Towards Rapid Multi-robot Learning from Demonstration at the RoboCup Competition

David Freelan, Drew Wicke, Keith Sullivan, and Sean Luke

Department of Computer Science, George Mason University
4400 University Drive MSN 4A5, Fairfax, VA 22030 USA
dfreelan@gmu.edu, dwicke@gmu.edu, ksulliv2@gmu.edu, sean@cs.gmu.edu

Abstract. We describe our previous and current efforts towards achieving an unusual personal RoboCup goal: to train a full team of robots directly through demonstration, on the field of play at the RoboCup venue, how to collaboratively play soccer, and then use this trained team in the competition itself. Using our method, HiTAB, we can train teams of collaborative agents via demonstration to perform nontrivial joint behaviors in the form of hierarchical finite-state automata. We discuss HiTAB, our previous efforts in using it in RoboCup 2011 and 2012, recent experimental work, and our current efforts for 2014, then suggest a new RoboCup Technical Challenge problem in learning from demonstration.

Imagine that you are at an unfamiliar disaster site with a team of robots, and are faced with a previously unseen task for them to do. The robots have only rudimentary but useful utility behaviors implemented. You are not a programmer. *Without coding them,* you have only a few hours to get your robots doing useful collaborative work in this new environment. How would you do this?

Our interest lies in rapid, real-time multi-robot training from demonstration. Here a single human trainer teaches a team of robots, via teleoperation, how to collectively perform tasks in previously unforeseen environments. This is difficult for two reasons. First, nontrivial behaviors can present a high-dimensional space to learn, yet one can only provide a few samples, as online training samples are costly to collect. This is a worst case for the so-called "curse of dimensionality". Second, when training multiple interactive robots, even if you can quantify the emergent *macro-level* group behavior you wish to achieve, in order to do learning, each agent needs to know the *micro-level* behavior he is being asked to do. One may have a micro→macro function (a simulator), but it is unlikely that one has the inverse macro→micro function, resulting in what we call the "multiagent inverse problem". These two challenges mean that real-time multi-robot learning from demonstration has proven very difficult and has a very sparse literature.

Over the past several years we have participated in the Kid-Size Humanoid League with a single objective: to successfully do a personal RoboCup-style technical challenge of our own invention, independent of those offered at RoboCup: **can we train multiple generic robots, through demonstration on the field, how to play collaborative soccer at RoboCup solely within the preparatory time prior to the competition itself?**

This is a very high bar: but over the past four years we have made major strides towards achieving it. In RoboCup 2011 we began by replacing a single hard-coded behavior in one attacker with a behavior trained on the field at the venue, and entered that robot into the competition. At RoboCup 2012 we expanded on this by training an attacker to perform *all* of its soccer behaviors (17 automata, Figure 1), again at the venue. This trained attacker scored our winning goal against Osaka. This year we intend to train multiple robots, and ideally all four robots on the team, to perform collaborative behaviors.

Our approach, HiTAB, applies supervised learning to train multiple agents to perform behaviors in the form of decomposed hierarchical finite-state automata. HiTAB uses several tricks, notably task decomposition both per-agent and within a team, to break a complex joint behavior into smaller, very simple ones, and thus radically reduce its dimensionality. Sufficient domain knowledge is involved that HiTAB may fairly be thought of as a form of *programming by demonstration.*

This paper documents our past efforts at applying HiTAB on the field at RoboCup. We also discuss related penalty-kick experiments using the technique, and detail our success so far towards our 2014 goal. Finally, we propose a new RoboCup Technical Challenge in multiagent learning from demonstration.

1 Related Work

Learning from demonstration (or LfD) has been applied to a huge range of problems ranging from air hockey [2] to helicopter trajectory planning [14], but rarely to the multi-robot case [1]. Most of the multi-robot learning literature falls under *agent modeling*, where robots learn about one another rather than about a task provided by a demonstrator. The most common multi-robot LfD approach is to dismiss the macrophenomena entirely and issue separate micro-level training directives to each individual agent [11]. Another approach is to train individual robots only when they lack confidence about how to proceed [4].

1.1 Machine Learning at RoboCup

To put our "personal technical challenge problem" in context, it's worthwhile to survey how machine learning has been used at RoboCup in the past. Machine learning has been applied to RoboCup since its inception, coming to slightly less than 100 papers and demonstrations since 1997. We mention only a small number of the papers here.

