person to teach multiple robots at the same time. This can occur for a variety of reasons, for example, when teaching collaborative robot behaviors in which the actions of one robot depend upon the state and actions of its teammate. Requiring an individual teacher for each robot is inefficient and impractical in many real-world settings. Instead, we would like a single person to teach 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)*.

In this paper, 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) demonstration learning algorithm [4],

HRI'09, March 11–13, 2008, San Diego, CA USA.

which enables a single 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*. Based on this single-robot algorithm, we contribute a robot-independent modular software architecture for robot learning, interaction and control. The presented system can be applied to a variable number of learners, from one to multiple, independent or collaborative, robots.

The *flexMLfD* system has been fully implemented and tested using multiple real-world domains and robotic platforms. In the following section, we discuss related work in the area of human-robot interaction (HRI), followed by an overview of the CBA algorithm in Section 3. We then present the complete multi-robot architecture in Section 4, and an overview of the demonstration learning process in Section 5. We conclude by presenting three real-world tasks that have been successfully learned using this system, each involving from two to seven robots.

2. RELATED WORK

Interfaces for the interaction and control of multiple robots have been studied in the HRI community for many robotics applications. A significant portion of existing work has focused on the human side of the interaction, examining the effects of various design elements on user performance [13, 11]. For example, closely related to our problem of multi-robot demonstration learning is an interface proposed by Glas et al. [10] for the control of multiple social robots. However, significant differences between the proposed social interface and our own work include the type and level of interaction and the need for policy learning. In this paper, we examine the interaction and control problem from the robot perspective, presenting a complete software system that enables robots to learn to perform tasks with complete autonomy. Usability and design studies of the proposed interface have been left for future work.

Much previous work has been devoted to the evaluation and study of multi-robot control from the systems perspective. Humphrey et al. [14] present a relational display designed for the control of a variable number of robots, which was successfully used to teleoperate up to nine simulated robots in a bomb defusing task. Fong et al. [8] present HRI/OS, a software framework for establishing interaction and dialog in human-robot teams. Targeted at execution of operational tasks, such as resource collection, this system supports a variety of user interfaces and robot platforms through an extensible API, making it applicable to a wide range of applications.

Wang and Lewis [16] present a study evaluating how the degree of operator control affects task performance in multi-robot tasks. The authors conclude that mixed initiative teams of robots performed more successfully at a simulated urban search and rescue task than either fully autonomous or manually controlled teams. Similar results are supported by other studies in adjustable autonomy research, motivating the use of partial autonomy in our multi-robot demonstration learning framework.

All of the above systems provide interfaces for robot control and navigation, but do not include a learning component. In particular, we are not aware of any system designed to address multi-robot interaction and control in the context of learning from demonstration. While demonstration-based systems have much in common with other interactive and semi-autonomous robotic systems, they have the following unique set of design requirements:

- *Policy Learning* The most important goal of the learning from demonstration system is to enable the robot to learn a policy representing the demonstrated behavior and to successfully reproduce this behavior in further trials.
- *Interaction* The system must provide a method of demonstration and interaction between the robot and teacher.
- *Adjustable Autonomy* To facilitate scalability to multi-robot applications, demonstration learning systems should be designed with adjustable autonomy such that the robot acts autonomously when it is familiar with the task and requests guidance in the form of a demonstration upon encountering an unfamiliar situation. In contrast to most adjustable autonomy systems, however, the autonomy of the robot should increase over time as the task is learned until full autonomy is reached.
- *Correction of Unwanted Behavior* Similarly to existing control systems, demonstration learning requires a method of correcting any mistakes made by the robot during autonomous execution. However, in addition to correcting the physical action, the system should allow the robot to learn from its mistake so that it will not be repeated in the future.
- *Control System Design* To be effective, the system must be applicable under a wide range of conditions. The system design must therefore maintain:

  platform neutrality: allowing the use of any robotic platform

  domain neutrality: applicable to a wide range of tasks

  location neutrality: executable on a wide range of computer systems, onboard or offboard the robot

- *Multi-Robot Learning* The teacher must be able to effectively monitor and interact with multiple robots at the same time.

