Teaching Robots by Moulding Behavior and Scaffolding the Environment

Joe Saunders*
J.2.Saunders@herts.ac.uk

Chrystopher L. Nehaniv
C.L.Nehaniv@herts.ac.uk

Kerstin Dautenhahn
K.Dautenhahn@herts.ac.uk

Adaptive Systems Research Group
University of Hertfordshire
Hatfield, AL10 9AB, United Kingdom

ABSTRACT

Programming robots to carry out useful tasks is both a complex and non-trivial exercise. A simple and intuitive method to allow humans to train and shape robot behaviour is clearly a key goal in making this task easier. This paper describes an approach to this problem based on studies of social animals where two teaching strategies are applied to allow a human teacher to train a robot by *moulding* its actions within a carefully *scaffolded* environment. Within these enviroments sets of competences can be built by building state/action memory maps of the robot's interaction within that environment. These memory maps are then polled using a k-nearest neighbour based algorithm to provide a generalised competence. We take a novel approach in building the memory models by allowing the human teacher to construct them in a hierarchical manner. This mechanism allows a human trainer to build and extend an action-selection mechanism into which new skills can be added to the robot's repertoire of existing competencies. These techniques are implemented on physical Khepera miniature robots and validated on a variety of tasks.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: [robotics]; I.2.6 [**Artificial Intelligence**]: [learning]; I.2.m [**Artificial Intelligence**]: [miscellaneous - imitation, programming by demonstration]

General Terms

Performance

Keywords

Social Robotics, Imitation, Teaching, Memory-based learning, Scaffolding, Zone of Proximal Development

1. INTRODUCTION

Imagine a scenario where your brand new domestic robot has just been delivered. The factory have pre-programmed it to carry out some useful tasks around the home e.g. collecting cutlery, cups and plates and placing them in a dishwasher or tidying up by picking up clothes left on the floor and placing them in a washing basket. You unpack the robot, press the "on" button, and the robot efficiently carries out these tasks whilst being safe to both you and itself. Later however you find that although the robot performs to the manufacturer's specifications there are some tasks which it does not carry out. It fails to tidy up the children's toys into the toy cupboard or it fails to recognise that a particular and expensive glass should not be placed in the dishwasher. After a call to the manufacturer you discover that there is another button on the robot marked "learn". When this button is pressed the robot can be taught additional skills. This paper presents research on how such a teaching mechanism might be implemented.

In section 2 we suggest that it is the social dimension of behaviour that holds the key to making robots behave more intelligently [6], an approach inspired from studies of social animals (e.g. apes) and the 'social intelligence hypothesis' [4], which proposes that intelligent behaviour in primates has its origins in dealing with complex social dynamics. We discuss how the social aspects of teaching, learning and imitation are used by some social animals to expand their repertoire of skills. From this work we extract the developmental concepts of "scaffolding" and "putting-through"/"moulding" as mechanisms which may prove useful for robot teaching. Section 3 discusses related work where observation, imitation and direct teaching are used. We conclude this review by outlining the computational techniques that we will use in creating a novel learning architecture which will allow new robot skills to be taught whilst retaining or improving existing skills. Section 4 details the realisation of this architecture on physical Khepera miniature robots. Section 5 gives the experimental validation of the work by showing examples of how behaviours can be created in a robot and how additional skills can then be added to an existing robot skill repertoire. Finally we discuss some of our findings and the possible directions for further research in this area.

2. TEACHING AND IMITATION IN ANIMALS

Moore [13] proposes a six-step hypotheses for the evolu-

*corresponding author

HRI'06, March 2–4, 2006, Salt Lake City, Utah, USA.

tion of imitation in nature The process starts with Thorndikian conditioning where existing motor actions are associated and reinforced based on particular environmental conditions. This step is later enhanced by operant (or Skinner) conditioning where novel motor responses are formed based on combinations of existing actions. The next evolutionary step is an implicit reinforcement cycle leading to "skills" where the animal is able to perfect the novel act. The fourth stage introduces the teacher. The teacher essentially guides the pupil by physically "moulding" or "putting-through" the actions of the pupil given particular environmental stimuli. This can be considered as self-imitation by the animal as it repeats the actions that it has experienced. Visual imitation of others is the next evolutionary stage. In this case the animal now only has to see an act to be able to repeat it. The final process is called cross-modal imitation where an animal is able to match features of its body with corresponding features of another animal. For example, human babies touch parts of the faces of their parents and can then locate the same features on their own face. In figure 1 we summarise and segment these stages into prime, taught and imitative sections.

The study presented in this paper bases its mechanisms for robot teaching on the *self-imitation* stage. However each of the earlier stages are also used. For example, we simplify the actions available from the *Thorndikian* stage by considering them to be part of the robot's existing repertoire of motor skills. This existing set of skills over and above basic motor actions are called "primitives". Explicit combinations of primitives can be specified by the teacher. We call these "sequences" but they are equivalent to novel sets of responses available at the *operant conditioning* stage. Skill learning is the essential building block upon which the teacher's directions are built. The *skill reinforcement* stage will therefore form the association between sensed stimuli and action. The fourth *self-imitation* stage is based on moulding or putting-through. This is where the training example is provided by the teacher by *putting* the robot *through* the range of actions required. Our previous work [21] considered aspects of observational/imitative learning at the *imitation* stage. Our current work develops these ideas in a further investigation of the spectrum of learner/pupil relationships.

