From exploration to imitation: using learnt internal models to imitate others

Anthony Dearden and **Yiannis Demiris**[1]

Abstract. We present an architecture that enables asocial and social learning mechanisms to be combined in a unified framework on a robot. The robot learns two kinds of internal models by interacting with the environment with no *a priori* knowledge of its own motor system: internal object models are learnt about how its motor system and other objects appear in its sensor data; internal control models are learnt by babbling and represent how the robot controls objects. These asocially-learnt models of the robot's motor system are used to understand the actions of a human demonstrator on objects that they can both interact with. Knowledge acquired through self-exploration is therefore used as a bootstrapping mechanism to understand others and benefit from their knowledge.

1 Introduction

A robot, like humans and other animals, can learn new skills and knowledge both asocially, by interacting with its environment, and socially, by observing the actions of other agents [23, 20]. Interaction enables a robot to learn basic low-level models about its own motor system - for example, the appearance of its motor system and how it is controlled [1]. There is, however, a limit to what a robot can learn efficiently just from its own actions. To learn higher-level models, involving sequences of actions or the position of interesting objects for example, the role of other agents in the robot's environment becomes important. Social learning mechanisms such as imitation have been shown to be a powerful way to transfer knowledge from one agent to another [5, 22]. In robotics this has the particular advantage of relieving the user of the necessity of programming hard-coded knowledge, and instead allowing them to teach actions or movements by demonstration.

Many existing asocial and social models of learning in robotics are based, to varying degrees, on psychological or neuroscientific models of learning in animals, and in particular humans, e.g. [24, 18, 4]. The benefit of turning to the biological sciences for inspiration in robotic learning architectures is clear. Human infants are capable of effortlessly combining learning from both their own interactions, and the actions of a caregiver. Both asocial and social learning methods have previously been studied separately in robotics. In this paper, we present an architecture that enables these learning mechanisms to be combined in a unified framework. The underlying components of this architecture are internal models, internal structures or processes that replicate the behaviour of the robot's environment [11]. In this work we describe how the robot can learn two specific kinds of internal models: Internal Object Models (IOMs), which model the state of objects such as the robot's or a demonstrator's motor system, and Internal Control Models (ICMs), which model how the state of these objects can be controlled by the robot.

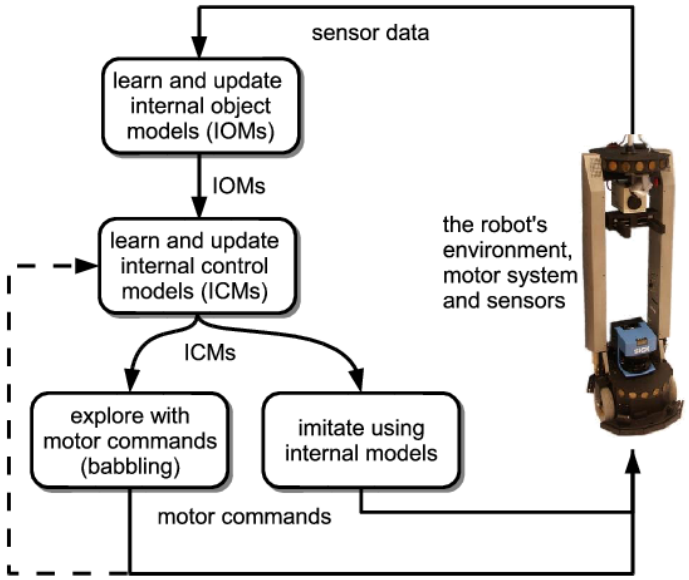

Figure 1. Overview of the learning software.

Drawing inspiration from motor babbling in infants [13], a system is presented that enables a robot to autonomously learn internal models with no *a priori* knowledge of its motor system or the external environment. Using the HAMMER architecture [5], the models that the robot learns of its own motor system are used to understand and imitate the actions of a demonstrator. Although learning is possible from observing movements, for example gestures, that do not involve interacting with objects, we are particularly interested in object manipulation.

Figure 1 shows an overview of the software components controlling the robot. Although the results are divided between the sections in this paper, each component runs simultaneously on the robot. Figure 2 shows the experimental setup. The robot used was an Activemedia Peoplebot, a mobile robot with a pan-tilt camera and a gripper.

[1] Department of Electrical and Electronic Engineering
BioART group, Imperial College London
E-mail: {anthony.dearden99, y.demiris}@imperial.ac.uk

in Proceedings of the AISB-2007, pp. 218-226, (2007)

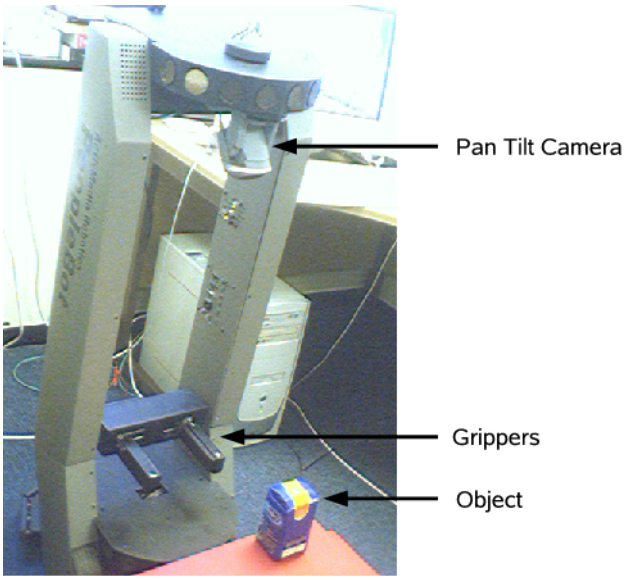

Figure 2. The experimental setup.

2 Discovering internal object models from visual data

Before a robot can learn *how* to control its environment, it needs to be able to *model* its environment. The robot's environment here is considered to consist of:

1. Its own motor system;
2. External, independent objects that its motor system can interact with;
3. The motor system of other agents.