The bulk of the machine learning RoboCup literature has involved single agents. This literature breaks down into three categories. First, learning algorithms have been applied about a dozen times to *sensor feature generation* tasks such as visual object recognition [13, 31] and opponent behavior modeling and detection (for example [8, 29]). Second, a equal amount of literature has applied machine learning to a robot's kinematics, dynamics, or structure. The lion's share of this work involves gait development (such as [19, 18]), with some work on kicking [6, 32], head actuation [5] and omnidirectional velocity control [17]. Third, about sixteen papers have concerned themselves with learning higher-level behaviors (for example [26, 28]).

Cooperative Multiagent Learning There have been approximately twenty five cooperative multiagent learning papers at RoboCup. The area breaks down into two categories. First, there is *team learning*, where a single learning algorithm is used to optimize the behaviors of an entire team. Some of this work has involved evolutionary computation methods to develop joint team behaviors (such as [15, 10]); reinforcement learning papers have instead usually developed a single homogeneous behavior (for example [7, 22]). In contrast the *concurrent learning* literature, where separate learners are applied per-agent, has largely applied multiagent reinforcement learning (such as [12, 21]).

It is useful here to mention why this area is dominated by optimization methods (reinforcement learning, evolutionary computation): as mentioned before, multiagent learning presents a difficult inverse problem, and optimization is the primary way to solve such problems. However, optimization generally needs many iterations for even moderately high-dimensional spaces, meaning realistically such methods must employ a simulator, and so are not optimal for real-time training.

Training Training differs from learning in that it involves a *trainer*, that is, a person who iteratively teaches behaviors, observes agent performance, and suggests corrections. This is a natural fit for soccer: but training is surprisingly rare at RoboCup. RoboCup has long sponsored a related topic, *coaching*, but the focus has more been on influencing players mid-game via a global view [27] than on training. One exception has used a coach to train action sequences as directed by human speech, then bind them to new speech directives [30]. This work resembles our own in that it iteratively trained behaviors as compositions of earlier ones. There is also work in *imitation learning,* whereby an agent learns by observing a (not necessarily) human performer [9, 16], though without any trainer correction.

We know of two examples. besides our own, where training or related iterative learning was done *at* RoboCup. The Austin Villa has fed the previous night's results into an optimization procedure to improve behaviors for the next day [20]. Using corrective demonstration, the CMurfs coached a robot to select the correct features and behaviors from a hard-coded set in an obstacle avoidance task during the open technical challenge [3].

We also note that, like our own work, [27] does hierarchical decomposed development of stateless policies, albeit built automatically and for single agents.

2 HiTAB: Hierarchical Training of Agent Behaviors

HiTAB is a multiagent LfD system which trains behaviors in the form of hierarchical finite state automata (or HFA) represented as Moore machines. The system is only summarized here: for a fuller description see [23].

In the single-agent case, an automaton contains some number of *states* which are each mapped to a unique *behavior*, plus a distinguished *start* state whose behavior simply idles. A behavior may be *atomic*, that is, hard-coded, or it may be another finite-state automaton trained earlier. Some atomic behaviors trigger built-in features: for example, transitioning to the *done* (similarly *failed*) state

immediately transitions to *start*, and further signals to the grandparent automaton that the parent HFA believes it is "done" with its task (or "failed"). Other built-in behaviors increment or clear counters. Every state has an accompanying *transition function* which tells HiTAB which state to transition to next time. Each iteration, HiTAB queries the current state's transition function, transitions as directed, then pulses the new state's behavior for an epsilon of time.

The trainer manually decomposes the desired task into a hierarchy of subtasks, then iteratively trains the subtasks bottom-up. In our experience, an experienced trainer need decompose only once. Training an automaton only involves learning its transition functions. In "training mode" the HFA transitions from state to state only when told to by the demonstrator. When the demonstrator transitions from state S to a new state $S' \neq S$, the automaton gathers the robot's current sensor feature vector $\vec{f}$, then stores a tuple $\langle S, \vec{f}, S' \rangle$ as a sample, and in many cases a "default sample" $\langle S', \vec{f}, S' \rangle$. A default sample says "as long as the world looks like $\vec{f}$, continue doing S'", and is added only when transitioning to a continuous behavior (such as *walk*), as opposed to a one-shot behavior (like *kick*).