Evidence for teaching in the animal kingdom comes mainly from studies of primates [4]. However there is also evidence from carnivores including domestic cats, tigers, cheetahs, otters, dolphins, orca whales and some bird species [24]. An example from cheetahs is where the mother rather than killing prey will capture and release the live prey to the cheetah cubs when they are about 3 months old. The behaviour is also selective, only prey species which the cubs are likely to catch are released. It appears that the cubs' experience results in faster learning and a more skilled performance. This was tested with domestic cats whose kittens were brought live mice by their mothers at an early age. By 6 months old the kittens showed superior skills to a test group who had not been exposed to the mice [23].

Compelling evidence of intentional teaching comes from studies of primate behaviour. Fouts *et al.* report on the chimpanzees Washoe and Loulis, Loulis being the adopted infant chimp of the mother Washoe. Washoe had previously been taught American Sign Language (ASL) however the human carers made no attempt to teach Loulis ASL and did not use ASL in Loulis' presence. However Washoe succeeded in teaching Loulis ASL both by demonstration and by *moulding* of Loulis' hands [18]. Moulding had also been used by the human carers to originally teach Washoe.

LEVEL	EVOLVED STAGE	TECHNIQUE
PRIME	Thorndikian Conditioning	*Primitives*
TAUGHT	Operant Conditioning	*Sequences*
IMITATIVE	Skill Reinforcement	*Skill Learning*
	Self-Imitation	*Moulding/Scaffolding*
	Imitation	*Observation/Imitation*
	Cross-Modal Imitation	*Correspondence*

Figure 1: Proposed evolutionary stages with techniques required to implement them. This paper deals with the *taught* skill set.

Scaffolding is where a physical situation is artificially modified, typically by the mother, to make it much easier for her child to complete the task when the child is at a developmental stage where it could not perform the appropriate acts or sequence its actions correctly. Scaffolding of tasks together with observational learning and moulding have been observed in wild chimpanzees [4]. Cracking nuts with a hammerstone is an especially difficult task for a chimpanzee to learn, taking up to 14 years to perfect in some cases. A number of observations have been recorded where the mother will clean the anvil, reposition the the nut or re-orient the hammerstone to favourable orientations for the infant. Scaffolding is also a familiar concept in human development and is emphasised in Vygotysky's idea of the "zone of proximal development" in his theory of the child in society [27]. Teaching and social interaction allow higher competence levels to be achieved through staged learning and building upon existing skills.

We take inspiration from these examples in social animals to study how *moulding/putting though* and *scaffolding* can be used to good effect in teaching robots new skills and allow existing skills to be modified.

3. RELATED WORK

Even with explicit programming robot control is hard due sensor noise, the non-deterministic state of the environment, the inability to ensure that the robots actions are deterministic and the need for real-time responses. There are generally a number of problems which need to be solved:

i) *how can the human teach the robot?* - what mechanisms can be used to make the robot match the intentions of the teacher? How can the robot learn when the task is complete?

ii) *what techniques can the robot use to learn?* - how can the machine generalise and execute the new task?

iii) *how can the robot incorporate the new experiences into its existing competencies?* - what sort of structure is necessary to ensure that new tasks can co-exist with existing tasks?

iv) *how can it select the right action at the right time?* - given a learned set of competencies which one is appropriate?

Approaches include topics such as programming by demonstration, imitation learning, learning from observation and robot shaping. Typically the observational and imitative approaches attempt to match the behaviour of the demonstrator and so construct an appropriate control policy. Schaal et al. [22] provide an overview where approaches to the problem are classified as follows:

i) *direct policy learning* - where supervised learning is used to learn a control policy directly.

ii) *learning policies from demonstrated trajectories* - this assumes that the task goal is known and uses sample trajectories to learn a control policy

iii) *model based policy learning* - where a predictive model of the control problem is constructed.

All of these approaches face two difficult problems. Firstly, that by observation alone the internal proprioceptive feedback that the teacher experiences cannot be directly experienced by the pupil [20] and secondly, there may be a mismatch between the external and internal sensorimotor spaces of the teacher and pupil - the correspondence problem [14].

In the *direct policy approach* these issues can be avoided by having the pupil experience the same set of actions and sensory states as the teacher with the correspondence problem solved by ensuring that both teacher and pupil have a similar embodiment. This approach has been used by a number of groups including Billard & Dautenhahn [3] and Hayes & Demiris [9]. In both cases a student robot followed a teacher robot and learned to associate imitated actions against perceived environmental state. Saunders *et al.* [20] however have demonstrated that there can be limitations in this approach due to reactive impersistence and teacher interference when using a pure following approach.

In recent work by Nicolescu *et al.* [15] a mobile robot tracks a teacher's movements matching predicted postconditions against the robot's current proprioceptive state. It then builds a hierarchical behaviour-based network based on "Strips" [17] style production rules. This work attempts to provide a natural interface between robot and teacher whilst automatically constructing an appropriate action-selection framework for the robot.