IOMs are used by the robot to track and represent the state of these objects. There are clearly more properties that could be modelled, such as the position of walls, but these are not needed by a robot to imitate actions applied to objects.

In this work we are interested in vision-based robots - vision offers the richest information about the scene, despite the complexities involved in processing. A visual tracking system such as colour histogram-based tracking or even a full 3D tracking system could be used to find and track objects. The robot is much more autonomous, however, if it can discover objects for itself. Instead of being told about the appearance of objects, it would be able to learn about their appearance from the low-level vision data it receives. In [6, 14], visual knowledge acquired through experimentation and segmentation of motion history images is used at the image processing level to find interesting regions, which can be classified as objects. The focus in this work, however, is not currently on how new objects could be discovered and classified through interaction, but how they can be controlled and and used for imitation.

Algorithm 1 runs online to learn IOMs, with low-level input from the movement of pixel-level features in the scene tracked using the KLT optical flow algorithm [12]. Instead of calculating the optical flow for every point in the image, which would be inefficient and inaccurate, only corner features are tracked; these points are the easiest to track robustly. New points are automatically tracked and dropped as the robot's camera moves or new objects enter the scene.

Algorithm 1 Learning IOMs from optical flow data

- The input is a list of tracked optical flow points. Each point, p, is defined by its position and velocity in 2D space, {x,y,dx,dy}.
- The output is a list of objects. Each object is defined as the mean and covariance of its state, O = {X,Y,DX,DY}.
- If objects have previously been detected:
  - Given the previous state of the object, O[t-1], estimate its current state, O[t]. This prediction can be done using basic dynamic information, or if they have already been learnt, using a forward prediction from the internal models given the previous motor commands.
  - For each optical flow point, on each existing object, O[t], calculate the probability this point is part of that object - P(p | O[t]).
  - If P(p | O[t]) is greater than a threshold probability, *pthresh*, assign it to object *O*.
- Whilst there are unassigned points:
  - Create a new object O_{new} using one unexplained point as a 'seed'.
  - Add other points for which P(p | O_{new}) is greater than the threshold probability, *pthresh.*
  - Update the mean and covariance of the object's state.
  - Repeat until all points are modelled, or no more points can be successfully modelled.
- Update the mean and covariance of each object's state with the new sensor data.

Algorithm 1 details how the IOMs are created and tracked by recursively clustering tracked points together. Unlike other clustering algorithms, such as K-means, the number of clusters does not need to be specified beforehand - this is important, because the robot should be capable of adapting to different numbers of objects. Instead, a probabilistic threshold of the variation in optical flow determines when points are added to or removed from IOMs - a value of 0.7 was found to work well.

The shape of objects can be estimated by fitting a convex hull to the clustered points, and by using the mean and the covariance of all optical flow points clustered to an object. The elements of the state vector of an IOM is defined by its position, size and shape. It is not just objects that can be tracked by this algorithm; the pan and tilt movement of the camera is tracked by clustering according to tracked points' velocities.

Clearly, objects cannot be detected unless they move. If the objects are part of the robot's own motor system, then it can discover them as it issues motor commands. If they are objects the robot could only interact with indirectly (such as the object in figure 3), then the robot has to either nudge into it, or be shown to it by a human teacher by shaking or waving the object.

Figure 4 shows the tracking of objects in an experiment. The robot's grippers are detected as soon as it starts to explore its motor system. The human hand and the object is detected when the human teacher moves. Figure 5 shows how the robot can also detect non-motor system objects by disturbing them with its own motor system.

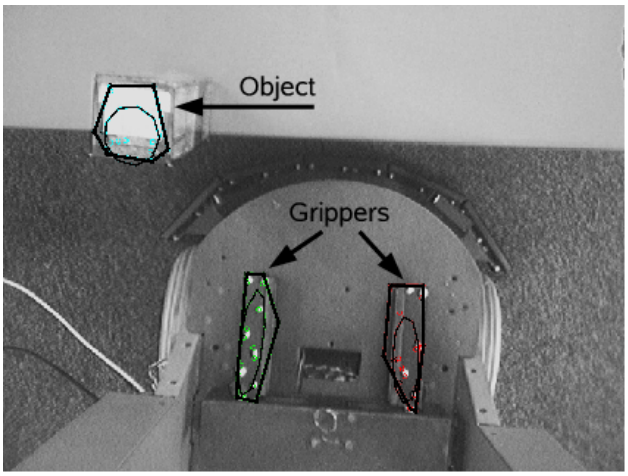

Figure 3. Moving image regions are clustered together; these regions are the robot's IOMs - internal models of where objects are in the scene. In this example, the grippers were moved by the robot, and the biscuit box object was shaken by a human demonstrator to make the robot aware of it. The thick black lines are the convex hull, and the thin ellipse shows represents the mean and covariance of the optical flow points' positions.

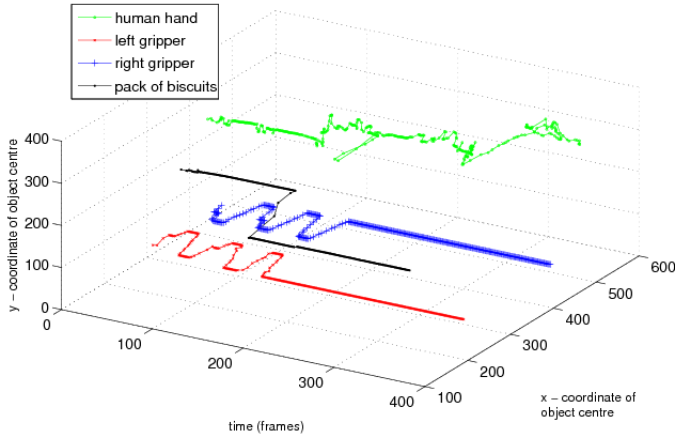

Figure 4. The movement of the IOMs in an experiment, as the grippers open and close and a human hand pushes a box of biscuits.