When training has concluded, the robot enters a "testing mode", at which point it builds an automaton from the samples. To do this, for each i the robot collects all tuples of the form $\langle S_i, \vec{f}, S' \rangle$, then reduces them to $\langle \vec{f}, S' \rangle$. These form data for a classifier $C_i(\vec{f}) \rightarrow S'$ which defines the transition function T_i accompanying state S_i. We use decision trees (C4.5) to learn these classifiers.

The trainer then observes the performance of the automaton. If he detects an incorrect behavior, he may correct it, adding a few new training samples, and then re-build the classifiers. HiTAB can also perform *unlearning*: use the corrective samples to determine which earlier samples had caused the erroneous behavior (either due to sensor noise or user error), then delete them [25]. Finally, the trainer can "undo" an incorrect sample he had just erroneously entered. When he is satisfied with the automaton, he can save it to the behavior library, at which point it becomes available as a behavior (and state) when training a later, higher-level automaton. A behavior saved to the behavior library can be revised in the future without retraining the entire HFA from scratch.

In HiTAB, both basic behaviors and sensor features may be parameterized: thus we may say "go to X" rather than "go to the ball"; and similarly "angle to X" rather than "angle to the nearest teammate". Use of parameterized behaviors or features in an automaton without binding them to ground values results in the automaton itself being parameterized as well. Of course, ultimately each parameter must be bound to a ground value somewhere in the hierarchy: the set of available ground values is, like basic behaviors, hard-coded by the experimenter.

HiTAB is adapted to multiagent scenarios in two ways. First, both homogeneous and heterogeneous interactive teams may be trained through a process we call *behavioral bootstrapping* [24]. The demonstrator starts with robots with empty behaviors, and iteratively selects a robot, trains it with a slightly more sophisticated behavior in the context of the current (simpler) behaviors running on the other robots, then distributes this behavior to similar robots, and repeats. Second, once sufficient joint interactive behaviors have been designed,