Another way to allow a robot to experience the appropriate sensory state is by allowing the teacher to manipulate the robot directly via a form of tele-operation and record the sensory state of the robot. Although not using a robot this method is closely related to Sammut's [19] "learning-to-fly" application where recordings of control parameters in a flight simulator flown by a number of human subjects were analysed using Quinlan's C4.5 induction algorithm [16]. The algorithm extracted a set of "if-then" control rules. Van Lent [26] also used this approach but provided a user interface which could be marked with goal transition information. This allowed an action-selection architecture to be constructed using "Strips" [17] style production rules. However, in both of these research areas the full "state" of the system (both internal and external) is available to the trainer. This may not be the case when teaching robots.

A long line of research into teaching service robots by observing humans has also been carried out by Dillman [7] where after observation production rules are generated to produce grammatical formalisms held in a knowledge database of actions.

Dorigo and Colombetti [8] use decomposition of tasks by a trainer to "shape" robot behaviour. We take a similar approach however we do not use either evolutionary or reinforcement learning techniques to create or modify robot behaviour.

Learning policies from demonstrated trajectories seems to be appropriate when the goals of the task are known and the task itself is self contained. For example when learning to duplicate human movements [11] or play tennis strokes [10] the goal of the task is already known or programmed into the learning mechanism. It is made explicit by the programmer for the specific (although mechanically complex) task to be solved. It is difficult to see how a new task could be incorporated into the existing learned policy without further explicit programming.

Bentivenga *et al.* [2] use *model based policy learning* to construct a learning framework using a memory-based approach. A humanoid robot learns to play games of "marble maze" and "air hockey" by recording exteroceptive data (ball angle/velocity, board tilt angles) and primitive type (roll ball away from corner, roll ball off wall) from a human demonstrator. The robot is able to select the appropriate primitive by analysing a memory model to find the nearest example to the current state. Parameters for the primitive are constructed using locally weighted regression on points nearest the selected primitive. This technique is also related to loose-perceptual matching methods described in [1].

Memory based learning approaches have a number of technical advantages. Firstly, complex functions can be learned by focusing on sets of less complex local approximations. Secondly, the local approximation for the target query (based on the current sensory state) is based on the training data at the time of the query and not on a pre-built function approximation. This means that additional training instances can be added immediately without the need to rebuild a target function (which would be the case for an inductive or neural network approach).

It is noticeable that many of the example applications described above have the ability to learn complex tasks based on some form of observation (where observation can be both direct and from post-processing of sensory data). However with the exception of [26, 7, 15] there are few mechanisms which allow another task to be both learned and included into the repertoire of previously learned functions. Our approach is to provide an interface which will both *learn a particular task* and have the ability to *add this task to an existing control mechanism.* This requires a number of steps.

i) a policy needs to be learned based on the sensory state of robot itself. The correspondence between the human teacher and robot also needs to be solved - both of these points we address by the simple process of *moulding* the robot by teleoperation.

ii) the robot needs to be aware of when tasks have a specific goal - we make this an explicit part of the training sequence.

iii) learning must be carried out in real-time and be subsequently modified or enhanced with additional learning

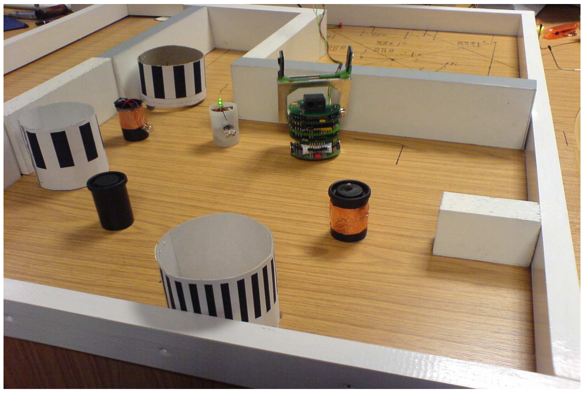


Figure 2: A typical environment showing a Khepera with vision sensor and gripper, objects with different electrical resistance and bar-coded containers.

experiences - this is made possible by using memory based learning methods.

iv) the new learning experience should not corrupt other previously learned experiences - we allow the construction of a hierarchy of memory models to provide this.

One of the key points in addressing many of the issues described is that a teacher constructs an appropriate learning environment for the robot. We do this while exploiting and extending some of the techniques already used by the practitioners above in a new framework.

4. FRAMEWORK

For this study we have used physical Khepera miniature robots (see figure 2) on a desk in a typical busy academic environment. Khepera's are 5cm diameter non-holonomic robots equipped with eight IR sensors placed at intervals around the base, an arm/gripper and a K213 linear vision system. The IR sensors are capable of detecting both ambient light and short range (10cm) obstacles. The arm/gripper arrangement can detect when an obstacle is within the gripper and also the electrical resistivity of the object grasped. The K213 vision system provides a one dimensional line of 64 grey scale values subtending an angle of 36° from the front of the robot. Commands to control the robot can be sent from a remote PC either via a radio signal or from a directly connected serial cable.

