

A Vision-Based Approach to Imitation Using Heterogeneous Demonstrators

Jeff Allen and John Anderson

Autonomous Agents Laboratory
Department of Computer Science
University of Manitoba
Winnipeg, Canada
jallen,andersj@cs.umanitoba.ca

Abstract

Imitation learning is a powerful mechanism used by humans and other creatures. In imitation learning, the actions of others form the basis for desirable behaviour, and an imitation learner must be able to recognize the outcomes of the actions of others, understand how these relate to its own abilities, and ultimately duplicate the final outcome of a series of actions. We are interested in supporting this type of learning in general populations of robots, where a two important complications arise. First, physical variation between demonstrator and learner may require the learner to carry out different action(s) from the demonstrator to achieve the same results. Second, since demonstrators' skills may differ as much as their physiology, agents must be able to compare the demonstrations of a number of different individuals, in order to give greater weight to better demonstrators. Being able to integrate multiple demonstrations from different demonstrators allows a learner to deal with these problems as well as encouraging the creation of more general behaviours, rather than simply mimicking the actions of a single agent with no ability to generalize. In this paper we describe an approach to imitation learning based on global vision, which deals with these problems.

Introduction

Imitation learning is a powerful mechanism, and evidence of learning from the demonstrations of others can be seen in primates, birds, and humans (Demiris & Hayes 2002; Matarić 2002; Billard & Matarić 2000). From an AI perspective, this is attractive in part in dealing with the knowledge acquisition problem: instead of programming a robot for each individual task, robots should ultimately be able to gather information from human demonstrations (Matarić 2000; Nicolescu & Matarić 2003; Breazeal & Scassellati 2002). Similarly, in a multi-robot setting, robots should be able to learn from one another's behaviour (Anderson, Tanner, & Baltes 2004; Breazeal & Scassellati 2002; Riley & Veloso 2001).

Matarić describes imitation as "...a complex process capable of creating arbitrary new motor skills by composing complex hierarchical combinations of primitives and other already learned skills" (Matarić 2002). In this context, and

in our work, primitives are the low level actions available to the imitator, while behaviours are compositions of primitive actions, allowing abstractions. In order to be able to learn a new activity, a learning agent must be able to watch a demonstration, relate the actions perceived in that demonstration to its own primitives, and put these primitives together into a behaviour that accomplishes the same objectives as the original demonstrations.

Action recognition is an important part of this: in a realistic domain, an agent benefitting from a demonstration is shown another agent accomplishing the task involved, rather than being told a sequence of primitives to be employed. The agent must be able to associate the visible effects of an action with the fact an action was performed, by concentrating on the environmental changes caused by the demonstrator (Crabbe & Dyer 2000; Calderon & Hu 2003; Jenkins & Matarić 2003).

If a demonstrator and observer have the same set of physical primitives, and accomplish these in the same way, then the task from the observer's perspective is to match the visible outcomes with those that would be achieved by the primitives available, and deal with any errors in perception that arise. Restricting imitation learning in this way is very limiting to a robot expected to operate and interact with others, however: a humanoid robot (or indeed, a human) demonstrating a task may use the same basic movement primitives (such as a *forward* motion) as a wheeled robot, but accomplish those in very different ways, allowing a human to step over or onto small obstacles, for example. An imitation learner should be able to recognize the overall outcome and accomplish that using its own primitives, and learn from any physical type of demonstrator (Calderon & Hu 2003; Billard & Matarić 2000; Billard & Dautenhahn 1999).

The abilities necessary to avoid assuming the same physical primitives or physiology are those necessary to promote generality in learning. While it is possible to approximate a demonstrator's effects on the environment and accomplish the same task, the easiest way of doing this is to duplicate precisely the sequence of environmental changes the demonstrator makes, with no added understanding of the task involved. This is a form of overfitting, where no general ability or model is learned, but rather a set outputs that fit the precise inputs the learner was shown. While the agent may perform as well as the demonstrator, the lack of ability to

generalize will not allow it to transfer this learning to even a slightly different situation.