2.1 Classifying IOMs

A robot cannot imitate until it knows:

1. What it should imitate with;
2. Who to imitate;
3. What objects the imitation should involve;

This is equivalent to classifying objects in the environment according to how they can be controlled. The three kinds of IOMs are: *self* IOMs, objects that are part of the robot's own motor system and can be directly controlled; *demonstrator* IOMs, objects that are part of the demonstrator's motor system and cannot be controlled; and *shared* IOMs, objects that both the demonstrator and the robot can control indirectly. The imitation task considered here is for the robot to replicate, using its own motor system, the actions that the demonstrator takes on a shared object.

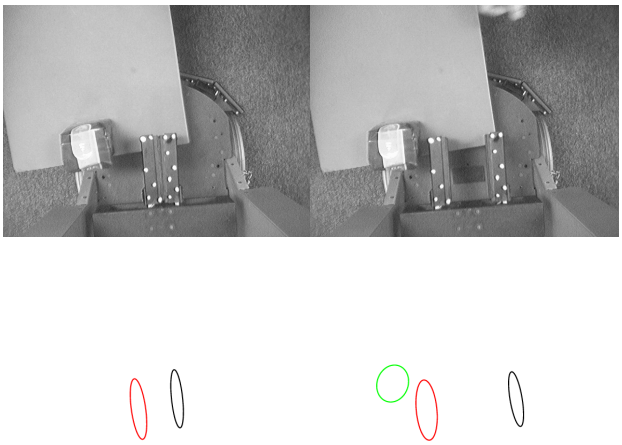

Figure 5. The robot can discover objects by moving them with its own motor system. The top images show frames from the robot 'babbling' in the environment. The bottom frames show the IOMs the robot has discovered before and after the movement.

The robot can learn to distinguish self IOMs from the other IOMs using the ICMs it has learnt for how to control IOMs. If a robot can directly control the state of an IOM, then it can classify it as its own motor system. Differentiating between active, *demonstrator* IOMs and passive, *shared* IOMs is more difficult because the robot can control neither. To solve this problem, the order in which objects are discovered is used. *Shared* IOMs do not move of their own accord, and therefore must be discovered by either being moved by the demonstrator or the robot. Therefore if an object is discovered close (less than 10 pixels) to the position of an existing object, it is classified as a *shared* IOM.

3 Internal control models

ICMs are used by a robot to model and learn how its motor command changes the state of IOMs. They are used as *forward models* to predict the consequences of its motor actions, or as *inverse models* to estimate the motor commands that will lead to a desired object state [1]. Coupling inverse and forward models gives a robot the ability to perform internal simulations of actions before physically executing them; through the *Simulation Theory* approach of the HAMMER architecture, these internal simulations can be used for action recognition and imitation [8, 17, 4].

A learnt ICM will not be able to completely accurately model a robot's motor system - errors will occur because of incorrect models, insufficient or noisy training data or the necessarily simplified internal representations of the model. The system that is being modelled may itself be stochastic. To overcome this uncertainty, it makes sense for an ICM to include information regarding not just its prediction, but how accurate it expects that prediction to be. This inaccuracy can be modelled by representing the internal model as a a joint probability distribution across the motor commands and and the state of elements of the robot's environment. The uncertainty in the model can be estimated from the variance of this distribution. Giving the robot information about the uncertainty of its internal models enables it to estimate how accurate, and therefore how useful, its internal models' predictions are - if multiple models are learnt, their predictive ability can be compared using the variance of their predictions. Section 5

shows how the robot can also use the variance in prediction to guide its exploration.

The basic elements of ICMs are the robot's motor commands and the state of the objects it has discovered - which are either part of its motor system or other objects. ICMs represent the causal structure

Random variable	Description
$M_{1:N}[t-d]$	Motor commands for *N* degrees of motor freedom, with different possible delays, *d*
$S_x[t]$, $S_y[t]$, $S_{dx}[t]$, $S_{dy}[t]$ ...	The state of each object - its position and velocity. For more complex objects, more statistical information can be calculated from its convex hull
$S_x[t-1]$, $S_y[t-1]$, $S_{dx}[t-1]$, $S_{dy}[t-1]$ ...	The state of each object at the previous time step
$P_1[t]$, $P_2[t]$	Proprioception information from other sensors, such as the touch sensors on the robot's grippers

Table 1. The variables the robot can use for its internal model. The robot has to learn Bayesian network structures and parameters using these variables as nodes on the network.

of how these elements interact as a Bayesian network [19]. Bayesian networks are used in [7] to model how infants develop and test causal relationships. Here, we have taken this idea and applied it to the motor system of the robot. Figure 8 in section 4 shows an example of the Bayesian network structures that the robot learns. The motor commands and state of the IOMs are the random variables (nodes) in the Bayesian network, and the causal relationships between them are represented with arcs. The Bayesian network represents a learnt probability distribution across *N* possible motor commands, $M_{1:N}[t-d]$, the current states and previous states of the each object $S_x[t]$, $S_y[t]$, $S_{dx}[t]$, $S_{dy}[t]$, and the state of the proprioception feedback from the robot (e.g. gripper touch sensors). The variable *d* represents the delay between a motor command being issued and robot's state changing; in real robotic systems it cannot be assumed that the effect of a motor command will occur after just one time-step, so this is a parameter that the robot must model and learn. Table 1 shows the possible components of each internal model's Bayesian network. A benefit of using Bayesian networks to represent internal models is that their causal structure is understandable by a human. They can therefore be used to verify the correctness of what the robot is learning.