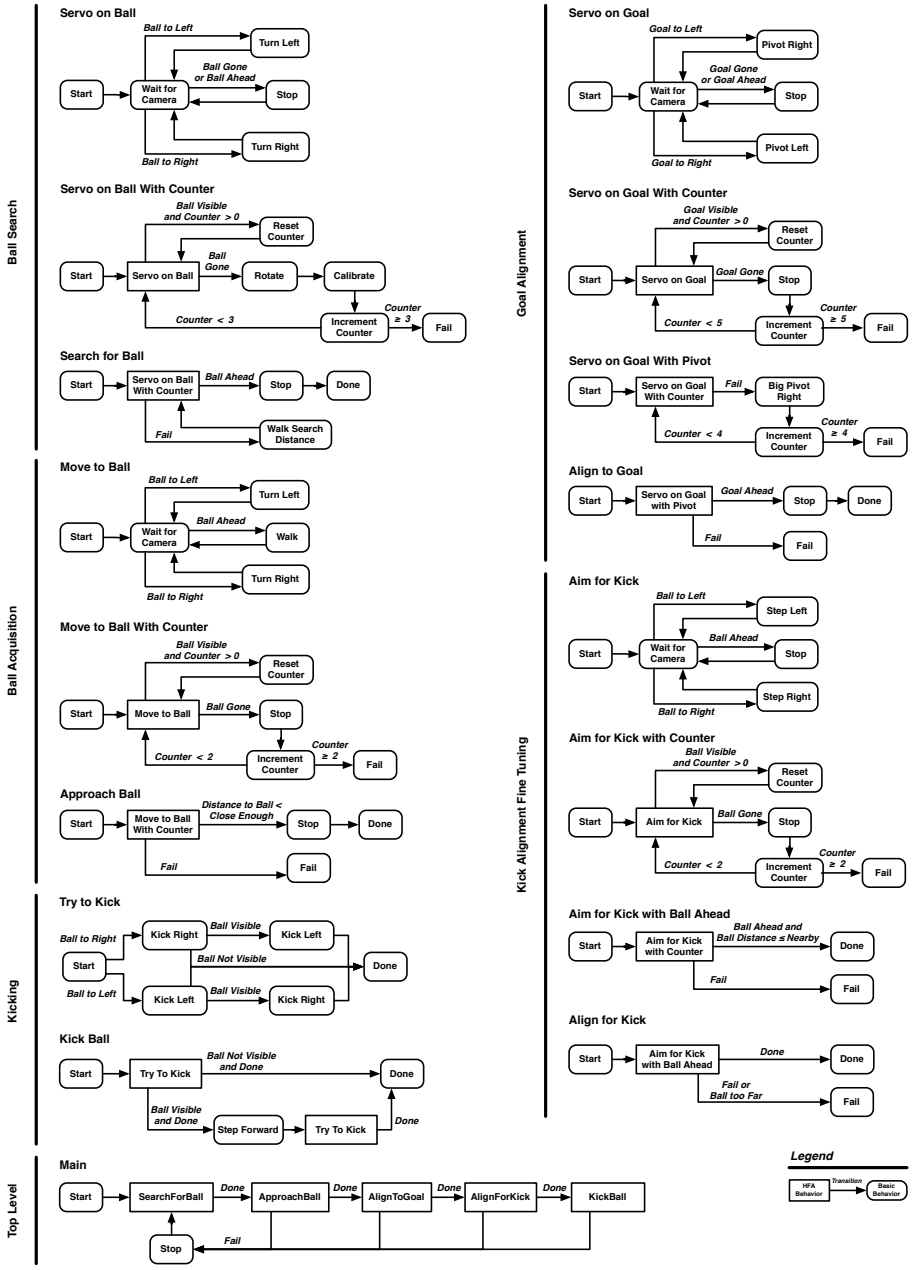


Fig. 1. Trained Hierarchical Finite-State Automaton for RoboCup 2012. Unlabeled transitions are always executed. Note the significant repetition in pattern: part of this is simply behavior similarity, but part is because the 2012 HFA interpreter did not support parameterized behaviors or features (see Section 2).

Is the ball visible?	X coordinate of the ball on the floor	Y coordinate of the ball on the floor
Bearing to the ball	Bearing to the attacker goal	Counter value
Is an HFA done?	Did an HFA fail?	

Table 1. Features in the Robocup 2011 and 2012 Experiments

Continuously turn left	Continuously turn right	Continuously walk forward
Walk forward one step	Sidestep one step left	Sidestep one step right
Stop	Re-calibrate gyros	Increment counter
Pivot left	Pivot right	Reset counter
Kick left	Kick right	Signal "Done"
Signal "Failed"	Wait for camera	

Table 2. Basic behaviors in the Robocup 2011 and 2012 Experiments

small teams of homogeneous or heterogeneous robots may be grouped together under a *controller agent* whose atomic behaviors correspond to the joint trained behaviors of its subordinates, and whose features correspond to useful statistical information about the subordinates. The controller agent is then trained using HiTAB. Homogeneous and heterogeneous controller agents may likewise be trained together, then put under their own controller agent, and so on, thus iteratively building entire swarms into a trained hierarchy of command. We have used HiTAB to train groups of many hundreds of agents [23].

3 Our Previous Efforts at RoboCup

The RoboPatriots have been GMU's entry in the RoboCup Humanoid League from 2009 to present. Initially the RoboPatriots focused on issues related to robot design, dynamic stability, and vision processing, and we exclusively used hand-coded behaviors. Then at RoboCup 2011, we demonstrated a HiTAB-trained robot as a proof-of-concept. The night before the competition, we deleted one of the hard-coded behaviors (ball servoing) and trained a behavior in its place through direct tele-operation of the robot on the field of play. We then saved out the trained behavior, and during the competition, one attacker loaded this behavior from a file and used it in an interpreter alongside the remaining hard-coded behaviors. This trained behavior was simple and meant as a proof of concept, but it worked perfectly.

In 2012 we had a much more ambitious goal: to train the entire library of behaviors of a single robot on the field immediately prior to the competition. Our attacker robots in 2012 used a decomposition of 17 automata which collectively defined a simple "child soccer" style of behaviors without localization: search for the ball, approach the ball, align to the goal, align for kicking, kick, and repeat. Two days before the competition, we deleted the entire behavior set and proceeded to train an equivalent set of 17 automata in its place (Figure 1), again through tele-operation of the robot on the competition field. The final HFA was saved to disk and run through an interpreter during game play.