The lack of ability to generalize will also result in an agent that can never perform better than an imitator, since precise duplication involves following both positive and negative demonstrator actions. An extreme example would be a demonstrator driving toward a soccer goal, and performing pointless loops during the trip. An imitator seeing only this demonstration would learn to approach the goal in exactly the same way, wasting time with the same looping action by not being able to see its redundancy.

We are interested in promoting generality and learning better behaviours by learning from multiple demonstrators. If the imitator in the example above were to observe the looping demonstrator as well as other demonstrators that simply drove in a straight line, the imitator could compare all demonstrators' performances and recognize the redundancy. The imitator could simply ignore poor demonstrations, or even better, could filter the demonstrators' actions, learning the good and ignoring the bad (here, ignoring the loop and driving straight to the goal, while still acquiring the relevant portions of the behaviour). Learning from multiple demonstrators in the latter fashion should enable the imitator to learn the most efficient method for obtaining the same major environmental effects common to the demonstrators. Multiple demonstrators should also be able to assist in overcoming the issues involved in physiological variation: if a human going to a door by climbing stairs was the only demonstration received, a wheeled robot would be unlikely to successfully follow, whereas seeing a different demonstrator take a nearby ramp would allow it to overcome this difficulty.

Being able to learn from multiple demonstrators is particularly useful when a robot is a new addition to an existing team. A new robot should be able to learn more quickly how to optimally perform the tasks required by observing and imitating all its teammates at the same time. If the team of robots are all performing the same task, the imitator would learn very quickly. If instead the robots are performing differing tasks, the imitator would learn all the tasks simultaneously, albeit likely more slowly than a single task (i.e. learning how to play offense, defense and goal in soccer). If a replacement robot was needed, it should be able to learn from the robots that it was to replace, preserving the experience that the original robot(s) have acquired.

A robot intended to learn in such a manner requires several layers of functionality. First, it must be able to use vision to associate the execution of primitives with their effects on the environment. In any realistic environment, agents will not associate their primitives by name or even implement them in the same way, so an agent's sole basis for comparison is the set of its own primitives and their effects. Second, an imitator must be able to observe the effects of others' actions over time in the form of individual demonstrations, and relate them as closely as possible to its own primitive actions. These are unlikely to be completely explained, both because of differing physiology and because of errors in perception. Third, it must be able to build sequences of these primitives into higher-level behaviours,

both so that these can be re-used elsewhere, but also so that these can serve as a more general model of the activity itself. Finally, the agent must be able to integrate the demonstrations from multiple demonstrators, to allow gaps due to physiological differences to be filled, to allow the filtering or discounting of agents that perform poorly, and to better create a general model of the activity itself.

We have developed an approach to multi-agent imitation learning that incorporates these four layers into an imitation learner designed to learn from multiple demonstrators with varied physiologies and skill levels. The domain in which we are working is robotic soccer, and within this, our initial implementation involves getting an agent to learn to position itself and kick a ball into a soccer goal, after being given multiple demonstrations by agents that differ in skill and physiology. The demonstrations are provided via a camera positioned at an oblique angle to the field and connected to global vision software that supports tracking of both the robot and the ball over time. The imitator is a wheeled robot, while demonstrations are provided by three different robots: one physiologically identical to the imitator, a smaller tank-treaded robot, and a humanoid.

In our work, the imitator's primitives are provided, but their visual effects are initially unknown. Once the imitator learns the visual phenomena associated with its own actions, these serve as the means of relating a demonstrators actions and ultimately learning the task demonstrated. The demonstrators' primitives are similarly hand-constructed, but differ from the imitators in some respects due to physiology (e.g. humanoid vs. treaded vs. wheeled motion). The atomic motor commands use as primitives here are 'forward', 'backward', 'turn left', and 'turn right', which execute atomic motor commands for a fixed time (a fraction of a second) on the imitator, and perform analogous but physiologically different actions on the demonstrator (e.g. a step in a given direction for a humanoid).

This paper focuses on the approach we have developed in terms of recognizing the actions of others and developing higher-level behaviours from demonstrations. We begin by reviewing background work on imitation learning, and then present the layers of functionality described above in the model we employ. Following that, we describe the ongoing implementation of this work.

