RoboPatriots: George Mason University 2014 RoboCup Team

David Freelan, Drew Wicke, Chau Thai, Joshua Snider, Anna Papadogiannakis, 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, cthai3@masonlive.gmu.edu, jsnider3@masonlive.gmu.edu, apapado2@masonlive.gmu.edu, sean@cs.gmu.edu

Abstract. The RoboPatriots are a team of four DARwIn-OP robots from George Mason University which participate in the Kid-Size Humanoid League. RoboCup 2014 marks the fifth year of participation for the RoboPatriots: in 2009 and 2010, we advanced to the second round, and in 2011 we were eliminated in the first round. Our approach is unusual in that we aim to train our robots how to play soccer using learning from demonstration, then field those robots in the competition.

1 Introduction

The 2014 RoboPatriots are a team of four DARwIn-OP humanoid robots which compete in the Kid-Size Humanoid League at RoboCup. Our goal for the competition this year is unusual: we will try to train the entire team how to play soccer, on the field at RoboCup, and then enter the trained behaviors in the competition. Because this year RoboCup has cut the setup days to one, we may not have enough time to train the robots at the venue: in this case some robots may use behaviors we had trained at George Mason University immediately prior to departure.

We have had past success using this technique at RoboCup, but in not as ambitious a goal as this year. In RoboCup 2011 we deleted a single hard-coded behavior (servoing on the ball) from one robot, then used successfully HiTAB to learn that same behavior on the RoboCup humanoid league soccer field the day before the competition. One attacker on the team used this learned behavior along with its hard-coded behaviors; for comparison, the other attacker used all hard-coded behaviors.

In RoboCup 2012 we went significantly further. We deleted all soccer behaviors from one of the robots, leaving behind only simple actions such as "move forward" or "turn", and basic sensor information such as the location of the ball in the robot's field of view. We then used HiTAB to train a hierarchy of seventeen finite-state automata which recreated the entire play behavior except for referee interaction.

This year our goal is to train all the soccer behaviors of multiple robots, and ideally all robots on the team. We do not intend to just train four separate robot behaviors, but rather train them as an interactive multiagent system: for example, training them to work together to perform set plays. If successful, we will have fielded an entire team of soccer robots which had been trained from scratch by a human demonstrator.

2 Hardware

We used four DARwIn OP humanoid robots with the following specifications:

- Height 454.5 mm
- Weight 2.9 Kg
- Number of DOF 20 (6 DOF per leg, 3 DOF per arm, 2 DOF head)
- Dynamixel RX-28M
- Main controller: Intel Atom Z530 CPU (@1.6GHz)
- Sub controller: STMicroelectronics Cortex-M3 STM32F103RE (@ 73MHz).
- Camera Logitech C905, HD video capture (up to 1600x1200@10fps, 1280x720@30fps)
- Walking Speed 24 cm/s

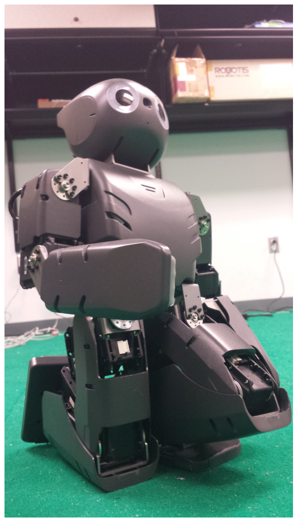
Fig. 1. GMU Darwin OP

3 Software Architecture

The RoboPatriots' software is a hybrid of GMU learning from demonstration system (HiTAB, discussed in Section 4), and the U Penn robot software which won in 2013. As presently envisioned, the architecture will use U Penn's localization, vision, and gait code, but the high-level soccer behavior code will be entirely replaced with an interpreter which performs learned soccer behaviors developed using HiTAB. Given time and need we will further modify U Penn code beyond this.

4 Multirobot Learning from Demonstration

HiTAB is a learning from demonstration system developed at George Mason University by which a human demonstrator or coach can train one or more robots or virtual agents to perform nontrivial behaviors in effectively real-time [2].

Single-Agent HiTAB Research in learning from demonstration may be roughly divided into two categories: research in learning trajectories or paths, and research in learning plans, automata, and behavioral policies. In the first case the learning algorithm may receive a great many samples, as every small movement is a data point. In the second case however, samples typically only arrive at a transition from behavior to behavior, and so are often very sparse. Our approach falls in this second category.

Sparsity is a serious problem, since the learning space is can be high dimensional. For example, if one were training a finite-state automaton describing all of soccer play, this might consist of some twenty states and an equal number of environment features on which transitions are based. Thus we must develop a way to reduce the dimensionality of the space, or otherwise simplify the problem, in order to make learning possible given the few samples available.