In the following sections we describe all components of the *flexMLfD* system and how they address these challenges.

3. SINGLE-ROBOT DEMONSTRATION LEARNING ALGORITHM

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 the algorithm, please see [4, 6].

Confidence-Based Autonomy is a single-robot demonstration learning algorithm that enables a robot[1] 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

[1]Physical or simulated.

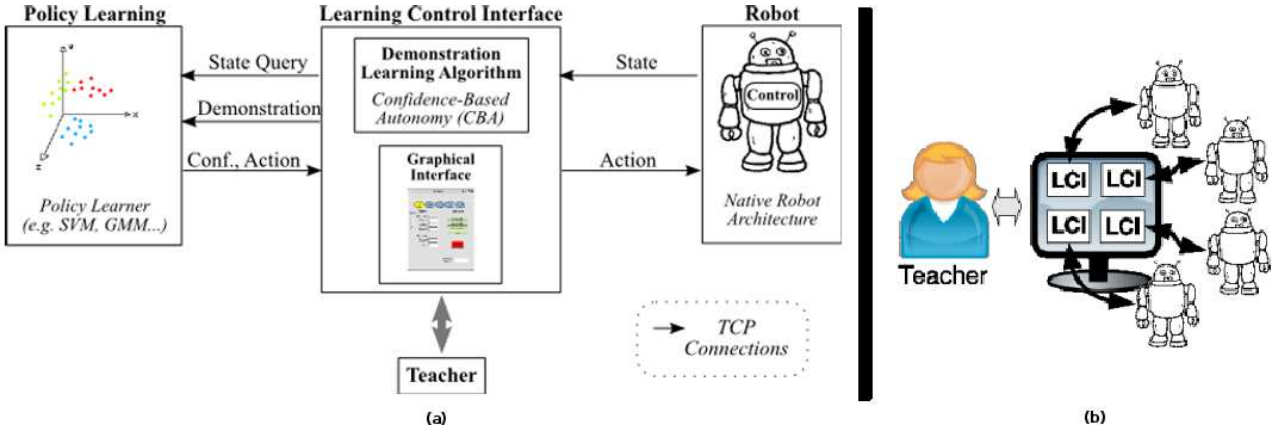


Figure 1: Overview of the *flexMLfD* demonstration learning system: (a) single robot learning architecture (b) high-level overview of multi-robot learning.

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 together 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 complimentary 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.

The CBA algorithm addresses several of the key design requirements for an effective multi-robot demonstration learning system, including interactive policy learning, adjustable autonomy and correction. Additionally, the generic state and action representation used by the algorithm provides the basis for a task-independent learning system. Next, we show how a complete multi-robot demonstration learning system is built around this algorithm.

4. MULTI-ROBOT SYSTEM DESIGN

In this section, we present *flexMLfD*, our multi-robot demonstration learning system. We believe this to be the first approach that enables a single teacher to teach multiple robots at the same time.

At the core of the *flexMLfD* system lies the CBA algorithm, which establishes a general state and action representation and provides a means for single-robot policy learning through adjustable autonomy. To utilize the CBA algorithm, we present the single-robot learning architecture shown in Figure 1. The single robot architecture consists of several software modules, at the heart of which is the Learning Control Interface (LCI). The LCI is a task-independent and robot-independent software component that provides a central control point for learning, managing the interaction between the teacher, a single robot and the policy learner. The LCI houses the CBA algorithm and provides interfaces to the policy learner and the robot's onboard software controller. Additionally, the LCI provides a standard graphical user interface (GUI) for interaction between the learning software and the teacher. In the following sections, we describe each component of the learning system in detail.