The basic sensor features and robot behaviors we relied on to build these automata are given in Tables 1 and 2 respectively: these were essentially the same basic sensor features and behaviors used in the hard-coded version. Note that not all features and behaviors were used in every HFA. The *Wait for Camera* behavior

Behavior	*Number of Samples*	*Number of Provided Samples*
ServoOnBall	11	11
ServoOnBallWithCounter	(estimate) 10	(estimate) 9
SearchForBall	10	8
MoveToBall	9	9
MoveToBallWithCounter	10	9
ApproachBall	15	11
ServoOnGoal	9	9
ServoOnGoalWithCounter	12	11
ServoOnGoalWithPivot	9	7
AlignToGoal	12	9
AimForKick	9	9
AimForKickWithCounter	10	9
AlignForKickWithBallAhead	22	14
AlignForKick	42	35
TryToKick	10	10
KickBall	9	6
Main	34	19
Total	243	195

Table 3. Number of data samples for each HFA trained at RoboCup 2012. *Provided Samples* are those directly provided by the user and do not include automatically inserted "default samples" for continuous sub-behaviors. The data for ServoOnBallWithCounter was not saved, so the estimate is based on other HFAs which used a counter.

ensured that we had new and complete vision information before transitioning (our vision system was slower than the HFA).

The top-level HFA behavior, *Main*, performed "child soccer" by calling the following second-level behaviors, which triggered additional hierarchical behaviors:

- *Search for Ball:* Using the bearing to the ball, the robot did visual servoing on the ball, with the additional constraint of performing a rotation if the ball was missing for several frames. If the robot had rotated several times, it then walked forward before resuming searching.
- *Approach Ball*: Using the bearing to the ball and distance to the ball, the robot moved towards the ball while performing course corrections en route.
- *Align to Goal:* Using the bearing to the goal, the robot oriented toward the goal while maintaining the ball near the robot's feet. The robot pivoted around the ball if it could not see the goal.
- *Align for Kick:* Using the $\langle X, Y \rangle$ position of the ball, the robot took small steps to get the ball in a box near its feet so a kick could be performed.
- *Kick Ball:* The robot kicked based on the X position of the ball. If after a kick the ball was still there, then the robot would kick with its other foot. If the ball was *still* there, the robot would take a step forward and repeat.

Issues such as referee box event response and recovery from falls were handled with hard-coded logic (in the second case, resetting to *Search for Ball*). The HFA included subroutines designed to handle high sensor noise: for example, *MoveToBallWithCounter* would robustly handle the ball disappearing due to a temporary camera error.

HiTAB can be used to rapidly retrain behaviors as needed. As an example, we had to train an additional HFA after the first day of competition. During our

early matches, we observed that the *Aim for Kick* sub-behavior assumed that the ball would consistently be near the robot's feet. However, due to sensor noise the robot might enter *Align to Goal* when the ball was far away, and so when *Aim for Kick* was entered, it would take many, many baby steps towards the ball. We then trained a new version, *Aim for Kick With Ball Ahead* to also include a failure situation for when the ball was outside a box centered at the robot's feet. The new HFA was then used in our later matches.

Table 3 shows the number of samples collected for all 17 trained HFAs. The first column includes automatically inserted default samples while the second column shows only the directly provided samples. Given the problem complexity, we were able to train on a remarkably small number of samples.

During our second match versus Team JEAP from Osaka University, **our trained robot scored the winning goal.** After discussion with colleagues at the competition, we believe that, to the best of our knowledge, this is the first time a competing robot at RoboCup has used a full behavior set trained in real time at the venue itself, much less scored a goal using those trained behaviors.

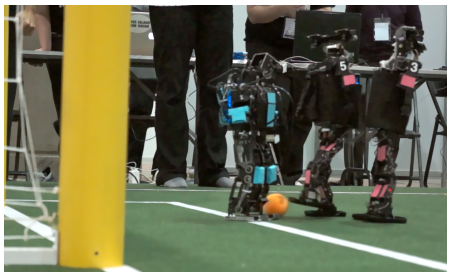


Fig. 2. GMU's trained Johnny-5 (magenta #5) kicks the winning goal against Osaka.