Our approach learns behaviors in the form of hierarchical finite-state automata (HFA) represented as Moore machines. Each state in an HFA maps to a behavior, and when the HFA is in that state, that corresponding behavior is performed by the robot. Behaviors may be hard-coded basic behaviors such as "walk forward" or "kick", or they may themselves be other HFA. No cyclic recursion is permitted. Associated with each state in an HFA is a corresponding transition function which tells the HFA which state it should transition to next. Transition functions are often based on the current sensor feature vector. Features may be parameterized, and so instead of "distance to the ball" or "distance to the goal", a feature may be defined simply as "distance to X", where X is left to be specified later. Similarly, basic behaviors may be parameterized, and all this in turn allows HFA to be parameterized, resulting in general-purpose behaviors such as "go to X" rather than "go to the ball". This parameterization allows us to train a general-purpose behavior once, then reuse it multiple times in different contexts, which helps in reducing dimensionality.

HFA are run by iteratively pulsing them. Each time an HFA is pulsed, it first calls the transition function associated with its active state, then transitions to the state returned by this function (which can of course be the same state). It then pulses the behavior associated with this new active state. If the behavior is itself another HFA, this process recurses, until ultimately a basic behavior is pulsed, which causes the robot to perform that behavior for a short period of time.

Our model learns the transition functions associated with each HFA, but not the states. Rather, each state is mapped to a unique behavior from the union of basic behaviors and currently learned HFA. This simplifying assumption is done in the name of dimensionality reduction. Because the behaviors are fixed, so are the number of transition functions to learn. Each transition function is essentially a mapping of the set of possible feature vectors to the (finite and unordered) set of states, and is so nothing more than a classifier. Thus to learn an HFA, HiTAB builds a set of classifiers, one per state in the automaton.

Training is done as follows. The demonstrator first determines the features to form the feature vector f for the HFA. He then iteratively selects behaviors from the current behavior library, and when he does so the robot performs those behaviors. When a new behavior b_t is selected, the robot stores a transition sample of the form $\langle b_{t-1}, f_t, b_t \rangle$, and also when appropriate a default ("keep on doing b_t") sample of the form $\langle b_t, f_t, b_t \rangle$. Ultimately the demonstrator asks the robot to learn from the demonstrations, at which point for each behavior b, the robot gathers all samples of the form $\langle b, f, b' \rangle$ for various f and b'. These are then reduced to samples of the form $b' \to f$ (that is, b' is the data point and f is is its class label), and from these samples the robot builds the classifier for the transition function associated with b.

Armed with its own transition functions, the robot then begins performing the HFA. At any point the demonstrator may jump in and correct errant behavior, resulting in additional samples. When the demonstrator is satisfied with the behavior, it is saved to the behavior library and becomes available as a corresponding state in a higher level HFA trained later.

Using this training approach it is theoretically possible to learn large, complex automata; but our intent is instead to permit the trainer to learn a hierarchy of simpler automata. This requires that the trainer manually decompose the behavior into simpler and simpler sub-behaviors, then iteratively learn each sub-behavior in turn bottom-up. In doing so, the trainer effectively projects the full joint space of the behavior into the much smaller subspaces of each of the sub-behaviors, dramatically simplifying the total learning space and dimensionality. Further, each sub-behavior may have its own reduced set of features appropriate for that sub-behavior, rather than requiring the full joint feature set.

Multiagent HiTAB Multiagent learning from demonstration presents a much more challenging problem than the single-agent case, because of the *multiagent inverse problem*. Learning from demonstration is fundamentally a supervised learning task: the demonstrator shows what must be done in various situations, and the agent learns from this. However in the multiagent case, even if the demonstrator can quantify what emergent macro-phenomenon he wishes the multiple agents to achieve in any given situation, in order to train them he must break this down into those individual micro-level agent behaviors which collectively achieve this. Unfortunately, while we can use an agent or robot simulator effectively as a "forward" function which tells us what macro-behavior arises from the combination of specific micro-level behaviors, we do *not* have the inverse function, that is, a function which tells us what micro-level behaviors are necessary to achieve a given macro-behavior. But this inverse function is exactly what the demonstrator needs.