4.1 Teaching Multiple Robots

Multi-robot learning is achieved by replicating instances of the single-robot architecture, as shown in Figure 1(b). 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 flexible design of this architecture has two great ben-

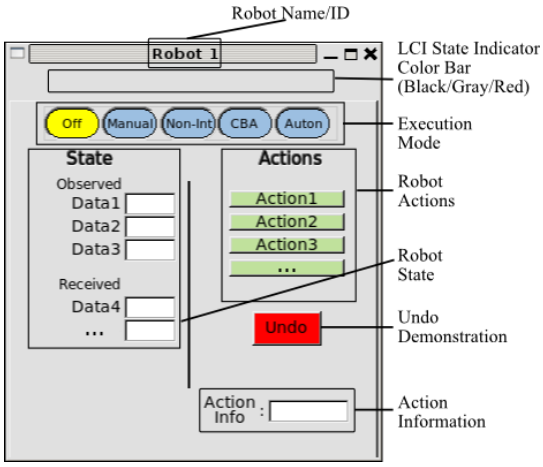


Figure 2: Screenshot of the LCI graphical user interface.

efits for the demonstration learning system. First, the flexibility of the underlying single-robot architecture enables this approach to be applied to a wide variety of tasks with minimal configuration. Second, the automatic regulation of each robot's autonomy, provided by the Confident Execution component, enables multiple robots to be taught at the same time. 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.

In this paper, we show how the proposed system can be successfully applied to training up to seven robots at the same time. Due to space limitations, we are unable to include a full analysis of the scalability of *flexMLfD*. For a detailed analysis of how the number of robots affects learning performance and demands on the teacher, please see [5].

4.2 Graphical Interface

The graphical user interface of the LCI serves as a two-way communication device between the robot and the teacher. The interface displays valuable system information, such as the robot's current state and the notification of a demonstration request. Using this interface, the teacher is able to use the interface to select among a set of possible actions for the robot to execute. Additionally, the GUI also provides the teacher with the capability to control the learning process by activating different execution modes and to undo incorrect demonstrations. A screenshot of the GUI is shown in Figure 2 with labels highlighting different regions of the display. Below we describe the function of each interface element:

Robot Name/ID Display – Window label indicating the name or ID number of the robot associated with this LCI.

LCI State Indicator Color Bar – Color bar indicating the current state of the LCI: not connected to the robot (black), connected and executing action (gray), connected and waiting for a demonstration (red). The color indicator enables the teacher to identify at a glance when a demonstration is required or when problems with network connectivity between the LCI and the robot occur.

Execution Mode – Set of buttons for selecting the current LCI execution mode. A detailed description of each mode are provided in Section 4.4.

Robot State – Display showing the current state of the robot. State features are organized by their source, separated into information observed locally through the robot's sensors and information received through communication from remote sources, such as teammates or external sensors. Communicated data can be especially difficult for the teacher to monitor without the GUI interface. The robot state display therefore provides vital information necessary to ensure that the teacher's view of the world matches that of the robot.

Robot Actions – Set of buttons specifying the list of robot actions available for demonstration.

Undo Demonstration – A button allowing the teacher to "undo" the last demonstration performed. This 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.

Action Information – When action execution is in progress, this display shows 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.

4.3 Software Architecture Design

The modular structure of the presented learning system has many benefits. Most importantly, it allows individual components, such as the physical robot platform or the policy learning algorithm, to be switched in and out freely. This 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 individual component.

The teacher configures the system for each learning task using an XML configuration file which specifies various parameters. These include robot information (name, IP, and port), choice of policy learning algorithm (e.g. SVM, GMM), 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.

Communication between software components occurs over Transmission Control Protocol (TCP) sockets. TCP sockets are a general communication method that enables platform and location neutrality. In our implementation, the LCI and policy learning components are located offboard the robot on a remote PC. This provides the user with a graphical interface, mouse and keyboard for interaction with the robot, and provides ample processing power for the policy learning algorithm. Many alternate solutions exist. For example, policy learning can be performed on a remote computing cluster for especially computationally challenging tasks, or all three components can be located onboard the robot if enough processing power and a suitable video display are available.

