RoboPatriots: George Mason University 2015 RoboCup Team

David Freelan, Drew Wicke, Chris Burns, Carl Walker, Laura Hovatter, Colin Ward, Daniel Lofaro, 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, cburns9@masonlive.gmu.edu, cwalke11@masonlive.gmu.edu, lhovatte@masonlive.gmu.edu, cward13@masonlive.gmu.edu, dlofaro@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 2015 marks the sixth year of participation for the RoboPatriots. Our approach is very unusual in that our goal is to train our robots how to play cooperative soccer, rather than program them. We do this not at our laboratory, but at the RoboCup venue during the preparatory period. Then, we enter the learned robot behaviors into the competition. In 2014 we trained all three attackers, pairing them with a hard-coded goalie. This year we intend to train a full team of all four robots.

1 Introduction

The 2015 RoboPatriots are a team of four DARwIn-OP humanoid robots which compete in the Kid-Size Humanoid League at RoboCup. Unlike all other teams in the competition, we attempt to train our robots how to play soccer *at the venue*, and then enter the result into the competition. We do this training during the preparatory period: if we run out of time, we may instead field behaviors we have trained at George Mason University immediately prior to departure, but in all past competitions we have not had to resort to this.

The approach we take to training robots is a learning from demonstration system called HiTAB, short for "Hierarchical Training of Agent Behaviors". In this method, we first manually decompose the joint and individual behaviors of the team into a hierarchy, then iteratively train this hierarchy bottom-up. This approach allows us to break large, complex behaviors into simpler behaviors, each with a much smaller dimensionality, thus permitting us to use many fewer samples and successfully train in real time. We also apply decomposition in a different way to organize teams of agents into hierarchies, and perform training bottom-up through the collective hierarchy as well. This permits us to overcome the inherent inverse problem which has stymied multiagent learning from demonstration in the past.

RoboCup 2011 was the first year we attempted any training using HiTAB. That year we deleted a single hard-coded behavior (ball servoing) from a single robot, then successfully used HiTAB to train that same behavior on the RoboCup humanoid league soccer field the day before the competition. We fielded the attacker with this learned behavior along with another attacker consisting of all hard-coded behaviors (for comparison).

In RoboCup 2012 we went significantly further. We deleted all soccer behaviors from one of the robots, leaving behind only basic hard-coded 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 robot successfully kicked a winning soccer kick against TeamOSAKA: we believe this is the first time a robot has ever been taught how to play soccer from scratch *at RoboCup*, and then won a match using what it had learned.

In 2014 we moved on to the multiagent case: we trained three attackers to play collaborative soccer, including set plays, using the basic behaviors provided by the UPennalizers', such as "achieve position X" or "kick to goal". This was again done during the preparatory period. This involved training 24 finite-state automata, which produced high level behaviors such as switching between offense and defense, and cooperative behaviors such as passing. Our goalie's behaviors were hard coded.

This year our goal is to train collaborative soccer behaviors for the entire team, including the goalie. 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
- DOF: 20 (6 per leg, 3 per arm, 2 in the head)
- Servos: Dynamixel RX-28M
- Controller: Intel Atom Z530 CPU (@1.6GHz)
- Subcontroller: STMicroelectronics Cortex-M3 STM32F103RE (@ 73MHz)
- Camera: Logitech C920
- Walking Speed: 24 cm/s, using 3D printed cleats

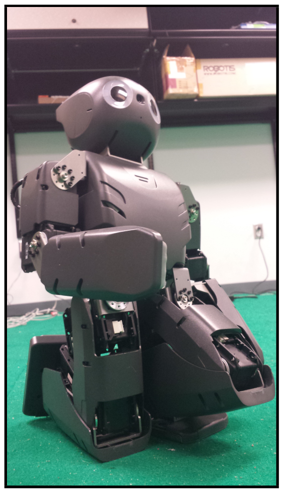
Fig. 1. A RoboPatriot

3 Software Architecture

The RoboPatriot software is a hybrid of GMU's learning from demonstration system (HiTAB, discussed in Section 5), and the UPennalizer software which won in 2013. All the higher level soccer behavior code is replaced with HiTAB. Furthermore, this year, gaits and vision architecture must be almost entirely replaced to handle the new carpet, ball changes, and white goal posts.

4 Robocup 2015 Challenges

In the 2015 Humanoid League, major changes have been made to the environment. For us, new concerns include the radically different floor surface, the ball color and size, and the goalpost color. To tackle the visual challenges, we must re-write the vision system entirely. We have changed our camera from the Logitech C905 to the Logitech C920, and are re-writing all previous tracking and identification algorithms: for example, we will detect balls by identifying a circular hole in the field rather than relying solely on blob tracking.