3.1 Learning through exploration

Practically any environment a robot works in will change, or have properties which cannot be modelled beforehand. Even if the environment is assumed to be completely predictable, endowing the robot with this knowledge may be beyond the abilities or desires of its programmer. A truly autonomous robot, therefore, needs to be able to learn and adapt its own internal models of its external environment. Unlike most machine learning situations, a robot has active control over the commands it sends to its as yet unknown motor system; this situation, where a learner has the ability to gather its own training data, is referred to as active learning [9]. Having the ability to interact with the system you are trying to model has the advantage that the data can be selected either to speed up the learning process, or to optimise the learnt model to be most useful for a particular task. The simplest way for a robot to learn about its environment through interaction is to issue random motor commands. This 'motor babbling' was used to learn internal models for a robot's grippers in [1]. A more sophisticated technique is to use an estimate of the ICM's prediction variance as function of motor command, $C(m,t)$. The actual motor command issued is the one expected to minimise this error. This technique was used to learn the control of a pan-tilt unit on both a real robot [2] and a camera in a football game simulation [3].

The decisions a robot makes about how to interact with the environment become more complex as more degrees of freedom (DOF) of the motor system or more exploration strategies are introduced. The robot has to decide what DOF or objects to learn about, not just what motor commands to send to its motor system. Instantly exploring all DOF at same time would take exponentially longer as the number of exploration possibilities increases. It would also lead to many more internal models having to be learnt simultaneously, which is computationally expensive. A developmental approach can be used to control how a robot explores its environment; more specifically, the robot needs to be able to decide on two things:

- When should the current exploration strategy be stopped?
- What should the next exploration strategy be?

We want the robot to realise when its current exploration strategy is not increasing the quality of the models it is learning. This information is available from the model learning system as the rate of change of the most accurate model's prediction variance, $C(m,t) - C(m,t-1)$. When this approaches zero, the robot knows the current exploration strategy is not improving the quality of the model. This is similar to using a 'meta-model' to estimate a predicting model's error to guide exploration [18].

The second question relates to what the robot should do next. The robot's goal is to learn models that explain how objects in its environment move. In the absence of any human intervention, the only cause of this can come from the robot's own interventions. In this situation, the robot can keep on exploring new degrees of freedom. Currently the degrees of freedom a robot explores are released in order of their distance from the vision system (camera movement, gripper movement then robot wheel movement).

3.2 Online learning of multiple internal models

ICMs consist of a structure, which represents how particular motor commands affect particular states of objects, and the parameters of the particular probability distribution being used for the model. Learning the parameters of a particular model is an online learning problem, with motor commands being the input data and IOMs' states being the output data. In the results here two types of distributions were used to represent the conditional probability distributions of the Bayesian network. For discrete motor commands such as the gripper controls, Gaussian distributions were used. The mean and the variance of the distribution are estimated recursively as:

$$\mu[t] = \frac{t}{t+1}\mu[t-1] + \frac{1}{t+1}S[t]$$

$$C[t] = \frac{t}{t+1}C[t-1] + \frac{1}{t+1}(S[t] - \mu[t])^2$$

For continuous motor commands such as the robot's pan-tilt unit control the conditional probability distributions can be represented using the non-parametric LWR algorithm [15]. The results of previous trials are stored in memory and used to predict the consequence of future trials by performing linear regression on the set of data in

Algorithm 2 Learning multiple ICMs

- For the current motor command(s) being explored, multiple internal models are formed for the motor system. Table 1 shows the search space for possible model structures for a given motor command.
- At each timestep, the state of objects, $s_1 ... s_n$, in the scene is estimated by the vision system using algorithm 1.
- Each model predicts what it expects the states of the objects and interactions to be given the previous motor command. This is given as a Gaussian distribution: $P\left(S_1 ... S_n \mid M\left[t-d\right]=m\right) \sim N\left(\mu, C\right)$
- The likelihood of each model's prediction is calculated: $P\left(S_1 ... S_n = s_1 ... s_n \mid M\left[t-d\right]=m\right)$. This gives a metric for how well each candidate model is performing.
  - If processing or memory resources are limited, models with consistently low scores can be removed, as they are unable to predict accurately.
  - Objects which are moving in an unpredictable way, such as humans or objects they are interacting with, will have low likelihoods for all model predictions. This can be used by the robot to find objects which are not part of its motor system, which it may want to interact with.
- If the variance of the most accurate model's prediction converges, i.e. $C\left(m,t\right) - C\left(m,t-1\right) \approx 0$, then the robot's exploration of this motor command is not improving the accuracy of model. This is the cue to try a new exploration strategy.

memory, which is weighted according to its distance from the query point. Various other distribution types exist that can be learnt online but these methods were chosen principally for their quick convergence properties and ease of implementation [16].

The learnt structure of the Bayesian network represents which motor commands control which objects. The task of the robot is to search through the space of structures connecting every possible random variable to find the one that maximises the likelihood of the sensor data given the evidence, which here is the state of the objects given the sensor data. In this situation, learning the structure is simplified by the fact that the most recently observed change can be most likely explained by the most recent motor command issued. Furthermore, motor commands are always the parent node of the Bayesian network, as none of the other variables being modelled can influence it.

The online internal model learning system works by simultaneously training multiple possible internal model structures, and is described in algorithm 2. One difference between the models learnt here and those learnt by similar systems such as mixture of experts [10], is that there is no need for a responsibility estimator module to decide when each individual internal model should be used. Instead, as each model learns to estimate what the variance of its prediction is, $C\left(m,t\right)$, the 'responsible' model is chosen as the one with the smallest variance for a given prediction.