The standard way to overcome inverse problems is through optimization. As such nearly the entire multiagent learning literature has consisted of optimization methods: stochastic optimization (genetic algorithms etc.) or reinforcement learning. Supervised techniques are rare. Furthermore, multiagent *learning from demonstration* cannot readily use optimization techniques because it typically has no simulator to provide the "forward" function to optimize over. As a result, this area has a very limited literature: of the few supervised methods, most fall instead in the category of *agent modeling*, where robots or agents learn about one another rather than about a task given to them by demonstrator. The most common multirobot learning from demonstration approach is to eliminate macro-behaviors entirely by issuing separate micro-level training directives to each individual agent [3–5]. This is very unlikely to scale. Another recent approach is to build up homogeneous behaviors through via confidence estimation rather than reinforcement learning [1].

We have taken a different tack: to once again use decomposition to simplify the inverse problem to the point where the gulf between micro- and macro-level behaviors is so small that it is obvious what micro-level behaviors must be learned. We do this by first manually breaking the swarm of agents into a hierarchy of smaller and smaller sub-swarms, ultimately down to individual agents. We then train individual agents with any needed fundamental single-agent behaviors. Then we train the smallest groups of agents (perhaps 2 or 3) to perform collective homogeneous interactive behaviors. Once these are learned, we can then train a virtual *controller agent* which directs the homogeneous behaviors of this group. The controller agent's "basic" behaviors map to the high-level interactive behaviors of its subordinates; whereas the controller agent's "features" are (hard-coded) statistical information about its subswarm (such as "percentage of agents who have fallen over" or "did someone score a goal?" or "centroid of the swarm").

This continues recursively: the controller agent develops various HFA behaviors as necessary, then we group controller agents together to learn interactive behaviors, and finally put that group under the control of a higher-level controller agent, and so on, until the entire swarm or team is joined.

This can also be done with heterogeneous behaviors: but in this case the controller must direct different behaviors to different agents rather than a single behavior. We do this by explicitly mapping basic controller behaviors to groups of heterogeous behaviors among its underlings.

5 Training Soccer Robots

The RoboPatriots are a four-robot heterogeneous team. Our goal this year will be to train them with interactive behaviors, demonstrating set plays, passing and receiving, and so on, using HiTAB. As there are four robots, we will use a controller agent to decide which set play to perform. We will train the behaviors for our four robots before the competition at George Mason University, and will then re-train them at the competition given sufficient time.

The trained architecture will probably be a flat swarm of four basic robots with no controller agent; though we may construct a controller hierarchy among the non-goalies if it proves useful.

For simplicity, training will not be on-board the robots: rather we will develop the trained behaviors on a remote laptop using one or more robots as clients. Sensor information from the robots will be communicated in real time to the laptop and used as features for HiTAB training, and likewise basic behaviors in HiTAB will be translated in real-tie to the robots to perform as actions (such as kicking or walking). Once the training is completed, we will transfer the learned behaviors to the robots and run them in an interpreter on-board.

6 Conclusions

We have described hardware and software of the RoboPatriots, a team of three humanoid robots developed at George Mason University. The RoboPatriots are a primary research platform for our learning from demonstration system, HiTAB, which is geared to training nontrivial hierarchical behaviors on teams of multiple cooperative robots.

Statement of Commitment

The RoboPatriots commit to participate in RoboCup 2014 in Brazil and to provide a referee knowledgeable of the rules of the Humanoid League.

Acknowledgments

This work is supported by NSF grants IIS-0916870 and IIS-1317813. We would be amiss if we did not acknowledge various students and faculty who have been involved in developing HiTAB over the past several years: notably Keith Sullivan, Bill Squires, Ahmed Elmolla, Ermo Wei, and Vittorio Ziparo

References

1. Chernova, S.: Confidence-based Robot Policy Learning from Demonstration. Ph.D. thesis, Carnegie Mellon University (2009)
2. Luke, S., Ziparo, V.: Learn to behave! rapid training of behavior automata. In: Grześ, M., Taylor, M. (eds.) Proceedings of Adaptive and Learning Agents Workshop at AAM AS 2010. pp. 61 – 68 (2010)
3. Martins, M.F., Demiris, Y.: Learning multirobot joint action plans from simultaneous task execution demonstrations. In: Proceedings of Autonomous Agents and Multi-Agent Systems Conference (AAMAS). pp. 931–938 (2010)
4. Martins, M.F., Demiris, Y.: Learning multirobot joint action plans from simultaneous task execution demonstrations. In: Proceedings of Autonomous Agents and Multi-Agent Systems Conference (AAMAS). pp. 931–938 (2010)
5. Takács, B., Demiris, Y.: Balancing spectral clustering for segmenting spatiotemporal observations of multi-agent systems. In: IEEE Internationa Conference on Data Mining (ICDM). pp. 580–587 (2008)