Two channels of communication exist over TCP connections in the *flexMLfD* system:

LCI-Robot Communication Communication between the LCI and the robot consists of state and action information.

The robot reports its state, consisting of raw and processed sensor values, to the LCI at a fixed time interval while it is not performing an action. This information is used by the policy learning component and the teacher to select actions for the robot to perform. During action execution, state information is not reported because action selection does not need to take place during this time.

The *flexMLfD* system makes no assumptions about the physical embodiment and onboard software architecture of the robot beyond the following requirements:

- The robot's onboard controller is able to establish a TCP connection with the LCI for data transfer.
- The robot's onboard controller is able to perform a set of predetermined actions the execution of which can be triggered by the LCI over the TCP connection.

Any robotic, or even software, system that meets these general requirements can be used for demonstration learning.

LCI-Policy Communication The LCI provides a general interface to the policy learning component, which can be viewed as a black box containing the classifier of the teacher's choosing. Communication between the LCI and policy learner consists of three types of information. State information acquired from the robot is used by the LCI to query the policy, which returns the policy-selected action and its action selection confidence. These values are used by the CBA algorithm within the LCI to regulate the robot's autonomy. Each time a demonstration is performed, the LCI communicates the new state-action pair to the policy learner to update the policy.

Using the above methods, the LCI coordinates the exchange of information between all system components. The LCI has multiple execution modes which determine the level of interaction and control that takes place, ranging from CBA demonstration learning to fully autonomous execution of the policy by the robot. Details of the possible execution modes are presented in the following section.

Finally, we highlight that the LCI is an interface for *learning* a task policy. Once the task is learned, the LCI begins to act simply as an intermediary between the robot and policy components. The teacher's involvement is no longer required as the robot is able to execute the task autonomously. When this final learning stage is reached, the teacher may continue to use the LCI in order to monitor the robot's status or to have additional control over the robot's execution mode. Alternatively, the LCI can be eliminated from the system entirely, allowing the robot to interact directly with the policy component over the established socket interface.

4.4 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 the learning system, in which the LCI is inactive. No communication occurs between components.

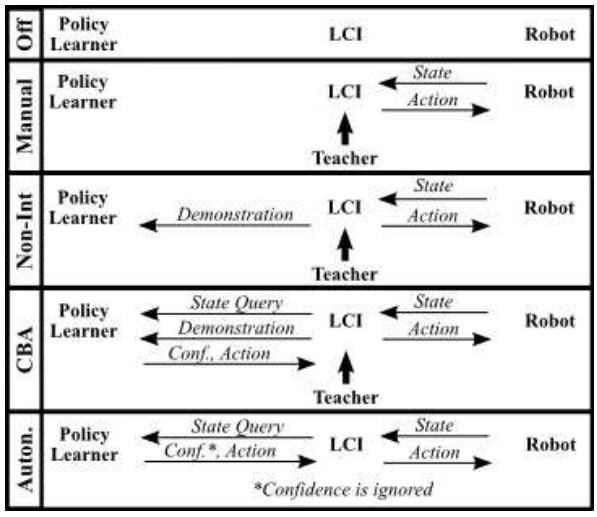


Figure 3: Interaction between the LCI and system components in each execution mode.

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 [4]. 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 the learning process and select demonstrations. Each time the LCI receives a new robot state, it queries the policy to obtain the highest confidence action and its action selection confidence. This information is then used by the CBA algorithm to select between demonstration and autonomy. If autonomy is selected, the policy-selected action is immediately communicated to the robot for execution. If a demonstration is required, the LCI notifies the teacher of the demonstration request using the LCI state indicator color bar. Once the teacher selects an action, the action is communicated to the robot and a new training datapoint, consisting of the robot's current state and the demonstrated action, is used to update the policy.