As multiple ICMs are trained, their prediction variance converges. In the experiments performed here, using models for estimating different delays in the motor-sensor system, the model which predicts most accurately is for the delay d=5 timesteps, equivalent to 0.33 seconds. This is reasonable given the latencies of the motor system and the lags which are present in the vision capture system. Figure 6 shows how this model's prediction varies as it is being learnt . The

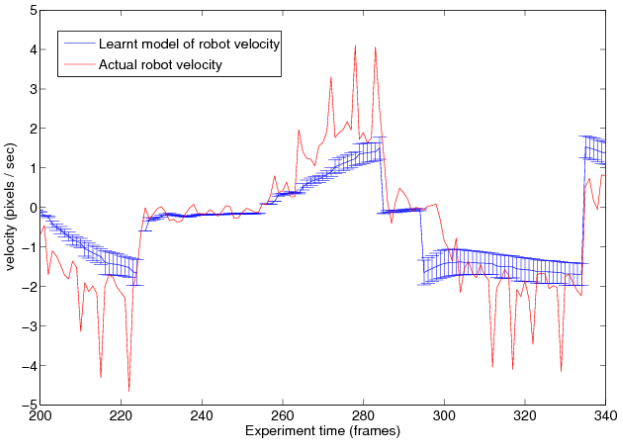

Figure 6. The robot learns online the mean and variance (shown with the error bars) of its velocity as it 'babbles' forwards and backwards. This is the prediction from the most accurate model, for which d=5. The large spikes in the actual data are because of dropped frames from the camera; the robot models this as noise.

error bars on the graph show the variance in the prediction, $C\left(m,t\right)$. Figure 7 compares two model structures being learnt for the *wheel velocity* motor command, which moves the robot forwards or backwards. Interestingly, the model it learns relates to how the motor command affects the position of objects in its environment: moving forward makes objects in front of it move closer. Figure 8 shows the structures of the internal models which the robot learns to be the most accurate for predicting the effects of its *gripper* and its *wheel velocity* motor commands.

This learning system is similar to the HAMMER architecture [5], used by the robot to perform imitation with learnt models in section 4, as it involves multiple competing internal models. The difference when learning is that the command fed to the motor system is not related to the models' predictions. Instead the predicted variance, $C\left(m,t\right)$, and its rate of change, $C\left(m,t\right) - C\left(m,t-1\right)$, is used by the active learning system to control how the robot interacts with the environment.

4 Imitating interactions using learnt internal models

The previous sections introduced the two types of internal models a robot learns from exploration: models of the objects in its environment, IOMs, and models of how to control them, ICMs. The HAMMER architecture presented here allows the robot to use these models to learn how to manipulate objects by observing the actions a demonstrator takes; we assume here that the robot has already learnt to classify IOMs, as discussed in section 3, so it knows the object to imitate (the demonstrator), the object to act with, and the *object* the action is performed on.

ICMs can be used directly as inverse models to imitate movements [1], but their usefulness is limited; they only model low-level motor commands and the sensory consequences over short time periods. The robot is unable to learn long term models from exploration because the motor commands it has available to explore with are all low-level commands: we are not assuming the existence of higher-level pre-programmed 'motor primitives' that control complex movements over multiple degrees of freedom.

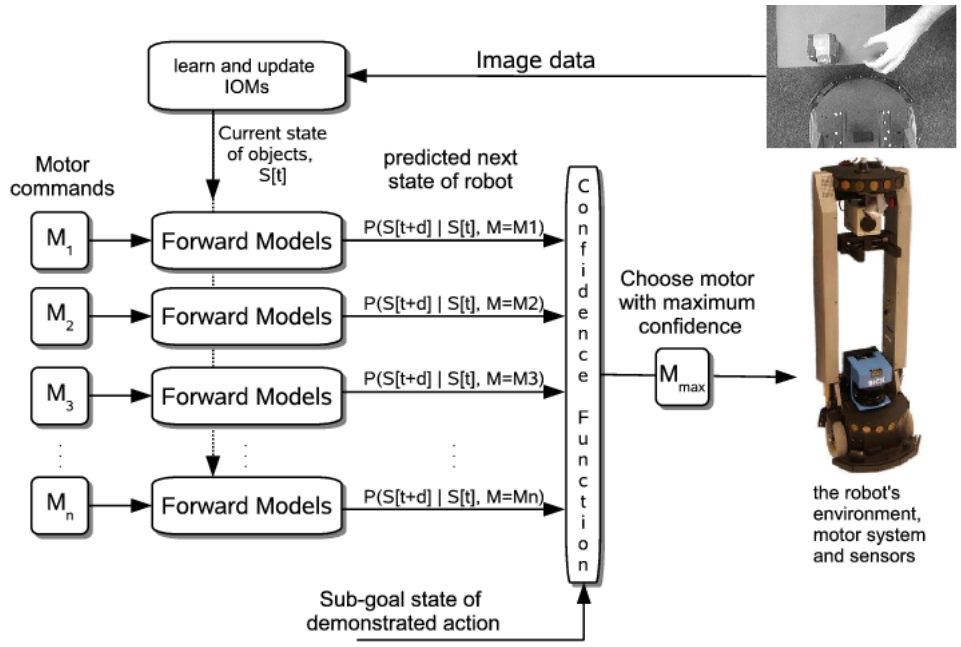

Figure 9. The imitation architecture, using internal models learnt through exploration.

Despite being of limited use on their own, asocially learnt internal models provide the building blocks of the imitation architecture, shown in figure 9 . A generative approach to imitation is used: the internal models of the robot's motor system are used to understand the observed movements of a demonstrator by generating motor commands that will produce the closest match to this movement. The most important part of the system is the forward models, which predict how a motor command will change the state of objects.

These forward models are created from the learnt ICMs, and enable the robot to simulate numerous different motor commands. In the current set of experiments, the total number of commands is sufficiently small that each possible motor command can be simulated. In general, with limited computational resources and more degrees of freedom, this will not be the case. Future work will use the ICMs as inverse models to provide a smaller subset of relevant motor commands to simulate.

Internal models are learnt relative to the robots own visual system, so it has no way of directly understanding the actions it perceives others taking. Indeed, the robot's own motor system may not be capable of imitating the complex gestures and actions of a human motor system because of the different morphology. To overcome this 'correspondence problem', the observed action is represented, not using the states of the objects, but by the *difference* between the states of IOMs. This enables the interaction between the demonstrator and the shared object to be modelled in the same coordinate system as the interaction between the robot and the shared object.