The learning environment we choose is based around the capabilities of the Khepera. To provide a reasonably complex learning environment the Khepera is placed in a walled "room" with various objects of different conductivity and some bar-coded containers.

4.1 Learning Mechanism

We use a memory based "lazy" learning method [12] to allow the robot to learn tasks. This is a simple k-nearest neighbour (kNN) approach where the value of each feature in the robot's state vector (see *Scaffolding* below) is regarded as a point in n-dimensional space, where n is the number of features in the state vector (see table 1). For each chosen task we collect a set of training examples (as described in *Moulding* below) together with their target primitives, each primitive being chosen by the human trainer when moulding the robot's actions. When the task is executed the robot continually computes its current state vector. It then computes the distance from the current state to each of the training examples. The distance between the state vector and the training example being the sum of the distances between the features in each, as follows:

Table 1: State Vector Used in experiments

State	Description
Repulsive Force	Vector of IR sensors
Repulsive Angle	Angle of IR Vector
Light Distance	Distance to light
Light Angle	Angle to light
Bars Seen	Number of bars seen by K213
Bar Size	Average bar size seen by K213
Bar.Std.Dev.	Std. Deviation of bar size
Arm Up/Down	Whether the arm is up or down
Gripper Open/Closed	If gripper is open or closed
Object in Gripper	If object is in the gripper
Resistivity	Resistivity of object

$$distance(X,S) = \sum_{i=1}^{n} W_i \, | \frac{x_i - s_i}{max_i - min_i} |$$

Where X is an instance of the training examples and S an instance of the robot's current sensory state. W is a non-negative vector of real numbers used to weight each of the dimensions. This weighting is discussed in the *scaffolding* section below. Setting k to 1 will result in the nearest point in the training examples being used and yield a single primitive as its target function. Where k is greater than 1 the algorithm will yield a set of primitives. We choose the most common primitive from the set as the target function. Note that this method will always result in a primitive being chosen. In environmental situations not previously experienced by the robot, generalisation occurs as the primitive nearest to the current state is chosen. Thus performance is based on the similarity of new situations to those already experienced.

In work to date the k value has been set experimentally to approximately correlate to the number of state vector entries in each memory table. For a small number of entries k is set to 1. For larger tables k has been set to higher values but not exceeding 5. We make use of the Tilburg University Memory Based Learner [5] to provide the kNN functionality. This system has the advantage of providing a very efficient tree-based coding structure for the training examples so as to speed up performance.

4.2 Moulding

The concepts of scaffolding and moulding can play an important part in animal learning. They support a form of self-imitation that may be the natural precursor to more complex forms of imitative learning. In our framework we use the idea of moulding or putting-through directly. The human has the ability to control the robot by remotely moving it through a set of pre-defined basic primitives. This set of primitives are basic actions available to the robot (see table 2). The human teacher has no access to the internal state of the robot. By manipulating the robot in this manner we also avoid both the problem of observation by the robot of the human actions and of the correspondence problem between the robot and human. During the robot moulding process a snapshot of the robots proprioceptive

Table 2: Pre-defined Primitives.

Primitive	Description
Move Forwards	Move Forward 1cm or continuously
Move Backwards	Move Backwards 1cm or continuously
Turn Right	Turn Right by 5° or continuously
Turn Left	Left Left by 5° or continuously
Raise Arm	Raise Arm, if not already raised
Lower Arm	Lower Arm, if not already lowered
Open Gripper	Open gripper if not already open
Close Gripper	Close gripper if not already closed

and exterioceptive state (see table 1) is recorded together with the directed primitive on each human command to the robot. For each human defined task we can therefore build a memory model of state/primitive combinations.

4.3 Scaffolding

All of the states perceived by the robot are recorded in the state vector however different attributes of this vector are relevant to different tasks. For example, to avoid obstacles the attributes of the IR sensors are of more importance than the position of the gripper, whereas to track an object the perceived orientation of the object is more relevant than the values of the IR sensors. Here we capture a pre-defined set of states some of which are numeric summaries pertinent to the expected applications and realisable by the sensor arrangement of the robot (see table 1). For example, rather than storing 64 grey-scale values for the K213 linear vision sensor we pre-process the K213 data to specifically detect bar-coded items. In this case when no such items can be detected the K213 values are set to zero. Repulsive and ambient light sources from the IR sensors are formed into vectors. Apart from the process of vector creation no further pre-processing is carried out. Thus these sensors are always "on" and not specifically programmed to detect particular objects or environmental items.

We use two mechanisms to ensure that the appropriate attributes are chosen. The first is a technical solution originally used in Quinlan's C4.5 Induction algorithm [16]. This is based on computing *information gain* to measure how well a given attribute separates the set of recorded state vectors according to the target primitive. This is defined as follows:

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{\mid S_v \mid}{\mid S \mid} Entropy(S_v)$$

where S is the collection of training examples, *Entropy(x)* is a function returning the entropy of x in bits, *Values(A)* is the set of all possible values for a particular state attribute A and S_v is the subset of S for which attribute A has value v. Further explanations of this metric can be found in [16, 12]. The information gain measurement allows particular attributes in the state vector to have greater relevance by using it to weight the appropriate dimensional axes in the kNN algorithm (by setting W_i above). This has the effect of either lengthening or shortening the axes in Euclidean space thus reducing the impact of irrelevant state attributes.