Background and Related Work

A number of important prior imitation learning projects have already been cited above. Few of these are designed to deal with multiple demonstrators, let alone demonstrators that are physiologically different and different in skill levels. The approach presented in this paper is designed from the bottom up to learn from multiple demonstrators that vary physically, in underlying control programs, and skill levels. Our approach also differs from most prior work in that rather than defining primitives abstractly (e.g. avoidance, following, homing, aggregation, dispersion, navigation, foraging, and flocking (Mataric 2000; 2002)), we operate with the lowest-level primitives possible given the design of the robot.

Some previous research has been performed involving imitating multiple demonstrators. The work of (Yamaguchi,

Tanaka, & Yachida 1997) allows agents on a team to imitate teammates that are considered to be more skilled, and in their absence, to learn from reinforcement learning. Our work differs from this in that while demonstrators are compared, the basis for this comparison is the actual behaviours learned from each demonstrator, and elements of good behaviour can still be learned even from teammates that are not well-skilled overall. (Price & Boutilier 2003) also used imitation learning with multiple demonstrators, coupled with reinforcement learning. In that work, however, the imitator extracts information about the demonstrators simultaneously and combines it without comparing their abilities. While this is a faster method of learning from multiple sources, it allows poor demonstrators to have too strong an influence.

In addition, the problems involved in imitation learning from multiple heterogeneous demonstrators are similar to those encountered when deciding who to interact with in a group of agents when a choice is available. This problem is explored in a soccer domain in (Anderson, Tanner, & Baltes 2004), where reinforcement learning is used to create a model of teammates, and is used to both select a passing target when options present themselves, and to allow an agent to cover for others on a team that are known to be weak. That work involved only keeping track of an agent's quality of performance, however, rather than attempting to actually learn domain behaviour.

Finally, an area of research closely related to imitation learning is traditional AI plan recognition. In classic plan recognition, a sequence of actions performed by an actor is used to infer the goals being pursued by that actor, and the action sequence is organized into a plan structure (Kautz 1991). In contrast, much imitation learning work, including this work, does not construct a formal plan structure, in the sense of a contiguous sequence of operators, but instead operates using a behaviour-based approach. Behaviours are abstracted hierarchically and consist of compositions of behaviours and primitive actions, but are not constructed using a traditional planning algorithm. The imitator can use its newly learned actions to achieve the same outcomes as the demonstrator; however, it can also improvise by working with abstracted behaviours instead of just following a detailed plan extracted from observing the demonstrator. The techniques of observation and abstraction of observed knowledge are analogous in both areas.

Methodology

An imitation learner dealing with multiple demonstrators differing in physiology and skill must be able to recognize the sequence of low-level actions of others through observation, abstract these to higher-level behaviours, reproduce them using its own action primitives, and be able to deal with differences in demonstrations from agents of different skill and physiology. An overview of our approach to accomplishing these tasks is shown in Fig. 1.

In order to understand the actions of others, the agent must have some basis, and in our approach, this is a model of effect constructed from the visual outcome of the agent's own low-level actions. This model is then used to convert the visual data stream into a sequence of the imitator's own prim-

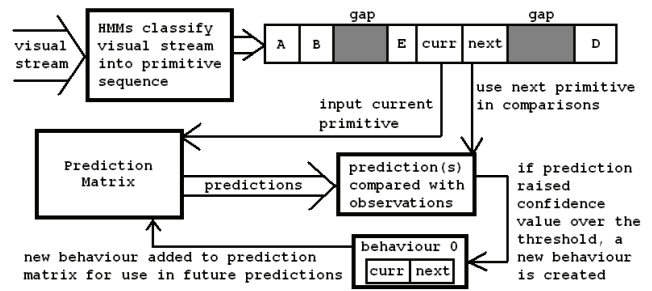


Figure 1: Imitation Learning Architecture

itive actions, to the degree this is possible given physiological differences between demonstrators. These sequences of imitator actions are then further abstracted and refined into more meaningful behaviours to help the imitator learn how its actions affect its environment. The subsections that follow deal with these aspects of our approach.