The information about object interactions is a continuous stream of data. To perform the imitation at a more abstract level the sequence is split into sub-goals using peaks in the spatio-temporal curvature of the interaction distance between objects, as shown in figure 10. This technique is used in [21] to perform gesture recognition of humans by splitting the action into a sequence of movements. It is used here to find a sequence of interactions between objects; each element in the sequence is a sub-goal for the robot to imitate. By breaking a continuous stream of interaction data up into a set of key points in the interaction, the represented action and imitation is now independent of the specific timings involved in the movement - for most actions, it is the sequence of states in the movements that are important, not the time between the movements. Splitting a demonstration into a sequence also means it can easily be recognised if demonstrated again. Figure 11 shows screen-shots of the first three sub-goals extracted from an object interaction.

The confidence function's role is to assign a value to each possible motor command according to how close the robot estimates it will move it to the current sub-goal state. The confidence of each motor command, *m*, is calculated as:

$$confidence(m) = exp\left(-\left(abs\left(\widehat{S}_{self,m} - \widehat{S}_{shared,m}\right) - G_n\right)^2\right)$$

where G_n is desired interaction distance of the current sub-goal, and $abs\left(\widehat{S}_{self,m} - \widehat{S}_{shared,m}\right)$ is the predicted distance between the *self* IOM state and the *shared* IOM state. Confidences are higher for motor commands that make the robot's predicted motor system interaction with an object closest to the desired interaction. The confidences displayed in the graphs are normalised to sum to 1 at each time step for easy visualisation. To imitate a demonstrated sequence, the robot uses the motor command with the highest confidence.

The imitation process can be carried out entirely in simulation and visualised to the demonstrator. Figure 12 shows the simulated consequences of the robot imitating the first two sub goals of a demonstrated sequence. The simulation enables the intentions of the robot to be communicated to the demonstrator before executing them. The demonstrator can use this information to stop the robot performing an incorrect imitation, and potentially find out what is incorrect in the robot's knowledge. Future work will involve looking at how the demonstrator can become a more active element in the robot's development by adapting his actions according to visualisations of the

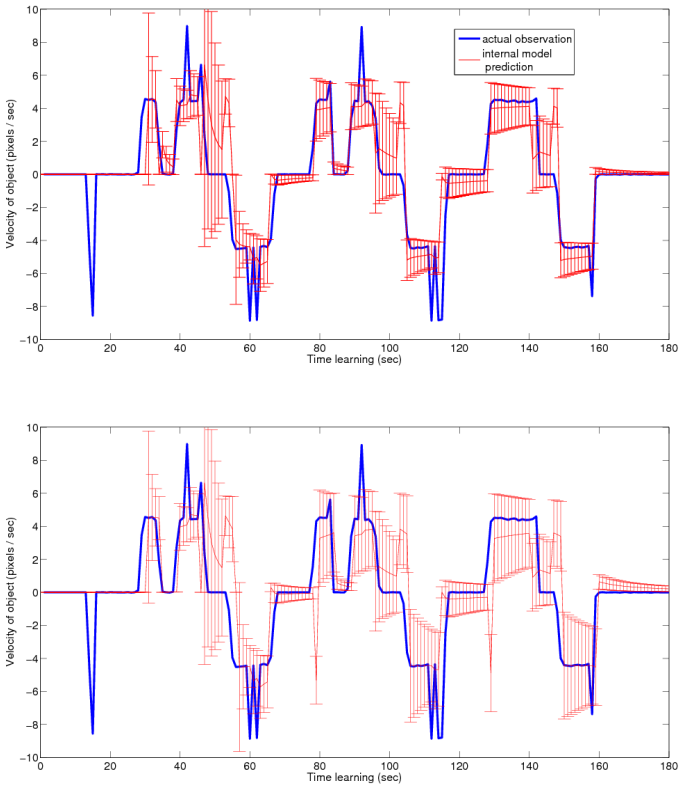

Figure 7. The predictions of two internal model structures for estimating the effect of the velocity motor command as the robot 'babbles' forwards and backwards. The top one can be seen to be the most accurate because it has the lowest estimated prediction variance, shown with the error bars. The structure of this model is shown in figure 8.

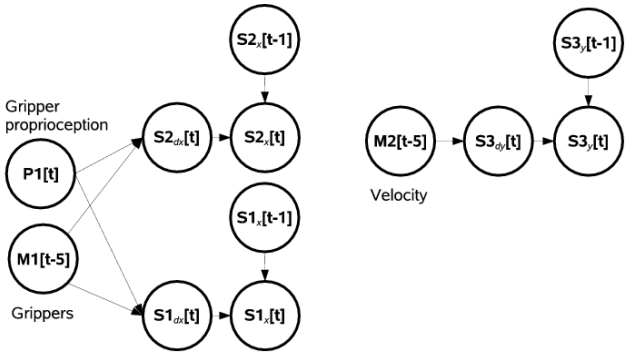

Figure 8. The most useful Bayesian network structures learnt for the gripper motor control (left) and the wheel velocity motor command (right). Both show that the motor commands affect the position of objects in the scene by changing their velocity. It has also learnt that the grippers' touch sensor can be used to predict how the grippers move.

robot's current knowledge. Figure 13 shows the confidence for multiple motor commands in simulation for the first two sub-goals: the robot moves forward, opens its gripper to touch the object, and then closes its gripper to move away.

The same architecture is used to make the real robot imitate an interaction with an object. Unlike the simulation, the state of the robot

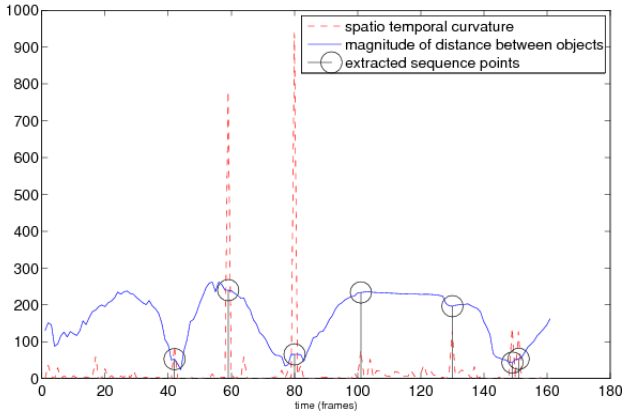

Figure 10. Extracting key points to imitate from an interaction sequence, shown in black circles. These points are extracted from peaks in the spatio-temporal curvature of the distance between the robot's motor system and the object it wishes to interact with.