The second mechanism for attribute selection is the human trainer. It is assumed that the trainer already understands the task (from an external viewpoint) that the robot must carry out and therefore is able to construct the training environment appropriately so as to ensure that irrelevant features are removed. This idea allows the technical selection of relevant state features to be enhanced as the other features will now tend to have constant values and therefore a low information gain.

As an example consider training the robot to perform a "wall following" behaviour. The teacher might remove extraneous objects from the training area such as the bar-coded containers. By moving the robot through a number of wall following experiences the set of sensory states recorded will then be primarily based on the IR sensors (which resolve to the repulsive vector/angle attributes). These attributes will then be automatically selected by the extended kNN algorithm based on their higher information gain. As discussed in section 2 above this process of scaffolding or creating favourable conditions for learning would seem a quite natural phenomenon in social animals and is of course fundamental to all forms of human teaching.

4.4 Learning New Tasks

We are now in a position to define the mechanisms available to the human trainer. The trainer directs the robot using a screen based interface which provides a number of buttons used to set operation modes such as "execute" and "start/stop learning" plus an edit field to label actions and a list from which to choose existing labelled actions and primitive operations.

The robot can be in one of three modes. The first is *execution mode*, which is its normal mode of operation where its current behaviour is executed. Alternatively the robot can be in *training mode* where the human trainer can mould, scaffold and create new activities for the robot to eventually use in execution mode. An *intermediate mode* is where the trainer can execute one of the set of available competencies. For example by selecting the primitive "Move Forward" in this mode the robot will execute the move forward primitive. This is useful for placing the robot in an appropriate state prior to training.

In "learning" mode the robot can learn new competences at one of three training levels determined by the trainer: *sequence*, *task* and *behaviour*. All three training levels are started by pressing a "start learning" button and terminated by pressing a "stop learning" button. For each new competence (either a behaviour, task or sequence) the trainer explicitly provides an appropriate label, for example "PickUpCup". When training is complete the label is added to the set of actions available to the trainer and thus can be used immediately for further training sessions. Existing labelled actions can also be modified with additional training episodes as required. In training mode the trainer has the option to execute the selected labelled competence so that the results of the robot's actions can be assessed immediately.

The first competence level is the *sequence*. This is where the robot can be directed through a given sequence of primitives which it records without reference to its state. An example of a sequence might be to lower the arm and close the gripper. This could, for example, be labelled as the 'grab' sequence. The grab sequence would then become part of the available set of competences available for the trainer to use. These new sequences could also then be used in combination with other primitives and other sequences to create further sequences. Note that sequences are entirely deterministic. When requested to perform a sequence the robot will sim-

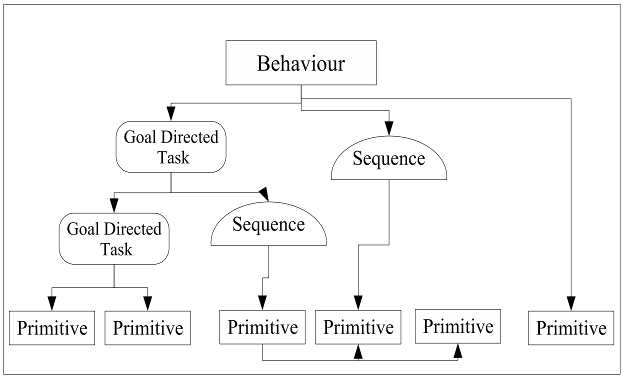


Figure 3: An example of a trained hierarchy of primitives, primitive sequences, learned goal-directed tasks and the final behaviour.

ply execute the recorded list of competences taught by the trainer sequentially. It will make no reference to the environmental state. Each primitive when executed by the robot can be run in two further modes - discrete or continuous. In discrete mode the primitive will execute followed immediately by a "stop" instruction. The continuous mode does not issue the "stop" instruction. Discrete mode allows the trainer put the robot though its range of actions step by step. Continuous mode is typically used after training is complete and enables the robot to execute the primitives without the jerkiness caused by the "stop" instructions above.

The second level for learning is called a *goal-directed task* or simply a *task*. This differs from a sequence in that during training the actions taken by the robot will depend on the environmental state at that time. The trainer now has the opportunity to select not only basic primitives, but sequences and other goal-directed tasks. The tasks are goal directed because the trainer also has the opportunity to inform the robot when the task has completed. This goal state is paired with the robot state and becomes a further training record in the memory model for that particular task. In execution mode the task is iterated until the environmental state is close to a goal state and the task will then terminate.As an example consider an obstacle avoidance behaviour. The trainer would place the robot in an obstacle facing situation, choose the "task" level, label it "Obstacle Avoidance" and press the "start learning" button. The robot can then be moulded into a non-obstacle avoidance situation. The trainer would then signal that the goal state was reached. This training regime would be repeated for many obstacle avoidance situations and thus many obstacle recognition states with appropriate avoidance actions and goal states being recorded into the Obstacle Avoidance memory model.