Understanding the Visual Effects of Primitives

In our approach, the actual recognition and classification of the actions of others in terms of the agent's own primitives is performed using Hidden Markov Models (HMMs), trained to recognize the visual changes in the environment manifested by the imitator's own primitives across a sequence of visual frames. HMMs model sequences where one aspect of the sequence is unobservable (hidden) and the other aspect is directly observable (Rabiner & Juang 1986; Yang, Xu, & Chen 1997). These have proven very useful in the past for recognizing sequences of events that have underlying stochastic elements (Rabiner & Juang 1986). In this work, the observable stochastic element of the sequence is the change in the imitator's orientation and position from state to state across visual frames, while the hidden stochastic element is the error in the accuracy of vision. HMMs are employed to compensate for this inaccuracy, resulting in more accurate classification of the primitives from the vision data than a simple comparison method. Recognizing primitives with Hidden Markov Models has been used in related research, such as assembly tasks learned from human demonstration (Hovland, Sikka, & McCarragher 1996), pointing to their potential utility in the area of imitation.

Vision in our implementation is provided by *Ergo* (Furgale, Anderson, & Baltes 2005), a system that allows a camera to be placed at any angle, and recognizes the motions of robots through patterns that require no predefined colors. Packets of visual data from *Ergo* are analyzed using a collection of Hidden Markov Models. Each HMM in the collection is trained to recognize a single primitive action performed by the imitator itself. Data for this training is gathered by placing the robot on a soccer field to record the vision data generated in response to executing individual primitive commands, and separating these by primitive and filtering out erroneous responses (e.g. if an infrared signal is lost and a robot does not move in response to a 'forward' command).

HMMs are then trained offline using the Baum-Welch

re-estimation method (Rabiner & Juang 1986; Yang & Xu 1994), which adjusts the HMM parameters until no more improvements in accuracy are made. This method is used to iteratively re-estimate the HMM parameters based on the probability that a partial observation sequence will end in a particular state given the HMM (Yang & Xu 1994). Each HMM will calculate the probability that the observation sequence matches the specific motor command it is trained to recognize, and the primitive symbol (i.e. “forward”, “backward”, etc) of the HMM with the highest probability will be selected. The imitator can then use these HMMs to convert the vision data into sequences of symbols, where each symbol represents one of the imitator’s primitive motor commands. Each primitive also records the average change in the imitating robot’s position when that primitive is executed by the imitator, which is also gathered during HMM training. This average change can then be used by the imitator to help it predict what will happen if it executes one of these primitives (the use of these predictions will be described shortly).

Recognition: Matching Primitives to the Visual Effects of Demonstrator Actions

After completion of the steps outlined above, the imitation learner has a repertoire of primitives with a single, unique HMM to provide a mapping between that primitive and its visual outcome in the world. A primitive represents a single motor command available to the imitating robot, and the initial set of these forms the boundary of the vocabulary for describing the actions of others. That is, agents in this approach never learn primitives they cannot execute themselves. In order for an imitator to understand a demonstrator in terms of its own capabilities, it must match its primitives to the visual effects of the demonstrator. To do this, the imitator analyzes the stream of vision data provided by Ergo during a demonstration, and converts this visual stream into a sequence of primitives that approximate the demonstration.

This sequence of primitives is initially empty, and grows as more of the visual sequence is processed by the HMMs associated with each primitive. Each HMM calculates the probability that its primitive will match with the start of the remaining visual stream (that portion yet to be processed by the HMMs). This match is based on the accuracy of approximating the state change. The primitive that has the highest probability of a match is appended to the end of the sequence of imitator primitives being built to approximate the demonstration. As the primitive sequence expands, the amount of the visual stream approximated by it increases, and thus the start of the unprocessed visual stream is moved incrementally down the stream (as shown in Fig. 2). The distance moved is dependent on the size of the selected HMM, since the start of the unprocessed visual stream is moved up to the point in the stream where the HMM approximated the state change.