4 Penalty Kick Experiments

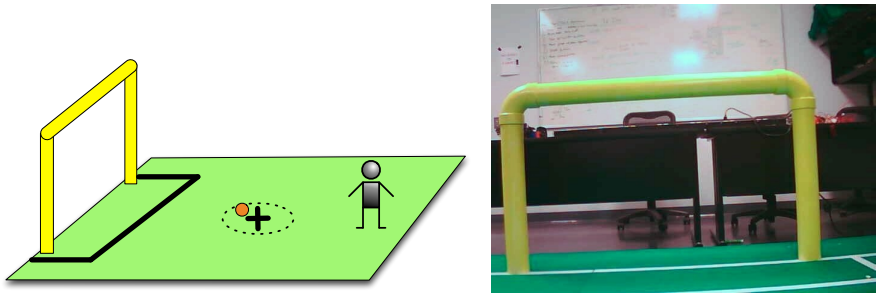
One claimed benefit of LfD is that the trained behaviors perform as well as hand-coded behaviors. After RoboCup 2012, we conducted experiments to verify this claim by comparing our trained soccer behavior with the hand-coded behavior deployed on our other attacker. The task was penalty kicks, similar to those used during the RoboCup competition.

The robot was placed 40 cm away from the penalty kick mark with a neutral head position and facing the goal. The ball was randomly placed within a 20 cm diameter circle centered on the penalty kick mark (see Figure 3(a)). Initially, the robot could see the goal, but not the ball, as shown in Figure 3(b). The metric was time to kick the ball, independent of whether a goal was scored. Both behaviors were run 30 times.

Figures 4(a)-(b) show histograms for the hard-coded and trained behaviors. For both behaviors, sensor noise caused one run to take significantly longer than the rest. The trained behavior had a mean execution time of 37.47 ± 5.51 seconds (95% confidence interval), while the hardcoded behavior had a mean of 35.85 ± 3.08. The means were not statistically significantly different.

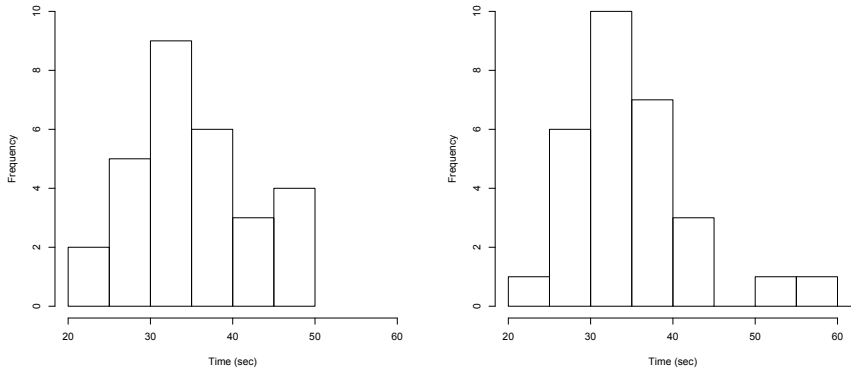
5 Set Plays: A Multiagent Training Proof of Concept

For RoboCup 2014 our goal is to train not just a single robot but a full team of humanoids to play interactive robot soccer. To that end we have begun with an experiment in multi-robot training on the soccer field: set plays.



(a) Penalty kick experimental layout (b) Robot starts facing the goal

Fig. 3. Penalty kick experimental layout. The robot cannot initially see the ball.



(a) Hardcoded results (b) Trained results

Fig. 4. Penalty kick results. In both experiments, one run took longer than 60 seconds.

Multi-robot training is notionally difficult because of the interaction among the robots and the challenges faced in coordinating them. To attack this problem at scale, HiTAB relies on manual decomposition of a swarm of agents under a hierarchy of trained "controller agents". However for small groups (two to four agents) we focus instead on developing joint behaviors among the agents. This is the case for the set-play scenario, which typically involves two agents.