The final mechanism for learning is a *behaviour*. This allows the trainer to construct the complete behaviour for the robot from the component set of tasks, sequences and primitives. The construction of a behaviour is the same as for a task except that no goal state is required. The behaviour will run continually in execute mode and base its decision of what task, sub-task, sequence or primitive to use based on the current environmental state. With careful training the trainer can now build a hierarchy of tasks, sequences and primitives as required (see figure 3).

4.5 Action Selection

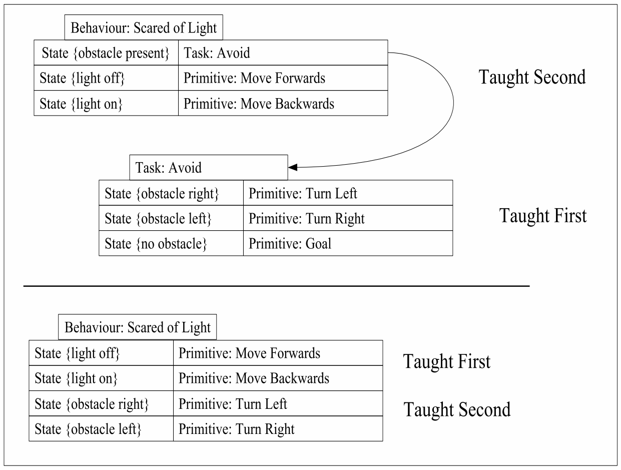
The trainer by constructing a hierarchy of tasks, sequences and primitives is now effectively building an action selection architecture for the robot. At the top behavioural level a decision is made based on the robot's current state as to what to execute next (based on the kNN selection). If the selection is a primitive or sequence these will be executed and the next state cycle will begin. Alternatively the selection could be a task. Within the task the robot state selects the next appropriate action, which again could be a primitive, sequence or task. Working down through the hierarchy eventually results in the execution of a primitive. Note that each *task* executed in the hierarchy will only terminate when its goal condition is selected based on the current robot state, thus within the lowest selected task the state will be polled after each executed primitive. This method of action-selection is similar to the extended feed-forward free-flow hierarchy proposed by Tyrrell [25], who demonstrates how hierarchical approaches to action-selection can often exhibit better performance than "Strips" style production rule methods. Precedence of one action over another is entirely based on current environmental state. The stored memory state most similar to the current state is chosen at each level within the framework.

5. VALIDATION OF FRAMEWORK

We illustrate the successful functioning of the implemented social learning architecture from using the system on two scaffolded behaviours. The first is simple and illustrates the different ways that a trainer could proceed in training the robot. The second is more complex and shows how a new skill can be added to an existing set of actions. Please note that for reasons of clarity the diagrams only show each unique sequence, task or primitive per memory model. In reality each memory model may have a great many instances of different states for the same sequence, task or primitive.

The first behaviour is called "Scared of Light" and was to train the robot to move forward when a light was off, move backwards when a light was on and avoid bumping into obstacles in all cases (note that the robot had no prebuilt competencies other than the basic set of primitives at this stage). Figure 4 shows two different approaches to the task, the first exploits the hierarchy by seperating the behaviour with an "avoid obstacles" sub-task. The second combines both competencies into one behaviour. Both training regimes are successful, however further training episodes may become more difficult with the latter approach. Training of the robot was carried out by two individuals who had not previously used the system. Observation of each person's approach is informal but illuminating. The first user pre-constructed a possible solution using one behaviour and one task before implementing it on the robot. The second user took an entirely different approach. She first trained the robot to correctly respond to the light and then subsequently added training episodes to cope with the obstacle avoidance behaviour. This resulted in a single behaviour with no sub-tasks. However both users successfully trained the robot to complete the task. As part of our future research we intend to carry out further trials of the system with increasingly complex tasks to ascertain if there is a natural point where users start to automatically construct sub-tasks and scaffold each task appropriately.

The second behaviour is called “Tidy Up”. This behaviour is a proxy for the household robot described in the introduction to this paper. We provide two containers. One we call the “cupboard”, the other we call the “basket”. There are a number of objects either plastic or with copper strips. The training regime is much more complex in this instance (see figure 5). This is not only because there is more to teach but also that we need some negative examples. This is important to ensure consistent behaviour. For example we had to train the robot to do something sensible if it dropped the object. This situation was scaffolded by initially running the “Tidy Up” behaviour (having already trained the robot to grasp the object) and then removing the object from the gripper. At this point we terminated execution and pressed the learning button. We then selected the “GripperOpenArmUp” sequence and then terminated learning. For the initial “Tidy Up” task seven steps, with up to three scaffolding experiences per task and up to fifteen moulding experiences per scaffold were needed. The robot however executed the behaviour successfully.