It is possible that a visual sequence converted into primitive symbols by the HMMs will contain gaps, indicating places where no primitive actions could sufficiently bridge the state changes observed from the visual stream. This

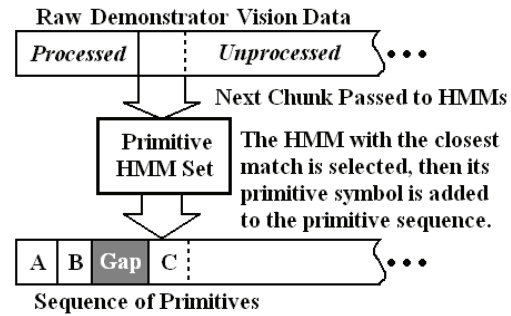


Figure 2: Hidden Markov Models Classify Raw Demonstrator Vision Data into Sequence of Primitives

would occur, for example, when the demonstrator performs actions unavailable to the imitator as primitives, such as a humanoid demonstrator stepping over the ball, when the imitator is a wheeled robot. Even if such gaps did not exist in a demonstration, using the full sequence on its own would only allow the imitator to approximate the demonstration precisely (i.e. mimic the demonstrator). To be able to generalize, fill in gaps that are left by physiological differences, and be able to learn from multiple demonstrator, these primitive sequences are used to construct a more meaningful abstraction of the demonstration using behaviours. This process is detailed in the next section.

Predicting and Constructing Behaviours

Behaviours are defined in this work as sequences or abstractions of primitive sequences that also take environmental changes into account, such as the position of the ball. Behaviours will have greater overall state changes than individual primitives, and allow the agent to reason about and recognize complex sequences of actions as a unit. At its most basic, the sequence of actions recognized from a given demonstration is a single behaviour, albeit a low level one that precisely duplicates a single demonstration.

Abstractions supported by behaviours allow gaps in the explainable visual sequence of a demonstration due to physiological differences between the imitator and demonstrator to be bridged. For example, consider the case where the imitator is a wheeled robot, and the demonstrator is a humanoid. If the demonstrator stepped over the ball, the resulting gap could be approximated by the wheeled imitator through a behaviour that drives around the ball. Our work only compares the state change of behaviours to the gaps, using the duration of behaviours to break ties when two or more behaviours have similar state changes. For example, one behaviour’s state change may cause the imitator to drive in a straight line to a destination. A separate behaviour might result in the same state change, although in a different manner (such as driving in a pointless loop before proceeding to the destination). Since these two behaviours produce the same final state changes, they can be compared, and the faster behaviour (the straight line behaviour) would be selected over the slower one.

Some behaviours, rather than encompassing a range of ac-

tions, merely consist of repeated primitive actions: moving forward a longer distance, or turning in a circle for example. The difficulty with these behaviours is that there can be any number of them based on the length of the repetitive sequence. To limit this, we also define *parameterized* behaviours, similar to those of (Jenkins & Matarić 2003). These accept a parameter to indicate the number of repetitions of the repeated primitive action they abstract, and thus allow a single behaviour to encompass any number of repetitions. Repetitive behaviours can also represent the repetition of behaviours as well as primitives.

At any point in time, an imitator will have abstracted a given set of behaviours, each of which represent a combination of primitive actions and/or other behaviours. Since the number of these combinations is essentially infinite, it is necessary to keep the number of existing behaviours reasonable by choosing to maintain only the most useful. To allow this, we define a *permanency* attribute for all behaviours. A behaviour with a permanency attribute of zero is removed, while exceeding an upper threshold allows the behaviour to be considered permanent (i.e. it is no longer evaluated for permanency).

The basic means by which a sequence of actions can be recognized and stored as a behaviour has been detailed above. Using this mechanism alone will only allow behaviours that exactly match the visual phenomena being observed (i.e. exactly as the demonstrator did it), and may leave gaps where matching is unsuccessful. To build truly useful behaviours, an imitator must be able to explore its options and also be able to consider how abstract behaviours it has already learned could be used to deal with a new demonstration. It must also be able to abstract behaviours from lower level behaviours, not just primitive actions. To do this, we use existing knowledge (primitive actions and behaviours) to attempt to *predict* what should happen next (i.e. what the imitator would do) given the observation of a demonstration thus far.