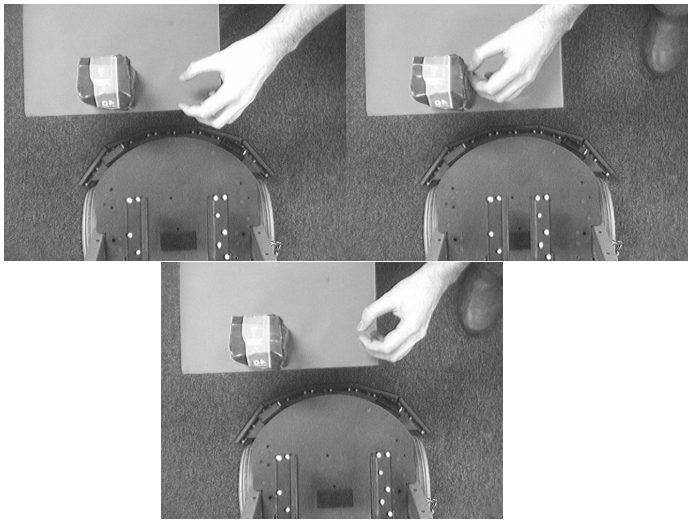

Figure 11. The first three sub-goals being imitated, extracted using the spatio-temporal curvature. Even though this action is occurring as the robot learns to control its gripper system, it is able to recognise it as an interesting action to imitate because neither the human hand nor the pack of biscuits can be accurately explained by its internal models.

and the objects are not updated using the simulation, but with feedback from its vision system. Figure 14 shows the confidence of each motor command as the robot imitates the demonstrated interaction. Figure 15 shows screen-shots from an imitation.

In both the simulation and on the robot the observed interaction is successfully imitated. There are some interesting differences between the real system and the system simulated with the internal models. The real robot finishes the interaction in less time than the simulation. This is due to drift in the simulation, as errors in the internal models accumulate over time. When the gripper is fully open on the real robot, the *open gripper* command receives a lower confidence. As figure 8 shows, the ICMs had learnt during babbling that the gripper proprioception sensor data affected how the grippers move - when the gripper is fully open, the *open gripper* motor

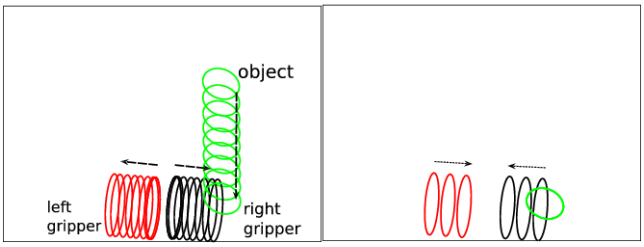

Figure 12. The simulated visualisation of the IOMs as the robot tries to touch the biscuits (left) and then moves away (right). The ellipses represent the means and covariances of the predicted objects' position, and the arrows show the direction of movement. Note that all aspects of this simulation, the appearance of the objects and their control with the motor system, are learnt from exploration. This is why the biscuits do not collide with the gripper; the robot has not learnt that objects can move when touched by other objects.

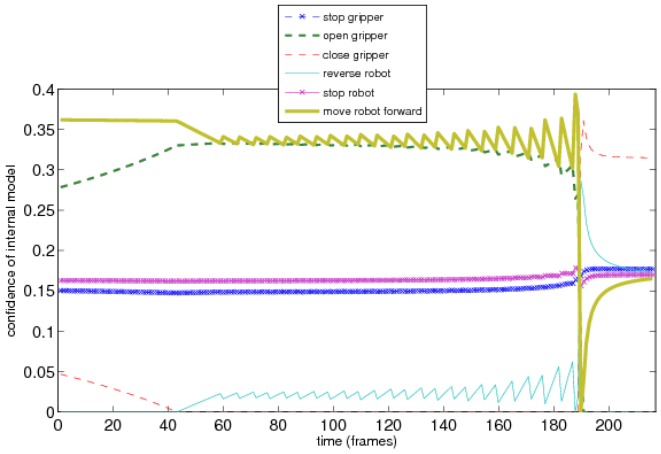

Figure 13. The progress of confidences of each learnt internal model in simulation as the robot tries to touch the object of interest. After this, the first goal state has been reached so the robot moves its grippers away to approach the next sub-goal.

command will not have any effect and will therefore not be useful in achieving the goal of moving the gripper closer to the object. This information is not available, however, in the simulation as the internal models do not currently learn *when* the proprioception information changes, just *how* to use it. The confidence values of the *open gripper* and *move forward* motor commands in the simulated imitation oscillate. This is because the simulation, unlike the robot, does not currently allow multiple motor commands to be issued simultaneously, so the two most appropriate motor commands end up being executed alternately.