Behaviour: Scared of Light

State {obstacle present}	Task: Avoid
State {light off}	Primitive: Move Forwards
State {light on}	Primitive: Move Backwards

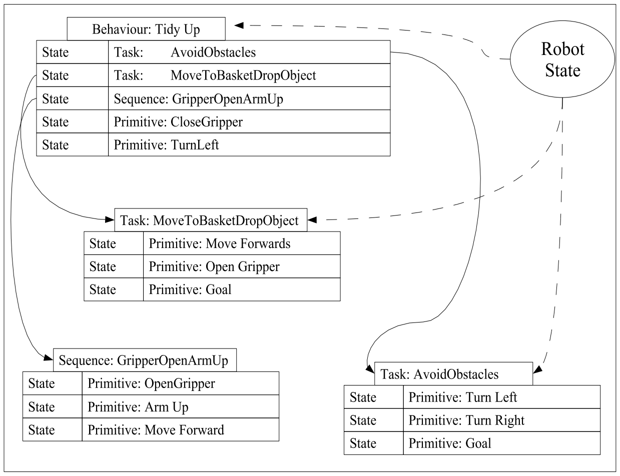
Task: Avoid

State {obstacle right}	Primitive: Turn Left
State {obstacle left}	Primitive: Turn Right
State {no obstacle}	Primitive: Goal

Behaviour: Scared of Light

State {light off}	Primitive: Move Forwards
State {light on}	Primitive: Move Backwards
State {obstacle right}	Primitive: Turn Left
State {obstacle left}	Primitive: Turn Right

Figure 4: Different teaching styles. The upper part of the diagram shows the avoid task being taught first, followed by scaffolding to recognise when to move forward and backward. In the lower part of the diagram the trainer made no attempt to segment the behaviour. All competencies are added to a single behaviour.



Behaviour: Tidy Up

State	Task: AvoidObstacles
State	Task: MoveToBasketDropObject
State	Sequence: GripperOpenArmUp
State	Primitive: CloseGripper
State	Primitive: TurnLeft

Task: MoveToBasketDropObject

State	Primitive: Move Forwards
State	Primitive: Open Gripper
State	Primitive: Goal

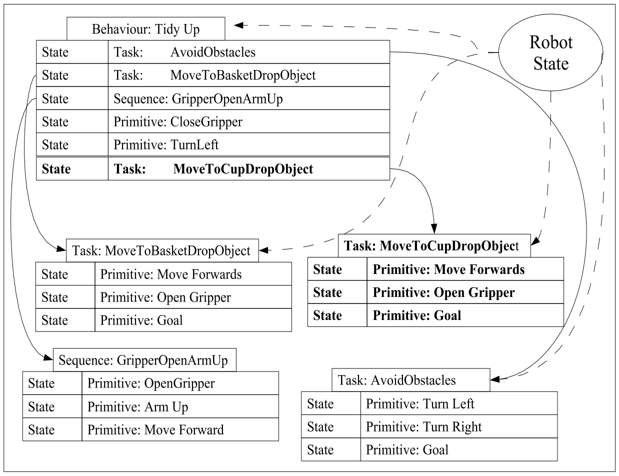
Sequence: GripperOpenArmUp

State	Primitive: OpenGripper
State	Primitive: Arm Up
State	Primitive: Move Forward

Task: AvoidObstacles

State	Primitive: Turn Left
State	Primitive: Turn Right
State	Primitive: Goal

Figure 5: The hierarchy created after successfully training the robot to place plastic containers into the basket (detailed states not shown).


Behaviour: Tidy Up
Robot State
Task: MoveToBasketDropObject
Task: MoveToCupDropObject
Sequence: GripperOpenArmUp
Task: AvoidObstacles

Behaviour: Tidy Up

State	Task: AvoidObstacles
State	Task: MoveToBasketDropObject
State	Sequence: GripperOpenArmUp
State	Primitive: CloseGripper
State	Primitive: TurnLeft
State	**Task: MoveToCupDropObject**

Task: MoveToBasketDropObject

State	Primitive: Move Forwards
State	Primitive: Open Gripper
State	Primitive: Goal

Task: MoveToCupDropObject

State	**Primitive: Move Forwards**
State	**Primitive: Open Gripper**
State	**Primitive: Goal**

Sequence: GripperOpenArmUp

State	Primitive: OpenGripper
State	Primitive: Arm Up
State	Primitive: Move Forward

Task: AvoidObstacles

State	Primitive: Turn Left
State	Primitive: Turn Right
State	Primitive: Goal

Figure 6: The hierarchy has been extended to allow the robot to succesfully place copper objects in the cupboard whilst still placing plastic objects in the basket (detailed states not shown).

Figure 6 shows the “Tidy Up” task extended by training the robot to recognise the copper objects and placing them in the “cupboard”. The training sequence here involved creating a new task “MoveToCupDropObject”, extending the “Tidy Up” task to execute the “MoveToCupDropObject” task if the robot could see the cupboard. Two negative examples were also required. The robot is trained to ignore the cupboard if it has the plastic object. Similarly it is trained to ignore the basket if has the copper object.