Previously, while studying infants mimicking facial expressions, (Meltzoff & Moore 1997) proposed an underlying system of body babbling, where babies practice movement through self-generated activity, and (Demiris & Hayes 2002) developed a computational model based on this phenomenon. They devised a system using *forward models* to predict the demonstrators behaviour. A forward model takes as input the state of an object and a control command that is to be applied to it, and predicts the next state. (Dearden & Demiris 2005) also implement analogous forward models, though they use Bayesian networks as the basis for them. In this work, we adapt the concept of a forward model for prediction to deal with integrating demonstrations from multiple demonstrators.

In our approach, we employ the concept of a forward model both as a predictive model, as well as the mechanism to trigger the creation of new behaviours. After a prediction is made, it is compared to the observed demonstrator action at the next time step. When the model makes frequent, accurate predictions about two primitives occurring in sequence (or a primitive and a behaviour, or two behaviours), it will trigger the creation of a new behaviour to represent this se-

quence.

The imitator maintains one main forward model for the activity being learned, and this forward model contains behaviours it has learned from the demonstrations provided, along with the primitives it started with. The imitator also employs an additional separate forward model for each demonstrator it observes. While imitating a demonstrator, the imitator will update the forward model specific to that particular demonstrator, and keeping these distinct allows them to be used to compare the quality of demonstrators when deciding which demonstrator skills to incorporate into the main imitator forward model. Further details on the process of a behaviour moving from the forward model for an individual demonstrator to the imitator's main forward model are detailed in the next section.

The imitator's forward model uses an n by n matrix to keep track of which behaviours (or primitives) follow each other in sequence, where n is the total number of behaviours and primitives (so a link from each behaviour/primitive to each other one). Each element in the matrix corresponds to the *confidence* that the forward model has that a behaviour (or primitive) will follow another (default confidence value will be $1/(\# \text{ of behaviours and primitives})$). To make a prediction of the demonstrator's next action, the forward model uses the last recognized behaviour/primitive, and looks up that row in the matrix. The behaviour/primitive with the highest confidence in that row is then predicted to occur next (for this reason we call the matrix the *prediction matrix*). The effects of the predicted behaviour/primitive are applied to the imitator's internal model of the current environmental state, then compared at the next time step to the state change (similar to the forward models used in (Demiris & Hayes 2002)). If the prediction is correct, the confidence value linking the two behaviours/primitives is increased, and its permanency increased by a small factor, while if the prediction is incorrect, it is decreased.

After a confidence value is modified, it is checked to see if it has surpassed a threshold indicating that the frequency of association merits creating a new behaviour. If the confidence value is above the threshold, a new behaviour is created that represents the sequence of those two behaviours. This leads to new behaviours containing sequences of smaller behaviours that are useful in sequence. When a new behaviour is created, the confidence value that triggered the creation is reset to the default value in the prediction matrix. If the new behaviour contains sub-behaviours, the new behaviour incorporates the sequence of primitives from the sub-behaviours, but not the actual sub-behaviour itself. The old sub-behaviour can then be removed from the forward model (if it is not used on its own anymore), which assists in keeping the overall number of behaviours manageable.

Dealing with Multiple Demonstrators

In order to deal with multiple demonstrators, we need to be able to evaluate demonstrators' performances relative to each other. An imitator can then learn the most efficient behaviours used by the demonstrators. This results in the imitator building a unique repertoire of behaviours that should

be able to perform as well as the best demonstrators, with the potential to outperform them. In addition to a unique forward model for each demonstrator, as described above, the imitator’s internal representation also maintains a rank of the overall quality of each demonstrator.

The quality of a demonstrator is represented by its *learning preference*, which is analogous to a learning rate. The learning preference (or LP) is between 1 and 0, and used when updating prediction matrix confidence values (if the confidence value is to be increased, the amount to increase it is multiplied by the LP before being applied). This makes a good demonstrator (higher LP) increase the confidence values in its matrix faster, thus allowing the agent to learn new behaviours more quickly from that demonstrator (and vice versa for a poor demonstrator). When observing a demonstrator or when the imitator is just choosing what it should do next (if all the demonstrations are already over), the imitator makes predictions with *each of its forward models* simultaneously (all demonstrator forward models and the imitator’s own main forward model). The imitator then updates the LP of each demonstrator based on the accuracy of their forward model predictions (increase for correct, decrease otherwise).