5 Discussion

The purpose of both exploration and imitation presented in the experiments here is to enable a robot's knowledge and motor control ability to develop. So far, the process we have described is one-directional: the robot learns basic internal models and uses these to copy interactions on objects that both it and a human demonstrator can control. We are currently looking into the next stages of this teacher-imitator relationship, whereby imitation is not the final goal of the robot, but another process in its developmental repertoire that

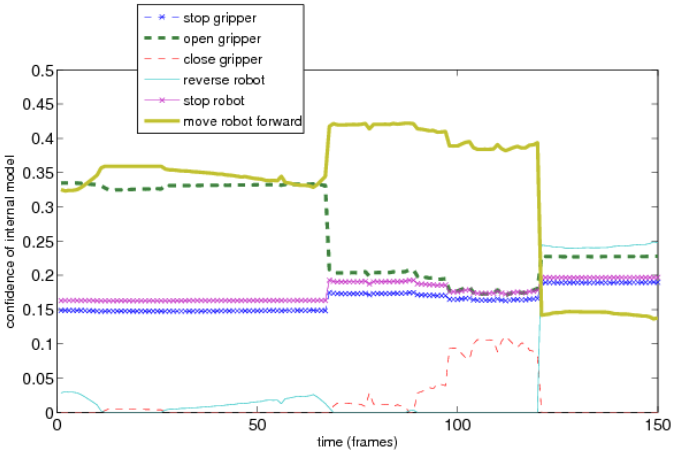

Figure 14. The progress of confidences of each motor command on the actual robot as the robot tries to imitate an interaction with the object.

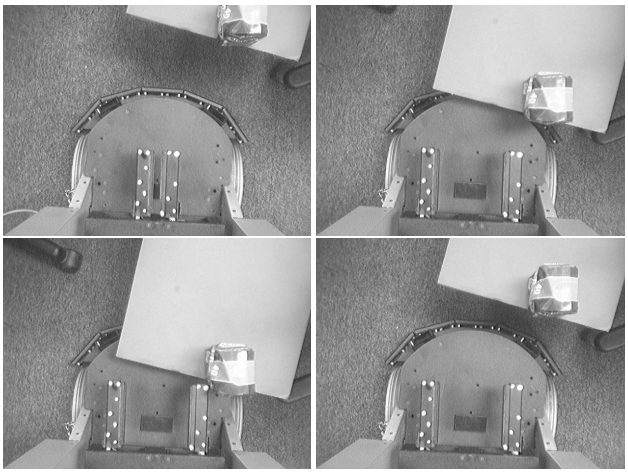

Figure 15. Frames 0, 50, 120 and 150 from the same imitation experiment as as figure 14.

is used to help it to learn.

Further results and experiments are currently being performed for more degrees of freedom in the robot's motor system, such as using its pan-tilt unit. With no *a priori* knowledge, the information available about the interactions is limited by the properties of objects the vision system can represent. Currently this is just the position and size of objects. This is why the only interaction the robot is currently capable of is object 'nudging'. Future work will involve investigating how the robot can attempt different interactions with the same objects so as to learn more detailed ways of interacting. This involves modelling more complex representations of objects and their interactions.

ACKNOWLEDGEMENTS

This work has been supported through a Doctoral Training Award from the UK's Engineering and Physical Sciences Research Council (EPSRC), and through a bursary from BBC Research. The authors would like to thank the members of the BioART team at Imperial College and Dilly Osbahr for their comments.

References

[1] A. Dearden and Y. Demiris. Learning forward models for robotics. In *Proceedings of IJCAI 2005*, pages 1440–1445, 2005.

[2] Anthony Dearden and Yiannis Demiris. Active learning of probabilistic forward models in visuo-motor developmen. In *Proceedings of the AISB*, pages 176–183, 2006.

[3] Anthony Dearden and Yiannis Demiris. Tracking football player movement from a single moving camera using particle filters. In *Proceedings of the 3rd European Conference on Visual Media Production (CVMP-2006)*, pages 29–37, 2006.

[4] Y. Demiris. Imitation, mirror neurons, and the learning of movement sequences. In *Proceedings of the International Conference on Neural Information Processing (ICONIP-2002)*, pages 111–115. IEEE Press, 2002.

[5] Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition (hammer). *Robotics and Autonomous Systems*, 54:361–369, 2006.

[6] Paul Fitzpatrick and Giorgio Metta. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences*, pages 2165–2185, 2005.

[7] A. Gopnik and A. N. Meltzoff. *Words, Thoughts, Theories*. MIT Press, 1st edition, 1998.

[8] R. M. Gordon. Simulation without introspection or inference from me to you. In M. Davies and T. Stone, editors, *Mental Simulation*, pages 53–67. Oxford: Blackwell, 1995.

[9] M. Hasenjager and H. Ritter. Active learning in neural networks. *New learning paradigms in soft computing*, pages 137–169, 2002.

[10] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[11] M Jordan and D Rumelhart. Forward models: Supervised learning with a distal teacher. In *Cognitive Science*, volume 16, pages 307–354, 1992.

[12] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[13] A. N. Meltzoff and M. K. Moore. Explaining facial imitation: A thoretical model. *Early Development and Parenting*, (6):179–192, 6 1997.

[14] G. Metta and P. Fitzpatrick. Better vision through manipulation. In *Proceedings of 2nd International Workshop on Epigenetic Robotics*, pages 97–104, 2002.

[15] A. W. Moore, C.G. Atkenson, and S. A. Schaal. Memory-based learning for control. Technical report, 1995.

[16] Richard E. Neapolitan. *Bayesian Structure Learning*. Prentice Hall, 2004.

[17] S. Nichols and S. P. Stich. *Mindreading*. Oxford University Press, 2003.

[18] P. Oudeyer, F. Kaplan, V. Hafner, and A. Whyte. The playground experiment: Task-independent development of a curious robot. In *proceedings of the AAAI Spring Symposium Workshop on Developmental Robotics*, 2005.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[20] Jean Piaget. *The Child and Reality*. Viking Press Ltd., 111 edition, 1974.

[21] C. Rao and M. Shah. View-invariant representation ans learning of human action. In *Conference on Computer Vision and Pattern Recognition (CVPR'01)*. IEEE, 2001.

[22] S Schaal, A Ijspeert, and A Billard. Computational approaches to motor learning by imitation. *Phil. Trans. of the Royal Society of London B*, (358):537–547, 2003.

[23] Jessica A. Sommerville, Amanda L. Woodward, and Amy Needham. Action experience alters 3-month-old infants' perception of others' actions. *Cognition*, 2005.

[24] Robert W. White. Motivation reconsidered: The concept of competence. *Psychological Review*, 66(5), 1959.