Note that while waiting for the demonstration, the robot's state may change due to changes in the environment. The LCI continues to update the state information and query the policy each time an update is received from the robot. The robot continues to wait until either a demonstration is performed, in which case it is associated with the most current state, or the robot finds itself in a state of high confidence, in which case the demonstration request is cancelled and the robot executes the policy-selected action autonomously.

In addition to responding to demonstration requests, the teacher may also initiate corrective demonstrations. If the teacher observes the robot performing an incorrect action, a correction is performed by selecting the correct action in the graphical interface. When this occurs, the LCI records the new demonstration, communicating it to the policy learner. The incorrect action already being executed by the robot 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.

5. OVERVIEW OF TEACHING PROCESS

This section presents an overview of a typical multi-robot learning process:

1. *flexMLfD* Configuration – The teacher initializes learning by configuring an instance of *flexMLfD* for each robot. This is performed by creating an XML configuration file specifying robot information, as well as selecting the set of state features and actions relevant for the task at hand. Once a connection to the robot is established, these values are displayed in the LCI GUI. Learning is initiated by activating the "CBA" learning mode.

2. Demonstration Learning – All robots begin with no prior knowledge about the task and initially request a demonstration. Demonstration requests remain frequent during the early stages of learning, and the teacher commonly has multiple pending demonstration requests from different robots. In the current implementation, the teacher selects arbitrarily among the demonstration requests. As training data is gathered over time, the quality of each robot's policy improves and its autonomy increases. By practicing the task, each robot refines its policy upon encountering rare or complex states. Learning is complete once the correct action is selected for all states with high confidence and the entire task is performed correctly without help from the teacher.

3. Autonomous Task Execution – Once policy learning is complete, the robot uses the policy to execute the task autonomously .

Figure 4 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. The data curve shows that following an initial burst of demonstration requests, the robot quickly achieves an average autonomy rate of 80–90% as it continues to refine the policy. The data in the graph was recorded in the beacon homing domain (see Section 6), but the trend seen here is is typical of most learning domains in which initial demonstrations provide the robot with the experience for handling most commonly encountered domain states.

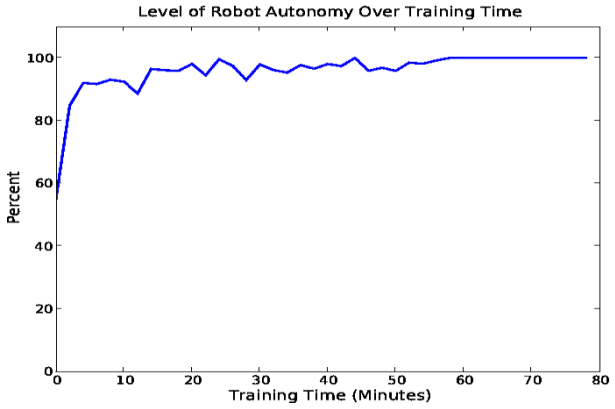


Figure 4: Average level of autonomy of a single robot over the course of training.