How might one use HiTAB to train an interactive joint behavior among two robots without a controller agent coordinating them? We see three possibilities:

- *Train the Robots Independently* We train one robot while tele-operating the other (the *dummy*), and vice versa. This is the simplest approach, but to us it does not intuitively feel like a match for multiagent training scenarios which involve a significant degree of interaction.
- *Bootstrap* We train one robot to perform a rudimentary version of its behavior with the other robot doing nothing. We then train the second robot to do a slightly more sophisticated version of its own behavior while the first

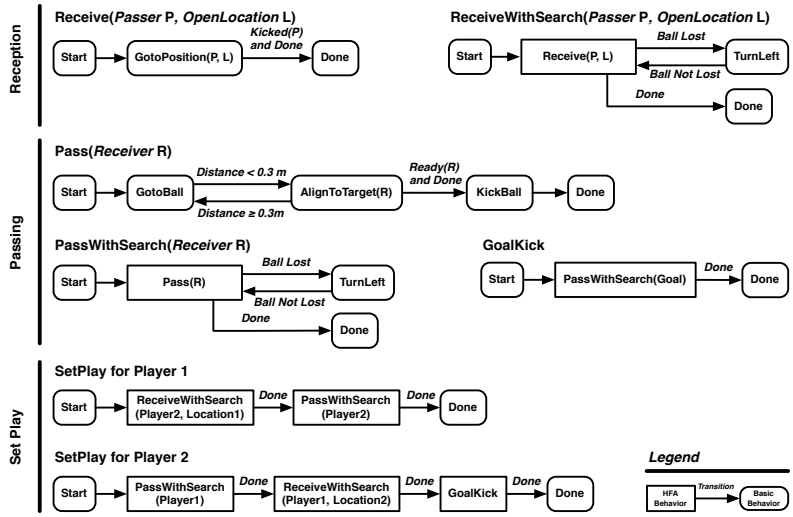


Fig. 5. Trained Hierarchical Finite-State Automata for the 2014 set-play experiments. Passing and reception automata are shared among both robots, but each robot executes a different top-level set play automata.

robot is performing its trained rudimentary behavior. This back-and-forth training continues until the robots have been fully trained.

– *Simultaneously Train* We use two HiTAB sessions, one per robot, to train the robots at the same time while interacting with one another. This obviously requires much more effort on behalf of the demonstrator (or multiple demonstrators working together).

For 2014 we have new robots (Darwin-OP humanoids) and so have decided to base our system on a heavily modified version of the UPennalizers's open-sourced 2013 champion software. This code provides localization and helpful behaviors which we use as the foundation for basic behaviors and features in the set plays:

– *GotoPosition(P, L)* goes to location L on the field, facing the location of player or object P, then broadcasts a "Ready" signal for five seconds.
– *GotoBall* goes to the ball position.
– *AlignToTarget(R)* orients around the ball until the robot is facing player or object R.
– *KickBall* kicks the ball and broadcasts a "Kick" signal for five seconds.
– *TurnLeft* rotates to the left.

Each robot was also equipped with the robot sensor features *Kicked(P)* (did P raise the Kick signal?), *Ready(P)* (did P raise the Ready signal?), *Ball Lost* (has the ball been lost for over three seconds?), and *Distance* (to ball). Note that the Goal, as a parameter, was considered to be always "Ready".

Clearly these behaviors and features are higher-level than those used in 2012, and the resulting automata are simple for a programmer to implement. We took

(and are continuing to take) such baby-steps on purpose: real-time training of multirobot behaviors is notionally nontrivial, and previous examples for guidance are few and far between. Our goal is to show that such a thing is even feasible.

Using this foundation, we trained the robots independently via dummies to perform the joint set play behaviors shown in Figure 5: Robot A would acquire the ball while B moved to a preset position. When both were ready, Robot A would then kick to B and move to a second preset position. Then Robot B would kick to A, which would then kick to the goal.

Though we had imagined that we would need to perform simultaneous training or bootstrapping, in fact we have been perfectly successful in training set plays separately using dummies. This surprising result is likely due to the small number (two) of robots involved, but it has nonetheless forced us to question the prevailing wisdom: does interaction *necessarily* complicate multi-robot learning?

Whether independent training will be sufficient for the remainder of the behaviors for 2014 remains to be seen: and ultimately we will need to train a virtual controller agent (likely residing on the goalie) to direct which behaviors and joint actions should be undertaken by the team at any given time.

6 Conclusion: A Technical Challenge Problem Proposal

In this paper we outlined our efforts so far towards an unusual and challenging goal: to successfully train a full robot soccer team on the field of play at the RoboCup competition. We think that a "personal technical challenge" like this is not only a useful research pursuit, but it also has direct impact on robot soccer. After all, coaching and training players is an integral part of the sport! People are not born with, nor hard-coded, to play soccer: they learn it from demonstration and explanation from coaches and through the imitation of other players.