The imitator also updates the LP based on the outcome of a demonstrator’s predicted behaviours. To increase the LP, a behaviour must result in the imitator (ordered from highest LP increase to lowest): scoring a goal, moving the ball closer to the goal, or moving closer to the ball. The opposite of these usefulness criteria decrease the LP (opposite of scoring a goal would be scoring in the wrong goal). We are using these criteria to shape and speed up the imitator’s learning, a technique that has been shown to be effective in other domains (Matarić 1997). Future work could shape the learning less and examine how much could be learned under those conditions. To shape the learning of the imitator towards learning more efficient behaviours, we compare individual behaviours in certain circumstances. Behaviours are compared if they are predicted at the same time, and they produce the same outcome. The behaviour that takes less time has its permanency increased, and the other behaviour’s permanency is decreased. This is only done for behaviours considered useful by the above criteria (no sense learning how to perform useless behaviours quickly).

Each demonstrator also has a *decay rate* associated with it. The decay rate for a demonstrator is simply $1 - LP$, so a good demonstrator has a low decay rate, and a bad demonstrator has a high decay rate. The decay rate is used to ensure that the total collection of behaviours doesn’t grow to an unreasonable size. The decay rate is applied to the permanency attribute of all behaviours in the demonstrator’s forward model at every prediction step. A good demonstrator’s behaviours should be correctly predicted (and their permanency increased) frequently enough that their increases in permanency will overcome their decreases from decay rate.

When a demonstrator’s behaviour becomes permanent (permanency past an upper threshold), it is permanent *to the forward model for that demonstrator*, not permanent to the imitator. Behaviours that achieve permanency within their demonstrator specific forward models are copied into a *candidate behaviour buffer*. This candidate buffer is the final

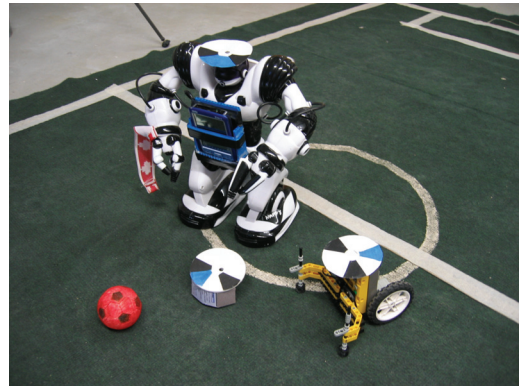


Figure 3: The robots used in our implementation, imitator on the right.

step for a behaviour to become permanent to the imitator’s main forward model. Once in the buffer, a behaviour’s decay rate becomes that of the imitator’s (which never changes), and its permanency attribute is reset to the default value (the midpoint between min and max values for permanency). This equalizes the chance behaviours have of becoming permanent once they reach the buffer. The buffer acts as an extension to the imitator’s main forward model, so the behaviours in it are added temporarily to the imitator’s prediction matrix. If a behaviour becomes permanent in the candidate behaviour buffer, it is permanently added to the imitator’s main forward model, and the demonstrator has its LP increased. If a behaviour’s permanency drops below zero, it is removed completely, and its demonstrator has its LP decreased.

Implementation

The implementation of the work described here is ongoing, and is currently focussed on building the behaviour-development routines described above. The robots employed in our implementation are shown in Fig. 3. The imitator employs a body constructed from Lego MindStorms, while the three demonstrator types are a) identical to the imitator; b) a treaded 2” tank, differing greatly in both movement and size, and c) a humanoid (Robosapien). The field is a standard RoboCup E-League field (2.7 x 1.5 m), and the hat patterns on the robots allow them to be tracked and identified individually using vision by Ergo.

As described above, the first step in our approach is gathering proper training data for each primitive to be used for training the HMMs. The imitator is placed on the soccer field and its movements are analyzed by Ergo and recorded as a series of observation sequences. Each primitive is recorded in a separate block of observations, and each time a primitive action is carried out, the starting location and orientation of the robot is randomly selected. Each primitive is recognized by the changes it causes over time. The changes that are important in this work are the changes in x and y coordinates, and the orientation of the robot. Trying to recognize primitives by focusing on a single variable