The new floor surface is underspecified, notionally 30mm artificial grass. To accommodate this, we have 3D printed special cleats for the robot to wear. The cleats settle into the grass to improve the robot's ability to grip the surface, and also allow the bottom of the foot to settle on the artificial grass more consistently. We are also developing new gaits to pair with the cleats and the new surface.

5 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 effectively in 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 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 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 a mapping of the set of possible feature vectors to the (finite and unordered) set of states, and is 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 that form the feature vector f for the HFA. He then iteratively selects behaviors from the current behavior library, and when he does 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 $f \rightarrow b'$ (that is, f is the data point and b' 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 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 progressively less complex sub-behaviors, then iteratively learn each sub-behavior bottom-up. In doing so, the trainer effectively projects the full joint space of the behavior into the much smaller subspaces of each 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 the 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 gap 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 progressively 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.

Heterogeneous Multiagent HiTAB This year we are focusing on heterogeneous collective behaviors. In some cases collective behaviors must be heterogeneous because the robots themselves are heterogeneous in design or capability (for example, attackers versus goalies). In other cases the behaviors are heterogeneous because even though the robots are identical, they must use different simultaneous behaviors to collaborate (for example, attackers and midfielders). The primary new challenge in heterogeneous agents is how to map them to the behaviors found in a single controller: whereas before a controller's basic behaviors would direct homogeneous behaviors in all of its underlings, now the controller's basic behavior must map to *joint behaviors* among different kinds of underlings.

To do this, we first train small heterogeneous groups to do certain basic joint behaviors. In a given joint behavior B, robots of each robot type i perform some unique behavior B_i designed such that all such B_i collectively achieve B. This behavior B is then mapped to a basic behavior in the controller agent, allowing us to train the controller as before. Though controller behaviors can be more or less automatically mapped, we are faced with hard-coding the *sensor* data of the controller agent to statistical information gathered from his subordinates, such as their average location; or how many robots are left on the field; or whether the goalie has fallen.

One possible sensor the controller might receive is a *signal* sent to it by a subsidiary agent indicating that it has (for example) completed a task or failed at it. This would allow the controller to coordinate behaviors such as set plays among agents. An alternative approach is to eliminate the controller entirely and have agents signal to one another, and be able to sense one anothers' signals as part of their sensor suite. For very small teams such as is found in RoboCup, this is a simpler method than the controller method, though it does not scale to larger numbers agents. We may try both of these approaches. Additionally, we will devise fall-back behaviors for the (now seemingly inevitable) situation where wireless does not work at RoboCup.

6 Training Soccer Robots

The RoboPatriots are a four-robot heterogeneous team. Our goal this year is to train interactive behaviors, demonstrating set plays, passing and receiving, goal tending, and so on, using HiTAB. We will train the behaviors for our four robots before the competition at George Mason University (just in case), and will then re-train them at the competition.

Last year, we succeeded in training three cooperative attackers. This year, we plan to train the goalie and the attackers collectively. However, there are a number of new challenges for HiTAB to face. The first is the unreliable wireless at the Robocup venue, which our previous HFAs relied on. This year we may add to the wireless-based signaling some kind of visual cueing, given by one robot to indicate either its current state, or the environment's state. For example, these cues may include "waiting for pass", or "ball is on our side of the field". We will also design the behaviors such that, if all communication pathways are unavailable, they can still play individually.

We will train the behaviors using a laptop with one or more robots as clients (essentially effectors of the laptop). 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-time to the robots to perform as actions (such as kicking or walking). Some sensor information, such as signaling "waiting for pass", will be done using a "dummy agent". A dummy agent is a virtual agent whose state can be artificially controlled, without the robot having to be in that state in the environment. For instance, suppose we are training Robot A to pass the ball when Robot B is in the state "waiting for pass". To make training both easier and faster, we create a dummy agent called Robot B, which can be toggled in and out of the state "waiting for pass". This way, we do not have to wait for a physical Robot B to walk to position and signal, and the trainer can simply tell Robot A to *assume* Robot B is signaling. Once training is completed, we will transfer the learned behaviors to the robots as our final competitive behavior.

7 Conclusions

We have described the hardware, software, and development approach for the RoboPatriots, a team of four 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.

Acknowledgments

The work presented in this paper is supported by NSF NRI Grant 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 spatio-temporal observations of multi-agent systems. In: IEEE Internationa Conference on Data Mining (ICDM). pp. 580–587 (2008)