In some of the training episodes there were indications that suggested that some tasks can be very difficult to demonstrate. For example, the alignment of the robot to successfully pick up a film canister must be precise. If the robot is too close the gripper cannot grasp it, if the robot is slightly misaligned the canister can be knocked over. Demonstrating this ability to the robot as a sub-task proved difficult as the range of possible state attributes was very small in this instance. We think that it may be that certain useful component tasks such as these may be better defined pre-coded as basic primitives i.e. as factory presettings.

6. DISCUSSION

We have described and implemented a robot social learning architecture, inspired from the study of social animals, that allows a human trainer to teach a physical robot without explicit programming. The teaching is based on building up hierarchical sets of reusable competences via interactive scaffolding. Each competence is based on the assumption that experiences captured by the robot as a result of directed human training can be re-applied when the robot experiences a new situation which is similar to those in its set of stored experiences. Thus it “self-imitates”, generalising by reproducing its own behaviour in new contexts. The training takes place in real-time and although relatively new the architecture appears to scale from simple to moderately complex tasks successfully. However further experimentation to access performance on tasks of very high complexity will be necessary.

To date we have also obtained limited feedback on the use of the system by non-roboticists where informal tests have indicated that it may not be obvious to a non-technical trainer that a robot may need a developmental program to learn to carry out complex tasks. Although this seems a natural assumption which is made when training other adults, children or animals. This may be simply due to inexperience with "intelligent" machines or that the robot itself does not "advertise" the fact that it lacks basic skills. Machines up to now have been engineered mostly to work precisely as specified, they are usually not expected to have to be taught or developed in any way.

In our future research we intend to further study these issues and also use the architecture to further investigate how robots could learn from each other.

7. ACKNOWLEDGMENTS

The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

8. REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders. An approach for programming robots by demonstration: Generalization across different initial configurations of manipulated objects. In *6th IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA'05)*, pages 61–66. IEEE, 2005.

[2] D. C. Bentivegna and C. G. Atkeson. A framework for learning from observation using primitives. In *Proc. RoboCup Int. Symp., Japan*, 2002.

[3] A. Billard and K. Dautenhahn. Experiments in learning by imitation - grounding and use of communication in robotic agents. *Adaptive Behaviour Journal*, 7(3/4), 1999.

[4] R. W. Byrne. *The Thinking Ape: Evolutionary Origins of Intelligence*. Oxford University Press, 1995.

[5] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. Timbl:tilburg memory-based learner. Technical Report ILK 04-02, Tilburg University, 2004. Available from http://ilk.uvt.nl/.

[6] K. Dautenhahn. Getting to know each other – artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 16:333–356, 1995.

[7] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47:109–116, 2004.

[8] M. Dorigo and M. Colombetti. *Robot Shaping: an experiment in behavior engineering*. MIT Press, 1998.

[9] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proc. Int. Symp. Intelligent Robotic Systems, Grenoble*, pages 198–204, 1994.

[10] H.Miyamoto and M.Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11:1331–1344, 1998.

[11] J.A.Ijspeert, J.Nakanishi, and S.Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE Int. Conf. Robotics and Automation*, 2002.

[12] T. M. Mitchell. *Machine Learning*. McGraw-Hill International, 1997.

[13] B. R. Moore. *Social Learning in Animals: The Roots of Culture*, chapter The Evolution of Imitative Learning, pages 245–265. Academic Press Inc., 1996.

[14] C. L. Nehaniv and K. Dautenhahn. The Correspondence Problem. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press, 2002.

[15] M. N. Nicolescu and M. M. Matarić. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 31(5):419–430, 2001.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] R.E.Fikes and N.J.Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[18] R.S.Fouts, D.H.Fouts, and T. Cantfort. *Teaching sign language to chimpanzees*, chapter The infant Loulis learns signs from cross fostered chimpanzees, pages 280–92. State University of New York Press, 1989.

[19] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proc. Ninth Int. Conf. on Machine Learning*, pages 385–393. Morgan Kaufmann, 1992.

[20] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. An experimental comparison of imitation paradigms used in social robotics. In *Proc. IEEE Robot and Human Interactive Communication (ROMAN '04)*, pages 691–696. IEEE Press, September 2004.

[21] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. An examination of the static to dynamic imitation spectrum. In *Proc. 3rd Int. Symp. on Animals and Artifacts at AISB 2005*, pages 109–118, 2005.

[22] S. Schaal, A.Ijspeert, and A. Billard. *The Neuroscience of Social Interaction*, chapter Computational approaches to motor learning by imitation, pages 199–218. 1431. Oxford University Press, 2004.

[23] T.M.Caro. Predatory behaviour in domestic cat mothers. *Behaviour*, 74:128–47, 1980.

[24] T.M.Caro and M.D.Hauser. Is there teaching in non-human animals? *Quarterly Review of Biology*, 67:151–74, 1992.

[25] T. Tyrrell. Computational Mechanisms for Action Selection, PhD Thesis. Technical report, Centre for Cognitive Science, University of Edinburgh, 1993.

[26] M. van Lent and J. E. Laird. Learning procedural knowledge through observation. In *K-CAP 2001: Proc. Int. Conf. Knowledge Capture*, 2001.

[27] J. V. Wertsch. *Vygotsky and the Social Formation of the Mind*. Harvard University Press, 1985.