To this end, we propose a new yearly challenge problem for RoboCup involving collaborative multiagent LfD (beyond just an open challenge). RoboCup teams would yearly be presented with a brand new task, and they would have four hours to train their robots to collectively perform that task. The robots might be asked to do a certain set play; or to collectively form a bucket brigade to convey balls from one corner of the field to the other. In earlier years teams might be informed of the task a month before; or the tasks might be restricted to single agents. But eventually the task should require multiple interacting agents and few clues provided beforehand except for the basic behaviors permitted. Differences in robot hardware or software architectures might constrain the available techniques, and so the challenge might need to be more a showcase than a judged competition.

Acknowledgments Research in this paper was done under NSF grant 1317813.

References

1. Argall BD *et al.* A survey of robot learning from demonstration. *Robotics and Autonomous Sytems*, 57, 2009.
2. Bentivegna DC *et al.* Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47(2-3):163–169, 2004.

3. Çetin Meriçli *et al.* Multi-resolution corrective demonstration for efficient task execution and refinement. *International Journal of Social Robotics*, 4, 2012.
4. Chernova S. *Confidence-based Robot Policy Learning from Demonstration.* Ph.D. thesis, Carnegie Mellon University, 2009.
5. Fountain J *et al.* Motivated reinforcement learning for improved head actuation of humanoid robots. *RC.* 2013.
6. Hausknecht M and Stone P. Learning powerful kicks on the Aibo ERS-7: the quest for a striker. *RC.* 2010.
7. Kalyanakrishnan S *et al.* Half field offense in RoboCup soccer: a multiagent reinforcement learning case study. *RC.* 2006.
8. Kaminka GA *et al.* Learning the sequential coordinated behavior of teams from observations. *RC.* 2002.
9. Latzke T *et al.* Imitative reinforcement learning for soccer playing robots. *RC.* 2006.
10. Luke S *et al.* Co-evolving soccer softbot team coordination with genetic programming. *RC.* 1997.
11. Martins MF and Demiris Y. Learning multirobot joint action plans from simultaneous task execution demonstrations. *AAMAS*, 931–938. 2010.
12. Merke A and Riedmiller M. Karlsruhe Brainstormers — a reinforcement learning approach to robotic soccer. *RC.* 2001.
13. Metzler S *et al.* Learning visual obstacle detection using color histogram features. *RC.* 2011.
14. Nakanishi J *et al.* Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91, 2004.
15. Nakashima T *et al.* Performance evaluation of an evolutionary method for RoboCup soccer strategies. *RC.* 2005.
16. Noda I. Hidden markov modeling of team-play synchronization. *RC.* 2003.
17. Oubbati M *et al.* Velocity control of an omnidirectional RoboCup player with recurrent neural networks. *RC.* 2005.
18. Saggar M *et al.* Autonomous learning of stable quadruped locomotion. *RC.* 2006.
19. Schwarz M and Behnke S. Compliant robot behavior using servo actuator models identified by iterative learning control. *RC.* 2013.
20. Stone P. Personal conversation, 2014.
21. Stone P and Veloso M. Layered learning and flexible teamwork in RoboCup simulation agents. *RC.* 1999.
22. Stone P *et al.* Keepaway soccer: From machine learning testbed to benchmark. *RC.* 2005.
23. Sullivan K and Luke S. Learning from demonstration with swarm hierarchies. *AAMAS.* 2012.
24. Sullivan K and Luke S. Real-time training of team soccer behaviors. *RC.* 2012.
25. Sullivan K *et al.* Unlearning from demonstration. *IJCAI.* 2013.
26. Takahashi Y *et al.* Behavior acquisition based on multi-module learning system in multi-agent environment. *RC.* 2002.
27. Takahashi Y *et al.* A hierarchical multi-module learning system based on self-interpretation of instructions by coach. *RC.* 2003.
28. Tuyls K *et al.* Reinforcement learning in large state spaces. *RC.* 2002.
29. Visser U and Weland HG. Using online learning to analyze the opponents behavior. *RC.* 2002.
30. Weitzenfeld A *et al.* Coaching robots to play soccer via spoken-language. *RC.* 2008.
31. Wilking D and Röfer T. Realtime object recognition using decision tree learning. *RC.* 2004.
32. Zagal JC and del Solar JR. Learning to kick the ball using back to reality. *RC.* 2004.