6. EXAMPLES OF MULTI-ROBOT DEMONSTRATION LEARNING

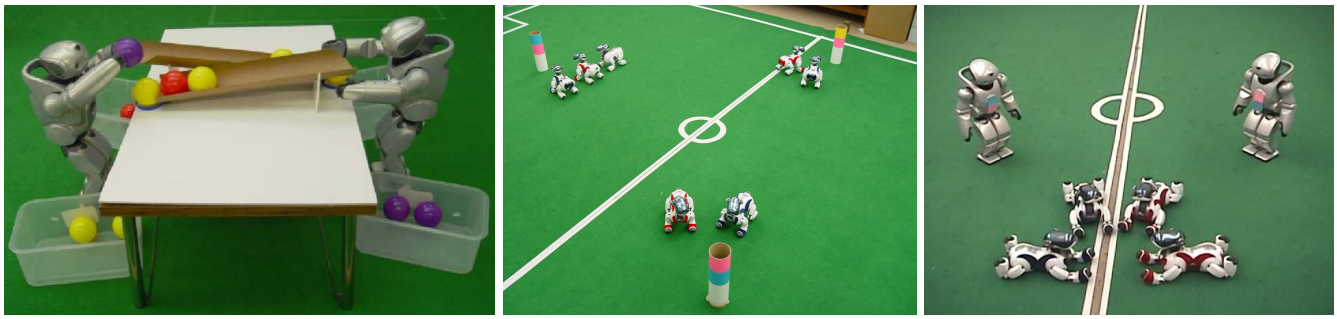
The *flexMLfD* system has been applied to a wide range of both single-robot and multi-robot domains [4, 7]. In this section, we present three example multi-robot tasks that showcase the flexible design of the *flexMLfD* system. These tasks were performed using the legged Sony AIBO and the humanoid Sony QRIO robots. We assume that each robotic platform has a set of sensing and acting abilities that are known to the user. In the case of the AIBOs, the robots are able to detect objects in the environment using their onboard camera, to calculate the relative distance and angle of these objects, and to communicate via the wireless network. The QRIO robots can similarly detect and locate objects and communicate wirelessly in addition to detecting sounds using an onboard microphone array. Both robotic platforms also support a wide range of actions.

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. 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.

Below, we present a brief summary of each task:

Ball Sorting Domain The ball sorting domain, shown in Figure 5(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 pass the ball to the teammate. 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 5(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



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

Figure 5: 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.

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 5(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.

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 $Talk$, which 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. Instead all task-specific elements are specified within the XML configuration file prior to training.

7. CONCLUSION

In this paper, we introduced a new class of policy learning algorithms, multi-robot learning from demonstration, and contributed *flexMLfD*, a robot-independent and task-independent demonstration learning system that supports a variable number of robot learners. 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.

8. ACKNOWLEDGMENTS

We thank SONY for their generosity in making the QRIOs and their motion and software features available for our research. This research was partly sponsored by BBNT Solutions under subcontract no. 950008572, via prime Air Force contract no. SA-8650-06-C-7606. The views and conclusions contained in this document are those only of the authors.

9. REFERENCES

[1] Microsoft robotics studio, 2008. http://msdn.microsoft.com/en-us/robotics.

[2] C. Breazeal, G. Hoffman, and A. Lockerd. Teaching and working with robots as a collaboration. In *3rd Int. Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1030–1037, Washington, DC, USA, 2004. IEEE Computer Society.

[3] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262, New York, NY, USA, 2007.

[4] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research (to appear)*.

[5] S. Chernova and M. Veloso. Scalability of confidence-based autonomy multi-robot demonstration learning. In *Under submission.*

[6] S. Chernova and M. Veloso. 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.

[7] S. Chernova and M. Veloso. 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.

[8] T. Fong, C. Kunz, L. M. Hiatt, and M. Bugajska. The human-robot interaction operating system. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 41–48, New York, NY, USA, 2006.

[9] B. P. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[10] D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. Simultaneous teleoperation of multiple social robots. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 311–318, New York, NY, USA, 2008.

[11] M. A. Goodrich, T. W. McLain, J. D. Anderson, J. Sun, and J. W. Crandall. Managing autonomy in robot teams: observations from four experiments. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 25–32, New York, NY, USA, 2007.

[12] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *2nd International Symposium on Intelligent Robotic Systems*, 1994.

[13] S. G. Hill and B. Bodt. A field experiment of autonomous mobility: operator workload for one and two robots. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 169–176, New York, NY, USA, 2007.

[14] C. M. Humphrey, C. Henk, G. Sewell, B. W. Williams, and J. A. Adams. Assessing the scalability of a multiple robot interface. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 239–246, New York, NY, USA, 2007.

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

[16] J. Wang and M. Lewis. Human control for cooperating robot teams. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 9–16, New York, NY, USA, 2007.